Jeremy T. Bradley (Ed.)

Computer Performance Engineering

LNCS 5652

6th European Performance Engineering Workshop, EPEW 2009
London, UK, July 2009
Proceedings

Springer

Lecture Notes in Computer Science 5652

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Jeremy T. Bradley (Ed.)

Computer Performance Engineering

6th European Performance Engineering Workshop, EPEW 2009
London, UK, July 9–10, 2009
Proceedings

Springer

Volume Editor

Jeremy T. Bradley
Imperial College London
Department of Computing
South Kensington, London SW7 2AZ, UK
E-mail: jb@doc.ic.ac.uk

Library of Congress Control Number: 2009929549

CR Subject Classification (1998): B.8, C.4, D.2.8, D.4.8, H.3.4, K.6.2

LNCS Sublibrary: SL 2 – Programming and Software Engineering

ISSN 0302-9743
ISBN-10 3-642-02923-X Springer Berlin Heidelberg New York
ISBN-13 978-3-642-02923-3 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12715992 06/3180 5 4 3 2 1 0

Preface

This volume of LNCS contains the proceedings of the 6th European Performance Engineering Workshop, held at Imperial College London during July 9–10, 2009. This was the first in the EPEW series to be held in the UK, following on from the highly successful workshops that were held in Toledo (2004), Versailles (2005), Budapest (2006), Berlin (2007) and Palma de Mallorca (2008).

As with previous EPEW workshops, the event was supported by submissions from all over the world, including Asia, the Middle East, North America, as well as Europe. There were 33 submissions in total of which 13 were selected for full papers and four as short papers. I would like to commend the diligent efforts of the Programme Committee, who returned a complete set of reviews – four per paper – which is most unusual. This enabled and enhanced the week-long programme discussion which selected the papers presented here.

The papers themselves maintained the tradition of diversity and quality that the European Performance Engineering Workshop has supported throughout. Papers representing the different fields of performance engineering and analysis, were broadly classified by applications, techniques and formalisms. In the applications domain, we had a significant contribution, I believe for the first time, in the modelling of auctions and markets. There were also contributions on hardware modelling of RAID systems, as well as five papers on performance aspects of cellular and fixed-line networks. New techniques presented included a novel approach to mean value analysis, an application of stochastic ordering to queueing networks and an interesting extension of passage-time analysis. Finally, EPEW has always been supported by researchers using or enhancing stochastic formalisms and this year we had: three papers based around the stochastic process algebra, PEPA; a significant extension to tagged customers in generalized stochastic Petri nets; and a paper looking at representation and analysis of generally distributed stochastic systems.

As Programme Chair, I would like to thank everyone involved in making EPEW 2009 a success: Springer for their continued support of the workshop series, the invited speakers, the Programme Committee and reviewers, and of course the authors of the papers submitted, without whom there could not be a workshop. I would particularly like to thank Uli Harder and Maria Vigliotti who worked extremely hard to bring together the LNCS volume and make all the local arrangements a success. We hope that you, the reader, find the papers in this volume interesting, useful and inspiring, and we hope to see you at future European Performance Engineering Workshops.

July 2009 Jeremy Bradley

Organization

Organizers

PC Chair	Jeremy Bradley
Publicity and Local Co-chairs	Uli Harder and Maria G. Vigliotti

Programme Committee

Jeremy Bradley	Imperial College London (UK)
Mario Bravetti	Università di Bologna (Italy)
Lucy Cherkasova	HP Labs (USA)
Michel Cukier	University of Maryland (USA)
Tadeusz Czachórski	Polish Academy of Sciences, Gliwice (Poland)
Jean-Michel Fourneau	Université de Versailles (France)
Stephen Gilmore	University of Edinburgh (UK)
Marco Gribaudo	Università degli Studi di Torino (Italy)
Uli Harder	Imperial College London (UK)
Armin Heindl	Universität Erlangen-Nürnberg (Germany)
Helmut Hlavacs	University of Vienna (Austria)
András Horváth	Università di Torino (Italy)
Carlos Juiz	Universitat de les Illes Balears (Spain)
Tomáš Kalibera	Purdue University (USA)
Helen Karatza	Aristotle University of Thessaloniki (Greece)
Leïla Kloul	Université de Versailles (France)
William Knottenbelt	Imperial College London (UK)
Samuel Kounev	Technische Universität Darmstadt (Germany)
Catalina M. Lladó	Universitat de les Illes Balears (Spain)
Aad van Moorsel	Newcastle University (UK)
Manuel Núñez	Universidad Complutense de Madrid (Spain)
Fernando L. Pelayo	University Castilla-La Mancha (Spain)
Marco Scarpa	Università di Messina (Italy)
Markus Siegle	Universität der Bundeswehr München (Germany)
Mark Squillante	IBM T.J. Watson Research Center, NY (USA)
Miklós Telek	Budapest University of Technology and Economics (Hungary)
Nigel Thomas	Newcastle University (UK)
Maria G. Vigliotti	Imperial College London (UK)
Sabine Wittevrongel	Universiteit Gent (Belgium)
Katinka Wolter	Humboldt Universität zu Berlin (Germany)
Soraya Zertal	Université de Versailles (France)

Additional Reviewers

Gregorio Díaz Descalzo
Nicholas J. Dingle
Alexander Gouberman
Carlos Guerrero
Michael Kuperberg
Isaac Lera
Luis Llana
Hermenegilda Macià
Mercedes G. Merayo
Philipp Reinecke
Martin Riedl
Zsolt Saffer
Johann Schuster
Connie U. Smith
Valentín Valero
Salvatore Distefano

Table of Contents

Tagged Generalized Stochastic Petri Nets

Gianfranco Balbo[1], Massimiliano De Pierro[1], and Giuliana Franceschinis[2]

[1] Dipartimento di Informatica – Università degli Studi di Torino, Italy
[2] Dipartimento di Informatica – Univ. del Piemonte Orientale, Alessandria, Italy

Abstract. This paper introduces an extension of the Generalized Stochastic Petri Net (GSPN) formalism in order to enable the computation of first passage time distributions of tokens. A "tagged token" technique is used which relies on net's structural properties to guide the correct specification of this extension. The extended model is suited for an automatic translation into an ordinary GSPN that can be used for the first passage time analysis. Scheduling policies of tokens in places, that are neglected in ordinary GSPNs, become relevant in the Tagged Generalized Stochastic Petri Net (TGSPN) formalism and specific submodels are proposed which are then used during the translation from TGSPNs to ordinary GSPNs. A running example inspired by a Flexible Manufacturing application is used throughout the paper to introduce the different concepts and to provide evidence of the relevance of the results.

1 Introduction

Generalized Stochastic Petri Nets (GSPNs) [1] are a modelling formalism very effective in the representation and evaluation of many real systems. Among the advantages of modelling with GSPNs are the precision and the expressive power of their constructs together with the automatic derivation of the Markov chains that represent their underlying stochastic processes and that are used to compute many classical performance indices. The standard analysis of GSPN models amounts to the computation of their transient and stationary state probability distributions that are used to assess their dynamics.

In this paper we report on an initial effort to augment the modelling power of GSPNs providing definitions, constructs and methodologies useful for an easy and straightforward computation of passage times among states of the model. Our work is both an extension and a refinement of that of Dingle and Knottenbelt [7] which use the tagged customer approach to assess the behaviour of GSPN models with "customer-centric" performance measures.

As observed in [7], prior work on this type of approach in the context of GSPN analysis is very limited. Most of the papers presenting the use of the tagged customer method, that is typical of queueing theory [11], in models developed with Petri net [10] or Process Algebra [5] based formalisms require a manual intervention on the original model that is often difficult to perform. Computing passage times between sets of states in Markov chains is a well known problem with limited theoretical difficulties. Disregarding for the moment the computational

J.T. Bradley (Ed.): EPEW 2009, LNCS 5652, pp. 1–15, 2009.

complexity of the solution in the case of very large models, for which specific and powerful tools exist [5,6], the actual difficulty consists in mapping the situations that arise when the tagged customer enters and leaves a specific portion of the model onto the corresponding sets of Markov chain states that represent the borders for the passage time computations. Driven by application purposes, in [7,12] the notion of "customer" is informally brought within the GSPN's domain making it useful for the analysis of specific cases, but leaving uncertain what to do in other situations when models generate peculiar behaviours that require more precise definitions.

In this paper we will recall the concept of "tagged token" introduced in [7] and provide proper and precise methods for defining the beginning and the end of passage times at the GSPN level. Tokens in GSPNs cannot be interpreted, in general, as entities that travel throughout the models, but are instead indistinguishable quantities consumed and generated at transition firing instants. The idea of selecting one of these tokens to make it "tagged" and of following it throughout the net is not a trivial task, unless certain structural properties of the net are exploited. Our proposal envisions that invariant properties, corresponding to conservation conditions within the net (p-semiflows), are computed first in order to identify "circuits" where tokens with indistinguishable behaviour are preserved. Tokens of this type can be treated similarly to the customers of QN models and can thus be tagged by the modeller in order to compute the distribution of the time they take to travel between points of these circuits.

The TGSPN (Tagged Generalized Stochastic Petri Nets) formalism proposed in this paper extends classical GSPNs with primitives that allow to identify the input/output interfaces of a submodel as well as the p-invariant that corresponds to a certain class of tokens. This p-invariant identifies the places where the tagged customer can be found and thus the portion of GSPN that needs to be manipulated in order to allow the first-passage-time analysis.

The tagged token approach allows a richer analysis of the model, but has an obvious computational cost due to the larger state space that it induces. Moreover, the introduction of the tagged customer makes the policies used to manage the tokens in the input places of the transitions relevant for the performance measures obtained with this analysis and it must be explicitly addressed. The way in which we account for this problem builds on the experience of embedding queueing policies in the input place of a timed transition of a GSPN that led to the proposal of Queueing Petri Nets (QPNs) [4,3,8]. We provide compact and parametric representations of the different input queueing policies, specifically tailored to the goal of tracking the tagged token position in the queue. The difference is that in QPNs the special queueing places embed the queue, the server and a sort of output buffer for the tokens that have already received service that we instead exclude from our representations.

The main contributions of this paper are thus (1) a precise extension of the GSPN formalism to guide the identification of the tagged tokens and to support the first-passage-time analysis; (2) the introduction of a number of queueing policies that can be properly analyzed.

The paper is so organized: Sec. 2 is a brief introduction to the GSPN formalism; Sec. 3 introduces the method to identify the "taggable" classes of tokens and provides the TGSPN formal definition; Sec. 4 introduces the method used to specify at the net level the measure of interest; Sec. 5 provides automatic rules which translate the TGSPN into a GSPN model in order to apply the analysis and Sec. 6 extends the specification with queue policies; Sec. 7 illustrates in steps how the analysis is done starting from the new GSPN generated in the previous step; finally, Section 8 compares the proposed approach with works published in the literature.

2 The GSPN Formalism

Generalized Stochastic Petri Nets (GSPNs) extend classical Place/Transition (Petri) nets with timing specifications. GSPNs are used in many application fields for the validation and evaluation of distributed systems characterized by concurrency, synchronization and congestion. A detailed introduction to the formalism and to its applications can be found in [1]. Here we recall the notation and the basic definitions that are useful for the rest of the paper.

Definition 1. A GSPN system is a tuple $(P, T, I^-, I^+, H, \Pi, \mathbf{w}, \mathbf{m_0})$ where:

- $P = \{p_i\}$ is a finite and non empty set of *places*.
- $T = \{t_i\}$ is a finite and non empty set of *transitions*.
  - $E \subseteq T$ (often denoted as $E = \{T_i\}$) is the set of *timed* transitions which fire with a random delay characterized by a negative exponential probability distribution; each of these transitions can be either of single-server, n-server, or ∞-server type;
  - $Z \subseteq T$ (often denoted as $Z = \{t_j\}$) is the set of *immediate* transitions which fire in zero time;
  - $P \cap T = \emptyset$; $Z \cap E = \emptyset$ and $T = E \cup Z$.
- $I^-, I^+ : T \times P \to \mathbb{N}$ are the *input* and *output* functions that correspond to the arcs of the net, and their multiplicity;
- $H : T \times P \to \mathbb{N}^+ \cup \{\infty\}$ is the *inhibition* function that corresponds to the inhibition arcs of the net; $H(t, p) = \infty$ means that no inhibition arc exists from p to t;
- $\Pi : T \to \mathbb{N}$ is the (absolute) *priority* function.
- $\mathbf{w} : T \to \mathbb{R}$ is a function (possibly marking dependent) that assigns to each transition of the net:
  - the *rate* of a negative exponential distribution of the firing delay when transition $T_i \in E$;
  - the *weight* (used in case of probabilistic choices) when transition $t_i \in Z$.
- $\mathbf{m_0} : P \to \mathbb{N}$ is a multiset on P representing the *initial marking* of the net.

A *marking* $\mathbf{m}$ (or state) of a GSPN is a multiset on P.
The set of input places to transition t (also referred to as the *preset* of t), denoted ${}^\bullet t$, the set of output places (or *postset*) of t, denoted $t^\bullet$, and the set of inhibitor places of t, denoted ${}^\circ t$, are defined as follows:

$${}^\bullet t := \{p \in P \mid I^-(t,p) > 0\},\ t^\bullet := \{p \in P \mid I^+(t,p) > 0\}$$
$${}^\circ t := \{p \in P \mid H(t,p) < \infty\}$$

A similar dot notation and similar definitions hold for the places of the net.

A transition t_i has *concession* at marking $\mathbf{m}$ iff $I^-(t_i,p_j) \leq \mathbf{m}(p_j) < H(t_i,p_j)$, $\forall p_j \in P$, where $\mathbf{m}(p_j)$ represents the number of tokens in place p_j in marking $\mathbf{m}$. If no higher priority transitions have concession in $\mathbf{m}$, it is said that t is *enabled* and it can *occur*, or *fire*.

The occurrence of transition t_i yields marking $\mathbf{m}[t_i\rangle = \mathbf{m} + I^+(t_i) - I^-(t_i)$. The occurrence of a *sequence* σ of transitions enabled at $\mathbf{m}$ and yielding $\mathbf{m}'$ is denoted similarly: $\mathbf{m}[\sigma\rangle\mathbf{m}'$.

The enabling degree of t_i in $\mathbf{m}$ (denoted $ed(t_i,\mathbf{m}) = k$) is the greatest natural k such that $kI^-(t_i,p_j) \leq \mathbf{m}(p_j) < H(t_i,p_j)\ \forall p_j$. In a given marking $\mathbf{m}$, a n-server timed transition T_i represents as many activities in parallel, as the $min(n, ed(T_i,\mathbf{m}))$.

A marking where no immediate transitions are enabled is called *tangible*, otherwise *vanishing*. In GSPN models, the tangible markings which are reachable from a given initial tangible marking $\mathbf{m}_0$ form the Tangible Reachability Graph (TRG) and are the states of an underlying Markov Chain (MC) describing the time behaviour of the system. Tangible and vanishing markings which are reachable from a given initial marking $\mathbf{m}_0$ form the Reachability Graph (RG). The transition rates among TRG states are defined in terms of the rates associated with timed transitions and of the weights of the immediate ones.

A p-*flow* $\mathbf{f}$ is a $|P|$-component vector solution of the system of linear equations $\mathbf{f} \cdot C = \mathbf{0}$, where C is the GSPN incidence matrix ($C[i,j] = I^+(t_i,p_j) - I^-(t_i,p_j)$) and $\cdot$ is the scalar product. A solution $\mathbf{f}$ whose coefficients are natural and non-negative numbers is called a p-*semiflow*. The support of a p-semiflow is the set of places corresponding to the positive components of the semiflow itself. In our work we are interested in *minimal* p-semiflows: a minimal p-semiflow is such that it can not be obtained as a positive linear combination of other p-semiflows, which also means that there are no other semiflows whose supports are properly included in the support of the minimal one. When the positive components of a p-semiflow are equal to one it is easy to understand the meaning of the p-semiflow which says that the number of tokens distributed on its support is unaffected (remains unchanged) by the firing of any transition of the net. Formally, this property is captured by the observation that for each p-semiflow of the GSPN $\mathbf{f} \cdot \mathbf{m} = \mathbf{f} \cdot \mathbf{m}_0 = N$ for any marking $\mathbf{m}$ reachable from $\mathbf{m}_0$. N that depends on the initial marking of the GSPN is called a p-*invariant*. The sum notation $\sum_i \mathbf{f}(i).p_i$ is often used to denote the p-semiflow $\mathbf{f}$.

Fig. 1 shows a GSPN model of a Flexible Manufacturing System (FMS). The FMS comprises four manufacturing stations, from M_1 through M_4, where two of them, M_2 and M_3, can fail. Raw parts are loaded on suitable pallets in the Load/Unload station represented by the pair (place p_0 / transition T_0) and are then manufactured being sequentially brought to the four machines. When the manufacturing is completed, the finished part is replaced on the pallet by a raw one and the cycle starts again. A set of repairmen (p_{19}) operate on failed

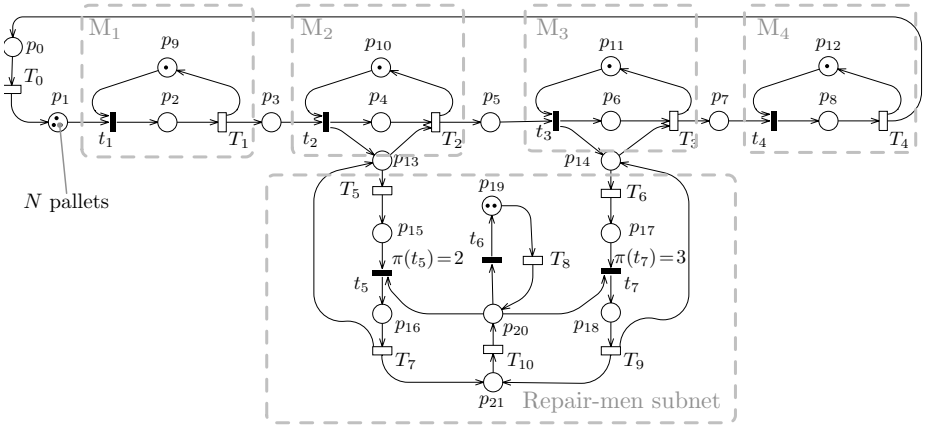


Fig. 1. A FMS with faulting machines and repair-men

machines as illustrated in the lower part of the picture. They cycle between vacation and repair periods. Upon return from a vacation (T_8) a repairman checks if a machine has failed (t_5,t_7) and in such case starts a repair activity (T_7, T_9), otherwise goes back to vacation (t_6). After the repairman ends working on a failed machine, he takes a rest (T_{10}) before starting a new cycle. Priority is given to machine M_3 when a repairman sees that have both failed. An example of minimal p-semiflow of such system is $1.p_2 + 1.p_9$ Which states that, given its initial marking, the system will always show $\mathbf{m}(p_2) + \mathbf{m}(p_9) = 1$.

3 Tagged Token Specification

We already pointed out that in GSPN tokens are "volatile" elements *created* and *destroyed* on transition firing and with no semantic relationship between destroyed and created tokens. First passage times are defined at the level of the CTMC that underlies a GSPN, as measures of the time taken by the chain to move from a set of *start* states into a set of *target* states for the first time (target states are *absorbing*). Since start and target states correspond to specific distributions of tokens on the net places, it is natural to associate passage times to *movements* of certain tokens through the net that thus assume a distinctive identity meaningful for the modeller. Consider, for instance, the FMS model of Fig. 1. Tokens in places p_0, p_1 up to p_8 can be interpreted as the N *pallets* that carry the parts from machine to machine in order to have them manufactured. Similarly, tokens in places p_{16}, p_{18}, p_{19}, p_{20}, and p_{21} represent the repairmen in different states of their activity.

When the model is not trivial, identifying the groups of tokens that represent classes of trackable and atomic objects, as well as the parts of the model where they can flow through, is not a simple task: hence it is useful to give some support to the modeller in this situation.

A rigorous notion of trackable objects can be provided looking for conservative flows of tokens in the net. This approach not only supports the identification of the classes of objects suited for a first passage time analysis, but it also guarantees the correctness of the results, since it defines both a *class of customers* that behave similarly and the *subnet* in which these customers flow.

The formal method that we use for this purpose is the computation of the minimal p-semiflows of the GSPN; to illustrate it let us refer again to the GSPN of Fig. 1. Default (automatically generated) names have been used to identify the places of this net to stress the conceptual separation between the mathematical properties derived from an automatic tool and their interpretation given by the modeller.

The net has eleven minimal p-semiflows related to the different classes of objects. Some of them regard the *machines*, for instance p-semiflow $p_2 + p_9$ concerns M_1 and means that M_1 can be either busy (places p_2 marked), or idle (place p_9 marked). Analogously $p_{10} + p_{13} + p_{15} + p_{16}$ states that machine M_2 can be in one of the following states: idle, busy, waiting for a repairman, being repaired. Instead $p_{16}+p_{18}+p_{19}+p_{20}+p_{21}$ p-semiflow is related to the *repairmen*, has invariant sum 2, and means that a repairman can be either in vacation (place p_{19} marked), in service (place p_{20} marked), working (places p_{16} or p_{18} marked), or resting after a repair (place p_{21} marked). For what concerns the raw pieces being manufactured, p-semiflow $p_0+p_1+p_2+p_3+p_4+p_5+p_6+p_7+p_8$ identifies the *pallets* serving the different machines; its invariant sum is 3.

With these interpretations, we can choose one of these p-semiflows to compute the first passage times of interest. For instance, we could assess the efficiency of the model from the point of view of the average length of the intervals between repairman vacations, or of the probability that the processing-time taken by the pallets to go through the part of the system that may fail (machines M_2 and M_3) remains below a certain threshold.

The distribution of the first passage time is computed for the submodel of interest by means of a tagging procedure which refines the subnet in order to single out the interaction of one of the tokens of a certain class with respect to the others. Starting from a GSPN extended with the indication of the p-semiflow of interest a modified and *equivalent* version can be built which accounts for the identity of one of these tokens and which explicitly encodes it into the net structure. The identified customer is denoted as the *tagged-customer*, while the related token, that carries its identity in the whole net, the *tagged-token*. The extended net captures the dynamics of the tagged customer with respect to the others and can thus answer the performance questions of interest.

The whole tagging procedure can be schematized with the following steps:

1. the minimal p-semiflows of the model are computed and shown to the modeller as eligible classes of customers;
2. the modeller, on the basis of his/her knowledge of the application, selects one of the semiflows, thus identifying the class of customers of interest;
3. the subnet corresponding to the chosen semiflow is modified in order to track one token out of the group for first passage time analysis (see Sec. 5).

The subnet modification foreseen in the last step of the procedure, might require additional information on the characteristics of the model such as the scheduling discipline used to manage the tokens of a queue place. By default a Random Order (RO) policy is adopted, however several other policies may be specified, as will be discussed in Sec. 6, that are irrelevant for the standard analysis, but that become important for the correct computation of the first passage time distribution.

The formal definition of Tagged GSPN (TGSPN) follows:

Definition 2. ***Tagged GSPN:*** *A Tagged GSPN, TGSPN, is a tuple* $\left(\mathcal{N}, \mathbf{f}, p_\circ, \mathcal{Q}\right)$ *such that:*

- $\mathcal{N} = \left(P, T, I^-, I^+, H, \Pi, \mathbf{w}, \mathbf{m}_0\right)$ *is a GSPN;*
- $\mathbf{f}$ *is a minimal p-semiflow of* $\mathcal{N}$*. It identifies the sets* $\mathcal{P} \subseteq P$ *and* $\mathcal{T} \subseteq T$ *such that* $\mathcal{P} = \{p_i \in \mathbf{f}\}$, $\mathcal{T} = \bigcup_{p_i \in \mathcal{P}} {}^\bullet p_i \cup p_i^\bullet$*;*
- $\mathbf{f}$ *is such that if* $p_i, p_j \in \mathcal{P}$ $(i \neq j)$ *then* ${}^\bullet p_i \cap {}^\bullet p_j = \emptyset$*;*
- $p_\circ \in \mathcal{P} : \mathbf{m}_0(p_\circ) > 0$ *is the place where the tagged-token is located in the initial marking* $\mathbf{m}_0$*;*
- *The places in* $\mathcal{P}$ *that have at least one output transition which is timed are called queue places.* $\mathcal{Q}$ *maps each queue place to one of the following scheduling policies: Random Order (RO), First Come First Served (FCFS), Last Come First Served with No Preemption (LCFS), Last Come First Served with Preemption-Resume (LCFS-PR);*
- *The default scheduling policy for queueing places is RO;*
- *Several transitions that share an input place which is a queue place with policy different from RO must satisfy the following conditions: (i) the queue place must be the only input place in their input set; (ii) they must all have the same number of servers; (iii) their rate can't be marking dependent.*

The third condition of Def. 2 is introduced mostly to minimize the amount of additional information that must be specified for the equivalent extended net to be constructed. Relaxing a few of this constraints is presently under investigation, but additional work is needed in order to enlarge the class of treatable models without hampering the robustness of the definition.

4 Net Definition of First Passage Time

One of the most challenging points of this work is to provide a clear and easy to use net level specification of passage times that enables the modeller to define the measures of interest on the net rather than on its underlying CTMC. The passage of customers throughout the net corresponds to *traversal paths* that they follow and along which timing intervals are measured. Focussing the attention on the class of customers identified by a p-semiflow $\mathbf{f}$, we can identify the subnet $\mathcal{N}_\mathbf{f}$ that is a restriction of $\mathcal{N}$ on the sets of places and transitions respectively identified by $\mathcal{P}$ and $\mathcal{T}$, given in Def. 2 of TGSPN. Passage time information is provided identifying, within $\mathcal{N}_\mathbf{f}$, an open (not cyclic) path. In the simplest form

a path is identified by start and end points, however forbidden ways along the route could also be specified. The formulation of start and end points is based on a notion of events that cause the tagged–customer to enter, and subsequently leave, the path of interest. This is obtained by identifying specific transitions of $\mathcal{N}_{\mathbf{f}}$ that we call *entry* and *exit points*, denoted respectively with sets $\mathcal{S}_{in}$ and $\mathcal{S}_{out}$

The connected component $\mathcal{S}$ of $\mathcal{N}_{\mathbf{f}}$ that is defined following the oriented arcs from the entry–points to the exit–points, constitutes the traversal–path along which to compute the passage time: actually the traversal–path $\mathcal{S}$ is a subnet of $\mathcal{N}_{\mathbf{f}}$. For instance, in the FMS model of Fig. 1, considering the activities of the repairmen, the modeller can specify to start the passage-time computation from the beginning of a working cycle (transition T_8) up to the end of a repairing cycle (transitions T_7 and T_9) without taking a vacation again (transition t_6). Possibly, he can specify also one exit–point only, T_7, if he is interested in the traversal time on machine M_2 only.

Forbidden paths are specified through *forbidden exit–point* transitions (denoted by $\mathcal{S}_{forb}$ set). These transitions reduce the size of subnet $\mathcal{S}$ preventing the algorithm that computes this connected component to include undesired paths in the result. Forbidden paths translate into forbidden states of the CTMC used for first passage time computation ([9]). In this work, places of $\mathcal{S}$ cannot be members of the preset ${}^\bullet t$ (respectively postset $t^\bullet$) of an entry–(exit–)point t to keep the derivations simpler and to provide an easier interpretation of the results.

Further refinements on the initial and final states can be provided by means of Boolean expressions on the local markings of places. Given a subnet traversal–path $\mathcal{S}$ consistent with the above definition, start states are characterized by a set of 3-tuples $START = \{\langle t, C_t^{pre}, C_t^{post}\rangle\}$ where $t \in \mathcal{S}_{in}$ and C_t^{pre}, C_t^{post} are Boolean conditions on the marking. The conditions C_t^{post}, C_t^{pre} that must be satisfied by start states and by their predecessors in the (tangible) RG are used to restrict the measure of the first passage time to some specific condition, e.g. one may be interested in the time spent in a given subnet by the tagged customer, conditioned on the fact that no other customer was in the subnet upon its arrival.

One possible specification of a traversal–path is:

- $START = \{\langle t_2, \mathbf{m}(p_6) = 1 \wedge \mathbf{m}(p_{19}) = 2, true\rangle\}$, $\mathcal{S}_{out} = \{T_3\}$: it specifies a traversal path $\mathcal{S}$ denoting the time of a raw piece (the customer of interest) takes to pass through machines M_2 and M_3, given that upon its arrival the machine M_3 was busy working on another piece and all the repairmen were in vacation.

5 Unfolding the TGSPN into a GSPN

In this section we discuss the method for constructing a GSPN *equivalent* to a TGSPN characterized by a (specific) p-semiflow $\mathbf{f}$. As we already pointed out, this p-semiflow identifies a class of customers (or tokens) that are the subjects of the tagging procedure. The expansion amounts to isolate one of these customers and to explicitly represent the ways in which it interacts with the others. For this

purpose, let us define a new subnet $\mathcal{N}_{\mathbf{f}}^{+}$, that includes $\mathcal{N}_{\mathbf{f}}$ and that is a restriction of $\mathcal{N}$ on the set of transitions $\mathcal{T}$ and on the new set of places $\mathcal{P}^{+} = \bigcup_{t\in\mathcal{T}}({}^{\bullet}t \cup t^{\bullet})$. In this section we assume that a random order policy is implicitly adopted for all the queue places of the net. In Sec. 6 we will discuss how to deal with some other scheduling policy.

We denote the process of refining the subnet $\mathcal{N}_{\mathbf{f}}^{+}$ in order to tag a token as the *unfolding* of the TGSPN. It considers, one at the time, all the transitions of the subnet $\mathcal{N}_{\mathbf{f}}^{+}$ (also called *tagged* transitions in the sequel) and is formally defined in the following.

To better illustrate the unfolding procedure, Fig. 2 is provided as a reference: it depicts the process for a single tagged transition. Part (a) depicts a TGSPN fragment, where transition t_i and places p_j, and p_l are tagged, i.e., $t_i \in \mathcal{T}$ and $p_j, p_l \in \mathcal{P}$. Part (b) presents the unfolded GSPN: transition t_i^t and places p_j^t, and p_l^t constitute part of the unfolding of t_i, p_j, and p_l, and specifically that related to the subnet involved in the movement of the tagged-token. Transition t_i^u and places p_j^u, and p_l^u represent instead is the part of the unfolding of t_i, p_j, and p_l related to the subnet involved in the movement of tokens that in the TGSPN are the "other" customers. Here, we illustrate the unfolding rules for tagged transitions having a single input tagged-place, however the procedure has been generalized to the case of multiple input tagged places, as explained in [2].

Let $(\mathcal{N}, \mathbf{f}, p_\circ, \mathcal{Q})$ be a TGSPN, with $\mathcal{N} = (T, P, I^-, I^+, H, \Pi, \mathbf{w}, \mathbf{m}_0)$, then it is possible to construct an equivalent GSPN $\mathcal{N}' = (T', P', I'^-, I'^+, H', \Pi', \mathbf{w}', \mathbf{m}'_0)$ as follows. Let $T = \{t_i\}$ and $P = \{p_i\}$, and assume $p_k \in \mathcal{P}'$ where $\mathcal{P}' = \mathcal{P}^{+}\backslash\mathcal{P}$. To avoid useless repetitions, let I^* represent both I^- and I^+, then:

- $T' = T\backslash\mathcal{T} \cup T^{+}$ where $T^{+} = \{t_i^t, t_i^u : t_i \in \mathcal{T}\}$
- $P' = P\backslash\mathcal{P} \cup P^{+}$ where $P^{+} = \{p_i^t, p_i^u : p_i \in \mathcal{P}\} \cup \{p_r \in \mathcal{P} : p_r^\circ \neq \emptyset\}$
- $I'^*(t_i^u, p_j^u) = I^*(t_i, p_j)\ \forall t_i^u, p_j^u, \quad I'^*(t_i^u, p_k) = I^*(t_i, p_k)\ \forall t_i^u, p_k$
  $I'^*(t_i^u, p_j^t) = 0\ \forall t_i^u, p_j^t$
- $I'^*(t_i^t, p_j^t) = 1, \forall t_i^t, p_j^t, \quad I'^*(t_i^t, p_k) = I^*(t_i, p_k)\ \forall t_i^t, p_k$
  $I'^*(t_i^t, p_j^u) = I^*(t_i, p_j) - 1, \ \forall t_i^t, p_j^u$
- $I'^*(t_i, p_j) = I^*(t_i, p_j)$
- $H'(t_i, p_j) = H(t_i, p_j)\ \forall p_j \in P\backslash\mathcal{P}, \forall t_i \in T\backslash\mathcal{T}$
- $H'(t_i^t, p_j) = H'(t_i^u, p_j) = H(t_i, p_j), \ \forall p_j \in P\backslash\mathcal{P}\ \forall t_i \in \mathcal{T}$
- $\forall p_j \in \mathcal{P}, \forall t_k \in T$ such that $H(t_k, p_j) \neq \infty$, place p_j is added to P' of $\mathcal{N}'$ in order to record the marking of the tagged place $p_j \in \mathcal{P}$. $p_j \in P'$ is such that ${}^{\bullet}p_j = {}^{\bullet}p_j^u \cup {}^{\bullet}p_j^t$ and $p_j{}^{\bullet} = p_j^{u\bullet} \cup p_j^{t\bullet}$. If $t_k \in T\backslash\mathcal{T}$ then $H'(t_k, p_j) = H(t_k, p_j)$ (Figure 2d); if $t_k \in \mathcal{T}$ then $H'(t_k^t, p_j) = H'(t_k^u, p_j) = H(t_k, p_j)$.

The initial marking $\mathbf{m}'_0$ of $\mathcal{N}'$ is:

- $\forall p_i \in \mathcal{P} : p_i \neq p_\circ\ \mathbf{m}'_0(p_i^u) = \mathbf{m}_0(p_i) \wedge \mathbf{m}'_0(p_i^t) = 0$
- $\mathbf{m}'_0(p_\circ^u) = \mathbf{m}_0(p_\circ) - 1 \wedge \mathbf{m}'_0(p_\circ^t) = 1$
- $\forall p_i \in P\backslash\mathcal{P}\ \mathbf{m}'_0(p_i) = \mathbf{m}_0(p_i)$

Always to avoid repetitions, let * denote either u or t. The firing weights and rates $\mathbf{w}'$ of $\mathcal{N}'$ are modified according to the rules that are discussed in the

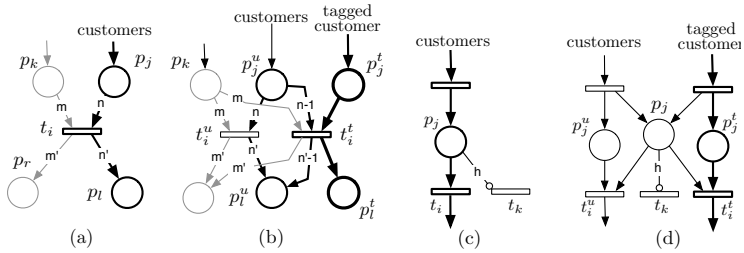


Fig. 2. Unfolding procedure for I^*: (a) TGSPN, (b) unfolded GSPN; and H: (c) TGSPN, (d) unfolded GSPN

following. Observe that the formulae below are valid for timed transitions whose input arc from a tagged place has multiplicity one. A general formula has been derived for the case of arc multiplicity greater than one, as illustrated in Fig. 2 (see [2]).

Immediate transitions: if transition $t_i \in \mathcal{T}$ is an immediate transition with weight $w(t_i)$, then its unfolded copies $t_i^t \in T'$ and $t_i^u \in T'$ in the new GSPN remain immediate with rates dependent from the marking:

$$w'(t_i^*, \mathbf{m}) = w(t_i) \frac{\mathbf{m}(p_j^*)}{\mathbf{m}(p_j^u) + \mathbf{m}(p_j^t)}$$

Timed transitions of type Infinite Server: if transition $t_i \in \mathcal{T}$ is a timed transition of type "Infinite Server" and rate $r(t_i)$, then its unfolded copies $t_i^t \in T'$ and $t_i^u \in T'$ in the GSPN remain timed of type "Infinite Server" with rates $r'(t_i^u) = r'(t_i^t) = r(t_i)$;

Timed transitions of type n-Server: if transition $t_i \in \mathcal{T}$ is a timed transition of type "n-Server" and rate $r(t_i)$, then its unfolded copies $t_i^t \in T'$ and $t_i^u \in T'$ in the new GSPN remain timed but of type "Marking Dependent" with rates:

$$r'(t_i^*, \mathbf{m}) = min\{\mathbf{m}(p_j^u) + \mathbf{m}(p_j^t), n\} r(t_i) \frac{\mathbf{m}(p_j^*)}{\mathbf{m}(p_j^u) + \mathbf{m}(p_j^t)}$$

The above expressions are valid only if, in the TGSPN, p_j is the only place in the input set of t_i; more generally the $ed(t_i, \mathbf{m})$ must be considered in place of $min\{\mathbf{m}(p_j^u) + \mathbf{m}(p_j^t), n\}$.

Timed transitions of type Marking Dependent: if transition $t_i \in \mathcal{T}$ is a timed transition of type "Marking Dependent", assume that the marking dependent rate of transition t_i is expressed as a function $g(\mathbf{m})$. Let $p_j \in \mathcal{P}$ the tagged place input of t_i. Then the unfolded copies of t_i, $t_i^t \in T'$ and $t_i^u \in T'$, in the new GSPN remain timed of type "Marking Dependent" with rates that are modified in the following manner: g^*, the marking dependent function of transition t_i^*, is obtained by g after an appropriate variables substitution for any value of k such that $p_k \in \mathcal{P}$:

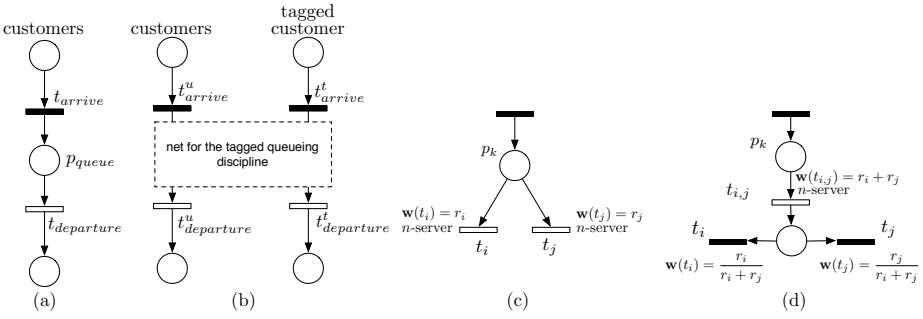


Fig. 3. (a) Untagged GSPN implicit queue; (b) general unfolding scheme. A queueing place with more servers: (c) original net; (d) intermediate translation

$$g^*(\mathbf{m}) = \frac{\mathbf{m}(p_j^*)}{\mathbf{m}(p_j^u) + \mathbf{m}(p_j^t)} g\big(\mathbf{m}(p_1), \ldots, \mathbf{m}(p_k^u) + \mathbf{m}(p_k^t), \ldots, \mathbf{m}(p_n)\big)$$

6 Queueing Policies

In passage time analysis the scheduling of tokens at transition firing is relevant. Up to now we assumed that queue places were managed with a RO policy. Several other scheduling policies can be specified in a TGSPN allowing to increase the modelling power of the formalism. The scheduling discipline is specified annotating a place of the TGSPN with the corresponding policy; not annotated places will be served in RO. We assume that scheduling policies are associated with places meaning that when several transitions withdraw tokens from the same place, they all use the same policy to choose the token upon firing.

Currently TGSPNs (see Def. 2) allow for the specification of the following scheduling disciplines at queue places in case of n-server transitions: FCFS, RO, LCFS, LCFS-PR. In this section we discuss the net unfolding for LCFS only. Similar translations have also been devised for the other cases and can be found in [2]. Fig. 3.a-b show the general pattern used for the translation: in part (a) is the TGSPN where the tagged transitions denoting the arrival and the departure of a customer are respectively named t_{arrive} and $t_{departure}$, whereas the tagged place subject to scheduling and annotated with the policy is named p_{queue}; part (b) shows the unfolded nodes and the unfolded block modelling the queue at place p_{queue}. The subnet in the block could explicitly model every position in the queue as well as the modifications occurring in the queue upon each arrival and departure, but then we would need to include a structure with size depending on the maximum number of tokens in the queue place. A different approach is possible and is discussed in the sequel where the queue is represented in a compact manner tracking only the position of the tagged token in the queue.

When a *queue place* is shared among several servers, the problem can be normalized as shown in Fig. 3.c-d where the sojourn of the tokens in the queue

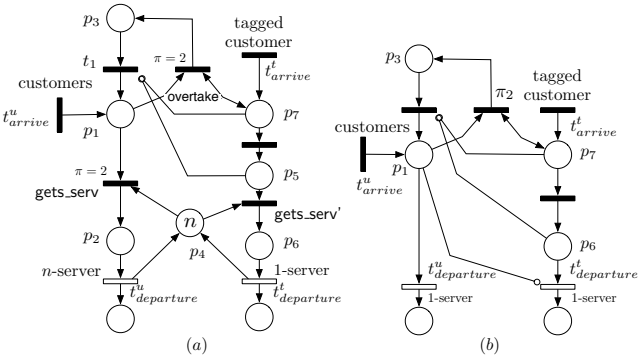


Fig. 4. GSPN model of the (a) n-server with LCFS without PR; (b) 1-server with LCFS with PR

is split from the identification of the server which actually triggers the change of state. An embedded MC argument can be used to show that the representations of Fig. 3.c and 3.d are equivalent from the point of view of the CTMC that underlies the model.

LCFS scheduling policy. Figure 4.a shows the GSPN for the LCFS scheduling policy (without preemption) when tagged with regard to a given customer. The server is assumed to be a n-server station. At any instant, the customers at the queue are classified into those arrived ahead of the tagged-customer (tokens in place p_3) and those arrived after the tagged-customer (tokens in place p_1). The tokens in place p_2 model the customers being served and a token in place p_6 models the tagged-customer that is being served. The logical queue is represented by the set of places p_1, p_3, p_5 and p_7. As soon as the tagged-customer enters the queue (p_7) all the customers, except those currently served, are preempted from place p_1 and put in place p_3 (higher priority immediate transition overtake implements this) since they become temporarily "older" than the tagged-customer: these customers stay in p_3 till the tagged-customer gets the server (p_6), in fact inhibitors arcs (p_5, t_1) and (p_7, t_1) prevent tokens in p_3 from leaving this place as long as the tagged-token is in the queue. Tokens arriving in place p_1, while p_5 is marked, represent the customers that joined the queue after the tagged-customer arrival: These tokens will overtake the tagged-customer at the server when one instance will be available, in the depicted net this is obtained by assigning priority to transition gets-serv over transition gets-serv′. Figure 4.b shows another example: the translation for a LCFS-PR with a single-server station.

7 Computing the Passage Time of the Unfolded TGSPN

Let $\mathcal{N}$ and $\mathcal{N}'$ be the TGSPN and the unfolded TGSPN respectively. The computation of the first passage time requires first the precise definition of the markings that represent the start and target sets.

Starting from the net level definition of a traversal–path $\mathcal{S}$, coupled with marking refinements, the derivation of the start and target states of the CTMC is straightforward.

A (tangible) marking $\mathbf{m}_s$ is a start state iff it is reachable from a tangible marking $\mathbf{m}'$ through the firing of a sequence σ made up of one timed transition followed by zero or more immediate transitions ($\mathbf{m}'[\sigma\rangle\mathbf{m}_s$) containing exactly one transition t_i^t that is the tagged expansion of transition t_i such that:

– $t_i \in \mathcal{S}_{in}$ (and thus $\langle t_i, C_i^{pre}, C_i^{post}\rangle \in START$),
– $\sum_{p_j^t \in \mathcal{S}'} \mathbf{m}'(p_j^t) = 0 \wedge \sum_{p_j^t \in \mathcal{S}'} \mathbf{m}_s(p_j^t) = 1$ (which ensures that the firing of t_i^t corresponds to moving the tagged token from a place outside $\mathcal{S}'$ to a place inside $\mathcal{S}'$, being $\mathcal{S}'$ the unfolded version of $\mathcal{S}$),
– $C_i^{pre} = true \wedge C_i^{post} = true$.

Similarly, a target state is any (tangible) marking $\mathbf{m}_e$ obtained by selecting all arcs $\mathbf{m}''[\sigma\rangle\mathbf{m}_e$ in the TRG annotated with a firing sequence σ including the firing of t_j^t, such that $t_j \in \mathcal{S}_{out}$, $\mathbf{m}''$ is reachable from a start state $\mathbf{m}_s$ through a path that doesn't contain any end state, and $\sum_{p_j^t \in \mathcal{S}'} \mathbf{m}''(p_j^t) = 1 \wedge \sum_{p_j^t \in \mathcal{S}'} \mathbf{m}_e(p_j^t) = 0$. Analogous reasoning can be made for the forbidden states. Attention should be paid to those start states that besides being reachable through arcs involving a subnet entry–point can also be reached through some "internal" arc of the path going from a start to an end state (details are in [2]).

The main steps of the computation procedure are:

1. Generation of the $\mathrm{MC}_{\mathcal{N}'}$ from the unfolded GSPN $\mathcal{N}'$ (see Sec. 5);
2. Selection of the start and target states: it consists in recognizing the states of $\mathcal{N}'$ that are start and target using the information provided by sets $START$, $\mathcal{S}_{out}$ and $\mathcal{S}_{forb}$;
3. Transformation of the MC: all the target states in the CTMC must be made absorbing (i.e., all transition rates out of these states must be set to 0);
4. Classical first passage time computation techniques [9] are applied to the MC generated in the previous step. When a set of starting states is defined an initial distribution over these states (to provide some relative weights) is needed. Several choices for this distribution are possible ranging from the portion of the steady state distribution computed for the whole GSPN (when there is one) up to an initial distribution specified by the modeller given the objectives of the study. It is also possible to optimize this step by computing the steady state probability of the $\mathrm{MC}_{\mathcal{N}}$ before applying the unfolding, and then properly distributing to the start "unfolded" states of $\mathrm{MC}_{\mathcal{N}'}$.

Presently we only have a prototype implementation of this solution technique, but several tools are available to perform such computation (e.g. [5,6]) and we are planning to investigate whether they can be easily interfaced to work with the method proposed in this paper.

8 Conclusions and Related Work

In this paper a new definition of Tagged GSPN has been introduced, in which tagged places and transitions are chosen with the support of methods that exploit the structural properties of the model and where queueing places implementing various scheduling policies may be specified.

Our work is an improvement of that reported in [7] which represents, in our opinion, the only paper addressing this topic within the GSPN context in a systematic manner.

The choice of the places and transitions to be tagged is in our proposal related with the possibility of computing the p-semiflows of the net that identify the locations where tokens are preserved and thus "circuits" where the movement of tokens can be interpreted as a flow of customers.

Ensuring that the tagged places and transitions are those identified on the basis of a p-semiflow overcomes a potential source of problems contained in the original definition proposed in [7]. Indeed, that definition does not seem to require that either none or all the arcs departing from a place must be tagged: this could be a potential source of problems, because without this restriction it is not possible to guarantee that the behaviour of the untagged GSPN is "equivalent" to (i.e. is an exact lumping of) that of the unfolding of the tagged GSPN (as long as only average performance indices need to be computed). For instance, considering the behaviour of the repairmen of the FMS of Fig. 1, our technique requires to tag all the nodes of the subnet comprising the places $p_{16}, p_{18}, p_{19}, p_{20}, and p_{21}$ (as already pointed out in Sec. 3). However, tagging only part of this subnet (specifically $p_{19}, T_8, p_{20}, t_5, p_{16}, T_7, p_{21}, T_{10}, and t_6$) would satisfy the restrictions provided in [7], but would also yield a sub-model that traps the tagged customer (one of the two repairmen) which is thus prevented from working at the repair of machine M_2. Obviously the state spaces of the two models would differ in this case making this tagged model not equivalent to the un-tagged one.

Moreover, compared to the way of specifying start and end states proposed in [12] and [7], the definition given in this paper was devised to be both more precise and more intuitive for the modeller. More precise because the characterization of start states should be related not only to some condition to be satisfied by the corresponding marking, but also to the event of the tagged customer arrival in the subnet under observation (this may have some subtle implications, in particular when computing the steady state distribution of the start states, as it is illustrated in [2]). More intuitive for the modeller because the "first passage time" notion at the level of an abstract model (such as a GSPN) where the system structure is represented rather than the detailed state-transition diagram, it can quite naturally be associated with the notion of "traversal time" of a given subnet.

Finally, this paper provides an original and efficient way to consider scheduling policies different from the random order one. An interesting feature of our proposal is the compact and parametric representation of the queue, specifically tailored to the goal of tracking the tagged token position in the queue. The semantics of TGSPNs with queue places is formally defined by means of an

unfolding and queue place substitution procedure, that allows to automatically generate a GSPN model, ready for generation of a CTMC on which existing passage time density computation techniques can be applied.

There are several possible developments of the work presented in this paper, mostly related with the use of High Level Petri Net formalisms. A first possibility could be that of exploiting the characteristics of these modelling languages to identify symmetries and to automatically generate reduced state spaces. The second obvious advantage could derive from the possibility of modelling in a natural manner several classes of customers flowing through common parts of the net and of isolating a customer of one class as tagged customer in order to obtain a detailed comparison of their behaviours.

References

1. Ajmone Marsan, M., Balbo, G., Conte, G., Donatelli, S., Franceschinis, G.: Modelling with Generalized Stochastic Petri Nets. J. Wiley, Chichester (1995)
2. Balbo, G., De Pierro, M., Franceschinis, G.: Tagged generalized stochastic Petri nets. Technical Report (2009), http://www.di.unito.it/~depierro/articoli.html
3. Bause, F., Buchholz, P., Kemper, P.: Hierarchically combined Queueing Petri Nets. In: Proc. 11th Int. Conf. on Analysis and Optimization of Systems, Discrete Event Systems, Sophia-Antipolis, pp. 176–182 (1994)
4. Bause, F., Kemper, P.: QPN-tool for qualitative and quantitative analysis of Queueing Petri Nets. In: Haring, G., Kotsis, G. (eds.) TOOLS 1994. LNCS, vol. 794, pp. 321–334. Springer, Heidelberg (1994)
5. Bodrog, L., Horvath, G., Racz, S., Telek, M.: A tool support for automatic analysis based on the tagged customer approach. In: Proc. of the 3rd int. conf. on the Quantitative Evaluation of Systems 2006, Washington, DC, USA, pp. 323–332. IEEE, Los Alamitos (2006)
6. Bradley, J.T., Dingle, N.J., Knottenbelt, W.J., Wilson, H.J.: Hypergraph-based parallel computation of passage time densities in large semi-Markov models. Journal of Linear Algebra and its Applications 386, 311–334 (2004)
7. Dingle, N.J., Knottenbelt, W.J.: Automated Customer-Centric Performance Analysis of Generalised Stochastic Petri Nets Using Tagged Tokens. In: Proc. of the 3rd Int. Workshop on Practical Applications of Stochastic Modelling (PASM 2008) (September 2008)
8. Kounev, S., Dutz, C., Buchmann, A.: QPME – Queueing Petri Net Modeling Environment. In: QEST 2006: Proceedings of the 3rd international conference QEST 2006, Washington, DC, USA, pp. 115–116. IEEE CS, Los Alamitos (2006)
9. Kulkarni, V.G.: Modeling and Analysis of Stochastic Systems. Chapman & Hall, London (1995)
10. Miner, A.S.: Computing response time distributions using stochastic Petri nets and matrix diagrams. In: Petri Nets and Performance Models 2003, p. 10. IEEE, Los Alamitos (2003)
11. Nelson, R.: Probability, Stochastic Processes, and Queueing Theory: The Mathematics of Computer Performance Evaluation. Springer, Heidelberg (1995)
12. Suto, T., Bradley, J.T., Knottenbelt, W.J.: Performance trees: Expressiveness and quantitative semantics. In: Proc. of the 4th International Conference on Quantitative Evaluation of Systems (QEST 2007), Edinburgh, UK. IEEE, Los Alamitos (2007)

Modelling Zoned RAID Systems Using Fork-Join Queueing Simulation

Abigail S. Lebrecht*, Nicholas J. Dingle, and William J. Knottenbelt

Department of Computing, Imperial College London,
South Kensington Campus, SW7 2AZ, United Kingdom
Tel.: +44 20 7594 8251
{asl102,njd200,wjk}@doc.ic.ac.uk

Abstract. RAID systems are ubiquitously deployed in storage environments, both as standalone storage solutions and as fundamental components of virtualised storage platforms. Accurate models of their performance are crucial to delivering storage infrastructures that meet given quality of service requirements. To this end, this paper presents a flexible fork-join queueing simulation model of RAID systems that are comprised of zoned disk drives and which operate under RAID levels 01 or 5. The simulator takes as input I/O workloads that are heterogeneous in terms of request size and that exhibit burstiness, and its primary output metric is I/O request response time distribution. We also study the effects of heavy workload, taking into account the request-reordering optimisations employed by modern disk drives. All simulation results are validated against device measurements.

1 Introduction

RAID[1] has revolutionised data storage because of its ability to synthesise a set of low-cost commodity storage devices into a single logical unit that can deliver high reliability with high performance. However, RAID system performance varies heavily in practice, depending on chosen configuration and operating context. Given a budget and an expected workload, it is therefore a major challenge for system designers and engineers to select RAID components and corresponding configurations capable of delivering a required level of quality of service. Performance models provide a low-cost means to evaluate the suitability of candidate system designs ahead of implementation.

In the above context, this paper introduces a queueing-based simulator for the analysis of RAID systems comprised of zoned disks. Our goal is to provide an elegant high-level framework that avoids very detailed low-level device simulation

* Corresponding author.

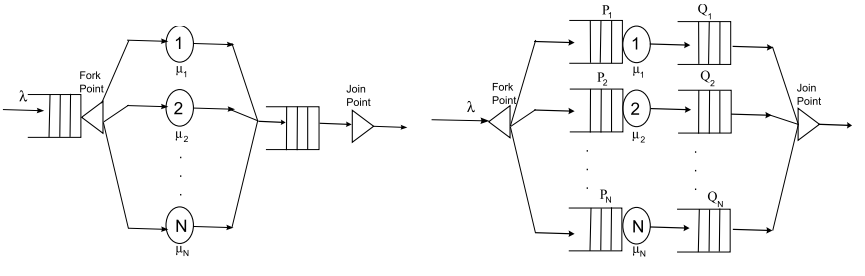
[1] Redundant Array of Inexpensive Disks [1]; RAID levels describe various ways of spreading data across multiple storage devices using striping, mirroring, and/or parity.

J.T. Bradley (Ed.): EPEW 2009, LNCS 5652, pp. 16–29, 2009.

(e.g. as performed by the DiskSim [2] and RaidSim [3] simulators) and which can be simply parameterised from disk drive technical specifications. The simulation generates as its primary output metric the cumulative distribution function of I/O request response time. From this, it is straightforward to calculate metrics typically encountered in Service Level Agreements, including response time quantiles and the mean, variance and higher moments of I/O request response time.

Simulation is often used to study RAID system performance, since there exist no exact analytical models of RAID of any level [4]. However, there are numerous analytical queueing network approximations (e.g. [5,6,7,8,9]). In [4,10,11] we have developed approximate analytical queueing models of RAID 01 and 5. Simulations are often used to validate the results of analytical models. Additionally, they provide the ability to replicate the details of the scheduling algorithms and mechanical behaviour of real systems, while analytical models must abstract these details. Thus, simulations can aid the development of more realistic analytical models.

In [12] we introduced a zoned RAID simulator for RAID level 0 (striping, no redundancy). This simulation is based on modelling each disk drive as an $M/G/1$ queue and approximates RAID 0 as a split-merge queueing system (see Figure 1(a)). In this system, a job (I/O request) splits into N subtasks which are serviced in parallel. Only when all the subtasks finish servicing and rejoin can the next job split into subtasks and start servicing. However, it is generally accepted that the queueing model which most accurately reflects the behaviour of RAID systems is the fork-join queueing network [5]. In a fork-join queue with N queues, (see Figure 1(b)), each incoming job is split into N subtasks at the fork point. Each of these subtasks queues for service at a parallel service node before joining a queue for the join point. When all N subtasks in the job are at the head of their respective join queues, they rejoin (synchronise) at the join point.



(a) Split-merge queueing model (b) Fork-join queueing model

Fig. 1. Split-merge vs. fork-join queueing models

This paper presents a fork-join simulation capable of modelling RAID levels 01 (mirror of stripes) and 5 (distributed parity). In order to simulate a RAID system, we must first implement an effective single disk simulation. We can then

use several instances of this single disk simulator as components in our disk array simulator. Section 2 summarises our single disk simulation, which utilises the *JINQS* Java-based queueing simulation library [13]. Section 3 details the fork-join extensions to the single disk simulator required to create an effective model for RAID. This involves firstly simulating a fork-join queueing network and then tailoring it to model the specific demands of RAID 01 and 5. Furthermore we enhance the model to accept various types of workload. We then validate the accuracy of this new simulator by comparing results from several different types of workload to device measurements taken on a real RAID system.

2 Single Disk Simulation

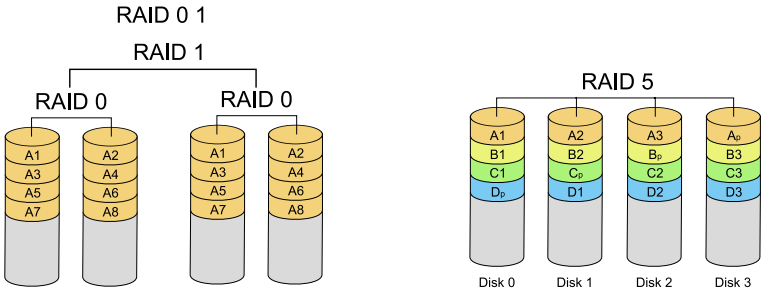
We model single disk drives as $M/G/1$ queues and use the *JINQS* Java-based simulation library [13] for $M/G/1$ queue simulation. The service time density of an access to a random location on a single disk drive is the convolution of the seek time, rotational latency and data transfer time probability density functions. In our model we use the seek time and rotational latency probability distributions defined in [14] and the data transfer time distribution from [4]. The seek, rotation and transfer times are sampled using the cumulative distribution function inversion method described in [12]. An important subtlety that needs to be taken into account is that modern disks are *zoned*, with more sectors on the outer tracks than inner tracks. Therefore, a random request is more likely to be directed to a sector on an outer track. Similarly, zoning means that it is faster to transfer data on a track close to the circumference than the centre of the disk. The seek time and data transfer models must take these factors into account. We assume that all requests are random accesses and therefore it is always necessary to position the disk head before transferring data.

3 RAID Simulation

Disk arrays organise multiple independent disks into a single logical disk unit. By striping data across multiple disks and accessing the disks in parallel, higher data transfer rates are achieved, especially with larger I/O requests. Data striping also ensures that data is balanced across the disks, avoiding data hot spots. Disk striping involves writing data blocks of a constant pre-defined size to successive disks in a cyclical pattern.

However, the larger the disk array, the more likely it is that a member disk will fail. In order to avoid data loss as a result of failures, redundancy can be employed using mirroring (see Figure 2(a)) or parity blocks (see Figure 2(b)). Parity is block-interleaved and distributed across all disks.

All these schemes involve striping of I/O accesses across disks in the disk array. A fork-join queue in which customers represent I/O requests provides a good foundation for an abstraction of this behaviour.



(a) RAID 01 (b) RAID 5

Fig. 2. RAID Configurations [15]

3.1 Fork-Join Simulation

As shown in the UML class diagram of Figure 3, our queueing simulation is specified in terms of *QueueingNode*, *Link* and *Customer* classes. *QueueingNode*s are connected by *Link*s to create a network of queues. Response times measurements are obtained by recording the time each *Customer* spends in the network. To extend this for fork-join queueing, we introduce *ForkLink* and *JoinLink* classes to extend the *Link* class and a *ForkedCustomer* class to extend the *Customer* class. *ForkedCustomer*s are created at a *ForkLink* when a customer forks into subtasks and removed at the *JoinLink*. A *ForkLink* creates a new *ForkedCustomer* for each subtask, each with a reference to the original *Customer*. These *ForkedCustomer*s are sent to one of the n single $M/G/1$ queues. When a *ForkedCustomer* leaves an $M/G/1$ queue, it is sent to *JoinLink*, which collects all *ForkedCustomer*s. When all the *ForkedCustomer*s for a particular *Customer* have arrived, the original *Customer* is sent on its way and all of its *ForkedCustomer*s are destroyed.

3.2 RAID 01 Simulation

There are certain extensions to the fork-join simulation described above that must be made to model a RAID 01 system accurately.

In particular, both the fork-join simulation described above and the RAID 0 simulation of [12] are limited to supporting requests consisting of a number of subtasks that is a multiple of the number of disks. We therefore extend the *ForkLink* class with *RAID01ReadForkLink* and *RAID01WriteForkLink* classes, both of which support striping of variable size subtasks across disks starting from a randomly selected disk. Additionally, we extend the *JoinLink* class with the *RAIDJoinLink* class to support joining of variable-sized requests.

In terms of subtask scheduling for RAID 01 read operations, we assume an efficient RAID controller which reads half the data from the primary disks and half the data from the mirror disks [7]. RAID 01 write operations send each subtask to both the primary and mirror disks and create double the number of *ForkedCustomer*s as for a read request of the same size.

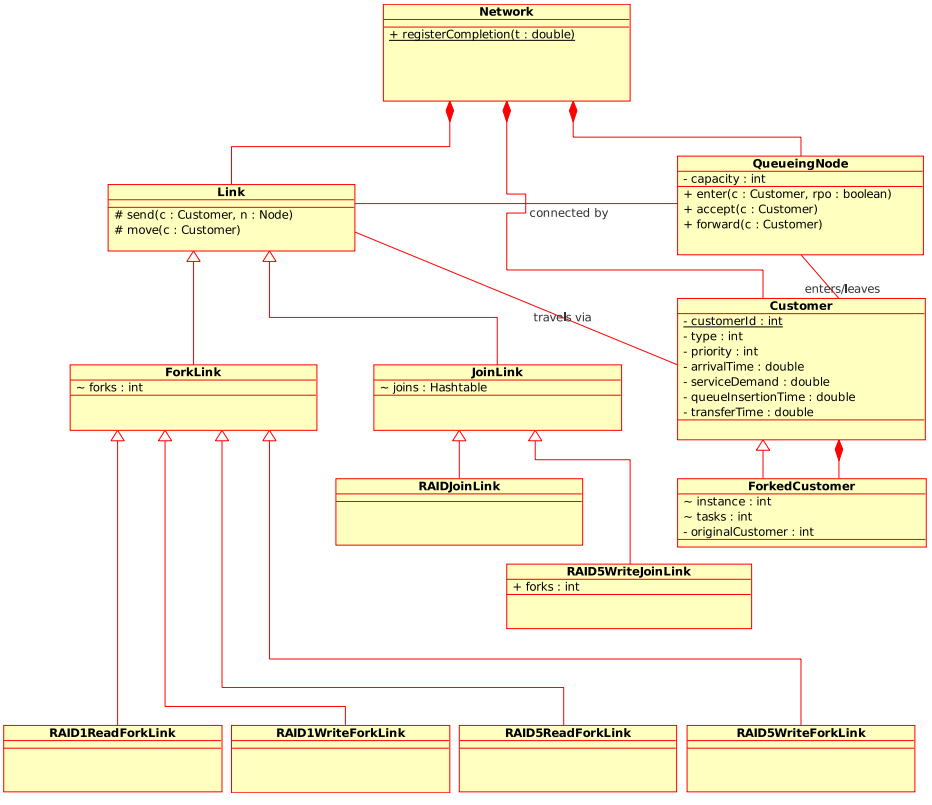


Fig. 3. RAID simulator class diagram

3.3 RAID 5 Simulation

Manufacturers of RAID controllers seldom reveal the mechanisms and scheduling strategies involved in their products. In the cases of RAID 0 and 01, the likely disk accesses are relatively straightforward to predict. However in RAID 5, particularly with operations involving pre-reads and parity updates, there are many possibilities for scheduling strategies and disk head positioning times within a request. Here we base the design of our RAID 5 simulation upon the operational assumptions of RAID 5 disk behaviour presented in [4,7,11].

In a manner analogous to the RAID 01 case, we extend the *ForkLink* class with *RAID5ReadForkLink* and *RAID5WriteForkLink* classes.

A RAID 5 read request will read only from the disks containing data blocks in a stripe and not the disk with the single parity block in each stripe. To simulate this, when forking each request, the position of the parity disk is randomly chosen as well as the starting disk. If a request accesses more than one stripe, then the position of the parity disk within the array is incremented (modulo the number of disks) at the end of each stripe.

The behaviour of a RAID 5 write is complex, with different parity-update schemes that depend on the size of the request. For simplicity, we assume requests are aligned to start striping from the first disk in the array.

Given a b-block write request on an n-disk RAID 5 system, the possibilities are:

If a request consists of a number of complete stripes (i.e. $b \bmod (n-1) = 0$), all the disks are utilised, with either the new data block or the new parity block written to each disk. Full stripe writes can be simulated by sending *ForkedCustomer*s to each disk and joining them at the *RAID5WriteJoinLink* when all subtasks have completed.

If a request consists of $b \bmod (n-1) < \frac{n-1}{2}$ blocks (i.e. it consists of zero or more full stripe writes followed by a small partial stripe write), then parity is calculated using [1]:

$$new\_parity = new\_data \oplus old\_data \oplus old\_parity$$

where $\oplus$ is the exclusive-or (XOR) operator. This is a read-modify-write operation. After transferring the full stripes, each of the $b \bmod (n-1)$ blocks and parity must be transferred twice, first to read the old data and parity, then to write the new data and parity. When the old data and parity have been read from all disks, a new request will be issued to write the new data and parity to the same disks. This request is given non-preemptive priority in the queue, so at least one disk (the last to complete the pre-read) will just have completed reading a data or parity block that now needs to be re-written.

If $\frac{n-1}{2} \leq b \bmod (n-1) < n-1$ (i.e. the request consists of zero or more full stripe writes followed by a large partial stripe write), then to minimise disk accesses the parity is calculated by pre-reading from the disks that are not being written to. The new parity is calculated by XOR-ing the data that will be written with the data from the disks that will remain unchanged. This is a reconstruct-write operation. After the full stripe transfers, $n-1-b \bmod (n-1)$ blocks of data are pre-read for the calculation of the new parity. When all $n-1-b \bmod (n-1)$ disks complete their pre-read, a new request is sent to the other $b \bmod (n-1)+1$ disks to write the new data and parity.

Simulation of the above operations is supported in the *RAID5WriteForkLink* and *RAID5WriteJoinLink* classes. The *RAID5WriteForkLink* subdivides any arriving request into full stripe subtasks followed by pre-read subtasks. These subtasks are then routed to the relevant $M/G/1$ queues. When the pre-read subtasks have completed and are accounted for at the *RAID5WriteJoinLink* then, instead of completing the request, the *RAID5WriteJoinLink* creates a new high priority request to send back to the *RAID5WriteForkLink*, where it splits into $b \bmod (n-1)+1$ subtasks (the number of blocks to write plus the parity). In order for the simulation to differentiate between full stripe writes and pre-reads and the following partial stripe write, the *ForkedCustomer*s are assigned classes representing the type of request.

The subtasks of the partial stripe write will have different service times depending on the nature of the previous request serviced by the disk. In the case that $b \bmod (n-1) < \frac{n-1}{2}$, there are four possible scenarios to be considered.

The first scenario is when the disk is busy at the arrival instant of any of the partial stripe write subtasks. Since the partial stripe write is accessing all the disks used for the pre-read, and all the pre-reads must complete before the partial stripe write is issued, it is not possible that the job currently servicing is a *ForkedCustomer* from the same *Customer*. Hence to simulate a return to the required disk position to transfer data, a random sample of seek and rotation time is taken.

If the disk is idle on arrival of a subtask, then there are a further three mutually exclusive scenarios with different positioning times:

- If another request has been in service between the pre-read and partial stripe write subtasks then the simulator needs to sample a new seek and rotation time.
- If the disk was the last to complete the pre-read, then it will be positioned on the correct track, but just past the rotational position. In this case, the simulator returns a positioning time of one full disk rotation.
- Otherwise, the disk is still positioned at the correct track and the simulator needs to sample from the rotational latency for positioning time.

If $b \bmod (n-1) \geq \frac{n-1}{2}$, there are again a number of scenarios to consider. Since the pre-read involves different disks than the partial stripe write, it is possible that previous full stripe subtasks from the same request could still be servicing on the disks required for the partial stripe write after the pre-read has completed.

In this context, if a subtask arrives to a busy disk, we consider whether the job currently in service is part of the same request. If it is, the subtask will follow on with no positioning time. If it arrives to an idle disk, the simulator checks if the previous job was part of the same request. If it was then the disk head is pointing to the correct track and the simulator needs to sample rotational latency only. In all other cases the positioning time is obtained by sampling both seek and rotation time.

Since we are simulating zoned disks, we must take into account that the transfer time must be same both for the full-stripe and pre-read and for the partial stripe write requests, since they are both accessing the same position on the disk. Therefore, the transfer time for each subtask to each disk is recorded in a hash table and referred to when the partial stripe write is serviced.

When all the partial stripe write subtasks complete, the *RAID5WriteJoinLink* sends the single request on its way and removes all *ForkedCustomer*s attached to that request.

3.4 Bulk Arrivals

Most queueing simulations assume that arriving requests are Markovian. However, over the last decade, there have been many studies of storage system I/O traces (e.g. [16,17,18,19,20,21]) which consistently show that real-life arrivals to storage systems exhibit burstiness and a variety of request size distributions.

Consequently, we have extended the simulator to support bulk arrivals of I/O requests at the RAID controller, making use of *JINQS*'s in-built support for arrivals that consist of a number of requests defined by a chosen probability distribution.

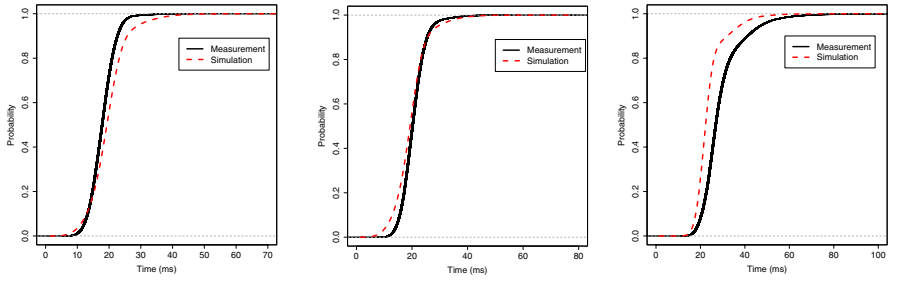
3.5 Rotational Positioning Optimisation

Bursty workloads [20] result in highly variable queue lengths. As queue length increases, response time suffers. To lessen this effect, many disk drives employ scheduling algorithms to reorder jobs in the queue to minimise head positioning time [22,23]. This reduces the time needed to service each job, which in turn reduces the waiting time for all jobs [24].

We incorporate this factor into our simulation by parameterising the service time distribution sampler according to the current queue length. The sampler then takes as many combined samples of seek and rotation time as there are jobs in the queue and chooses the minimum of these to be the positioning time of the request starting service. This can be used for either single disk simulation or RAID simulation.

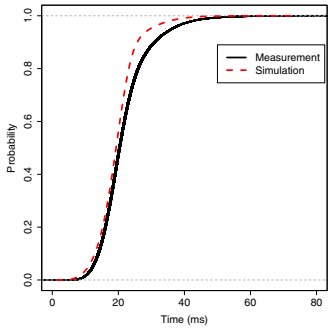
4 Validation

Our experimental platform consists of an Infortrend A16F-G2430 RAID system containing four Seagate ST3500630NS disks. Each disk has 60 801 cylinders. A sector is 512 bytes and we have approximated, based on measurements from the disk drive, that the time to write a single physical sector on the innermost and outermost tracks are 0.012064ms (t_{max}) and 0.005976ms (t_{min}) respectively. The stripe width on the array is configured as 128KB, which we define as the block size. Therefore there are 256 sectors per block. The time for a full disk revolution is 8.33ms. A track to track seek takes 0.8ms and a full-stroke seek requires 17ms for a read; the same measurements are 1ms and 18ms respectively for a write [25].

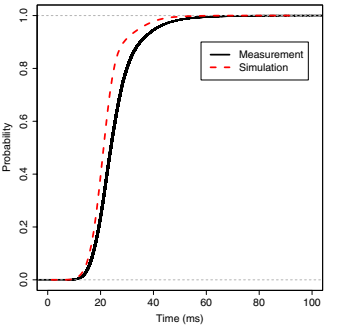


(a) 2-block read request (b) 4-block read request (c) 3-block write request

Fig. 4. Cumulative distribution functions of RAID 01 I/O request time on a 4 disk RAID system ($\lambda = 0.01$ requests/ms)



(a) 4-block mean read request



(b) 2-block mean write request

Fig. 5. Cumulative distribution functions of RAID 01 I/O request time on a 4 disk RAID system with request sizes chosen from a geometric distribution ($\lambda = 0.01$ requests/ms)

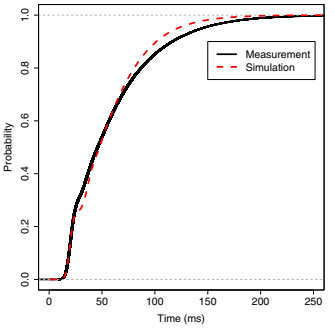


Fig. 6. Cumulative distribution functions of RAID 01 I/O read request time on a 4 disk RAID system with 4-block requests and geometrically distributed bulk arrivals with mean size 3 ($\lambda = 0.01$ requests/ms)

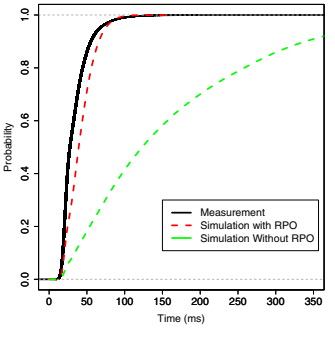
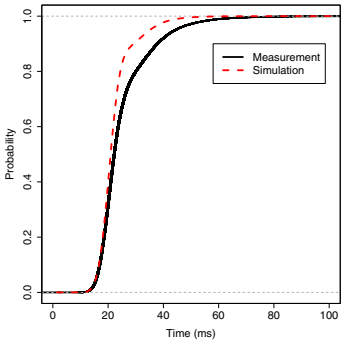
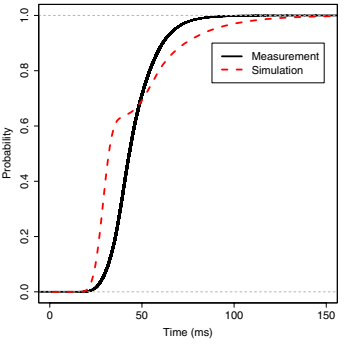


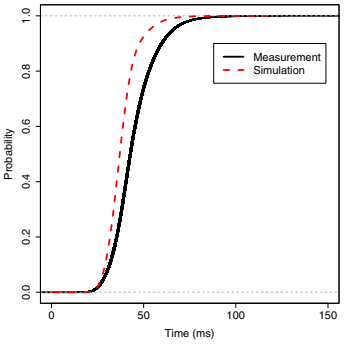
Fig. 7. Cumulative distribution functions of RAID 01 I/O read request time on a 4 disk RAID system with 4-block requests ($\lambda = 0.06$ requests/ms)



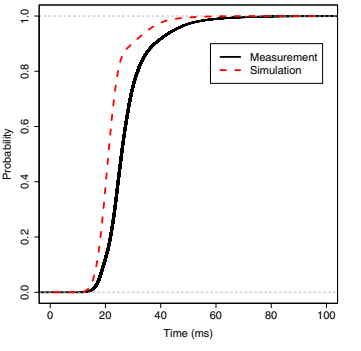
(a) 5-block read request $\lambda = 0.02$



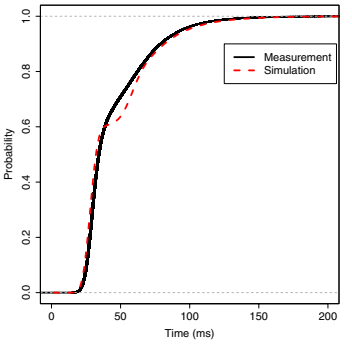
(b) 1-block write request $\lambda = 0.01$



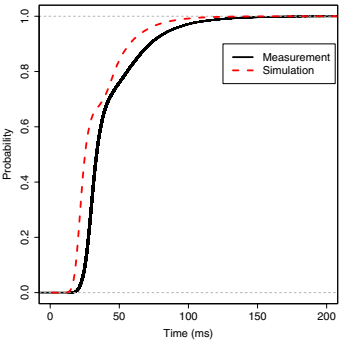
(c) 2-block write request $\lambda = 0.01$



(d) 3-block write request $\lambda = 0.01$

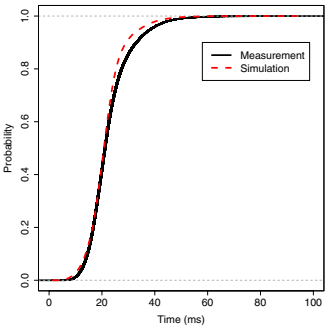


(e) 4-block write request $\lambda = 0.01$

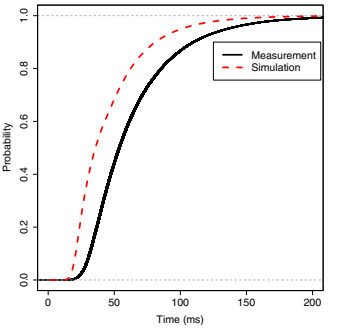


(f) 5-block write request $\lambda = 0.01$

Fig. 8. Cumulative distribution functions of RAID 5 I/O request time on a 4 disk RAID system (λ requests/ms)

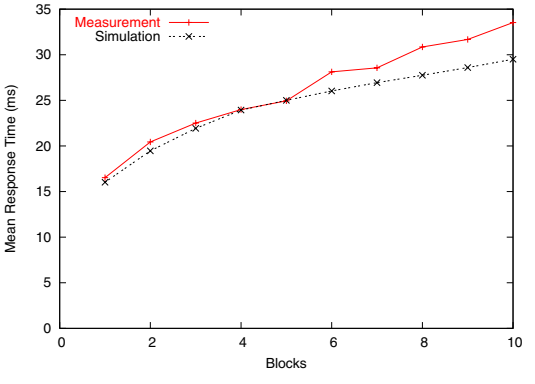


(a) 5-block mean read request, $\lambda = 0.01$

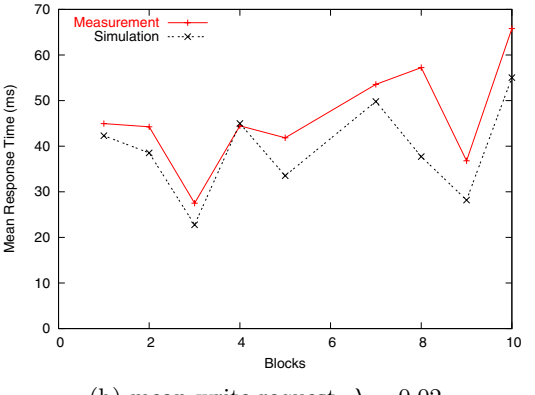


(b) 4-block mean write request, $\lambda = 0.02$

Fig. 9. Cumulative distribution functions of RAID 5 I/O request time on a 4 disk RAID system with request sizes chosen from a geometric distribution (λ requests/ms)



(a) mean read request, $\lambda = 0.01$



(b) mean write request, $\lambda = 0.02$

Fig. 10. Plot of mean response time against request size on a 4 disk RAID 5 system (λ requests/ms)

To obtain response time measurements from this system, we implemented a benchmarking program that issues read and write requests using a master process and multiple child processes. These child processes are responsible for issuing and timing I/O requests, leaving the master free to spawn further child processes without the need for it to wait for previously-issued operations to complete.

In order to validate the simulation model effectively, it was necessary to minimise the effects of buffering and caching as these are not currently represented in the model. We therefore disabled the RAID system's write-back cache, set the read-ahead buffer to 0 and opened the device with the `O_DIRECT` flag set. For each of the experiments presented below (both measurement and simulation), 100 000 requests were issued. We present a selection of comparisons of cumulative distribution functions (cdf). The single disk simulation is validated in [12], so here we only present RAID validations.

4.1 RAID 01

In Figure 4 we compare measurement and simulation cdfs for RAID 01 with Markovian arrivals at a rate of 0.01 requests/ms for different request type and size. We generally observe good agreement between model and measurement, particularly in Figure 4(b), in which a full stripe read is taking place. Figure 5 considers the same conditions, except in this case the request size is variable and sampled from a geometric distribution with a specified mean request size. We observe excellent agreement between model and measurement in these cases.

Figures 6 and 7 validate RAID 01 with more interesting workloads. In Figure 6 simulation and measurement cdfs are compared for a RAID 01 system with constant-size full stripe requests, which arrive in bursts. Each request arrives as part of a batch. The number of requests in each batch is decided by a geometric distribution. We continue to see excellent agreement between model and measurement. Figure 7 involves a high arrival rate at the array (0.06 requests/ms), such that rotational positioning optimisation (RPO) should be expected. We plot two simulation cdfs, one with RPO enabled on the simulator and the second with RPO disabled. It is clear from the graph that for large arrival rates (and hence long queue lengths) incorporating RPO into any model is crucial.

4.2 RAID 5

In Figure 8 we compare measurement and simulation cdfs for RAID 5 systems with Markovian arrivals with arrival rate λ requests/ms and constant size requests. We can use these validations to help judge the accuracy of our RAID 5 models. We observe in Figure 8(a) that the read simulation appears to agree well with the measurements. Figures 8(b) and 8(e) are small partial stripe writes. While mean values appear to agree well, the shapes of the cdf curves for the simulation differ somewhat from the measurement curves, particularly in the case of a small partial stripe write that does not follow a full stripe write (Figure 8(b)). Figures 8(c) and 8(f) are large partial stripe writes. These show better agreement than the small partial stripe equivalents. However, they appear

to consistently underestimate the measurements, as does the full stripe write request in Figure 8(d). It is possible that this underestimation can be attributed to not factoring into the simulation RAID controller overheads including parity computation time.

Similarly, Figure 9 compares simulation and measurements for RAID 5 requests with size decided by a geometric distribution. We again observe excellent agreement for read requests in Figure 9(a). Write requests in Figure 9(b) tend to underestimate the measurements since both full stripe requests and large partial stripe write requests of constant size underestimate the measurement. Figure 10 compares mean response times for simulation and measurement for up to 10-block jobs. The model predicts effectively the qualitative characteristics of mean RAID 5 response times as block size varies.

5 Conclusion

This paper has presented a RAID simulation based on fork-join queueing networks. We have presented extensive validations of this simulation against device measurements, generally observing excellent agreement for RAID 01 and 5 with request streams of both constant and variable size and bursty arrivals. We have also incorporated rotational positioning optimisations into our simulation and have shown in our validations that this is fundamental to any accurate representation of disk drives or RAID systems operating under heavy load.

In future work, we hope to further relax the constraints on the simulation. In particular, it is straightforward to modify the simulator to represent other RAID levels and to accept arrival streams that consist of both read and write requests, and both random and sequential accesses. Furthermore, we intend to discover more about RAID controller overheads and incorporate these into our models. We also intend to apply our simulator in validating and improving current and future analytical queueing models of RAID systems. Finally, caching is an interesting and practically useful aspect that merits further investigation and integration into our model.

References

1. Patterson, D.A., Gibson, G., Katz, R.H.: A case for redundant arrays of inexpensive disks (RAID). In: Proc. International Conference on Management of Data (SIGMOD) (1988)
2. Bucy, J.S., Ganger, G.R.: Contributors: The DiskSim Simulation Environment Version 3.0 Reference Manual. School of Computer Science, Carnegie Mellon University. 3.0 edn. (January 2003)
3. Chen, P.M., Lee, E.K.: Striping in a RAID level 5 disk array. SIGMETRICS Performance Evaluation Review 23(1), 136–145 (1995)
4. Lebrecht, A.S., Dingle, N.J., Knottenbelt, W.J.: A response time distribution model for zoned RAID. In: Al-Begain, K., Heindl, A., Telek, M. (eds.) ASMTA 2008. LNCS, vol. 5055, pp. 144–157. Springer, Heidelberg (2008)
5. Lee, E.K.: Performance Modeling and Analysis of Disk Arrays. PhD thesis, University of California at Berkeley (1993)

6. Chen, S., Towsley, D.: A performance evaluation of RAID architectures. IEEE Transactions on Computers 45(10), 1116–1130 (1996)
7. Harrison, P.G., Zertal, S.: Queueing models of RAID systems with maxima of waiting times. Performance Evaluation 64(7-8), 664–689 (2007)
8. Varki, E.: Response time analysis of parallel computer and storage systems. IEEE Transactions on Parallel and Distributed Systems 12(11), 1146–1161 (2001)
9. Varki, E., Merchant, A., Xu, J., Qiu, X.: Issues and challenges in the performance analysis of real disk arrays. IEEE Transactions on Parallel and Distributed Systems 15(6), 559–574 (2004)
10. Lebrecht, A.S., Dingle, N.J., Knottenbelt, W.J.: Modelling and validation of response times in zoned RAID. In: 16th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS) (September 2008)
11. Lebrecht, A.S., Dingle, N.J., Knottenbelt, W.J.: Validation of large zoned RAID systems. In: 24th UK Performance Engineering Workshop (UKPEW), July 2008, pp. 246–261 (2008)
12. Wan, F., Dingle, N.J., Knottenbelt, W.J., Lebrecht, A.S.: Simulation and modelling of RAID 0 system performance. In: 22nd Annual European Simulation and Modelling Conference (ESM), September 2008, pp. 145–149 (2008)
13. Field, A.J.: JINQS: An Extensible Library for Simulating Multiclass Queueing Networks. Imperial College London (August 2006)
14. Zertal, S., Harrison, P.G.: Multi-RAID queueing model with zoned disks. In: High Performance Computing and Simulation Conference (HPCS) (June 2007)
15. Wikipedia: Standard RAID levels (April 2009), http://en.wikipedia.org/wiki/Standard_RAID_levels
16. Gomez, M., Santonja, V.: Analysis of self-similarity in I/O workload using structural modeling. In: Proc. 7th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), October 1999, pp. 234–243 (1999)
17. Gomez, M., Santonja, V.: Characterizing temporal locality in I/O workload. In: Proc. International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS) (2002)
18. Wang, M., Ailamaki, A., Faloutsos, C.: Capturing the spatio-temporal behavior of real traffic data. Performance Evaluation 49(1-4), 147–163 (2002)
19. Riska, A., Riedel, E.: Disk drive level workload characterization. In: Proc. USENIX 2006 Annual Technical Conference (ATEC), Boston, MA (2006)
20. Ruemmler, C., Wilkes, J.: Unix disk access patterns. In: Proc. Usenix Winter Conference, San Diego, CA, pp. 405–420 (1993)
21. Shriver, E., Merchant, A., Wilkes, J.: An analytic behavior model for disk drives with readahead caches and request reordering. In: Proc. ACM SIGMETRICS, pp. 182–191 (1998)
22. Jacobson, D.M., Wilkes, J.: Disk scheduling algorithms based on rotational position. Technical Report HPL-CSP-91-7rev1, HP Laboratories (1991)
23. Seltzer, M., Chen, P., Ousterhout, J.: Disk Scheduling Revisited. In: Proc. USENIX Winter Technical Conference, USENIX Association, pp. 313–324 (1990)
24. Hsu, W.W., Smith, A.J.: The performance impact of I/O optimizations and disk improvements. IBM Journal of Research and Development 48(2), 255–289 (2004)
25. Seagate: Barracuda ES Data Sheet (2007), http://www.seagate.com/docs/pdf/datasheet/disc/ds_barracuda_es.pdf

Performance of Auctions and Sealed Bids

Erol Gelenbe[1,⋆] and László Györfi[2,⋆⋆]

[1] Intelligent Systems and Networks Group
Department of Electrical and Electronic Engineering
Imperial College, London
SW7 2BT
e.gelenbe@imperial.ac.uk
[2] Dept. Computer Science & Information Theory
Technical University of Budapest
1521 Stoczek u. 2, Budapest, Hungary
gyorfi@szit.bme.hu

Abstract. We develop models of automated E-commerce techniques, which predict the economic outcomes of these decision mechanisms, including the price attained by a good and the resulting income per unit time, as a function of the rate at which bidders provide the bids and of the time taken by the seller to decide whether to accept a bid. This paper extends previous work in two main directions. Since automated E-commerce mechanisms are typically implemented in software residing on the Internet, this paper shows how network quality of service will impact the economic outcome of automated auctions. We also analyse sealed bids which can also be automated, but which differ significantly from auctions in the manner in which information is shared between the bidders and the the party that decides the outcome. The approach that we propose opens novel avenues of research that bring together traditional computer and system performance analysis and the economic analysis of Internet based trading methods.

Keywords: Auctions, Sealed Bids, Networked Economics, Economic Performance, Quality of Service.

1 Introduction

Recent events in finance and the stock market have shown that our understanding of the automated computer and network based economy is not sufficiently advanced to be able to predict the dramatic changes that occur in the price of securities, commodities and the overall behaviour of prices. Thus more analysis is needed about how prices are established in the presence of automated

⋆ Research undertaken as part of the ALADDIN (Autonomous Learning Agents for Decentralised Data and Information Systems) project, jointly funded by a BAE Systems and EPSRC (Engineering and Physical Research Council) strategic partnership under Grant No. EP/C548051/1.

⋆⋆ Supported by the Computer and Automation Research Institute of the Hungarian Academy of Sciences.

J.T. Bradley (Ed.): EPEW 2009, LNCS 5652, pp. 30–43, 2009.

trading patterns. We feel that it would be presumptuous to try to model such systems in general and we propose to begin from modest premises and simple models, trying to build up to large systemic representations from solid building blocks based on first principles. This paper pursues the study of such systems using methods which has been successfully used in the performance analysis of computer systems and networks, and in operations research.

Among the techniques used in commerce and E-commerce, auctions have been the subject of some investigation [3]. They have also been used as tools for decision making in resource allocation problems [13] and for the coordination of software agents. The advantage of auctions is that the mechanisms that they use for negotiation between economic agents, and for decision, are clearly defined so that precise models of the resulting behaviour can be constructed. Models of auctions and sealed bids are amenable to rigorous analysis, as shown in early work on Martin Gardner's "Secretary, or Sultan's Dowry, Problem" where a recruiter selects the best candidate from a sequence of applicants, the quality of successive candidates are random variables, and the recruiter must make the irrevocable choice of a candidate from an initial sequence, without the possibility of further candidates being considered [1, 11] after the decision is made. Other analysis of auctions can be found in [8–10, 14, 15]. Of course there also direct links between auctions and networks due to the sale of wireless spectrum; however virtual auctions have also been suggested as a means of allocating network bandwidth [18] and the wireless spectrum, in real time, to competing users [19].

In this paper we pursue the work begun in [17, 20] by studying price formation in auctions. In simple terms, the analysis aims at obtaining the economic performance of an auction in terms of the price that is attained, which is of interest to the buyers, and the income per unit time, which is of interest to sellers. This paper first recalls some of the earlier work in [17], and then shows how it can be extended in several directions, and in particular to:

- (i) Include the effect of network QoS on the economic performance criteria, and in particular the impact of packet or message loss and delay.
- (ii) Discuss the effect of items which may not be available for sale, or whose arrival to the auction may be delayed.
- (iii) Economic mechanisms which differ from auctions, such as sealed bidding schemes, where (contrary to auctions) the value of the offers are not known to everyone until *all* the offers have been made, and a given deadline or time-out has expired.

In the following sections we first discuss the probabilistic structure that we use has inter-arrival times for bids, random variables for the increments or values of successive bids, and a probabilistic representation of the time that the seller takes to make his decision after a bid is received. We then compute measures of interest such as the income per unit time resulting from repeated auctions, and the price attained by the good being sold. Since we focus on systems that are built in software on top of computer networks, we analyse the impact that the network quality of service (QoS), such as packet losses and delays, will have on the economic outcome of the auction. Finally, we study similar models for sealed

bids, where the successive bidders may not be aware of the price attained by the preceding bids. Thus our work also provides a handle to examining the links between economic performance and the performance of the underlying computer networks that support this economic activity, and we hope that this paper can provide a novel link between trading systems that reside on the Internet and computer system and network performance evaluation, combining traditional performance analysis together with the economic performance at the application level.

2 Analysis of Auctions with Competitive Bids

As mentioned in the introduction, auctions are very common and many will have a direct personal experience of such mechanisms. In this section we focus on English auctions with a known fixed *reserve price* s so that the bidding process only really begins when a bid arrives that exceeds this price. The notion of reserve price is that the seller will only sell the good if the bid is higher than s which is known in advance both to the bidders and to the seller.

As with all English auctions, successive bidders have to make a bid which is *higher* than the previous bid, and the bidder cannot renege: i.e. after a bid is made, if the seller accepts it then the price is that of the highest bid. We also consider an auction centre which is running many successive auctions, which we model as unlimited sequence of statistically identical auctions, with a rest period between successive auctions. This approach will also allow us to compute the relevant quantities for a single auction, i.e. the first and last one, if auctions do not repeat themselves. We assume that the first auction starts at time $t = 0$, and that bids arrive at random times $0 < T_1 < T_2 < \ldots$. Denote by B_i the bid generated at T_i. The $(i+1)th$ bid is of value $B_{i+1} = B_i + I_{i+1}$ where the increment I_{i+1} is a positive random variable representing the increment over the previous bid, while the first bid needs to be equal to the reserve price $B_1 = s$.

Decisions on the part of the seller during the first auction are represented by the *decision delays* $D_i \geq 0$ which are random variables. If $T_{i+1} < T_i + D_i$ then the ith bid is superseded by the $(i+1)th$ bid, while when $T_{i+1} \geq T_i + D_i$, then the seller accepts the ith bid at instant $T_i + D_i$ and the first auction ends at that instant. Thus the $1st$ auction ends at the Nth bid where N is the positive random variable:

$$N = \min\{i : T_i + D_i \leq T_{i+1}\} \tag{1}$$

and hence the *seller's income* brought by this sale is:

$$S = B_N \tag{2}$$

After this first sale is completed, the administrative details required by the sale will take another time duration $R \geq 0$, so the duration of the auction is

$$\tau = R + T_N + D_N. \tag{3}$$

We have to calculate the expected income $\mathbf{E}\{B_N\}$, and the expected duration of the auction $\mathbf{E}\{R + T_N + D_N\}$. Once the first auction period is over, the whole

auction repeats itself indefinitely and independently of the previous auction. Let (S_i, τ_i) denote the pair of the income and duration of the ith auction, so that after the nth auction we have the total *income per unit time* resulting from this sequence of auctions is:

$$\phi_n = \frac{\sum_{i=1}^n S_i}{\sum_{i=1}^n \tau_i}.$$

By the strong law of large numbers, we have that

$$\phi_n = \frac{\frac{1}{n}\sum_{i=1}^n S_i}{\frac{1}{n}\sum_{i=1}^n \tau_i} \to \phi = \frac{\mathbf{E}\{S\}}{\mathbf{E}\{\tau\}} = \frac{\mathbf{E}\{B_N\}}{\mathbf{E}\{R + T_N + D_N\}}. \tag{4}$$

In order to characterize the process that we have just described, some assumptions will be needed about these random variables. We characterize the assumptions as follows: assume that $\{(T_i, B_i, D_i)\}_{i=1}^{\infty}$ is a Markov process such that the transition probabilities have a special form:

$$\begin{aligned}
&\mathbf{P}\{T_{i+1} - T_i \le t, I_{i+1} \le u, D_{i+1} \le d \mid B_i = b\} \\
&= \mathbf{P}\{T_{i+1} - T_i \le t \mid B_i = b\} \cdot \mathbf{P}\{I_{i+1} \le u \mid B_i = b\} \cdot \mathbf{P}\{D_{i+1} \le d \mid B_i = b\} \\
&= A(t \mid b) \cdot I(u \mid b) \cdot D(d \mid b),
\end{aligned}$$

so that we recognise the fact that the increments I_{i+1} will depend on the price that has been attained in the most recent bid $B_i = b$, the inter-arrival time of the bids $T_{i+1} - T_i$ and the decision delay D_{i+1} also depend on the value of the most recent bid $B_i = b$; indeed as the price of the bid increases, we may expect that new bids arrive more slowly while the seller may also become more anxious to sell. Finally we assume that R is a random variable with some general distribution function

$$R(t) = \mathbf{P}\{R \le t\}$$

and that it is independent of all the preceding random variables. It is the model of the situation, where there is an infinite buyers' population, and each buyer makes bids sequentially such that they have feedback information from the seller on the bids, moreover each buyer knows the initial and end time points of the auction period.

2.1 The Case with Markovian Bid Arrivals, Exponential Decision Times, and Unit Increments

A special case of the model we have described was analyzed in [17] under the assumption of unit increments for successive bids $I_{i+1} = B_{i+1} - B_i = +1$ $(i \ge 1)$, while $B_1 = s$. That analysis also assumed that the bids' successive interarrival times $T_{i+1} - T_i$ are independent and exponentially distributed with parameter λ_b $(i \ge 1)$, dependent on b the value of the most recent bid, and that T_1 has exponential distribution with parameter λ_0. The decision times D_{i+1} are also mutually independent and exponentially distributed with parameter δ_b dependent on the value b of the most recent bid. Assume also that the rest periods R

are exponentially distributed with average value 1, so that all times and rates have been normalized to the unit value of the rest period.

Under these assumption the price S_t is a birth process, therefore we construct a state-space, where state 0 is the state in which the auction has started or restarted and the seller is waiting for the $1st$ bid, state l is the state in which the current bid is at level $l \geq s$, where s is as before, the "reserve price" or the minimum acceptable level for a bid. Finally $a(l)$ is state in which the bid of level l has been accepted, and the auction 'rests" for a time whose average value is 1 before entering the 0 state where the auction starts again. In the steady-state the corresponding probabilities $p(0),\ p(l),\ p(a(l))$ will satisfy the following balance equations:

$$[\lambda_l + \delta_l]p(l) = \lambda_{l-1}p(l-1),\ l > s > 0,$$

and

$$[\lambda_s + \delta_s]p(s) = \lambda_0 p(0),$$

and

$$p(a(l)) = \delta_l p(l), l \geq s > 0,$$

and

$$\lambda_0 p(0) = \sum_{l=s}^{\infty} p(a(l)),$$

so that

$$p(0) = \sum_{l=s}^{\infty} \frac{\delta_l}{\lambda_0} p(l).$$

These equations together with the condition that the sum of the probabilities is one, give us:

$$p(l) = p(s) \prod_{k=s+1}^{l} \frac{\lambda_{k-1}}{\lambda_k + \delta_k}, l > s,$$

$$1 = p(0) + \sum_{l=s}^{\infty} [p(l) + p(a(l))] = p(0) + \sum_{l=s}^{\infty} p(l)[1 + \delta_l],$$

$$1 = \frac{\delta_s}{\lambda_0} p(s) + p(s)(1 + \delta_s) + p(s) \sum_{l=s+1}^{\infty} \left[1 + \delta_l + \frac{\delta_l}{\lambda_0}\right] \prod_{k=s+1}^{l} \frac{\lambda_{k-1}}{\lambda_k + \delta_k}$$

so that:

$$p(s) = \left[1 + \frac{\delta_s}{\lambda_0} + \delta_s + \sum_{l=s+1}^{\infty} [1 + \delta_l + \frac{\delta_l}{\lambda_0}] \prod_{k=s+1}^{l} \frac{\lambda_{k-1}}{\lambda_k + \delta_k}\right]^{-1}$$

$$p(0) = \frac{\delta_s}{\lambda_0} + \sum_{l=s+1}^{\infty} \frac{\delta_l}{\lambda_0} \prod_{k=s+1}^{l} \frac{\lambda_{k-1}}{\lambda_k + \delta_k} 1 + \frac{\delta_s}{\lambda_0} + \delta_s$$

$$+ \sum_{l=s+1}^{\infty} \left[1 + \delta_l + \frac{\delta_l}{\lambda_0}\right] \prod_{k=s+1}^{l} \frac{\lambda_{k-1}}{\lambda_k + \delta_k}$$

From these expressions, many of the measures of interest can be derived. For instance, if the total duration of an auction from when it starts to when it ends, excluding the rest time, is denoted by $T = \tau - R$, then

$$p(0) = \frac{1 + \frac{1}{\lambda_0}}{1 + \frac{1}{\lambda_0} + \mathbf{E}\{T\}} \tag{5}$$

so that the expected duration of the auction until a sale occurs is:

$$\mathbf{E}\{T\} = \left[1 + \frac{1}{\lambda_0}\right] \left[\frac{1}{p(0)} - 1\right] \tag{6}$$

Similarly, we are interested in knowing what the average sale price is, and it is given by

$$\mathbf{E}\{S\} = \sum_{l=s}^{\infty} lp(a(l)) = s\delta_s p(s) + p(s) \sum_{l=s+1}^{\infty} l\delta_l \prod_{k=s+1}^{l} \frac{\lambda_{k-1}}{\lambda_k + \delta_k}, \tag{7}$$

so that the average income per unit time for the seller becomes:

$$\phi = \frac{\mathbf{E}\{S\}}{1 + \frac{1}{\lambda_0} + \mathbf{E}\{T\}} \tag{8}$$

2.2 The Effect of Network QoS

We imagine that an "auction centre", i.e. a computer system located somewhere in the Internet which is used by various buyers and sellers to run their economic transactions, is currently used by our seller to run his auction. Bidders then have to access this auction centre via the network when they wish to make a bid. Similarly the seller may be located elsewhere in the Internet and also receives information about bids from the auction centre and then communicates its decision to seal or to wait via the Internet to the auction centre.

As indicated in the introduction, when both bidders and the seller access the auction centre and the software that is running the auction, the network quality of service (QoS), which typically includes packet losses, delay and jitter, will also affect the economic performance of the auction. Thus the network introduces additional effects on the auctions, including:

- Delaying the arrival of the bids due to normal packet travel delays and also potentially because of congestion,
- Reversing the order of some of the bids and thus introducing unfairness with later bidders being taken into account before earlier bidders due to the fact that some bidders' paths through the network may be shorter or less congested than other bidders' paths, which can be exacerbated by "jitter" or significant variance in the packet delays,
- Losing some of the bids due to packet losses so that the price increases are smaller and hence further delayed, and

– Delaying the arrival of information concerning the seller's acceptance of a bid due to normal delays or congestion, including a "round-trip" delay due to the information needing to reach the seller from the auction centre about the most recent bid, and the seller having to convey its decision back to the auction centre.

Such adverse effects can occur when the network is congested and when a given auction centre is very "popular" and hence highly utilised or sought after. Here we will consider the latter two effects. Let us first consider ϕ in the right-hand-side of (4).

Let p_i be the probability that the ith bid is lost by the network before it arrives at the auction centre, and let $\hat{D}_i = D_i + r_i$ be the net decision delay which includes the round-trip delay from auction centre to seller and back. Let the bid inter-arrival times be independent and identically distributed with density without packet losses $f(x)dx = \mathbf{P}[x \leq T_{i+1} - T_i < x + dx]$ with Laplace–Stieltjes Transform (LST) $f^*(s)$. Now if $\{T'_i\}_{i=1}^{\infty}$ is the sequence of effective bid arrival instants excluding those bids which have been lost, if the losses occur in an independent and identically distributed manner with probability and p, then the LST of the density function of the random variable $(T'_{i+1} - T'_i)$ representing the effective inter-arrival times of bids is obviously:

$$g^*(s) \equiv \int_0^\infty \mathbf{P}[x \leq T'_{i+1} - T'_i < x + dx]e^{-sx} \tag{9}$$

$$= \sum_{j=0}^{\infty} (f^*(s))^{j+1} p^j (1-p) = \frac{(1-p)f^*(s)}{1 - pf^*(s)}, \tag{10}$$

and the expected value is:

$$\mathbf{E}[T'_{i+1} - T'_i] = \frac{\mathbf{E}[T_{i+1} - T_i]}{1-p} \tag{11}$$

In the case of Poisson arrivals, $f(x) = \lambda e^{-\lambda x}$ for $\lambda > 0$, giving $g(x) = (1 - p)\lambda e^{-\lambda(1-p)}$. Just for the sake of evaluating the effect of both packet losses and of delayed decision, suppose that decisions are taken after a constant time D and that the network will delay a decision from reaching the auction centre by a further round-trip constant value r which includes the time it takes the bidder to be informed of the current price of the good, and the time it takes the bid to reach the auction centre. Then a decision *not* to sell is taken with probability $q = \mathbf{P}\{D + r > T'_{i+1} - T'_i\} = 1 - e^{-(D+r)(1-p)\lambda}$ which is the probability that a new bid arrives before the decision to sell is taken, and it includes the effect of loss. If all bids have an average increment of $E[I]$ over the previous bid, and if the expected value of the first bid is $E[I_s]$, then

$$\phi = \frac{\mathbf{E}[I_s] + \frac{q\mathbf{E}[I]}{1-q}}{R + D + r + \frac{q}{1-q}\mathbf{E}\{(T'_{i+1} - T'_i)I_{\{r+D>T'_{i+1}-T'_i\}}\}}, \tag{12}$$

where I is the indicator function. Note that before a bid is accepted, a total average number of $1/(1-q)$ bids occur, including the last one. The total auction

time includes the time for the acceptance of the final bid which is $(D+r)$, and the time for an average number $q/(1-q)$ bids to arrive and these are all turned down because a new bid arrives before the previous bid can be accepted. The bids that are turned down wait on average for a time $\mathbf{E}\{(T'_{i+1}-T'_i)I_{\{r+D>T'_{i+1}-T'_i\}}\}$ before they are superseded by a higher bid. On the other hand, the average income per auction is obviously:

$$\Upsilon = \mathbf{E}[I_s] + \frac{q}{1-q}\mathbf{E}[I] \tag{13}$$

Clearly, Since

$$\mathbf{E}\{(T'_{i+1} - T'_i)I_{\{r+D>T'_{i+1}-T'_i\}}\} = \frac{q}{\lambda(1-p)} - (1-q)(r+D) \tag{14}$$

we obtain an expression that relates the economic "success" of the auction and the network QoS parameters q and r:

$$\begin{aligned} \phi &= \frac{\mathbf{E}[I_s] + \frac{q}{1-q}\mathbf{E}[I]}{R + D + r + \frac{q^2}{\lambda(1-p)(1-q)} - q(r+D)} \\ &= \frac{(1-q)\mathbf{E}[I_s] + q\mathbf{E}[I]}{R(1-q) + (D+r)(1-q)^2 + \frac{q^2}{\lambda(1-p)}} \end{aligned} \tag{15}$$

2.3 The Case When Goods Are Not Always Available for Sale

Up to now, we have implicitly assumed that there is an unlimited backlog of goods to be sold, waiting in the "seller's store room" for a sale to occur. This was equivalent to assuming that during a rest period, a new good is brought to the seller so that the next auction can start after the previous sale is completed and the rest period ends. Here we will modify the assumption, so that goods also arrive to the auction according to an arrival process, and an auction cannot start until there is at least one item for sale available. Thus we will suppose that goods for sale arrive singly to the seller at instants $0 < a_1 < a_2 < \ldots$, so that bids can only start after the good is actually available for sale. We will also define the instants $0 < d_1 < d_2 < \ldots$ when the successive items are sold.

In this case, if we look at things from the point of view of whoever *owns* the goods and is anxious to sell them, a significant measure of interest is $W_j = (d_j - a_j)$ the total time that a good has to wait before it actually ends up being sold. As before, let $\tau_j = \sum_{i=1}^{n(i)} A_{ji} + D_{j,n(j)} + R_j$. We then have:

- $d_1 = a_1 + \tau_1 - R_1$,
- $d_{j+1} = d_j + R_j + \tau_{j+1} - R_{j+1}$ if $a_{j+1} \leq d_j + R_j$,
- $d_{j+1} = a_{j+1} + \tau_{j+1} - R_{j+1}$ if $a_{j+1} > d_j + R_j$.

or equivalently:

- $W_1 = \tau_1 - R_1$,

- $W_{j+1} = W_j + a_j + R_j + \tau_{j+1} - R_{j+1} - a_{j+1}$ if $a_{j+1} \leq d_j + R_j$ or $a_{j+1} - a_j \leq W_j + R_j$,
- $W_{j+1} = \tau_{j+1} - R_{j+1}$ if $a_{j+1} > d_j + R_j$ or $a_{j+1} - a_j > W_j + R_j$.

which gives us an equational form similar to Lindley's equation [7]:

$$W_{j+1} = \tau_{j+1} - R_{j+1} + [W_j + R_j - (a_{j+1} - a_j)]^+ \tag{16}$$

where $[X]^+ = X$ if $X > 0$ and $[X]^+ = 0$ if $X \leq 0$.

3 The Analysis of Sealed Memoryless Bids

Sealed bids differ from auctions in the essential point that the seller does *not know* the amount of each of the bids it receives until it stops the bidding process and it opens the "sealed envelopes" that contain the details of each bid. In some cases, the seller may also not know *how many* bids have been submitted, so that it must make its decision to close a bid and accept the highest bid without knowing either its amount nor the number of bids. This section will be devoted to deriving analytical results for this economic mechanism.

Let $0 < T_1 < T_2 < \ldots$ be the random points of a Poisson process with intensity λ, representing the arrival instants of the bids. Denote by B_i the bid generated at T_i, and assume that the $B_1, B_2, \ldots$ are independent, identically distributed random variables with distribution function

$$F(z) = \mathbf{P}\{B_1 < z\}.$$

Here, the marked Poisson process $\{T_i, B_i\}_{i=1}^{\infty}$ (cf. Karr [5]) models a situation where where there is an infinite buyers' population, and each buyer *bids once in his/her life* and the buyers "don't see" each other, i.e., they have no feedback information on the successful bid, contrary to the model in the previous section, and they therefore generate bids independently of each other.

As previously, the seller has a decision delay time D *which is fixed in advance once and for all*, which is assumed to be an exponentially distributed random variable with parameter $\delta > 0$:

$$G(z) = \mathbf{P}\{D \leq z\} = 1 - e^{-\delta z},$$

and we assume that $\{B_i\}$, $\{T_i\}$ and D are independent. The seller has a minimum sale price s:

$$\mathbf{P}\{B_1 \geq s\} = 1 - F(s) = 1.$$

No sale occurs if there are no bids that arrive before time D. On the other hand, we assume that the seller can observe the bids and accepts a bid that arrives before D; so in this latter case, finding the best strategy is a version of the secretary problem.

As a *lower bound* on the performance of the best strategy, consider the case when the seller accepts the first bid within the decision delay D, and let $\hat{S}$ be the price that is then obtained. We then have

$$\hat{S} = B_1 I_{\{T_1 \le D\}},$$

where I is the indicator function. We can obtain $\mathbf{E}\{\hat{S}\}$ from:

$$\begin{aligned}\mathbf{E}\{\hat{S}\} &= \mathbf{E}\{B_1 I_{\{T_1 \le D\}}\} = \mathbf{E}\{B_1\}\mathbf{E}\{\mathbf{P}\{T_1 \le D \mid T_1\}\} \\ &= \mathbf{E}\{B_1\}\mathbf{E}\{e^{-\delta T_1}\} = \mathbf{E}\{B_1\}\frac{\lambda}{\lambda+\delta}.\end{aligned}$$

An *upper bound* of the best strategy when the seller chooses the successful buyer by selecting the maximum valued bid within the decision delay D can also be obtained as follows. If N_D is the number of bid arrival instants in $[0, D]$:

$$N_D = \#\{n,\, T_n \le D\}$$

then the price obtained $\tilde{S}$ is:

$$\tilde{S} = \max_{1 \le i \le N_D} B_i.$$

Since $\{B_i\}$, $\{T_i\}$ and D are independent:

$$\mathbf{P}\{N_D = n\} = \int_0^\infty \mathbf{P}\{N_D = n \mid D = t\} dG(t) = \int_0^\infty \frac{(\lambda t)^n}{n!} e^{-\lambda t} dG(t),$$

to obtain $\mathbf{E}\{\tilde{S}\}$, we calculate the tail distribution of $\tilde{S}$ as follows. For any $z \ge s$:

$$\begin{aligned}\mathbf{P}\{\tilde{S} \ge z\} &= \sum_{n=0}^{\infty} \mathbf{P}\{\tilde{S} \ge z \mid N_D = n\}\mathbf{P}\{N_D = n\} \\ &= \sum_{n=0}^{\infty} \mathbf{P}\left\{\max_{1 \le i \le N_D} B_i \ge z \mid N_D = n\right\}\mathbf{P}\{N_D = n\} \\ &= \sum_{n=1}^{\infty} \mathbf{P}\left\{\max_{1 \le i \le n} B_i \ge z\right\}\mathbf{P}\{N_D = n\}\end{aligned}$$

therefore

$$\begin{aligned}\mathbf{P}\{\tilde{S} \ge z\} &= \sum_{n=1}^{\infty} \left(1 - \mathbf{P}\left\{\max_{1 \le i \le n} B_i < z\right\}\right)\mathbf{P}\{N_D = n\} \\ &= \sum_{n=1}^{\infty} (1 - F(z)^n) \int_0^\infty \frac{(\lambda t)^n}{n!} e^{-\lambda t} dG(t) \\ &= \sum_{n=0}^{\infty} (1 - F(z)^n) \int_0^\infty \frac{(\lambda t)^n}{n!} e^{-\lambda t} dG(t) \\ &= 1 - \int_0^\infty e^{-\lambda(1-F(z))t} \delta e^{-\delta t} dt \\ &= 1 - \frac{\delta}{\lambda(1-F(z)) + \delta} \\ &= \frac{\lambda(1-F(z))}{\lambda(1-F(z)) + \delta}.\end{aligned}$$

On the other hand for $0 < z < s$,

$$\begin{aligned}\mathbf{P}\{\tilde{S} \geq z\} &= \mathbf{P}\{\tilde{S} \geq s\} = 1 - \mathbf{P}\{N_D = 0\} \\ &= 1 - \int_0^\infty e^{-\lambda t} dG(t) = 1 - \int_0^\infty e^{-\lambda t} \delta e^{-\delta t} dt = \frac{\lambda}{\lambda + \delta},\end{aligned}$$

implying that

$$\begin{aligned}\mathbf{E}\{\tilde{S}\} &= \int_0^\infty \mathbf{P}\{\tilde{S} \geq z\} dz \\ &= \int_0^s \mathbf{P}\{\tilde{S} \geq z\} dz + \int_s^\infty \mathbf{P}\{\tilde{S} \geq z\} dz \\ &= s\frac{\lambda}{\lambda + \delta} + \int_s^\infty \frac{\lambda(1 - F(z))}{\lambda(1 - F(z)) + \delta} dz.\end{aligned}$$

A simple stopping rule can now be introduced by a sequence of thresholds $v_1 \geq v_2 \geq \ldots$ such that the bid B_i is accepted if it exceeds the threshold v_i and there was no prior bid with this property. Let S be the price attained in this manner. Then:

$$S = \sum_{i=1}^\infty I_{\{B_1 < v_1, \ldots, B_{i-1} < v_{i-1}, B_i \geq v_i\}} I_{\{T_i \leq D\}} B_i.$$

Since the duration of the auction is:

$$\tau = R + \sum_{i=1}^\infty I_{\{B_1 < v_1, \ldots, B_{i-1} < v_{i-1}, B_i \geq v_i\}} \min\{T_i, D\}.$$

Then

$$\mathbf{E}\{S \mid \{T_i\}, D\} = \sum_{i=1}^\infty \prod_{j=1}^{i-1} F(v_j) \int_{v_i}^\infty b dF(b) I_{\{T_i \leq D\}},$$

and

$$\mathbf{E}\{S\} = \sum_{i=1}^\infty \prod_{j=1}^{i-1} F(v_j) \int_{v_i}^\infty b dF(b) \mathbf{P}\{T_i \leq D\},$$

If $Y_i = T_i - T_{i-1}$ denotes the ith inter-arrival time of bids then we have:

$$\begin{aligned}\mathbf{P}\{T_i \leq D\} &= \mathbf{E}\{\mathbf{P}\{T_i \leq D \mid T_i\}\} \\ &= \mathbf{E}\{e^{-\delta T_i}\} \\ &= \mathbf{E}\{e^{-\delta \sum_{j=1}^i Y_j}\} \\ &= (\mathbf{E}\{e^{-\delta Y_1}\})^i \\ &= \left(\frac{\lambda}{\lambda + \delta}\right)^i,\end{aligned}$$

implying that

$$\mathbf{E}\{S\} = \sum_{i=1}^\infty \prod_{j=1}^{i-1} F(v_j) \int_{v_i}^\infty b dF(b) \left(\frac{\lambda}{\lambda + \delta}\right)^i.$$

Similarly,

$$\mathbf{E}\{\tau \mid \{T_i\}, D\} = 1 + \sum_{i=1}^{\infty} \prod_{j=1}^{i-1} F(v_j)(1 - F(v_i)) \min\{T_i, D\},$$

therefore

$$\mathbf{E}\{\tau\} = 1 + \sum_{i=1}^{\infty} \prod_{j=1}^{i-1} F(v_j)(1 - F(v_i))\mathbf{E}\{\min\{T_i, D\}\}.$$

Because:

$$\min\{T_i, D\} = D - (D - T_i)^+,$$

one can verify that

$$\begin{aligned}\mathbf{E}\{\min\{T_i, D\} \mid T_i\} &= \mathbf{E}\{D\} - \mathbf{E}\{(D - T_i)^+ \mid T_i\} \\ &= \mathbf{E}\{D\} - e^{-\delta T_i} \int_{T_i}^{\infty} \delta e^{-\delta(t - T_i)} dt = \frac{1}{\delta}\left(1 - e^{-\delta T_i}\right),\end{aligned}$$

therefore

$$\mathbf{E}\{\min\{T_i, D\}\} = \frac{1}{\delta}\mathbf{E}\left\{1 - e^{-\delta T_i}\right\} = \frac{1}{\delta}\left(1 - \left(\frac{\lambda}{\lambda + \delta}\right)^i\right),$$

and so

$$\mathbf{E}\{\tau\} = 1 + \frac{1}{\delta}\left(1 - \sum_{i=1}^{\infty} \prod_{j=1}^{i-1} F(v_j)(1 - F(v_i)) \left(\frac{\lambda}{\lambda + \delta}\right)^i\right).$$

Introducing the notation

$$p_i = \prod_{j=1}^{i-1} F(v_j)(1 - F(v_i))$$

then

$$\mathbf{E}\{S\} = \sum_{i=1}^{\infty} p_i \mathbf{E}\{B \mid B \geq v_i\} \left(\frac{\lambda}{\lambda + \delta}\right)^i$$

and

$$\mathbf{E}\{\tau\} = 1 + \frac{1}{\delta}\left(1 - \sum_{i=1}^{\infty} p_i \left(\frac{\lambda}{\lambda + \delta}\right)^i\right).$$

This leads to the numerical problem of choosing the thresholds $\{v_i\}$, which maximize the income per unit time:

$$\frac{\mathbf{E}\{S\}}{\mathbf{E}\{\tau\}} = \frac{\sum_{i=1}^{\infty} p_i \mathbf{E}\{B \mid B \geq v_i\} \left(\frac{\lambda}{\lambda+\delta}\right)^i}{1 + \frac{1}{\delta}\left(1 - \sum_{i=1}^{\infty} p_i \left(\frac{\lambda}{\lambda+\delta}\right)^i\right)}.$$

4 Conclusions

In this paper, we have considered models of economic activities which can be represented as auctions and as sealed bids. Our purpose has to show how such economic activities can be studied using techniques which are commonly used in computer and network performance analysis and in areas of operations research, such as queueing and inventory theory. We have shown how such models can be constructed from first principles, and how they can lead to analytical solutions which provide insight into price formation and how they can be used for the optimisation of economic performance.

In particular, this paper has extended the previous work of one of the authors to include the impact of network QoS parameters such as message loss and delay, and to show how sealed bids, which differ significantly from auctions in the information that is available both to the parties making the offers and to the decider, can also be analysed with a similar approach. It is hoped that our results can motivate further work by attracting the attention of the computer system performance evaluation community to the study of some of the important "applications" which run on the Internet, namely those which involve automated economic transactions such as networked auctions.

Acknowledgements

We thank the anonymous referees for comments which helped us clarify the contributions of this paper, and also for their detailed comments concerning typos and omissions.

References

1. Chow, Y.S., Moriguti, S., Robbins, H., Samuels, S.M.: Optimal selection based on relative rank (the Secretary problem). Israel J. Math. 2, 81–90 (1964)
2. Gelenbe, E., Mitrani, I.: Analysis and Synthesis of Computer Systems. Academic Press, New York (1980); Imperial College Press & World Scientific, Singapore and London (2nd edn.) (in press, 2009)
3. Milgrom, P.R., Weber, R.: A theory of auctions and competitive bidding. Econometrica 50, 1089–1122 (1982)
4. McAfee, R.P., McMillan, J.: Auctions and bidding. J. Economic Literature 25, 699–738 (1987)
5. Karr, A.F.: Point Processes and their Statistical Inference. Marcel Dekker, Inc., New York (1991)
6. Medhi, J.: Stochastic Processes, 2nd edn. Wiley Eastern Ltd., New Delhi (1994)
7. Gelenbe, E., Pujolle, G.: Introduction to Networks of Queues, 2nd edn. J. Wiley & Sons, Chichester (1998)
8. Shehory, O.: Optimality and risk in purchase from multiple auctions. In: Klusch, M., Zambonelli, F. (eds.) CIA 2001. LNCS (LNAI), vol. 2182, pp. 142–153. Springer, Heidelberg (2001)
9. Guo, X.: An optimal strategy for sellers in an online auction. ACM Trans. Internet Tech. 2(1), 1–13 (2002)

10. Shehory, O.: Optimal bidding in multiple concurrent auctions. International Journal of Cooperative Information Systems 11(3–4), 315–327 (2002)
11. Finch, S.R.: Optimal stopping constants. Mathematical Constants, pp. 361–363. Cambridge Univ. Press, Cambridge (2003)
12. Gelenbe, E.: Sensible decisions based on QoS. Computational Management Science 1(1), 1–14 (2003)
13. Dash, N.R., Jennings, N.R., Parkes, D.C.: Computational mechanism design: a call to arms. In: IEEE Intelligent Systems, November-December 2003, pp. 40–47 (2003)
14. Hajiaghayi, M.T., Kleinberg, R., Parkes, D.C.: Adaptive limited-supply online auctions. In: Proc. 5th ACM Conference on Electronic Commerce, May 17–20, pp. 71–90. ACM Press, New York (2004)
15. David, E., Rogers, A., Schiff, J., Kraus, S., Jennings, N.R.: Optimal design of English auctions with discrete bid levels. In: Proc. of 6th ACM Conference on Electronic Commerce (EC 2005), Vancouver, Canada, pp. 98–107 (2005)
16. Fatima, S., Wooldridge, M., Jennings, N.R.: Sequential auctions for objects with common and private values. In: Proc. 4th Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems, Utrecht, Netherlands, pp. 635–642 (2005)
17. Gelenbe, E.: Analysis of automated auctions. In: Levi, A., Savaş, E., Yenigün, H., Balcısoy, S., Saygın, Y. (eds.) ISCIS 2006. LNCS, vol. 4263, pp. 1–12. Springer, Heidelberg (2006)
18. Dramitinos, M., Stamoulis, G., Courcoubetis, C.: An auction mechanism for allocating the bandwidth of networks to their users. Computer Networks 51(18), 4979–4996 (2007)
19. Kovacs, L., Vidacs, A., Heder, B.: Spectrum auction and pricing in dynamic spectrum allocation networks. The Mediterranean Journal of Computers and Networks 4(3), 125–138 (2008)
20. Gelenbe, E.: Analyis of single and networked auctions. ACM Trans. Internet Technology 9(2), Article 8 (May 2009)

Applying Symbolic Techniques to the Representation of Non-Markovian Models with Continuous PH Distributions

Francesco Longo and Marco Scarpa

Dipartimento di Matematica, Università di Messina, 98166 Messina, Italy
{flongo,mscarpa}@unime.it

Abstract. Among the proposed techniques for the analysis of non-Markovian models the state space expansion approach showed great flexibility in terms of modelling capacities. The principal drawback is the explosion of the state space. An attempt to alleviate such problem has been made in [1] but the storing of the reachability graph of the untimed system, augmented with information about active but not enabled events, still remains a bottleneck. This paper suggests a method for storing such an augmented reachability graph by the use of a Multi-terminal Multi-valued Decision Diagram and few Kronecker matrices. All the needed information is collected by applying a Saturation based algorithm that represents the main contribution of the work. An estimation of the memory occupation is also reported.

1 Introduction

Markovian models have been widely used to represent any kind of systems with the purpose of studying their performance and reliability. Their success is due to the simplicity and effectiveness of the analysis methods to evaluate *continuous time Markov chain* (CTMC) underlying the models. However there are practical situations in which the Markovian hypothesis is not applicable. In fact, many activities in computer, communication and manufacturing systems are more likely represented by non-exponentially distributed random variables that have a real impact on the derived measures, especially when transient analysis is performed.

The introduction of non-exponentially distributed events within stochastic models presents two major drawbacks: 1) events can be characterized by different memory policies [18] [3], 2) the numerical techniques used to solve the derived *non-Markovian models* are very complex. An extensive work has been carried out to solve such problems [8] [11] [10] [2] and an effective solution has been to represent non-exponentially distributed events via *Phase Type* (*PH*) distributions [15]. Such technique approximates the original non-Markovian process by mean of a Markov chain defined over an *expanded state space*. Each state in the non-Markovian process is expanded into a set of states within the Markov chain, called *macrostate*, with the purpose to capture the evolution of the non-exponentially distributed events [2].

J.T. Bradley (Ed.): EPEW 2009, LNCS 5652, pp. 44–58, 2009.

The main advantage of *state space expansion* approach is represented by the capacity of modelling any possible combination of memory policies and any number of concurrently enabled events. The drawback is the explosion of state space. The *state space explosion* problem is well known in the field of asynchronous systems and it limits the dimension of the manageable reachability sets even in the case of Markovian models. It is straightforward that it becomes a bottleneck especially in the contest of the state space expansion approach given that such approach is based on the multiplication of the system states.

An attempt to alleviate the memory consumption of the expansion technique has been done in [1], where *Stochastic Petri net* (SPN) models with PH distributed firing time transitions has been considered. In such work, the expanded reachability graph is symbolically described through macrostates, built by combining the reachability graph of the original untimed *Petri net* (PN), augmented with additional information about the active but not enabled transitions, and the matrices representing the discrete PH distributions. The expanded macrostates are not explicitly stored in memory but they are algorithmically evaluated on-the-fly through the use of Kronecker operators. In such a way only the *augmented reachability graph* of the untimed PN and the matrices associated to the PH distributions are explicitly stored. Even if the obtained results were encouraging, the storing of the augmented reachability graph remains a bottleneck.

The problem of managing huge state spaces is not new and it has been faced in [5] and [6] by Ciardo et al. Their method, called *Saturation*, exploits symbolic techniques for the generation and the representation of the reachability graph of asynchronous systems by using *Multi-Valued Decision Diagrams* (MDDs) and Kronecker matrices in order to improve memory consumption. However, Saturation is not directly applicable to the generation of the augmented reachability graph because it is not able to find and store the information about the active but not enabled events, necessary when the macrostates want to be algorithmically evaluated on-the-fly.

In this paper, we get inspired by the results obtained in [1], about the symbolical description of the macrostates, and in [5] and [6] about the symbolical representation of state spaces, and propose a new method for computing and storing in an efficient way the reachability graph of a system, augmented with the information about active but not enabled events, for all the system states. The goal of our work is achieved by using *Multi-terminal Multi-valued Decision Diagrams* (MTMDDs) [13] as data structure, and by collecting all the necessary information by applying a Saturation based algorithm. The main advantage of our approach is that both the expanded reachability graph and the augmented reachability graph of the untimed system are symbolically stored through the combination of an MTMDD and a certain number of Kronecker matrices.

The paper is organized as follows: in section 2, we introduce the reference model and recall some useful basic concepts. In section 3, we introduce the MTMDDs and present the algorithm to represent in a symbolic fashion the augmented reachability graph. Section 4 describes some numerical results, and, finally, Section 5 gives conclusions remarks and possible future work.

2 Model Definition and Basic Concepts

In this paper, we will refer to the discrete-state discrete-event model with generally distributed firing times, and individual memory policies defined in [12] for PNs. In such a model, timing is assigned to the events. Here, we assume that the random firing time of each event is described by a Continuous PH (CPH) distribution [15] accordingly to the state space expansion approach. We will refer to such kind of model as *CPH model*.

Definition 1. *A* CPH-model *is a tuple* $CPHM = (\hat{S}, S^{init}, \mathcal{E}, \mathcal{N}, \mathcal{C}, \mathcal{P})$*, where:*

- $\hat{S}$ *(of cardinality* $\left\|\hat{S}\right\|$*) is the potential state space;*
- $S^{init} \in \hat{S}$ *(of cardinality* $\left\|S^{init}\right\|$*) is the set of initial states;*
- $\mathcal{E}$ *(of cardinality* $\|\mathcal{E}\|$*) is the set of possible asynchronous events;*
- $\mathcal{N} : \hat{S} \to 2^{\hat{S}}$ *is the next-state function specifying the states that can be reached from a given state in a single step;*
- $\mathcal{C}$ *(of cardinality* $\|\mathcal{E}\|$*) is the set of CPH random variables associated to each event;*
- $\mathcal{P} : \mathcal{E} \to \{enabling, age\}$ *is a function that assigns a* preemption memory policy *to each event.*

An *age variable* a_e is assigned to each event e to keep track of the time during which it has been enabled. In a CPH-model, a_e represents the current stage in the CPH representation associated to e. The way in which a_e is related to the past history determines the different *preemptive memory policies*. In this work, we adopt the same names and semantic defined in [18] [3] to characterize the generally distributed firing time transitions of a non-Markovian SPN.

The preemption memory policies affect the age variable a_e in the following way: in the case of *enabling memory policy* a_e is reset to zero each time the event e either fires or is disabled in a new state; in the case of *age memory policy* a_e is reset to zero only when the event e fires otherwise it maintains its value.

If an event e presents an age variable $a_e > 0$ in a given state, it is said to be *active* in that state. We define $T_e^{(s)}$ and $T_a^{(s)}$ as the set of enabled events and *active but not enabled* events in state s respectively. At the entrance in a new state s, the residual firing time is computed for each event belonging to $T_e^{(s)}$, given its age variable. The next firing event is determined by the minimal residual firing time among the enabled events (*race policy* [12]).

2.1 Expanded Reachability Graph and Its Kronecker Representation

Let $S \subseteq \hat{S}$ be the system *state space*, i.e. the set of all the system states reached from the initial states S^{init}, and let $RG(S^{init})$ be the system *reachability graph* whose nodes are represented by system states and whose edges are labelled with system events. The system reachability graph can be explicitly represented in a

matrix form $\mathbf{R}$ of dimension $\|S\| \times \|S\|$ in which each entry $\mathbf{R}_{ij}$ contains the labels of the events whose firing changes state s_i into state s_j.

According to the state space expansion approach [16], the evolution of the stochastic process underlying the CPH-model can be represented by an expanded CTMC $\{Z(t) : t \geq 0\}$ whose states are defined over the pairs (s,A), where $s \in S$ is a reachable system state and A is the vector containing the values of the age variables a_e assigned to system events. All the states (s,A), characterized by the same value of s, form a *macrostate* in which the original state s is expanded within $Z(t)$.

The main features of the expansion algorithm are:

- let $\mathbf{Q}$ be the *infinitesimal generator matrix* of the expanded CTMC $Z(t)$, then $\mathbf{Q}$ can be represented as an $\|S\| \times \|S\|$ block matrix;
- non-null blocks of $\mathbf{Q}$ correspond to non-null elements in $\mathbf{R}$;
- The generic diagonal block $\mathbf{Q}_{ii}$ $(i = 1, \ldots, \|S\|)$ is a square matrix that describe the evolution of $Z(t)$ inside the macrostate related to the state s_i.
- The generic off-diagonal block $\mathbf{Q}_{ij}$ $(i, j = 1, \ldots, \|S\|, i \neq j)$ describes the transition from the macrostate related to s_i to the macrostate related to s_j.

Accordingly to the work presented in [1], the infinitesimal generator matrix $\mathbf{Q}$ of the expanded process doesn't need to be generated and stored as a whole but can be algorithmically evaluated on-the-fly when needed. This means that we do not encode the whole expanded state space (in other words the macrostates) generated by taking into account the different phases of all the CPHs modelling the system events, and, as a consequence, we encode each CPH distribution through the generator matrix of the CTMC associated to it. The generic non-null block $\mathbf{Q}_{ij}$ $(i, j = 1, \ldots, \|S\|)$ of $\mathbf{Q}$ can be symbolically represented through Kronecker expressions that take into consideration the matrix representations of the CPHs associated to the enabled and active but not enabled events either in s_i or in s_j. For a complete description of such expressions, in the case of discrete-time models, see [1]. Here we are interested in the knowledge of the reachability graph of the untimed system and of the sets $T_e^{(s)}$ and $T_a^{(s)}$, for all $s \in S$, since they need to be known when the method has to be implemented into a software tool. The sets $T_e^{(s)}$ are known from the model description, whereas $T_a^{(s)}$ have to be computed and stored in the most effective way along with the reachability graph of the untimed system.

2.2 Encoding S through MDDs

Great deal of work has been spent in recent years into developing space and time efficient techniques for the generation and the storage of huge state spaces. *Symbolic approaches* are the most successful to this purpose [4]. In contrast to explicit or enumerative techniques, where the entire set is explicitly stored and manipulated, symbolic techniques focus on generating compact representations of the set by exploiting model's structure and regularity.

A model has a structure when it is composed of K submodels, for some $K \in \mathbb{N}$. In this case a global system state can be represented as a K-tuple $(s^1, \ldots, s^K)$,

where s^k is the local state of submodel k, and the potential state space $\hat{S}$ can be considered as the cross product of each local state space S^k (having some finite size n^k): $\hat{S} = S^1 \times \cdots \times S^K$. In PNs, as an example, the set of places can be partitioned into K subnets and the marking can be written as the composition of the K corresponding submarkings. When identifying S^k with the initial integer interval $0, \ldots, n^k - 1$ it is possible to represent the state space $S \subseteq \hat{S}$ encoding its *characteristic function*, i.e. the function $\chi : \hat{S} \to \{0, 1\}$ where $\chi(s) = 1$ if $s \in S$ and 0 otherwise, for all $s \in \hat{S}$.

Numerous data structures have been employed to this purpose and the use of MDDs [17] have been introduced by Miner and Ciardo in [14]. MDDs are rooted, directed, acyclic graphs associated with a finite ordered set of integer variables. When K variables are used, the MDDs are able to represent Boolean functions in the form $f_M : \mathbb{Z}^K \to \mathbb{B}$. When used to encode a state space, an MDD has the following structure:

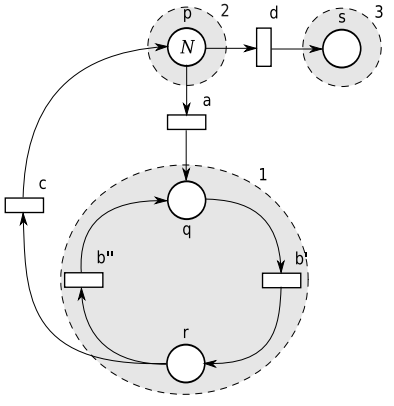
- nodes are organized into $K + 1$ levels, where K is the number of submodels;
- level K contains only a single non-terminal node, the root, whereas levels $K - 1$ through 1 contain one or more non-terminal nodes;
- a non-terminal node at level k has n^k arcs pointing to nodes at level $k - 1$;
- level 0 consists of two terminal nodes 0 and 1; they represent the return value of the characteristic function.

The value of the Boolean function f_M, represented by MDD M, for a given valuation of its integer variables, can be determined by tracing a path from its root node to one of the two terminal nodes. At each non-terminal node, the choice of the arc to be followed is determined by the value of the variable corresponding to that node. When the MDD encodes the characteristic function of a state space S then a state $s = (s^1, \ldots, s^K)$ belongs to S if and only if a path exists from the root node to the terminal node 1, such that at each node the arc corresponding to the local state s^k is followed.

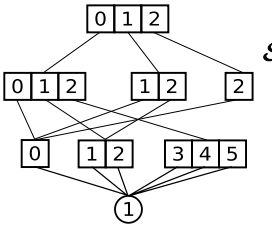
Fig. 1(a) shows the Petri net we will use as running example with a partitioning of its places into three subnets. Fig. 1(b) shows the associated MDD, the local state space of each subnet and the encoded global state space in the case of $N = 2$. In the MDD, the paths to terminal node 0 are not shown and they even don't need to be explicitly stored in memory.

2.3 Using Saturation to Compute $RG(S^{init})$

In [5], and then in [6], Ciardo et al. proposed an efficient algorithm for the generation of reachability graphs using MDDs. They started from the observation that in many cases, such as PNs and process algebras, a model expresses its next-state function as a union $\mathcal{N} = \bigcup_{e \in \mathcal{E}} \mathcal{N}_e$, where $\mathcal{N}_e$ is the next-state function associated with event e. Moreover asynchronous systems often exhibit a product-form behaviour such that, for each event e, the next-state function $\mathcal{N}_e$ can be written as a cross-product of K local functions, i.e. $\mathcal{N}_e = \mathcal{N}_e^1 \times \cdots \times \mathcal{N}_e^K$ where $\mathcal{N}_e^k : \hat{S}^k \to 2^{\hat{S}^k}$, for all $1 \leq k \leq K$.



(a)



$\mathcal{S} = \{000, \quad 011,$
$012, \quad 023,$
$024, \quad 025,$
$110, \quad 121,$
$122, \quad 220\}$

$\mathcal{S}^3 = \{0 \equiv (s^0), 1 \equiv (s^1), 2 \equiv (s^2)\}$
$\mathcal{S}^2 = \{0 \equiv (p^2), 1 \equiv (p^1), 2 \equiv (p^0)\}$
$\mathcal{S}^1 = \{0 \equiv (q^0, r^0), 1 \equiv (q^1, r^0), 2 \equiv (p^0, r^1),$
$3 \equiv (p^2, r^0), 4 \equiv (p^1, r^1), 5 \equiv (p^0, r^2)\}$

(b)

Fig. 1. An example of PN with a partition of its place (a) and the MDD associated to it in the case of $N = 2$ (b)

Under the above mentioned hypothesis each component of the state vector can be updated somewhat independently of the others. The system's state space may then be built by iterating the $\mathcal{N}_e$ functions in any order. Saturation, the iteration strategy proposed by Ciardo et al, exhaustively fires all events affecting a given MDD node bringing it to its final saturated shape. Moreover, nodes are considered in a bottom-up fashion, i.e., when a node is processed, all its descendants are already saturated. Such an iteration strategy improves both memory and execution-time efficiency.

The main drawback of the first version of Saturation proposed in [5] is the necessity to know a priory the local state spaces of the submodels in which the input model has been partitioned. In [6], Ciardo et al. proposed a new version of Saturation called *Unbound* that produces an MDD representation of the final state-space and a separately stored representation of the minimal local state spaces. The algorithm interleaves symbolic exploration of the global state space with explicit local explorations of each submodel. It incrementally discovers a set $\hat{S}^k$ of locally-reachable local states, of which only a subset $\mathcal{S}^k$ is also globally reachable. When a globally reachable state is identified, it is labelled as confirmed and an explicit local reachability analysis starts. During this phases the description of the model is consulted and an efficient encoding of the next-state function is build in the form of a set of Kronecker matrices. By defining matrices $\mathbf{W}_{e,k} \in \{0,1\}^{n^k \times n^k}$, where $\mathbf{W}_{e,k}[i_k, j_k] = 1 \iff j_k \in \mathcal{N}_{e,k}(i_k)$, in fact, the next-state function is encoded as the incidence matrix given by the Boolean sum of Kronecker products $\sum_{e \in \mathcal{E}} \bigotimes_{K \geq k \geq 1} \mathbf{W}_{e,k}$.

3 Computation of the Augmented $RG(S^{init})$

The Kronecker representation of a CPH-model introduced in section 2.1 can be considered as a possible method to face the state space explosion problem in the

case of non-Markovian models. However such an approach is still incomplete since it is yet necessary to explicitly store the system reachability graph $RG(S^{init})$, in the form of matrix $\mathbf{R}$, augmented with the information about the sets $T_e^{(s)}$ and $T_a^{(s)}$ for all the system states. Such an explicit representation can become unmanageably large to fit in a computer's memory especially in the case of asynchronous systems.

Answering the above mentioned problem represents the principal contribution of the present work. The possibility of encoding in a implicit way both the matrix $\mathbf{R}$ and the sets $T_a^{(s)}$, combined with the solution presented in [1], can be considered the final step toward a complete symbolic representation of the process underlying a CPH-model.

3.1 Encoding Sets $T_a^{(s)}$ through MTMDDs

MTMDDs [13] are an extension of MDDs. The main differences with respect to MDDs are that: 1) more than two terminal nodes are present in an MTMDD, and 2) such nodes can be labelled with arbitrary values, rather than just 0 and 1. In typical usage, these values are real. However, in our work, we consider a particular family of MTMDDs in which the terminal nodes are labelled with integer values so that an MTMDD represents functions in the form $f_M : \mathbb{Z}^K \rightarrow \mathbb{Z}$.

In such a way, we can use an MTMDD to efficiently store both the system state space S and the sets $T_a^{(s)}$ of active but not enabled events for all $s \in \mathcal{S}$ that are necessary in our approach for the evaluation of non-null blocks of matrix $\mathbf{Q}$. In fact, while an MDD is only able to encode the characteristic function of a state space, an MTMDD is also able to associate an integer to each state. Consequently by mean of an MTMDD it is not only possible to implicitly encode the characteristic function, considering that a state, whose path from the root to a terminal node different than 0 exists, belongs to the state space, but it is also possible to store the information about the set $T_a^{(s)}$ for that particular state. This can be done associating to each possible set $T_a^{(s)}$ an integer code that unequivocally represents it.

In this work, we exploit a very simple code. Let us suppose that the set $\mathcal{E}$ is ordered and let us call $\mathcal{E}_0$ the ordered set. Moreover let us associate to each event an unique index n such that $1 \leq n \leq \|\mathcal{E}_0\|$. Then the integer value associated to one of the possible sets $T_a^{(s)}$ can be computed starting from the indexes associated to the system events that belong to it in the following way:

$$b_{\|\mathcal{E}_0\|} \cdot 2^{\|\mathcal{E}_0\|} + \cdots + b_n \cdot 2^n + \ldots b_1 \cdot 2^1 + 1 = \sum_{i=1}^{\|\mathcal{E}_0\|} b_i 2^i + 1$$

where:

$$b_i = \begin{cases} 1, & \text{if event } e_i \in T_a^{(s)} \\ 0, & \text{otherwise} \end{cases}$$

Fig. 2 shows as an example the MTMDD associated to PN of Fig. 1(a) and the sets $T_a^{(s)}$ for each marking in the net in the case that all the transitions in the net present an age memory policy associated and that $\mathcal{E}_0 = \{a, b', b'', c, d\}$.

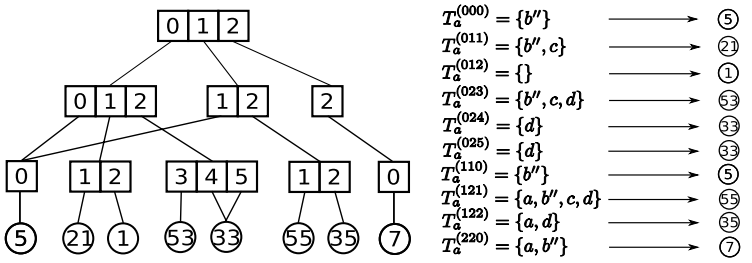


Fig. 2. The MTMDD associated to PN of Fig. 1(a) in the case of $N = 2$

3.2 Computation of Sets $T_a^{(s)}$

The combination of the MDD and the $K \cdot |\mathcal{E}|$ Kronecker matrices produced by Saturation Unbound is a very effective way to represent the reachability graph of the analyzed model. Since $RG(S^{init})$ is the data input to algorithmically generate the infinitesimal generator matrix $\mathbf{Q}$ of the expanded CTMC, we will use Saturation Unbound to generate it. Combined to the Kronecker representation of the expanded state space exposed in Section 2.1 the use of Saturation allows to further improve memory consumption when compared to the approach presented in [1].

However, as we already said, the knowledge of the reachability graph of the untimed system is not enough for the generation of the matrix $\mathbf{Q}$. Considering that the information about the enabled events for all the system states is contained in the high level description of the model and can be evaluated on the fly when needed with a negligible overhead, the only further needed information is the knowledge about the sets $T_a^{(s)}$. Using Saturation for the evaluation of the reachability graph bring in this case to the necessity of applying a further analysis step for the computation of such an information.

According to the definition given in Sec. 2, an event $e \in \mathcal{E}$ belongs to the set $T_a^{(s)}$ if it is not enabled in state s and if, for some path within the reachability graph, it is possible for its age memory variable to be greater than 0 in s. As a consequence of such behaviour, we can introduce the following property for an event to be active but not enabled in a system state:

Property 1. - Given a state $\overline{s} \in \mathcal{S}$ and an event $\overline{e} \in \mathcal{E}$ with an age memory policy associated, then $\overline{e} \in T_a^{(\overline{s})}$ iff one of the following statements holds:

1. $\exists\ s_1 \neq \overline{s} \in \mathcal{S}, \exists\ e_1 \neq \overline{e} \in \mathcal{E} \mid \overline{s} \in \mathcal{N}_{e_1}(s_1)\ \wedge\ \overline{e} \in T_e^{(s_1)}\ \wedge\ \overline{e} \notin T_e^{(\overline{s})}$,
2. $\exists\ s_1 \neq \overline{s} \in \mathcal{S} \mid \overline{s} \in \mathcal{N}(s_1)\ \wedge\ \overline{e} \in T_a^{(s_1)}\ \wedge\ \overline{e} \notin T_e^{(\overline{s})}$.

In fact, if event $\overline{e}$ is enabled in state s_1 and it is disabled by the firing of another conflicting event e_1, then it will retain its memory level in the reached state $\overline{s}$ (statement 1 of Property 1); moreover, if event $\overline{e}$ has memory in s_1 but it is not enabled, then it will retain its memory level in each state $\overline{s}$ reached

by the process, till it becomes enabled again and has the possibility of firing (statement 2 of Property 1).

Using Property 1, it is possible to compute the sets $T_a^{(s)}$ for all the system states by visiting the reachability graph along all the possible paths, and by storing the information as it is discovered in an explicit data structure. Such a computation can be performed even on-the-fly, during state space generation, if classical approaches are used because they add states visiting the reachability graph in a breadth-first-search (BFS) or in a depth-first-search (DFS) order. Since Saturation algorithm works in a highly localized manner, adding states out of the BFS or DPS order, the computation of all the sets $T_a^{(s)}$ by directly applying the Property 1 becomes inefficient or, in most of the cases, impossible.

In [7], Ciardo et al. applied Saturation to efficiently evaluate computation tree logic (CTL) formulae. In this subsection, we will first enunciate a theorem that shows how a CTL formula can be used to compute the sets $T_a^{(s)}$ through Property 1 and then we will define an algorithm that allows to take advantage of the efficiency of the Saturation algorithm to store such information in a MTMDD.

A CTL formula allows to evaluate whether a given condition holds on the states over a path within the reachability graph of a discrete state process. As extensively presented in [9], a CTL formula is written as a couple of operators: a path quantifier followed by a tense operator. The path quantifier is chosen between A (*all path*) and E (*there exist a path*), while the tense operator is one of the following: X (*next*), F (*future*, or *finally*), G (*globally* or *generally*), U (*until*). Here we are interested to the EU operator whose definition is as follows:

Definition 2. *Let $s_0 \in \mathcal{S}$ be a state of a discrete state process with state space $\mathcal{S}$, and let p and q be two logical conditions on the states. Then s_0 satisfies the formula $E[pUq]$, and we will write $s_0 \vDash E[pUq]$, iff $\exists\ n \geq 0, \exists\ s_1 \in \mathcal{N}(s_0), \ldots, \exists\ s_n \in \mathcal{N}(s_{n-1}) \mid (s_n \vDash q) \wedge (\forall m < n, s_m \vDash p)$.*

Let us suppose that, during the evaluation of a CTL formula, we also need to take into account constraints or special properties over the visited paths. Such a generalization can be obtained by using a generic next-state function $\mathcal{F}(s)$ different from $\mathcal{N}(s)$ to visit the states in the reachability graph and by ensuring that $\forall s \in \mathcal{S} | \mathcal{F}(s) \subseteq \mathcal{N}(s) \cup \mathcal{N}^{-1}(s)$. For example if we are interested to paths that are generated by the firing of event e we can consider as next-state function $\mathcal{N}_e(s)$. The Definition 2 can be extended to use a generic next-state function $\mathcal{F}(s)$ as follows.

Definition 3. *Let $s_0 \in \mathcal{S}$ be a state of a discrete state process with state space $\mathcal{S}$, and let p and q be two logical conditions on the states. Let also $\mathcal{F}(s) \subseteq \mathcal{N}(s) \cup \mathcal{N}^{-1}(s)$ be a reachability relationship between two states in $\mathcal{S}$ that defines a desired condition over the paths. Then s_0 satisfies the formula $E_{\mathcal{F}}[pUq]$, and we will write $s_0 \vDash E_{\mathcal{F}}[pUq]$, iff $\exists\ n \geq 0, \exists\ s_1 \in \mathcal{F}(s_0), \ldots, \exists\ s_n \in \mathcal{F}(s_{n-1}) \mid (s_n \vDash q) \wedge (\forall m < n, s_m \vDash p)$.*

Fig. 3(a) shows an example of a path of states satisfying the formula $E_{\mathcal{F}}[pUq]$: state s_0 satisfies the formula $E_{\mathcal{F}}[pUq]$ because there exists a path starting from

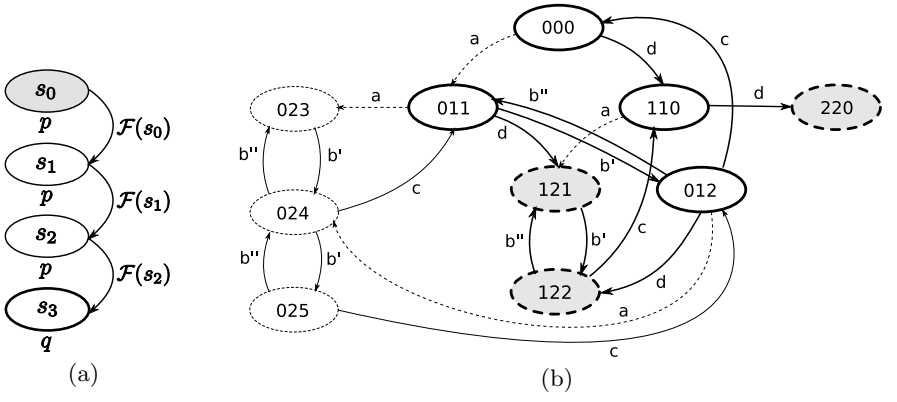


Fig. 3. A graphical example of (a) $s_0 \vDash E_{\mathcal{F}}[pUq]$ and (b) the algorithm of Tab.1

state s_0 to state s_3 where logical condition p holds on s_0, s_1, s_2, q holds on s_3, and each state in the path is reached according to the next-state function $\mathcal{F}(s)$.

Upon Definition 3, we build the following theorem which our algorithm is built on:

Theorem 1. *An event $\overline{e} \in \mathcal{E}$, with an age memory policy associated, belongs to $T_a^{(s_0)}$, with $s_0 \in \mathcal{S}$, iff $s_0 \vDash E_{\mathcal{F}}[pUq]$ over a path at least long one, where p and q are the statements "$\overline{e}$ is not enabled" and "$\overline{e}$ is enabled" respectively, and $\mathcal{F}(s) = \mathcal{N}^{-1}(s) \setminus \mathcal{N}_{\overline{e}}^{-1}(s)$.*

Theorem 1 states that it is possible to compute the states in which event $\overline{e}$ is active but not enabled simply evaluating CTL formula $E_{\mathcal{F}}[pUq]$ over the system state space considering condition p as "$\overline{e}$ *is not enabled*" and condition q as "$\overline{e}$ *is enabled*". Moreover the considered next-state function $\mathcal{F}(s)$ is the inverse of the system next-state function $\mathcal{N}(s)$ but dropping the paths due to the event $\overline{e}$. Due to lack of space the proof of the theorem is not given.

The traditional computation of the set of states satisfying $E[pUq]$ uses a least fixed point algorithm that starting with the set $\mathcal{Q}$ of states satisfying q, iteratively adds all the states that reach them on paths where property p holds. The set of states in which property p holds is called $\mathcal{P}$. In [7] Ciardo et al. proposed a new approach to compute the EU operator based on Saturation called *EUsat*. Such approach starts with the classification of the system events between two classes: *safe* and *unsafe*. Events are safe if their firing is guarantee to preserve the validity of the condition p. They are unsafe otherwise. In the case of unsafe events Saturation can't be used due to the great overhead produced for dropping the states outside $\mathcal{P}$. For this reason, after the events classification, a global fixed point iteration starts that interleaves two backward steps: BFS with filtration on unsafe events followed by Saturation on safe events. Both steps make use of the backward next-state function $\mathcal{N}^{-1}(s)$ to compute states in $\mathcal{P}$ that reach states in $\mathcal{Q}$ with one or more event firings.

Let us apply the *EUsat* algorithm to the specific case described in Theorem 1. The following considerations can be done:

- it is straightforward that in our case $p \vee q = true$ and therefore $\mathcal{P} \bigcup \mathcal{Q} = \mathcal{S}$; as a consequence all our events are safe and the *EUsat* algorithm would perform just one Saturation step and stop (Note 6 in [7]);
- given that the next-state function we are considering is $\mathcal{F}(s) = \mathcal{N}^{-1}(s) \setminus \mathcal{N}_{\overline{e}}^{-1}(s)$ in our case the backward next-state function used by *EUsat* would became $\mathcal{F}^{-1}(s) = \mathcal{N}(s) \setminus \mathcal{N}_{\overline{e}}(s)$. So the Saturation step necessary for *EUsat* evaluation would became a standard forward Saturation with the exception that event $\overline{e}$ would have to be dropped.

We can assert that the set of all the system state in which event $\overline{e}$ is active but not enabled can be obtained applying a forward Saturation step to the set of states in which event $\overline{e}$ is enabled neglecting $\overline{e}$ itself. Fig. 3(b) shows an example applied to the PN shown in Fig. 1(a). States in double non-dashed line have transition a enabled and are used as starting point for the computation. States in double dashed line are obtained through a forward Saturation step performed without considering the firing of a as a possible event. Transition a is active but not enabled in them. States in normal dashed line belong to the state space but transition a is neither enabled nor active in them. They can't be obtained through Saturation because we ignore event a.

3.3 The Algorithm

From Theorem 1 and from the above consideration it is possible to derive the algorithm presented in Tab. 1 in the form of pseudo-code. The exploited data types are: *model* (model description by mean of an high-level formalism), *partitioning* (model partitioning), *state* (model state), *event* (model event), *mdd* (complete data structure implementing an MDD), *mtmdd* (complete data structure implementing an MTMDD).

GenerateMTMDD is the main function. It takes as input parameters the considered model m, its partitioning p, the set of initial system states S_i (we consider the general case in which there can be more than one initial state) and the set of events that present an age memory policy $\mathcal{E}_m$. We suppose that the information about all the system events is contained in the high-level description of the model and that it is sufficient to inform the function about the events that present an age memory policy. The function builds the MTMDD encoding the state space S and the sets $T_a^{(s)}$ for the considered model taking into consideration the particular partitioning provided.

First of all an MDD is build that encode the set of initial system states (function *InitializeMDD*). Then a forward step of Saturation Unbound is performed to build the entire system state space. The procedure *ExplorerSaturationUnbound* recursively explores the MDD encoding the set of initial system states saturating its node. The MDD is modified to encode the entire state space. Saturation Unbound is used because we suppose that no information about local state spaces is initially provided. The additional information gathered by

Table 1. Pseudocode for the algorithm

$GenerateMTMDD$(in $m : model,\ p : partitioning,\ S_i$: set of $state,\ \mathcal{E}_m$: set of $event$) : $mtmdd$
Build the MTMDD encoding the state space S and the sets $T_a^{(s)}$ for model m considering partitioning p and the set $\mathcal{E}_m$ of events with age memory policy associated.
1. declare $a, t : mdd$; 2. declare $u : mtmdd$; 3. $a \leftarrow InitializeMDD(m, p, S_i)$; 4. $ExplorerSaturationUnbound(p, a, m)$; 5. $u \leftarrow a$; 6. foreach $e \in \mathcal{E}_m$ do 7. $\quad t \leftarrow ExtractMDD(m, p, e, a)$; 8. $\quad ExplorerSaturationClassical(m, p, e, t)$; 9. $\quad Sum(t, e, u)$; 10. return u;

$InitializeMDD$(in $m : model,\ p : partitioning,\ S_i$: set of $state$) : mdd
Build the MDD encoding the set of initial states S_i for model m considering partitioning p.

$ExplorerSaturationUnbound$(in $p : partitioning$, inout $a : mdd,\ m : model$)
Recursively explore the MDD a saturating its node and generating the state space S for model m considering the partitioning p. Saturation Unbound is used. Information about the local state spaces and the next-state function are store in m.

$ExtractMDD$(in $m : model,\ p : partitioning,\ e : event,\ a : mdd$) : mdd
Extract from MDD a a new MDD encoding the set of states enabling event e for model m considering partitioning p.

$ExplorerSaturationClassical$(in $m : model,\ p : partitioning,\ e : event$, inout $a : mdd$)
Recursively explore the MDD a saturating its node and generating the set of states reachable from states initially encoded by a considering all the event in $\mathcal{E}_m$ except event e. The first version of Saturation is used.

Sum(in $t : mdd,\ e : event$, inout $u : mtmdd$)
For every state s encoded by MDD t if s doesn't enable event e then add 2^n to the value associated to s in MTMDD u where n is the integer associated to event e. If s enables event e then skip to the following state.

Saturation Unbound are stored in the model data structure ready to be used later during the algorithm. In particular sets of local states S^k and matrices $\mathbf{W}_{e,k}$ are generated and stored.

An MTMDD is then build to initially contain the same information stored in the MDD. Such an MTMDD will be returned at the end of the function. Subsequently for each event in $\mathcal{E}_m$ a sequence of operation is performed to obtain the information about the sets $T_a^{(s)}$. From the MDD encoding the entire state space S is extracted a temporary MDD encoding the set of states in which the considered event is enabled (function $ExtractMDD$). A forward step of Saturation is then performed to obtain the states in which the event is active (procedure $ExplorerSaturationClassical$). Such procedure can take advantage from the information about local state spaces and next-state function gathered during the procedure $ExplorerSaturationUnbound$ so the first version of Saturation can be exploited. The procedure Sum finally accumulates the new information in

the final MTMDD filtering states where the considered event is enabled. It adds s^n to the value associated to s whether the nth event e is not enabled in s, 0 otherwise. Finally the MTMDD is returned.

4 Numerical Results

The algorithm presented in section 3.2 has been implemented and experimental results have been gathered applying it to the PN of Fig. 1(a) and varying the initial number of tokens N in place p from 1 to 280. We consider the number of nodes in the data structures created by the algorithm as an index of the memory occupation. Obtained results are reported in Fig. 4 and Tab. 2. Fig. 4 shows the number of nodes within the MTMDD u, returned by function $GenerateMTMDD$, and the number of nodes within the MDD a, created by

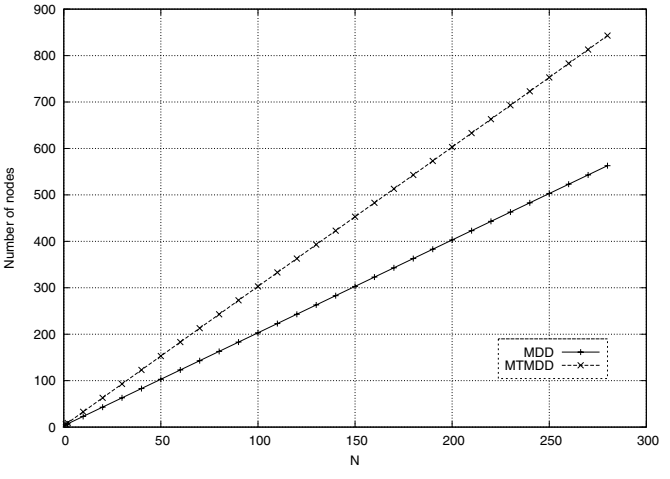


Fig. 4. Number of nodes for the MDD and the MTMDD with respect to N

Table 2. Experimental results for the PN of Fig. 1(a)

N	MDD nodes	MTMDD nodes	Markings	N	MDD nodes	MTMDD nodes	Markings
1	5	6	4	140	283	423	477191
2	7	9	10	150	303	453	585276
10	23	33	286	160	323	483	708561
20	43	63	1771	170	343	513	848046
30	63	93	5456	180	363	543	1004731
40	83	123	12341	190	383	573	1179616
50	103	153	23426	200	403	603	1373701
60	123	183	39711	210	423	633	1587986
70	143	213	62196	220	443	663	1823471
80	163	243	91881	230	463	693	2081156
90	183	273	129766	240	483	723	2362041
100	203	303	176851	250	503	753	2667126
110	223	333	234136	260	523	783	2997411
120	243	363	302621	270	543	813	3353896
130	263	393	383306	280	563	843	3737581

function *ExplorerSaturationUnbound*, versus N. Tab. 2 compares the number of nodes in the MDD and MTMDD with the number of markings in the net.

Observing Tab. 2 it is possible to state that the memory overhead determined by the additional information about sets $T_a^{(s)}$ is irrelevant. Such a consideration is particularly evident if we compare the difference between the number of nodes within the two data structures with the dimension of the system state space represented by the number of markings in the net. In the worst case the increment in terms of number of nodes is about 0.1 per millisecond. Moreover, as observed in [6] the number of MDD nodes linearly grows with N, and this important property is preserved from the MTMDD, even if a greater amount of information is stored into it (Fig. 4).

We don't take into consideration the intermediate memory occupation of the algorithm because the goal of the present work is to shows the validity of this new approach without taking into consideration any possible optimization, that can be performed in future works.

5 Conclusions and Future Work

The paper has presented a new symbolic technique for the generation and the storage of the reachability graph of asynchronous systems augmented with the information about active but not enabled events. Such an information is necessary for the symbolical description of the non-Markovian process underlying a system in which events are described by CPH distributed random variable.

The goal is achieved by the use of a Multi-terminal Multi-valued Decision Diagram and a set of Kronecker matrices as data structures and by applying a Saturation based algorithm for the collection of the needed information. Such an algorithm has been derived from a theorem inspired by works in symbolic model checking. The reported numerical results have shown the effectiveness of the proposed algorithm.

Much work remains to be done, however. We are studying the possibility to optimize our algorithm for what concerns both execution time and run-time memory occupation. In particular locality could be exploited in a much deeper way trying to perform an in-place update of the MDD nodes to build the MTMDD. Our goal is to avoid the necessity of the intermediate MDD for the subsequential application of the Saturation algorithm. An implementation of all the steps, from an high level model description to the measure computation, is under construction. A possible extension of this work is the application to the expansion method with the use of Discrete PH distributions. In such a contest the simultaneous firings of enabled events has to be explicitly addressed in the application of the algorithm.

References

1. Bobbio, A., Scarpa, M.: Kronecker representation of stochastic Petri nets with discrete PH distributions. In: Proc. Third IEEE Ann. Int'l Computer Performance and Dependability Symp. (IPDS 1998) (1998)

2. Bobbio, A., Tavella, P., Montefusco, A., Costamagna, S.: Monitoring the calibration status of a measuring instrument by a stochastic model. IEEE Transactions on Instrumentation and Measurement 46(4), 747–751 (1997)
3. Bobbio, A., Telek, M.: Combined preemption policies in MRSPN. In: Mittal, R., et al. (eds.) Fault Tolerant Systems and Software, pp. 92–98. Narosa Pub. House, New Dehli (1995)
4. Burch, J.R., Clarke, E.M., McMillan, K.L., Dill, D.L., Hwang, L.J.: Symbolic model checking: 10^{20} states and beyond. In: Fifth Annual IEEE Symposium on Logic in Computer Science, LICS 1990, June 1990, pp. 428–439 (1990)
5. Ciardo, G., Luttgen, G., Siminiceanu, R.: Saturation: an efficient iteration strategy for symbolic state space generation. In: Margaria, T., Yi, W. (eds.) TACAS 2001. LNCS, vol. 2031, pp. 328–342. Springer, Heidelberg (2001)
6. Ciardo, G., Marmorstein, R., Siminiceanu, R.: Saturation unbound. In: Garavel, H., Hatcliff, J. (eds.) TACAS 2003. LNCS, vol. 2619, pp. 379–393. Springer, Heidelberg (2003)
7. Ciardo, G., Siminiceanu, R.: Structural symbolic CTL model checking of asynchronous systems. In: Hunt Jr., W.A., Somenzi, F. (eds.) CAV 2003. LNCS, vol. 2725, pp. 40–53. Springer, Heidelberg (2003)
8. Cinlar, A.: Introduction to Stochastic Processes. Prentice-Hall, Englewood Cliffs (1975)
9. Clarke, E.M., Emerson, E.A.: Design and synthesis of synchronization skeletons using branching time temporal logic. In: Kozen, D. (ed.) Logic of Programs 1981. LNCS, vol. 131, pp. 52–71. Springer, Heidelberg (1982)
10. Cox, D.R.: The analysis of non-Markovian stochastic processes by the inclusion of supplementary variables. Proceedings Cambridge Philosophical Society 51(3), 433–441 (1955)
11. Kulkarni, V.G.: Modeling and Analysis of Stochastic Systems. Chapman and Hall, Boca Raton (1995)
12. Marsan, M.A., Balbo, G., Bobbio, A., Chiola, G., Conte, G., Cumani, A.: The effect of execution policies on the semantics and analysis of stochastic Petri nets. IEEE Trans. Softw. Eng. 15(7), 832–846 (1989)
13. Miner, A.S.: Symbolic representations and analysis of large probabilistic systems. In: Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.-P., Siegle, M. (eds.) Validation of Stochastic Systems. LNCS, vol. 2925, pp. 296–338. Springer, Heidelberg (2004)
14. Miner, A.S., Ciardo, G.: Efficient reachability set generation and storage using decision diagrams. In: Donatelli, S., Kleijn, J. (eds.) ICATPN 1999. LNCS, vol. 1639, pp. 6–25. Springer, Heidelberg (1999)
15. Neuts, M.: Probability distributions of phase type. In: Liber Amicorum Prof. Emeritus H. Florin, pp. 173–206. University of Louvain (1975)
16. Puliafito, A., Horvath, A., Scarpa, M., Telek, M.: Analysis and Evaluation of non-Markovian Stochastic Petri Nets. In: Haverkort, B.R., Bohnenkamp, H.C., Smith, C.U. (eds.) TOOLS 2000. LNCS, vol. 1786, pp. 171–187. Springer, Heidelberg (2000)
17. Srinivasan, A., Ham, T., Malik, S., Brayton, R.K.: Algorithms for discrete function manipulation. In: IEEE International Conference on Computer-Aided Design, ICCAD 1990. Digest of Technical Papers, November 1990, pp. 92–95 (1990)
18. Telek, M., Bobbio, A.: Markov regenerative SPN with non-overlapping activity cycles. In: International Computer Performance and Dependability Symposium, IPDS 1995, pp. 124–133. IEEE Computer Society Press, Los Alamitos (1995)

Mean Value Analysis for a Class of PEPA Models

Nigel Thomas and Yishi Zhao

School of Computing Science, Newcastle University, UK
{Nigel.Thomas,Yishi.Zhao}@ncl.ac.uk

Abstract. In this paper a class of closed queueing network is modelled in the Markovian process algebra PEPA and solved using the classical Mean Value Analysis (MVA). This approach is attractive as it negates the need to derive the entire state space, and so certain metrics from large models can be obtained with little computational effort. The class of model considered includes models which are not obviously classical closed queueing models. The approach is illustrated with three examples.

1 Introduction

There have been many attempts to find efficient solutions to large stochastic process algebra (SPA) models. SPA models suffer from the well known problem of state space explosion, where each additional component cause a multiplicative increase in the size of the global state space. This problem is particularly significant when there are many instances of the same type of component (so-called *massively parallel systems*). Such models may be extremely concise to specify, but even when the state space is folded or lumped, it may still far exceed the capacity available for solution.

Many of the approaches to efficiently solving SPA models have been based on concepts of decomposition originally derived for queueing networks [4]. Applying such approaches to stochastic process algebra allows the concepts to be understood in a more general modelling framework and applied to non-queueing models. Hillston [5] took an alternative, inspired by systems biology, approach by deriving a fluid approximation based on ordinary differential equations. Recently, Thomas [12] showed that such a fluid approximation is equivalent to a well known asymptotic solution for a class of closed queueing network (similar to the class of model considered in this paper). Traditionally this asymptotic solution was used as a computationally cheap alternative to mean value analysis [8] for very large populations. As such, it is clear that the class of model considered in [12] is also amenable to solution by mean value analysis. In this paper we make such an application and in doing so consider an extension to the class of model studied earlier [11,12].

Mean value analysis (MVA) is a method for deriving performance metrics based on steady state averages directly from the queueing network specification,

J.T. Bradley (Ed.): EPEW 2009, LNCS 5652, pp. 59–72, 2009.

without the need to derive any of the underlying Markov chain. As such it is relatively computationally efficient as long as the population size is not excessively large.

This paper is organised as follows. In the next section a brief overview of PEPA is given. The subsequent section then defines the class of model under consideration and gives the MVA solution of this class. Three examples are then used to illustrate the approach and to explore some numerical results. Finally some conclusions are drawn and some further work discussed.

2 PEPA

A formal presentation of PEPA is given in [3], in this section a brief informal summary is presented. PEPA, being a Markovian Process Algebra, only supports actions that occur with rates that are negative exponentially distributed. Specifications written in PEPA represent Markov processes and can be mapped to a continuous time Markov chain (CTMC). Systems are specified in PEPA in terms of *activities* and *components*. An activity (α, r) is described by the type of the activity, α, and the rate of the associated negative exponential distribution, r. This rate may be any positive real number, or given as unspecified using the symbol $\top$. It is important to note that in this paper the unspecified rate is not used.

The syntax for describing components is given as:

$$A \mid (\alpha, r).P \mid P + Q \mid P/L \mid P \underset{\mathcal{L}}{\bowtie} Q$$

$A \stackrel{def}{=} P$ gives the constant A the behaviour of the component P. The component $(\alpha, r).P$ performs the activity of type α at rate r and then behaves like P. The component $P + Q$ behaves either like P or like Q, the resultant behaviour being given by the first activity to complete.

Concurrent components can be synchronised, $P \underset{\mathcal{L}}{\bowtie} Q$, such that activities in the cooperation set $\mathcal{L}$ involve the participation of both components. In PEPA the shared activity occurs at the slowest of the rates of the participants and if a rate is unspecified in a component, the component is passive with respect to activities of that type. The shorthand notation $P||Q$ is used to mean $P \underset{\emptyset}{\bowtie} Q$ and $\prod_{i=1}^{N} P_i$ is used to mean the parallel composition of the components P_i where i takes the values 1 through to N, i.e. $P_1||\ldots||P_N$. In addition, we employ a further shorthand for synchronisation over many identical components, first introduced in [5], whereby $P[N]$ is taken to mean N copies of the component P, synchronised on the empty set, i.e. $P||\ldots||P$ where there are N instances of component P.

The component P/L behaves exactly like P except that the activities in the set L are concealed, their type is not visible and instead appears as the unknown type τ. In this paper we do not make use of hiding, although non-shared actions could be hidden in a model if that was desirable. The action set $\mathcal{A}(P)$ is defined as the set of sections which are currently enabled in the derivative P.

In this paper we consider only models which are cyclic, that is, every derivative of components P and Q are reachable in the model description $P \underset{\mathcal{L}}{\bowtie} Q$. Necessary conditions for a cyclic model may be defined on the component and model definitions without recourse to the entire state space of the model.

3 A Class of Closed Queueing Networks in PEPA

Now consider a model of a closed queueing network of N jobs circulating around a network of M service stations, denoted $1, \ldots, M$; each station is either a queueing station or an infinite server station. There are M_q queueing stations. Let $\mathcal{M}$ be the set of all queueing stations. At each queueing station, i, there is an associated queue (bounded at N) operating a FCFS policy and K_i servers. The servers are able to serve jobs of only one type; each job type, j, is served at rate r_j. At each infinite server station, i, jobs of type i experience a random delay with mean $1/r_i$. All services are negative exponentially distributed.

There are J job types. Each job type can be served at most one station. When a job of type j completes a service of at a given station, it will proceed to service at a station (possibly the same station) as a job of type k according to some routing probability p_{jk}.

In PEPA a queue station can be modelled as

$$QStation_i \stackrel{def}{=} (service_i, r_i).QStation_i \ , \ \forall i \in \mathcal{M}$$

Note that r_i is always specified as finite, and not $\top$. This is because passive actions are subject to the *apparent rate* in PEPA. The infinite server stations are not represented explicitly.

Each job will receive service from a sequence of stations determined by a set of routing probabilities,

$$Job_i \stackrel{def}{=} \sum_{k=1}^{J} (service_j, p_{jk} r_j).Job_k \ , \ 1 \leq i, j \leq J$$

Where, $0 \leq p_{jk} \leq 1$ and

$$\sum_{k=1}^{J} p_{jk} = 1 \ , \ 1 \leq j \leq J$$

Denote $\mathcal{S}_i$ to be the set of all job types which perform $service_i$ actions, i.e. $\mathcal{S}_i = j$ if $service_i \in \mathcal{A}(Job_j)$.

The entire system can then be represented as follows:

$$\left(\prod_{\forall i \in \mathcal{M}} (QStation_i[K_i]) \right) \underset{\mathcal{L}}{\bowtie} Job_1[N] \tag{1}$$

Where

$$\mathcal{L} = \bigcup_{\forall i \in \mathcal{M}} \{service_i\}$$

3.1 Mean Value Analysis

We now consider the *arrival theorem*, first derived independently by Sevcik and Mitrani [9] and Lavenberg and Reiser [7], applied to this class of PEPA model.

Theorem 1 ***Arrival Theorem.*** *Consider a component Job_i evolving into its successor derivative, Job_j in a system given by (1). The steady state distribution of the number of components behaving as any component Job_k at that moment is equal to the steady state distribution of the number of components behaving as Job_k in a system without the evolving job.*

The arrival theorem is as profound as it is simple and seemingly intuitive. It consequently gives rise to the well known *mean value analysis*, whereby the average behaviour of a system of N components may be derived from the average behaviour of a system of $N-1$ components. Therefore it is never necessary to derive a solution to the full CTMC if we are only concerned with the average behaviour of systems of this kind. This follows from the following set of relationships, derived following the pattern of Haverkort [2] pp. 241-245.

Theorem 1 implies that the average time a component spends in behaviour Job_j, denoted $W_j(N)$, where $\mathcal{A}(Job_j) = service_i$ and $i \in \mathcal{M}$, is given by the average number of Job_k ($\forall k \in \mathcal{S}_i$) components in a system with one fewer Job_l, $\forall i$, components in total. Denote $L_j(N)$ to be the steady state average number of components behaving as Job_j in a system with N jobs in total. If $\sum_{\forall i \in \mathcal{S}_j} L_i(N-1) \leq K_j - 1$ and $j \in \mathcal{M}$ then

$$W_i = \frac{1}{r_j}\ ,\ \forall i \in \mathcal{S}_j \tag{2}$$

Otherwise, if $\sum_{\forall i \in \mathcal{S}_j} L_i(N-1) > K_j - 1$ and $j \in \mathcal{M}$ then

$$W_i = \frac{1 + \sum_{\forall i \in \mathcal{S}_j} L_i(N-1)}{K_j r_i} \tag{3}$$

Clearly, if $j \notin \bigcup_{\forall i \in \mathcal{M}} \mathcal{S}_\rangle$, then $W_j(N)$ is a constant, given as $W_j(N) = 1/r_j$.

We now need to compute a quantity generally referred to as the *visit count*, and denoted V_i. The visit count is the number of times derivative Job_i is visited, relative to the number of times some reference derivative Job_I is visited, where $1 \leq I \leq J$. The actual value of V_i is not crucial, rather its value relative to the value of V_I. As such the choice of I is strictly arbitrary.

We can compute the visit count from the routing probabilities p_{ij}. Define the probability that a component will evolve from Job_i to Job_j, without revisiting Job_i, as follows:

$$P_{ij}(\sigma) = p_{ij} + \sum_{\forall k \notin \sigma} p_{ik} P_{kj}(\sigma)$$

The set σ here contains only the starting and ending behaviours of interest, in this case i and j, i.e. it is used to tell us if we reach Job_j or first return to Job_i. For convenience define the shorthand,

$$P_{ij} = P_{ij}(\{i, j\})$$

By definition, $P_{ii} = 1$. Clearly the system is irreducible if

$$P_{ij} > 0 \; \forall i, j \; , \; i \neq j$$

Now we choose some reference point I, such that,

$$V_i = \frac{P_{Ii}}{P_{iI}} \; , \; \forall i \neq I$$

and $V_I = 1$. Thus, V_i gives the number of times a component assumes the behaviour Job_i, relative to the number times it assumes the behaviour Job_I.

Given the quantity V_j, we can now compute the average response time per passage for a component behaving as Job_j.

$$\hat{W}_j(N) = V_j W_j(N) \tag{4}$$

From Little's theorem we know that

$$L_j(N) = X_j(N)W_j(N) = X(N)V_j W_j(N) = X(N)\hat{W}_j(N) \tag{5}$$

Where $X_j(N)$ is the observed rate of activity $service_j$ when the population size is N, and $X(N)$ is the sum of all possible $X_j(N)'s$.

Summing (5) over all behaviours Job_i, $i = 1, 2, \ldots, J$ gives,

$$\sum_{j=1}^{J} L_j(N) = X(N) \sum_{j=1}^{J} \hat{W}_j(N) = X(N)\hat{W}(N) = N$$

where $\hat{W}(N) = \sum_{j=1}^{J} \hat{W}_j(N)$. Thus,

$$X(N) = \frac{N}{\hat{W}(N)}$$

Hence, with Little's law applied for a given behaviour Job_j,

$$L_j(N) = X_j(N)W_j(N) = X(N)V_j W_j(N) = \frac{N}{\hat{W}(N)}\hat{W}_j(N) \tag{6}$$

We are now in a position to calculate $L_j(N)$ for any value of N if we can calculate $L_j(1)$. A solitary Job_i component will never compete for cooperation over the actions in $\mathcal{L}$, and so will experience a delay of $1/r_i$ in each derivative Job_i. Hence, the average number of components behaving as Job_j when $N = 1$, $L_j(1)$ is given by the proportion of time a component spends in that behaviour.

$$L_j(1) = \frac{V_j}{r_j \sum_{i=1}^{J} \frac{V_i}{r_i}} \tag{7}$$

We now apply the following iterative solution.

1. Calculate $L_j(1)$ for $j = 1, 2, \ldots J$, using (7).
2. $n = 2$
3. Compute $\hat{W}_j(n)$ for $j = 1, 2, \ldots J$, using (2), (3) and (4) and $L_j(n-1)$ from 1 above.
4. Compute $\hat{W}(n) = \sum_{j=1}^{J} \hat{W}_j(n)$.
5. Compute $L_j(n)$ for $j = 1, 2, \ldots J$, using (6) and $\hat{W}(n)$ from 4 above.
6. Increment n.
7. If $n \leq N$ return to step 3 else end.

Clearly this solution is not complicated to implement. For a system of J job types and N jobs it is necessary to compute $(2J+1)N$ distinct quantities. Hence, this will generally only be costly when N is extremely large.

4 Examples

In this section we explore the class of models introduced above, through three example PEPA models. Each example depicts a different aspect of this class. The first model is an abstract queueing model, with probabilistic branching on completion of service at one of the stations. The other two examples are practical models drawn from an ongoing area of study into performance modelling of security protocols. The first of these is a model of the classic Needham–Schroeder key distribution protocol. This model has no branching and so mean value analysis is applied easily. The second practical model is of a non-repudiation protocol. This model involves a single queueing station processing separate requests from two participants in an exchange.

4.1 Example 1: A Three Node Closed Queueing Network

Consider the following PEPA specification of a simple closed queueing network

$$
\begin{aligned}
Node1 &\stackrel{def}{=} (service_1, \xi).Node_1 \\
Node_2 &\stackrel{def}{=} (service_2, \mu).Node_2 \\
Node_3 &\stackrel{def}{=} (service_3, \eta).Node_3
\end{aligned}
$$

$$
\begin{aligned}
Request_1 &\stackrel{def}{=} (service_1, \xi).Request_2 \\
Request_2 &\stackrel{def}{=} (service_2, (1-p)\mu).Request_1 + (service_2, p\mu).Request_3 \\
Request_3 &\stackrel{def}{=} (service_3, \eta).Request_1
\end{aligned}
$$

The entire system is then specified as

$$
(Node_1[K_1] || Node_2[K_2] || Node_3[K_3]) \underset{\left\{ \substack{service_1, service_2 \\ service_3} \right\}}{\bowtie} Request_1[N]
$$

This system depicts a three node closed queueing network where all three nodes are queueing stations. After completing service at node 1, all requests

proceed to node 2. Following service at node 2, a proportion of requests, p, will return to node 1, whilst the remainder will be directed to node 3. All requests completing service at node 3 will return to node 1.

In this example it is a simple matter to compute the visit count for each node.

$$V_1 = 1$$
$$V_2 = 1$$
$$V_3 = p$$

There are clearly many possible approaches to implementing the iterative solution given above. For convenience this model has been solved in an Excel spreadsheet. Solutions with population sizes of over 10000 have been derived without any problems, although clearly a more efficient implementation is desirable for larger N when a range of parameter values are being considered.

Figure 1 shows the average queue size at node 3 varied with population size N for various values of p.

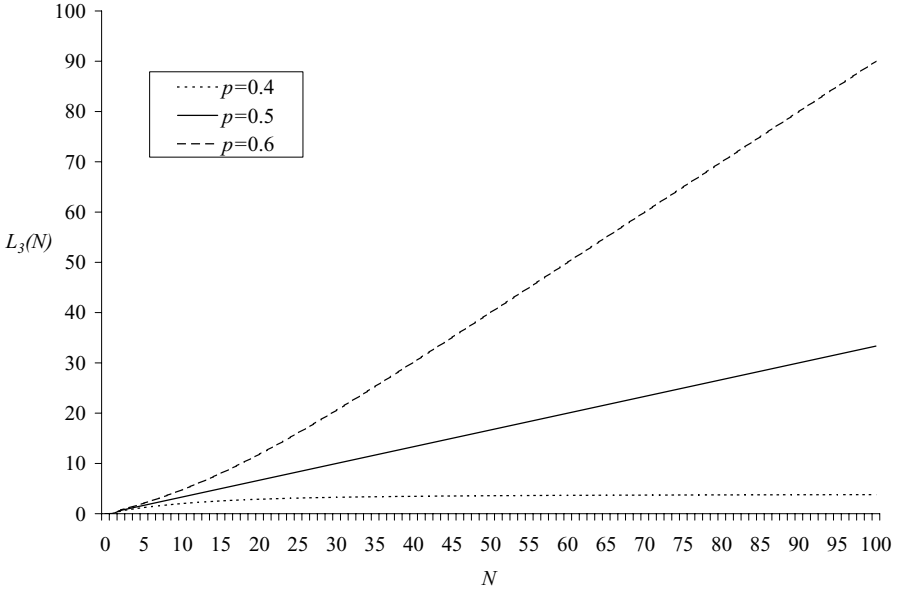


Fig. 1. Average queue length at node 3 varied with population size ($\xi = \mu = 10, \eta = 5$)

When $p = 0.5$ all three nodes have the same load, hence their queue sizes will be equal, i.e. $L_i = N/3$. Obviously, if p is less than 0.5 then node 3 will have a lower load than the other two nodes, hence it will have a smaller average queue length. In fact, the average queue length at node 3 will tend to a fixed value ($L_3(N) \to 4$ as $N \to \infty$ when $p = 0.4$). Conversely, if $p > 0.5$ then the third node will become the bottleneck of the system, and the majority of jobs will be queueing there. In the case $p = 0.6$, the average queue length at node 3 will tend to $N - 10$.

4.2 Example 2: A Secure Key Distribution Centre

Consider a model of the classic Needham-Schroeder key distribution protocol (taken from [14]) specified as follows:

$$KDC \stackrel{def}{=} (response, r_p).KDC$$

$$\begin{aligned}
Alice_0 &\stackrel{def}{=} (request, r_q).Alice_1 \\
Alice_1 &\stackrel{def}{=} (response, r_p).Alice_2 \\
Alice_2 &\stackrel{def}{=} (sendBob, r_B).Alice_3 \\
Alice_3 &\stackrel{def}{=} (sendAlice, r_A).Alice_4 \\
Alice_4 &\stackrel{def}{=} (confirm, r_c).Alice_5 \\
Alice_5 &\stackrel{def}{=} (usekey, r_u).Alice_0
\end{aligned}$$

The system is then defined as:

$$KDC[K] \underset{\{response\}}{\bowtie} Alice_0[N]$$

Where, K is the number of *KDC*'s and N is the number of client pairs (*Alices*'s).

In this model the component $Alice_i$ represents the actions of a pair of clients (normally referred to as *Alice* and *Bob*). The sequence of actions includes *Alice* requesting a session key from a secure server, known as the *key distribution centre* (KDC). This results in competition for the resources of the KDC amongst the various client pairs. Once the key has been issued to *Alice*, *Alice* and *Bob* exchange messages to confirm their mutual trust (established by shared trust of the KDC), before using the provided session key.

Clearly there is no branching, and so $V_i = 1$, $\forall i$. Furthermore there is only one queueing station, so this is always the bottleneck of the system unless K is large relative to N.

Figure 2 shows the average response time at the *KDC*, W_{KDC} for this system when there is one server for various service rates. Clearly, when the service rate is smaller, the response time is larger and its rate of increase is larger.

Figure 3 shows the average queue length at the *KDC*, L_{KDC} for this system when there is either one fast server or K slower servers. When the population size is large ($N > 30$ in this case) the *KDC* becomes saturated and there is consequently no difference in the service rate offered between the two cases shown. However, when N is smaller, there will be periods where one or more of the K servers will be idle, thus reducing the overall service capacity offered. Hence, for smaller N, a single fast server will outperform multiple slower servers with the same overall capacity, as is well known from queueing theory.

4.3 Example 3: A Non-repudiation Protocol

Non-repudiation protocols are used to prevent participants in a communication from later falsely denying that they took part in that communication. There

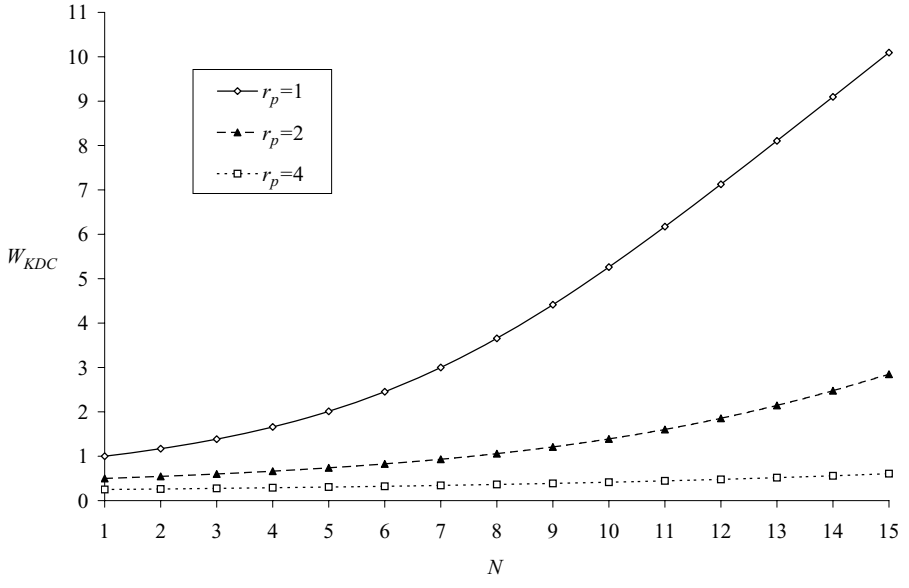


Fig. 2. Average response time at the *KDC* varied with population size ($r_q = r_B = r_A = r_c = 1$, $r_u = 1.1$, $K = 1$)

are many such protocols, with different properties. The one depicted here, first proposed by Zhou and Gollmann [15], utilizes a secure server, known as a *Trusted Third Party*, or TTP. Consider the following PEPA specification.

$$
\begin{aligned}
TTP &\stackrel{def}{=} (publish, rp).TTP \\
AB_0 &\stackrel{def}{=} (request, rq).AB_1 \\
AB_1 &\stackrel{def}{=} (publish, rp).AB_2 \\
AB_2 &\stackrel{def}{=} (getByA_1, rga_1).AB_3 \\
AB_3 &\stackrel{def}{=} (sendB, rb).AB_4 \\
AB_4 &\stackrel{def}{=} (sendTTP, rttp).AB_5 \\
AB_5 &\stackrel{def}{=} (publish, rp).AB_6 \\
AB_6 &\stackrel{def}{=} (get, rgb).AB_7 + (get, rga_2).AB_8 \\
AB_7 &\stackrel{def}{=} (getByA_2, rga_2).AB_9 \\
AB_8 &\stackrel{def}{=} (getByB, rgb).AB_9 \\
AB_9 &\stackrel{def}{=} (work, rw).AB_0 \\
System &= TTP \underset{\{publish\}}{\bowtie} AB_0[N]
\end{aligned}
$$

The model depicts a single TTP with N client pairs (*Alice* and *Bob*, denoted AB). The protocol utilises publishing in a public space (e.g. a bulletin board) by the TTP, from where the clients each download. In the specification

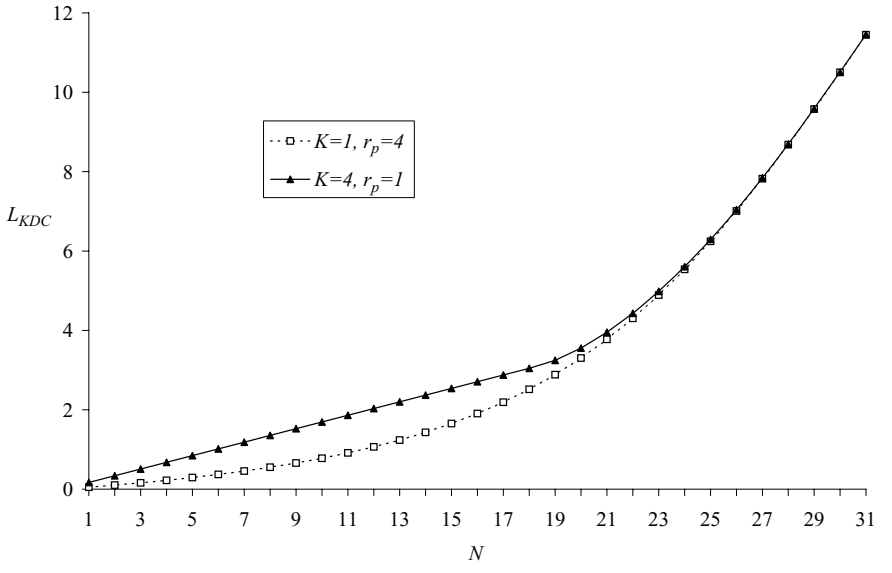


Fig. 3. Average queue length at *KDC* varied with population size ($r_q = r_B = r_A = r_c = 1$, $r_u = 1.1$)

above we are only concerned with contention on the publication by the TTP, although it would also be possible to specify an additional component to act as a web server.

In this model there is branching through a race on the *get* action in AB_6, and on service at the TTP (depending on which job type is served). Thus, the average number of components behaving as AB_1 when there are N client pairs, $L_1(N)$, depends the number of AB_1 and AB_5 when there are $N-1$ client pairs, $L_1(N-1)$ and $L_5(N-1)$. Likewise, the average number of components behaving as AB_6. The branching in AB_6 is specified over two *get* actions with different rates. These actions depict a race condition on *Alice* and *Bob* independently downloading from the bulletin board. In theory these could be given different names (they should ideally be called $getByB$ and $getByA_2$ respectively) but that is not possible in the current characterisation of the class of model given in Section 3.

The visit count is identical for each behaviour AB_i, $V_i = 1$, except V_7 and V_8, given by,

$$V_7 = \frac{rgb}{rgb + rga_2}$$
$$V_8 = \frac{rga_2}{rgb + rga_2}$$

Note that the visit count for behaviours AB_1 and AB_5, V_1 and V_5, are 1. Since *publish* actions from both AB_1 and AB_5 are served by the TTP, the

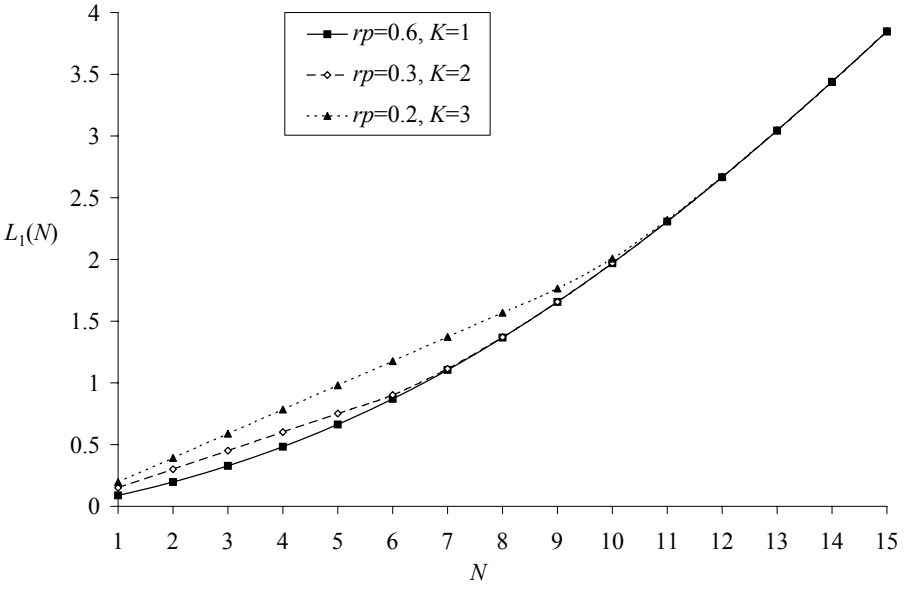


Fig. 4. Average number of AB_1 components varied with population size ($rq = rga_1 = rb = rttp = rga_2 = rgb = 1$, $rw = 0.01$)

'apparent visit count' of the TTP is effectively double that of the other stations, although we are not concerned with this quantity in our derivation of mean value analysis.

Figure 4 shows the average number of AB_1 components awaiting service at the TTP, for different values of service (publishing) rate, rp, varied with population size. Two parameter sets are used, identical except for the values of rp, although in each case the average service time is the same. Clearly, the values of $L_1(N)$ and $L_5(N)$ are the same, and so only $L_1(N)$ is shown. Clearly, the more jobs that are waiting in the queue, the longer an arriving job will have to wait. Thus, we find that the queue becomes longer at the TTP and that *proportionally* less time is spent performing the other actions, and so the throughput decreases. The figure shows a comparison between a single server and multiple servers with the same total service capacity. When the queue size is small then not all servers will be utilised; the more servers there are the more likely they are to be idle. Hence, there is linear growth of the queue when $K = 3$ and $N < 10$. In contrast, initially queue grows less quickly when there is only one (faster) server. However, once the population grows sufficiently for all servers to be highly utilised, then the three cases will show identical performance.

This situation is more clearly illustrated when looking at the response time for AB_1 components in Figure 5. When $K > 1$ the response time is static

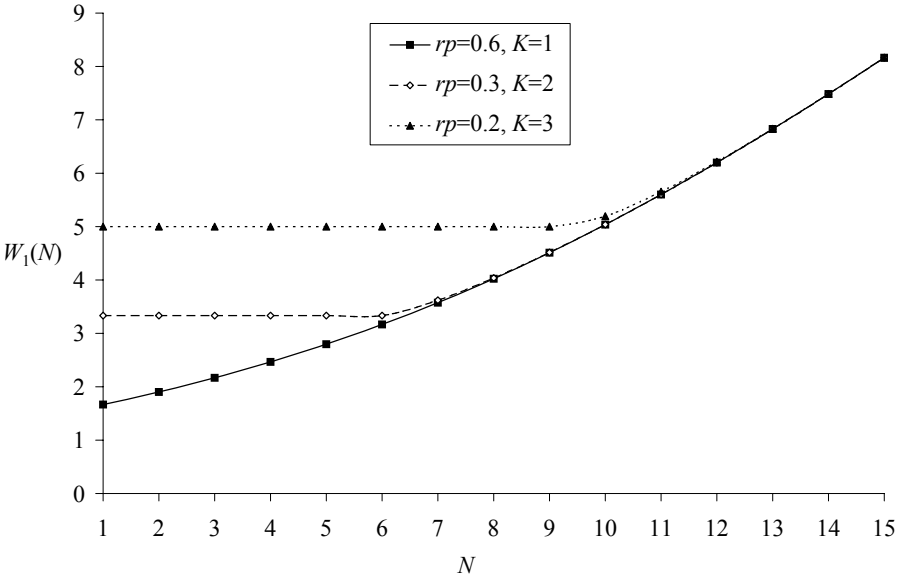


Fig. 5. Average response time for AB_1 components varied with population size ($rq = rga_1 = rb = rttp = rga_2 = rgb = 1$, $rw = 0.01$)

with N when the population is small, but once the population is large enough, the performance is once again shown to be the same in all cases.

5 Conclusions and Further Work

This paper demonstrates the solution of a class of PEPA models using classical Mean Value Analysis [8]. This gives a relatively computationally cheap method for solving large models without need to derive the state space of the underlying Markov chain. The CTMC for this class of model, even when lumped using standard techniques, is still large and generally grows at an exponential rate. Thus, when the number of instances of the Job_i components is extremely large, a CTMC solution is not going to be a feasible proposition. In contrast we have derived solutions to the example models here with over 10000 Job_i components, using a very simple (and not very efficient) spreadsheet based implementation on a relatively old laptop PC, in a fraction of a second. In the case of example 1 (Section 4.1) the lumped CTMC would have in excess of 50 million states when there are 10000 $Request_i$ components.

Clearly, the approach is limited in both the metrics that can be derived and also the class of model that is considered. The former limitation is a feature of mean value analysis (hence the name). However, the class of model could be extended in a number of ways. Mean value analysis applies to multiple classes of jobs in closed queueing network. Therefore it should be straightforward to

define a class of model with different groups of components, each with potentially different action rates and routing probabilities. This would be a relatively simple extension of the current class, but would involve careful use of notation to distinguish classes in a meaningful way.

The final example considered in this paper, a non-repudiation protocol, highlights a situation where we may wish an action type to appear in more than one *job* component, and potentially different action types in the same *job* component. Such a limitation is largely cosmetic, as an equivalent model can be specified in the existing class, as is done here. However, it should be possible to incorporate this option with a further adjustment to the notation used in this paper.

Finally, it should be noted that there can only be one service action type at a station and that must be given the same rate in any job type where it is enabled. Although intuitively it is possible to model the case where there are more job types enabled at a queueing station, doing so potential introduces race conditions and therefore distorts the effective service rate. We are still considering how such a situation might be best specified and it may be more feasible to consider approximate solutions in this scenario.

References

1. Clark, G., Gilmore, S., Hillston, J., Thomas, N.: Experiences with the PEPA Performance Modelling Tools. IEE Proceedings–Software 146(1), 11–19 (1999)
2. Haverkort, B.: Performance of Computer Communication Systems: A Model-based Approach. Wiley, Chichester (1998)
3. Hillston, J.: A Compositional Approach to Performance Modelling. Cambridge University Press, Cambridge (1996)
4. Hillston, J.: Exploiting Structure in Solution: Decomposing Compositional Models. In: Brinksma, E., Hermanns, H., Katoen, J.-P. (eds.) EEF School 2000 and FMPA 2000. LNCS, vol. 2090, p. 278. Springer, Heidelberg (2001)
5. Hillston, J.: Fluid-flow approximation of PEPA models. In: Proceedings of QEST 2005, pp. 33–43. IEEE Computer Society Press, Los Alamitos (2005)
6. Mitrani, I.: Probabilistic Modelling. Cambridge University Press, Cambridge (1998)
7. Lavenberg, S., Reiser, M.: Stationary state space probabilities at arrival instants for closed queueing networks with multiple types of customers. Journal of Applied Probability 17(4), 1048–1061 (1980)
8. Reiser, M., Lavenberg, S.: Mean value analysis of closed multichain queueing networks. JACM 22(4), 313–322 (1980)
9. Sevcik, K., Mitrani, I.: The distribution of queueing network states at input and output instants. JACM 28(2), 358–371 (1981)
10. Stallings, W.: Cryptography and Network Security: Principles and Practice. Prentice-Hall, Englewood Cliffs (1999)
11. Thomas, N.: Mean value analysis for a class of PEPA models, Technical Report CS-TR-1128, School of Computing Science, Newcastle University (2008)
12. Thomas, N.: Using ODEs from PEPA models to derive asymptotic solutions for a class of closed queueing networks, Technical Report CS-TR-1129, School of Computing Science, Newcastle University (2008)

13. Thomas, N., Zhao, Y.: Fluid flow analysis of a model of a secure key distribution centre. In: Proceedings 24th Annual UK Performance Engineering Workshop, Imperial College London (2008)
14. Zhao, Y., Thomas, N.: Approximate solution of a PEPA model of a key distribution centre. In: Kounev, S., Gorton, I., Sachs, K. (eds.) SIPEW 2008. LNCS, vol. 5119, pp. 44–57. Springer, Heidelberg (2008)
15. Zhou, J., Gollmann, D.: An efficient non-repudiation protocol. In: Proceedings of the l0th Computer Security Foundations Workshop (CSFW 1997). IEEE Computer Society Press, Los Alamitos (1997)

Automatic Generation of Performance Analysis Results: Requirements and Demonstration

Connie U. Smith[1], Catalina M. Lladó[2], and Ramon Puigjaner[2]

[1] Performance Engineering Services, P.O. Box 2640, Santa Fe, New Mexico, 87504-2640 USA
www.spe-ed.com

[2] Universitat de les Illes Balears. Departament de Ciències Matemàtiques i Informàtica. Ctra de Valldemossa, Km. 7.6, 07071 Palma de Mallorca, Spain
cllado@uib.es, putxi@uib.es

Abstract. This paper presents a performance model interoperability framework that brings together performance model interchange formats and experiment specifications with the automatic generation of performance analysis results for presentation and publication. We define the Use Cases and requirements and survey output and results used in practice. We present the output specification, the issues in the output-to-results transformation, the results specification schema extension, and a prototype implementation. A proof of concept example demonstrates the framework.

Keywords: Performance modelling, Tool interoperability, Queueing networks, Results.

1 Introduction

The concept of performance model interoperability was first introduced in 1995 [6]. Methods and tools supporting model interchange formats have evolved rapidly since 2004 with the introduction of XML as a viable mechanism for supporting model interchange [4].

Performance model interchange formats (PMIF) provide a mechanism for automatically moving performance models among modelling tools. Use of the PMIF does not require tools to know about the capabilities of other tools, internal data formats, or even existence. It requires only that the importing and exporting tools either support the PMIF or provide an interface that reads/writes model specifications from/to a file. Interchange formats have also been defined for layered queueing networks (LQN), UML, Petri Nets and other types of models.

Another interchange format, the Experiment Schema Extension (Ex-SE), allows the user to define a set of model runs that vary parameters. The Ex-SE provides a means of specifying performance studies and the output desired from them that is independent of a given tool paradigm. Each tool generates the performance metric output, such as response time, utilization, etc., specified for each experiment. A performance analyst typically studies this output to form

J.T. Bradley (Ed.): EPEW 2009, LNCS 5652, pp. 73–78, 2009.

conclusions about the results of the experiments, then prepares a presentation and/or report to explain the results.

Other work (see [5] for its description) has recognized the need for this last step. Our work develops the concept and provides a concrete realization of it. This paper streamlines the last step by defining a Results Schema Extension (Results-SE) that enables a user-customized transformation from the output of an experiment into the desired results.

The contributions of this work are:

- A definition of the most frequent Use Cases for this model interoperability framework
- A review of the typical types of output and results produced for queueing network based performance models for these Use Cases
- Definition of a modelling-paradigm independent schema for specifying the output of experiments and the transformation to results
- Implementation of a prototype demonstrating the feasibility of the approach
- Demonstration that the approach works.

2 Requirements for Producing Results

First we discuss the typical situations, or *Use Cases,* for conducting modelling experiments and analyzing results. Next we identify typical output and results that are needed for those Use Cases. Then we present our approach to providing the output and results.

QNM may be used in a variety of fields from computer performance evaluation to any other field that is interested in the behaviour of queues and servers. This paper addresses computer performance evaluation; other applications of QNM may require extensions to the analysis and results.

The most common reasons that performance analysts build and analyze QNM models are to:

1. Monitor and report on operational system performance
2. Analyze capacity requirements for future workload volumes
3. Evaluate problematic systems, identify causes and study options
4. Compare model results to measurements
5. Conduct technical investigations to compare results from: multiple tools, different solution algorithms, or even different types of solutions.

The next step is to determine the output metrics and results that are most often desired for these Use Cases.

We examined a sample of papers from the Computer Measurement Group 25th anniversary edition of the proceedings (1974 through 1999) [2] - the main source of practitioner modelling papers. Research results in other publications are similar.

We found three types of results: *tables*, graphs or *charts* in spreadsheet tools, and metric values embedded in the text of the paper. Some combinations of performance metrics occur frequently; examples are: service times and response times for several workloads; and throughput, response time and CPU utilization for several workloads.

Our conclusion is that the primary results are tables and charts. Charts are derived from tables, so they can be combined into one "result." Since there are many common combinations of both tables and charts, the specifications for those should be streamlined.

The most common format for tables and charts is xls [1] as in spreadsheet tools such as Excel and OpenOffice, and imported by most presentation and word processing packages. However, the most common document preparation system for research publications is LaTeX. Our approach transforms the output metrics to tables and charts in xls and LaTeX.

Additionally, we support two transformation modes: create a new table/chart and update an existing one. The update mode is convenient because it is unlikely that final results will be produced with one pass. It is also convenient when tables involve output from multiple tools. More importantly, it is easier to define table and chart formats by typing column and row headings or chart specifications directly into the spreadsheet rather than specifying transformation commands to create them.

This work does not address the metric values that are embedded in text. They have no tedious formatting requirements, and they might be best suited to the performance tree question/answer approach [7].

3 Model Transformation Approach

The Output Schema Extension is in Fig. 1. The "ValueUsed" applies to *Ranges* or other variables used in the experiment specification and reports the value used for that particular solution.

The metrics that may be produced are in the OutputWorkload (overall results by workload), OutputNode (overall results by Node), and Output NodeWorkload (results by Workload for Nodes). For more information see www.spe-ed.com/pmif/

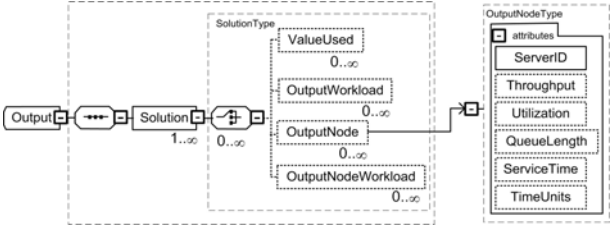


Fig. 1. Output Schema Extension

The output desired is specified in the Experiment specification. For each solution, the user may specify: WriteVariable, WriteOutput, or a ToolCommand that is passed to the tool unchanged. This allows users to print custom reports, visualization output, etc. particular to the tool, see [5].

The next step is to provide for an automatic conversion of the output into the table and chart results. We considered 2 options related to how those tables and charts would be expressed:

1. To use a "standard" xsd schema for spreadsheets for the results specification and transform the output into the xml format that follows such a schema.
2. To develop a transformation specification from output into the standard elements of a spreadsheet: rows, columns, and charts and transform the output into xls or LaTeX format.

Some spreadsheet tools, such as OpenOffice, do not yet support xml import and export, and the "standard" schema does not include chart specifications. Option 1 would require an additional schema to specify the transformation. Thus we chose the second option because it provides a specification of tables and charts using familiar notation, e.g., numeric rows and alphabetic columns. Java tools support the creation of a spreadsheet in xls format.

Fig. 2 shows the Results-SE schema. The Output-SE has a collection of outputs for each OutputSolutionSpec (or Solve) in the Experiment-SE. So the Results-SE specifies how to process each of those, and specifies the file/s containing the output. It can specify one or more tables (in xls tables go into different worksheets). WriteResult specifies the type of output metric to use (Node, Workload, etc.), the metric (such as response time), and where to place the values in the table (row, column, etc). There is a placeholder for Chart specifications to be added in future work.

The Transform prototype has been implemented in Java, using the Document Object Model (DOM) to read and validate the xml files (Output and ResultsSpec). We have also used the Apache POI APIs for manipulating MS Excel and OpenOffice file formats using pure Java.

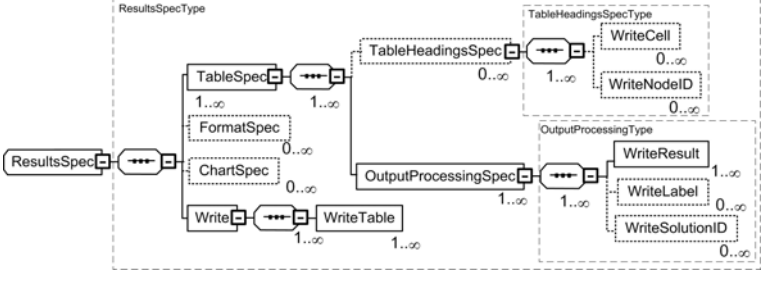


Fig. 2. Results-SE schema

4 Proof of Concept

The proof of concept uses a case study previously published and replicates it with the model interoperabilty experimental framework. It is a technical paper (Use Case) that compares a published solution to solutions derived automatically from experiment specifications.

The example was published in Jain's book [3] and subsequently used as an example of the experimental framework in [5]. It shows how to manually create a table, specify formats, enter the results from another source, then update the remaining values with the output from the experiment. The demonstration seeks to replicate the table in [5].

We run Qnap to produce the Output file of performance metrics specified in the experiment in [5]. We manually create an xls file with the formatting and Jain results taken from [3]. We then update the file using the results specification (an excerpt is in Fig. 3) to produce the table in Fig. 4.

```
<OutputProcessingSpec RowIncrement="3"FileToProcess="Jain574.xml">
  <WriteSolutionID Format="5" Row="3" Col="1" />
  <WriteLabel Value="Qnap" Row="5" Col="1" />
  <WriteResult Type="OutputWorkload" Metric="ResponseTime" Row="5" Col="2"/>
  <WriteResult Type="OutputNodeWorkload" Metric="ResidenceTime" Row="5" Col="3"
               ColIncrement="1"/>
  <WriteResult Type="OutputNode" Metric="Utilization" Row="5" Col="6" ColIncrement="1"/>
<OutputProcessingSpec>
```

Fig. 3. Excerpt of Results Specifications

	A	B	C	D	E	F	G	H
1				Residence Time			Utilization	
2	Experiment	Response	DISKB	DISKA	CPU	DISKB	DISKA	CPU
3	Run1							
4	Jain	1.406	0.107	0.0345	0.0192	0.72	0.42	0.48
5	Qnap	1.406	0.107	0.0345	0.0192	0.7200	0.4200	0.4800
6	Run2							
7	Jain	6.763	0.750	0.0455	0.0278	0.96	0.56	0.64
8	Qnap	6.763	0.750	0.0455	0.0278	0.9600	0.5600	0.6400
9	Run3							
10	Jain	1.013	0.0546	0.0345	0.0346	0.4	0.42	0.624
11	Qnap	0.754	0.0546	0.0345	0.0244	0.3960	0.4200	0.4680
12	Run4							
13	Jain	3.310		0.2	0.0192		0.90	0.48
14	Qnap	3.308	0.0000E+00	0.2	0.0192	0.0000E+00	0.9000	0.4800
15								

Fig. 4. Xls file automatically produced for Jain's case study

5 Conclusions

This paper has tied together previous work on performance model interchange formats and experiment specifications. It adds an output-to-results transformation to produce performance analysis results for presentation and publication.

Our general purpose approach was demonstrated with PMIF, however it also applies to other modelling paradigms, tools, and even measurement tools. It supports the automation of model studies from the creation of the performance

model specification, the experiments to be conducted with the model, the execution of models and transformation of output to tables and charts for presentation and publication. It supports the Use Cases in Section 2 (i.e., analyzing capacity requirements, evaluating problematic systems, etc.) and streamlines typical tasks such as exploring output and identifying results for presentation. It is a standard format that can be used by multiple tools.

Future work will develop additional templates for the most frequent results and implement additional prototypes for updating tables and creating charts. We will apply the framework to other tools, and extend it to apply to real time systems. An interesting extension might include creating rules for specifying threshold values and highlighting results in tables that exceed the threshold. We also envision the integration of Performance Trees by relating the queries to the output in order to produce results.

Acknowledgements. This work is partially funded by the TIN2006-02265 QUASIMODO project of the *Ministerio de Educacion y Ciencia*, Spain. Smith's participation is sponsored by US Air Force Contract FA8750-C-09-0086.

References

1. Microsoft Office binary (doc, xls, ppt) file formats, www.microsoft.com/interop/docs/OfficeBinaryFormats.mspx
2. CMG. Computer Measurement Group, www.cmg.org
3. Jain, R.: The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling. John Wiley, Chichester (1991)
4. Smith, C.U., Lladó, C.M.: Performance model interchange format (PMIF 2.0): XML definition and implementation. In: Proc. of the First International Conference on the Quantitative Evaluation of Systems, September 2004, pp. 38–47 (2004)
5. Smith, C.U., Lladó, C.M., Puigjaner, R., Williams, L.G.: Interchange formats for performance models: Experimentation and output. In: Proc. of the Fourth International Conference on the Quantitative Evaluation of Systems, September 2007, pp. 91–100 (2007)
6. Smith, C.U., Williams, L.G.: Panel presentation: A performance model interchange format. In: Proc. of the International Conference on Modeling Techniques and Tools for Computer Performance Evaluation (1995)
7. Suto, T., Bradley, J.T., Knottenbelt, W.J.: Performance trees: A new approach to quantitative performance specification. In: Proc. 14th Intl. Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2006). IEEE Computer Society, Los Alamitos (2006)

Analytical Model of Traffic Compression in the UMTS Network

Maciej Stasiak, Janusz Wiewióra, Piotr Zwierzykowski, and Damian Parniewicz

Poznan University of Technology
Chair of Communications and Computer Networks
ul. Polanka 3, Poznań 60965, Poland
piotr.zwierzykowski@put.poznan.pl

Abstract. The paper presents a new simple analytical method for a determination of the blocking probability in a full-availability group carrying a mixture of different multi-rate traffic classes with the compression property. The proposed model can be directly used for modelling of the Iub interface in the UMTS network servicing the Release 99 and HSDPA traffic classes. The described model can be applied for the validation of the efficiency of the Iub interface measured by the blocking probability and the average carried traffic for particular traffic classes.

Keywords: analytical model, traffic compression, UMTS, HSDPA.

1 Introduction

The increase in popularity of data transfer services in mobile networks of the second and the third generations has been followed by an increase in the interest in methods for dimensioning and optimization of networks servicing multi-rate traffic. In traffic theory, the issues concerning the problem are in full swing. This situation is primarily caused by the special conditions in the construction of these networks, and by the construction of the infrastructure of the access radio network in particular – as its development, or extension, needs a precise definition and assessment of clients' needs and is relatively time-consuming. Cellular network operators define, on the basis of SLA (Service Level Agreement), a set of the KPI (Key Performance Indicator) parameters that serve as determinants in the process of network dimensioning and optimization. Dimensioning can be presented as an unending and on-going process of analyzing and designing of the network. To make this work effective it is thus necessary to work out algorithms that would, in a reliable way, model the parameters of a designed network.

The dimensioning process for the 3rd generation UMTS (Universal Mobile Telecommunications System) system should make it possible to determine such a capacity of individual elements of the system that will secure, with the assumed load of the system, a pre-defined level of Grade of Service (GoS). With dimensioning of the UMTS system, the most characteristic constraints are: the radio interface and the Iub interface. When the radio interface is a constraint, then, in order to increase the capacity, access technology should be changed or

J.T. Bradley (Ed.): EPEW 2009, LNCS 5652, pp. 79–93, 2009.

subsequent branches of the system should be added (another NodeB). If, however, a constraint on the capacity of the system results from the capacity of the Iub interface, then a decision to add other stations (nodes) can be financially unfounded, following an incomplete or incorrect analysis of the system. This means that in any analysis of the system, a model that corresponds to the Iub interface should be routinely included.

The Iub interface can carry two types of traffic: the so-called Realse 99(R99) [1] and HSDPA (High-Speed Downlink Packet Access) [2,3] traffic. Release 99 defines several services offered in UMTS networks such as speech, videocall and data transmission (up to 384 kb/s). HSDPA is an extension of the UMTS network which allows to transmit high-speed data in the downlink direction (up to 7,2 Mb/s). Several papers have been devoted to traffic modelling in cellular systems with the WCDMA radio interface [4,5,6,7,8,9,10,11,12,13,14,15]. However, to the best knowledge of the authors, the relevant literature, proposes only one analytical model of the Iub interface [16]. In this paper the authors discuss the influence of the organization scheme of Iub on the efficiency of the interface. The paper describes two analytical models corresponding to the static and the dynamic organization scheme of the Iub interface. In the static scheme it was assumed that Iub was divided into two separated links and one of them carried a mixture of R99 whereas the other HSDPA traffic stream. In this scheme each of the links was modelled by the full-availability group with multi-rate traffic. The second organization scheme assumed dynamic constraint of the Iub interface resources for R99 traffic accompanied by unlimited access to the resources for HSDPA traffic. The dynamic organization scheme of Iub is analytically modelled by the new model of the full-availability group with limitation proposed by the authors. In both models the influence of the compression mechanism for HSDPA traffic classes was not taken into consideration and the average throughput per a HSDPA user was not discussed. The relevant literature discusses some analytical models for multi-rate traffic with compression (i.e. [17,18]) which can be applied for modelling HSDPA traffic. These models are quite simple under the assumption that all classes of the carried traffic are characterized by the compression property. In any other way, when the system services simultaneously classes which undergo and do not undergo compression, the methods are characterized by a high complexity, which limits their practical application.

The paper presents a new effective analytical model that can be used for the blocking probability determination in cellular systems with the Iub interface carrying a mixture the Release 99 and HSDPA traffic classes with adopted compression functionality. The paper has been divided into five sections. Section 2 briefly recalls the basic model of a full-availability group with multi-rate traffic which serves as the model presented in Section 3. Section 3 describes the analytical model of full-availability group with traffic compression. Section 4 shows an application of the model in the UMTS network for modelling of the Iub interface carrying R99 and HSDPA traffic classes. This section also includes the results obtained in the study of the system. The final section sums up the discussion.

2 Model of Full-Availability Group with Multi-rate Traffic

Let us assume that the total capacity of a full-availability group (FAG) with multi-rate traffic is equal to V Basic Bandwidth Units (BBUs). The group is offered M independent classes of Poisson traffic streams having the intensities: $\lambda_1, \lambda_2, ..., \lambda_M$. The class i call requires t_i BBUs to set up a connection. The holding time for calls of particular classes has an exponential distribution with the parameters: $\mu_1, \mu_2, ..., \mu_M$. Thus, the mean traffic offered to the system by the class i traffic stream is equal to:

$$A_i = \lambda_i / \mu_i. \tag{1}$$

The demanded resources for servicing particular classes in the group can be treated as a call demanding an integer number of (BBUs) [19]. The value of BBU, i.e. t_{BBU}, is calculated as the greatest common divisor of all resources demanded by the traffic classes offered to the system:

$$t_{BBU} = \mathrm{GCD}(R_1, ..., R_M), \tag{2}$$

where R_i is the amount of the resources demanded by class i call in *kbps*.

The multi-dimensional Markov process in FAG can be approximated by a one-dimensional Markov chain which can be described by the Kaufman-Roberts recursion [20,21]:

$$n\,[P_n]_V = \sum_{i=1}^{M} A_i t_i\,[P_{n-t_i}]_V\,, \tag{3}$$

where $[P_n]_V$ is the probability of state n BBUs being busy, and t_i is the number of BBUs required by a class i call:

$$t_i = \lfloor R_i / t_{BBU} \rfloor. \tag{4}$$

On the basis of formula (3), the blocking probability E_i for class i stream can be expressed as follows:

$$E_i = \sum_{n=V-t_i+1}^{V} [P_n]_V, \tag{5}$$

where V is the total capacity of the group and is expressed in BBUs ($V = \lfloor V_{phy}/t_{BBU} \rfloor$, where V_{phy} is the physical capacity of group in kbps).

The diagram in Fig. 1 corresponds to formula (3) for the system with two call streams (M=2, t_1=1, t_2=2). The $y_i(n)$ symbol denotes the so-called *reverse transition rate* of a class i service stream outgoing from state n. This parameter can be calculated on the basis of the local equations of equilibrium in the Markov chain [20,22]:

$$y_i(n) = \begin{cases} A_i\,[P_{n-t_i}]_V \,/\, [P_n]_V & \text{for } n \le V, \\ 0 & \text{for } n > V. \end{cases} \tag{6}$$

The reverse transition rate determines the average number of class i calls serviced in the state n.

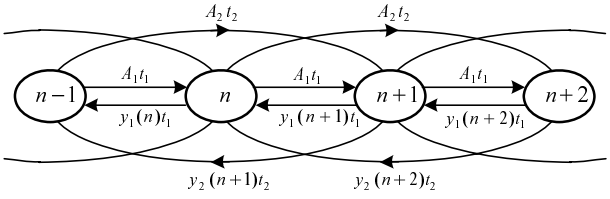


Fig. 1. Section of a diagram of the one-dimensional Markov chain in a multi-rate system (M=2, t_1=1, t_2=2)

3 Model of Full-Availability Group with Compression

Let as assume now that a full-availability group services a mixture of different multi-rate traffic streams with the compression property. This means that in the traffic mixture there are such calls in which a change in demands (requirements) is followed uniformly by the overload of the system.

In this group it is assumed that the system services simultaneously a mixture of different multi-rate traffic classes, while these classes are divided into two sets: classes whose calls can change requirements (demands) while being serviced and classes that do not change their demands in their service time.
In the considered model the following notation is used:

- $\mathbb{M}_k$ denotes a set of classes capable of compression, while $M_k = |\mathbb{M}_k|$ is the number of compressed traffic classes,
- $\mathbb{M}_{nk}$ is a set of classes without compression, and $M_{nk} = |\mathbb{M}_{nk}|$ denotes the number of classes without compression.

It was assumed in the model that all classes undergoing compression were compressed to the same degree. The measure of a possible change in requirements is *the maximum compression coefficient* that determines the ratio of the maximum demands to minimum demands for a given traffic classes. The coefficient $K_{\max}$ can be determined on the basis of the dependence:

$$\forall_{j\in\mathbb{M}_k} \quad K_{\max} = \frac{t_{j,\max}}{t_{j,\min}}, \tag{7}$$

where $t_{j,\max}$ and $t_{j,\min}$ denote respectively the maximum and the minimum number of basic bandwidth units (BBUs) demanded by a call of class j.

We assume that the system will be treated as a full-availability group with multi-rate traffic. The occupancy distribution in such a system can be expressed by the recursive Kaufman-Roberts formula (3), under the assumption that the amount of required resources by calls of the classes with the compression property is minimum. In the case of a system carrying a mixture of traffic streams that

undergo and do not undergo compression, the occupancy distribution (3) will be more conveniently expressed after dividing the two types of traffic[1]:

$$n\,[P_n]_V = \sum_{i=1}^{M_{nk}} A_i t_i\,[P_{n-t_i}]_V + \sum_{j=1}^{M_k} A_j t_{j,\min}\left[P_{n-t_{j,\min}}\right]_V, \tag{8}$$

where $t_{j,\min}$ is the minimum number of demanded BBUs in a given occupation state of the system by a call of class j that belongs to the set $\mathbb{M}_k$.
The blocking (loss) coefficient in the full-availability group will be determined by the dependence (6) that, in the considered case, will take on the following form:

$$E_i = B_i = \begin{cases} \sum\limits_{n=V-t_i+1}^{V} [P_n]_V & \text{for } i \in \mathbb{M}_{nk}, \\ \sum\limits_{n=V-t_{i,\min}+1}^{V} [P_n]_V & \text{for } i \in \mathbb{M}_k. \end{cases} \tag{9}$$

In Equations (8) and (9), the model is characterized by the parameter $t_{i,\min}$ which is the minimum number of BBUs demanded by a call of class i under the conditions of maximum compression. Such an approach is indispensable in determining the blocking probabilities in the system with compression since the blocking states will occur in the conditions of maximum compression. The maximum compression determines such occupancy states of the system in which further decrease in the demands of calls of class i is not possible.

In order to determine a possibility of the compression of the system it is necessary to evaluate the number and the kind of calls serviced in a given occupancy state of the system. For this purpose we can use Formula (5) that makes it possible to determine the average number of calls of class i serviced in the occupancy state n BBUs. This dependence, under the assumption of the maximum compression, can be written in the following way:

$$y_i(n) = \begin{cases} \frac{A_i\left[P_{n-t_i}\right]_V}{[P_n]_V} & \text{for } i \in \mathbb{M}_{nk}, \\ \frac{A_i\left[P_{n-t_{i,\min}}\right]_V}{[P_n]_V} & \text{for } i \in \mathbb{M}_k. \end{cases} \tag{10}$$

On the basis of Formula (10), knowing the demands of individual calls, we can thus determine the total average carried traffic in state n, under the assumption of the maximum compression:

$$Y_{\max}(n) = Y^{nk}(n) + Y^{k}_{\max}(n) = \sum_{i=1}^{M_{nk}} y_i(n) t_i + \sum_{j=1}^{M_k} y_j(n) t_{j,\min}, \tag{11}$$

where $Y^{k}_{\max}(n)$ is the average number of busy BBUs in state n occupied by calls that undergo compression, whereas $Y^{nk}(n)$ is the average number of busy BBUs in state n occupied by calls without compression.

[1] Further on in the paper, the terms “a set of classes with the possibility of compression" and “class with the possibility of compression", will be simplified to “a set of classes with compression" and, respectively, a “class with compression", respectively.

Let us assume that the value of the parameter $Y^{nk}(n)$ refers to non-compressed traffic and is independent of the compression of the remaining calls. The real values of carried traffic, corresponding to state n (determined in the condition of maximum compression), will depend on the number of free BBUs in the system. We assume that the real system operates in such a way as to guarantee the maximum use of the resources, i.e. a call of a compressed class always tends to occupy free resources and decreases its maximum demands in the least possible way. Thus, the real traffic value $Y(n)$, carried in the system in a given state corresponding to state n (determined in maximum compression) can be expressed in the following way[2]:

$$Y(n) = Y^{nk}(n) + Y^{k}(n) = \sum_{i=1}^{M_{nk}} y_i(n)t_i + \sum_{j=1}^{M_k} y_j(n)t_j(n). \tag{12}$$

The parameter $t_j(n)$ in Formula (12) determines the real value of a demand of class j in state n:

$$\forall_{j\in\mathbb{M}_k} \;\; t_{j,\min} < t_j(n) \leq t_{j,\max}. \tag{13}$$

The measure of the degree of compression in state n is the compression coefficient $\xi_k(n)$, which can be expressed in the following way:

$$t_j(n) = t_{j,\min}\xi_k(n). \tag{14}$$

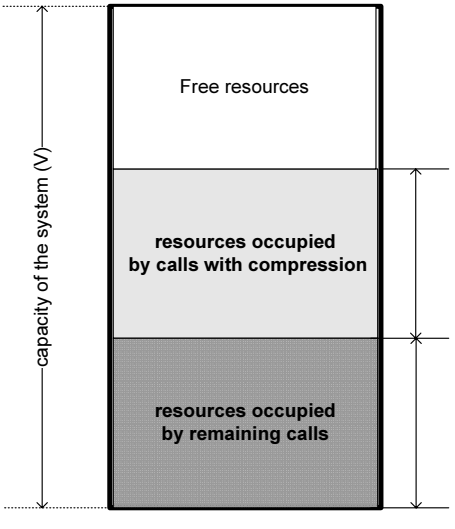


Fig. 2. Exemplary system with compression

[2] Further on in the description, to simplify the description, we will use the term "in state n" instead of "a given state n in maximum compression".

Taking into consideration (14), the average number of busy BBUs occupied by calls with compression can be written thus:

$$Y^k(n) = \sum_{j=1}^{M_k} y_j(n) t_j(n) = \xi_k(n) \sum_{j=1}^{M_k} y_j(n) t_{j,\min}. \tag{15}$$

We assume in the considered model that the system operates in such a way that it guarantees the maximum use of available resources and this means that calls that undergo compression will always tend to occupy free resources, decreasing their demands in the least possible way. The other parameter of the considered system, beside the blocking (loss) probability, is the average number of busy BBUs in the system occupied by calls with compression (Formula (15)). In order to determine this parameter, the knowledge of the compression coefficient $\xi_k(n)$ is indispensable. This coefficient can also be defined as the ratio of potentially available resources for the service of calls with compression to the resources occupied by these calls in the state of maximum compression. Thus, we can write (Figure 2):

$$\xi_k(n) = \frac{V - Y^{nk}(n)}{Y^k_{\max}(n)} = \frac{V - Y^{nk}(n)}{n - Y^{nk}(n)}. \tag{16}$$

The numerator in the Formula (16) expresses the total amount of resources of the system which can be occupied by calls of the class with compression, whereas the denominator can be interpreted as the the amount of resources which can be occupied by calls of the class with compression, under the assumption that the system (FAG) is in the state n of busy BBUs. A constraint to the value of the coefficient (16) is the maximum compression coefficient determined on the basis of the dependence (7). This constraint can be taken into account by defining formally the compression coefficient in the following way:

$$\xi_k(n) = \begin{cases} K_{\max} \text{ for } & \xi_k(n) \geq K_{\max}, \\ \xi_k(n) \text{ for } & 1 \leq \xi_k(n) < K_{\max}. \end{cases} \tag{17}$$

The compression coefficient determined by Formula (17) is not dependent on the traffic class. This results from the adopted assumption in the model of the same degree of compression for all traffic classes that undergo the mechanism of compression.

Knowing the value of the compression coefficient in every state n, we can determine the average resources occupied by calls of class j with compression:

$$Y^k_j = \sum_{n=0}^{V} y_j(n) \left[\xi_k(n) t_{j,\min}\right] [P_n]_V. \tag{18}$$

On the basis of the average resources occupied by calls of class j, we can determine the average resources occupied by calls of all traffic classes with compression:

$$Y^k = \sum_{j=0}^{M_k} Y^k_j. \tag{19}$$

Let us note that the value Y^k in Formula (19) is the average carried traffic in the system by calls which undergo compression.

4 Application of the Model in the UMTS Network

4.1 Architecture of the UMTS Network

Let us consider the structure of the UMTS network presented in Fig. 3. The presented network consists of three functional blocks designated respectively: UE (User Equipment), UTRAN (UMTS Terrestrial Radio Access Network) and CN (Core Network). The following notation has been adopted in Fig. 3: RNC is the Radio Network Controller, WCDMA is the radio interface and Iub is the interface connecting Node B and RNC. In the dimensioning process for the

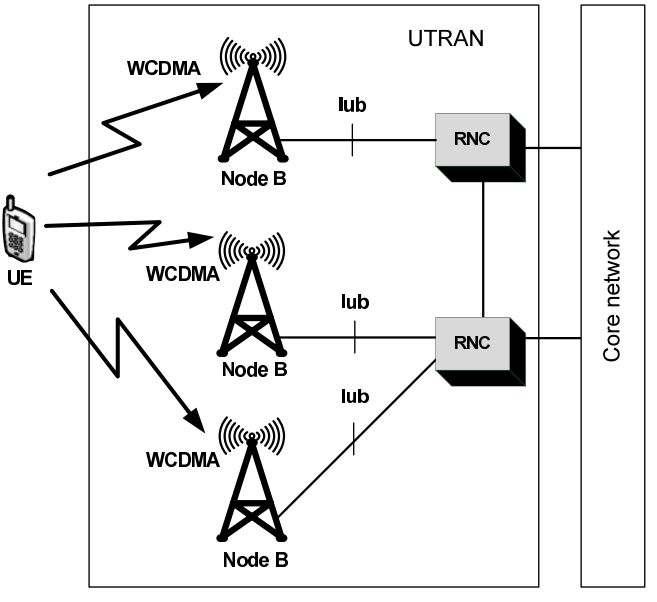


Fig. 3. Elements of the UMTS network structure

UMTS network, an appropriate dimensioning of the connections in the access part (UTRAN), i.e. the radio interface between the user and NodeB, and the Iub connections between NodeB and the RNC (Radio Network Controller), has a particular significance.

Having in mind the duration time of the expansion of a network and huge relevant costs involved as well as the following possible savings in expenditures, operators of cellular networks are more than eager to implement technological solutions that optimize investments but, nevertheless, make it possible to retain

the complex quality of service. One of such solutions, frequently used in real networks, is a separation of links on the Iub interface. The operator is in position to configure two virtual paths (VP - virtual path) of ATM (Asynchronous Transfer Mode) on the Iub interface and assign them to, respectively, real-time traffic and best effort traffic. Assuming that best effort VC (Virtual Channel) will not allocate the maximum demanded bandwidth in the same time, the total bandwidth can be co-shared between the VCs, which results in its better utilization. The application of the method should be thus even recommended for distinguishing parameters needed for designing/dimensioning of networks with different QoS requirements for different clients. Obviously, in the case of bandwidth overload, part of ATM cells will be lost. An example of a physical realization of this type of a solution on the Iub interface with the application of IMA (Inverse Multiplexing for ATM) [23] is shown in Fig. 4. The application of IMA makes it possible to create two logical ATM paths on the basis of separate physical links[3]. Table 1

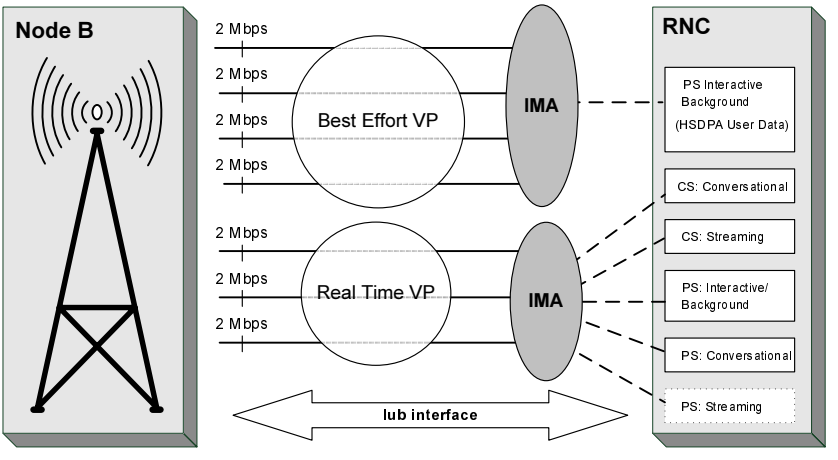


Fig. 4. One of the most common ways of carrying out a connection between the UMTS base station (NodeB) and RNC (Radio Network Controller) with the application of IMA technology

shows an example of UMTS Packet Switched (PS) and Circuit Switched (CS) services carried out by logical ATM paths dedicated to servicing “best effort" traffic and “real-time"traffic, respectively, and corresponding to Figure 4.

Additionally, it should be mentioned that the application of this solution paves the way for further optimization of capacity since with the application of traffic concentration devices between NodeB and RNC, the paths of the type “Real Time" will be carried by the concentrating device in the capacity ratio 1:1, while the paths of the “best effort" type can be carried, for example, in the ratio 2:1 (a twofold higher capacity on the input of the concentration device than on

[3] Figure 4 assumes that the links constituting IMA have throughput of 2Mbps.

Table 1. An example of service class mapping into ATM classes

ATM class of service	UMTS class of service	Exemplary service
Best Effort VP	Interactive Background (HSDPA User Data)	Web Browsing
Real Time VP	CS: Conversational	Voice
Real Time VP	CS: Streaming	Modem Connection
Real Time VP	PS: Interactive/Background	FTP, Realtime Gaming
Real Time VP	PS: Conversational	Speech (VoIP)
Real Time VP	PS: Streaming	Mobile TV

the output). Using the properties of offered traffic (e.g. different Busy Hours) we can reach further savings, at least with the development or expansion of RNC that has a limited number of input ports. A very good technology that ensures successful realization of the task, simultaneously facilitating the construction of the Iub interface, is LMDS (Local Multipoint Distribution Service) [24].

Regrettably, this rapid pace in the development of relevant technologies is not appropriately matched by mathematical models that would enable us to plan and dimension networks in accordance with required service predictions. Though there are many published studies on the optimization and dimensioning of the radio interface - for example the earlier works of the present authors e.g. [7, 8, 12, 15], still, a successive interface developed on such a large scale has not been exhaustively described mathematically. The method presented in this article fits well then in the process of optimization of the development and dimensioning of networks.

4.2 Calculation Algorithm

The Iub interface in the UMTS network can be treated as a full-availability group with multi-rate traffic (Sect.2). It is adopted in the paper that Iub is offered a mixture of different multi-rate traffic including HSDPA traffic. The proposed model with traffic compression (Sect.3), can therefore be used for modelling the Iub interface carrying HSDPA traffic in which a change in demands follows uniformly for all compressed classes.

On the basis of the considerations presented in Sects. 2 and 3, the algorithm of blocking probability E_i and average occupied traffic Y_i^k calculations for the Iub interface may be written as follows:

1. Calculation of offered traffic load A_i of class i (Eq. (1)).
2. Designation of the value of t_{BBU} as the greatest common divisor (Eq. (2)).
3. Determination of the value of t_i as the integer number of demanded resources by class i calls (Eq. (4)).
4. Calculation of state probabilities $[P_n]_V$ in FAG (Eq. (8)).
5. Designation of the blocking probability of class i (Eq. (9)).
6. Determination of the reverse transition rate for class i (Eq. (10)).
7. Calculation of the average compression coefficient (Eq. (17)).
8. Determination of the average traffic of class i carried by Iub (Eq. (18)).

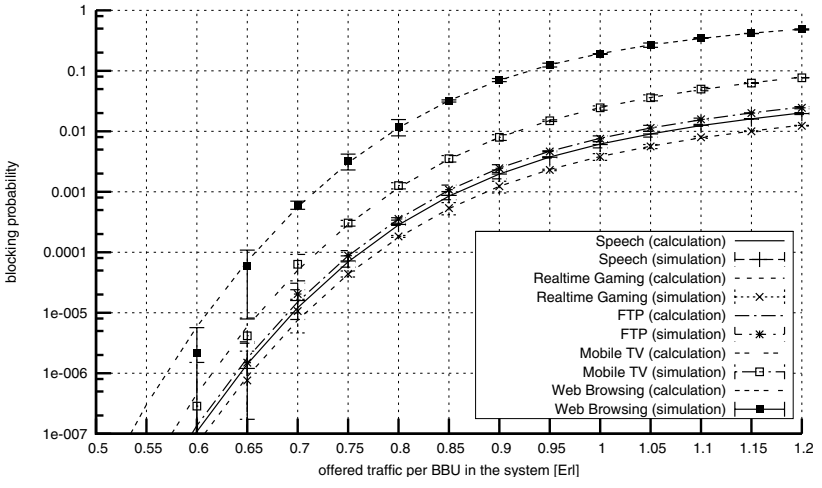


Fig. 5. Blocking probabilities for all traffic classes carried by the Iub Interface

4.3 Numerical Study

The proposed analytical model of the Iub interface is an approximate one. Thus, the results of the analytical calculations of the Iub interface were compared with the results of the simulation experiments. The study was carried out for users demanding a set of following traffic classes (services) in the downlink direction:

- class 1: Speech (VoIP,non-compressed) - $t_1 = 16$ kbps (16 BBUs),
- class 2: Realtime Gaming - $t_{2,\min} = 10$ kbps (10 BBUs),
- class 3: FTP - $t_{3,\min} = 20$ kbps (20 BBUs),
- class 4: Mobile TV - $t_{3,\min} = 64$ kbps (64 BBUs),
- class 5: Web Browsing - $t_{4,\min} = 500$ kbps (500 BBUs).

In the presented study, it was assumed that:

- t_{BBU} was equal to 1 kbps,
- the maximum compression coefficient was equal to 10 ($K_{\max}$=10),
- the considered Iub interface carried traffic from three radio sectors, and a physical capacity of Iub in the downlink direction was equal to: V_{Iub}= 3 × 7,2 Mbps = 21,6 Mbps (≅ 21 000 BBUs),
- the traffic classes were offered in the following proportions:

$$A_1t_1 : A_2t_2 : A_3t_3 : A_4t_4 : A_5t_5 = 1 : 3 : 10 : 5 : 10.$$

It was assumed that the main part of traffic was generated by FTP and Web Browsing services followed by Mobile TV and Realtime Gaming services, while the smallest part of traffic comes from VoIP service.

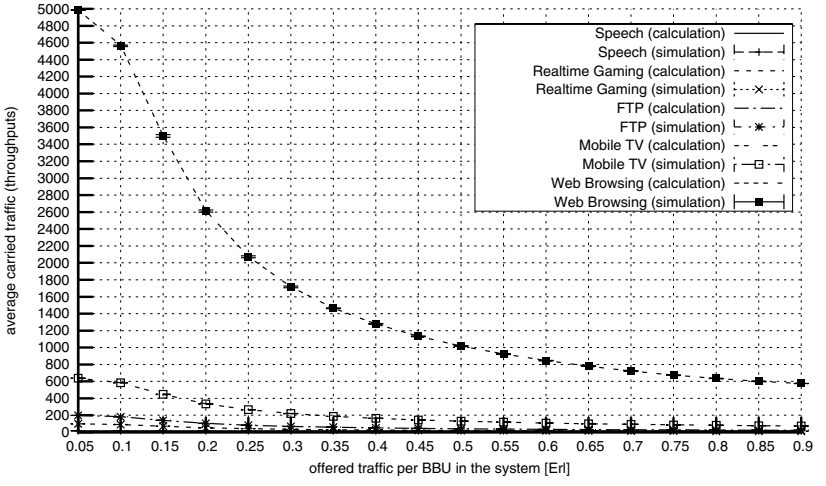


Fig. 6. Average carried traffic for particular classes serviced by the Iub interface

Figure 5 shows the dependence of the blocking probability in relation to traffic offered per BBU in the Iub interface. The presented results were obtained for a minimum value of required (demanded) resources for traffic classes with compression property.

Figures 7 and 6 present the influence of traffic offered per BBU in the Iub interface on the average carried traffic by Iub (Fig. 6), and on the value of the compression coefficient (Fig. 7). The results presented in Fig. 6 for Speech, Realtime Gaming, FTP and Mobile TV are very similar and, therefore, analytical results for the average carried traffic, for selected values of offered traffic per BBU in the Iub interface, were also presented in Tab. 2. It can be noticed that the plots corresponding to the traffic classes with compression in both figures have exponential character. The linear relation between compression coefficient and the average carried traffic (see Eq. (18)) explains the similar character of the curves in both figures. The results confirm a strong dependence between the average carried traffic (throughput) and the load of the system – the more overloaded system, the lower value of throughput. The character of results from a decrease in the amount of the resources required by a call of classes with compression: the more overloaded system, the smaller demands for calls with

Table 2. Average carried traffic for particular classes carried by Iub

Class of service	$a = 0.1$	$a = 0.3$	$a = 0.5$	$a = 0.7$	$a = 0.9$	$a = 1.2$
Speech (VoIP)	16.0	16.0	16.0	16.0	16.0	16.0
Realtime Gaming	91.4	34.4	20.4	14.5	11.5	10.3
FTP	182.9	68.7	40.7	29.0	23.0	20.7
Mobile TV	585.2	219.9	130.4	92.6	73.6	66.2

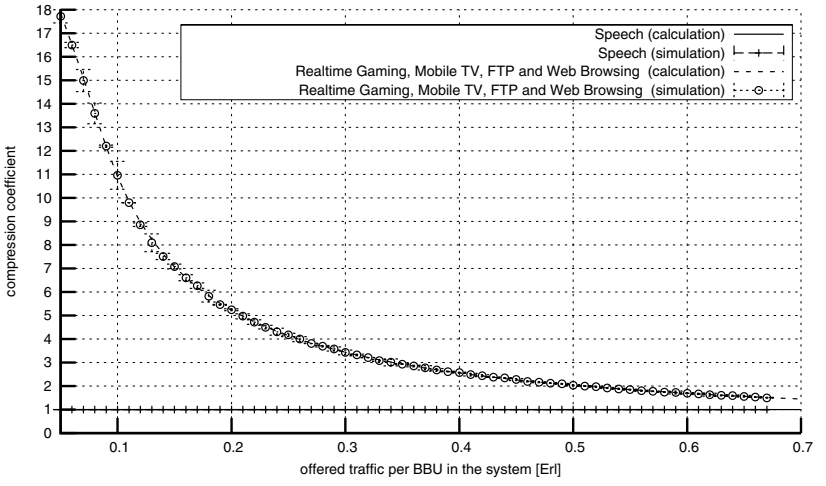


Fig. 7. Compression coefficient in relation to traffic offered to the Iub interface

compression. It is noticeable that the value of throughput for particular traffic classes from the value 0.9 of traffic offered per BBU decreases very slowly and, therefore, these values are not presented in the figures. This phenomenon corresponds to obtaining a minimum value of demanded resources by the calls with compression .

The results of the simulations are shown in the charts in the form of marks with 95% confidence intervals calculated after the t-Student distribution. 95% confidence intervals of the simulation are almost included within the marks plotted in the figures.

5 Conclusions

This paper proposes a new analytical model with compression that finds its application in modelling the Iub interface in the UMTS network carrying a mixture of different multi-rate Release 99 and HSDPA traffic classes, such as background (i.e. Web Browsing, FTP) and interactive (i.e. Mobile TV, Realtime Gaming) traffic classes.

The presented analytical method allows to determine the blocking probability for all traffic classes serviced by the Iub interface. It should be noted that in the model we assume the “worst case" approach in the Iub modelling and dimensioning that makes our calculations independent of the way of operation of the HSDPA scheduler [2], which underlines the universal character of the method.

It is worth emphasizing that the described analytical model could be used for a determination of the average carried traffic for particular traffic classes serviced by the Iub interface.

KPI, being an indispensable element of SLA, can be defined differently depending on the kind of the receiver of information. Thus, KPI will be defined

differently for engineering staff and differently for non-technical staff often involved in decision making concerning expenditures that are to ensure appropriate quality of services. While such parameter as the blocking probability is well understood by engineers, clients and non-technical staff may have some problems with the interpretation of the indicator and this group of users will rather prefer the average value as being more intuitive. The average value of carried traffic is also very characteristic for some services such as FTP or Web Browsing. With regards to the above factors, a necessity appeared of a skillful use of the average value of carried traffic as the initial value in the process of designing and dimensioning of the UMTS-HSDPA networks without violating the basic merits of the adopted model that are necessary for a system to operate successfully. Thus, this parameter is an important factor in 3G network capacity calculations, i.e. in dimensioning and optimization of WCDMA and Iub interfaces.

References

1. Holma, H., Toskala, A.: WCDMA for UMTS. Radio Access For Third Generation Mobile Communications. John Wiley and Sons, Ltd., Chichester (2000)
2. Holma, H., Toskala, A.: HSDPA/HSUPA for UMTS: High Speed Radio Access for Mobile Communications. John Wiley and Sons, Ltd., Chichester (2006)
3. Nawrocki, M., Aghvami, H., Dohler, M.: Understanding UMTS Radio Network Modelling, Planning and Automated Optimisation: Theory and Practice. John Wiley and Sons, Ltd., Chichester (2006)
4. Staehle, D., Mäder, A.: An analytic approximation of the uplink capacity in a UMTS network with heterogeneous traffic. In: 18th International Teletraffic Congress (ITC18), Berlin, pp. 81–91 (2003)
5. Iversen, V., Benetis, V., Trung, H.: Evaluation of multi-service CDMA networks with soft blocking. In: ITC 16th Specialist Seminar on Performance Evaluation of Mobile and Wireless Systems, Antwerp, Belgium, pp. 212–216 (2004)
6. Kwon, Y.S., Kim, N.: Capacity and cell coverage based on calculation of the Erlang capacity in a WCDMA system with multi-rate traffic. IEICE Transactions on Communications E87-B(8), 2397–2400 (2004)
7. Stasiak, M., Wiśniewski, A., Zwierzykowski, P.: Blocking probability calculation in the uplink direction for cellular systems with WCDMA radio interface, pp. 65–74. VDE Verlag GMBH, Berlin (2004)
8. Głąbowski, M., Stasiak, M., Wiśniewski, A., Zwierzykowski, P.: Uplink blocking probability calculation for cellular systems with WCDMA radio interface, finite source population and differently loaded neighbouring cells. In: Asia-Pacific Conference on Communications, pp. 138–142 (2005)
9. Ishikawa, Y., Onoe, S., Fukawa, K., Suzuki, H.: Blocking probability calculation using traffic equivalent distributions in SIR-based power controlled W-CDMA cellular systems. IEICE Transactions on Communications E88-B(1), 312–324 (2005)
10. Koo, I., Kim, K.: Erlang capacity of multi-service multi-access systems with a limited number of channel elements according to separate and common operations. IEICE Transactions on Communications E89-B(11), 3065–3074 (2006)
11. Vassilakis, V.G., Logothetis, M.D.: The wireless Engset multi-rate loss model for the handoff traffic analysis in W-CDMA networks. In: Proccedings of 19th International Symposium on Personal, Indoor and Mobile Radio Communications, Cannes, France, pp. 1–6 (2008)

12. Stasiak, M., Zwierzykowski, P., Wiewióra, J., Parniewicz, D.: An Approximate Model of the WCDMA Interface Servicing a Mixture of Multi-rate Traffic Streams with Priorities. In: Thomas, N., Juiz, C. (eds.) EPEW 2008. LNCS, vol. 5261, pp. 168–180. Springer, Heidelberg (2008)
13. Kallos, G.A., Vassilakis, V.G., Logothetis, M.D.: Call blocking probabilities in a W-CDMA cell with fixed number of channels and finite number of traffic sources. In: Proceedings of 6th International Conference on Communication Systems, Networks and Digital Signal Processing, Graz, Austria, pp. 200–203 (2008)
14. Głąbowski, M., Stasiak, M., Wiśniewski, A., Zwierzykowski, P.: Uplink blocking probability calculation for cellular systems with WCDMA radio interface and finite source population. Information Science and Technology. In: Performance Modelling and Analysis of Heterogeneous Networks, pp. 301–318. River Publishers (2009)
15. Stasiak, M., Wiśniewski, A., Zwierzykowski, P., Głąbowski, M.: Blocking probability calculation for cellular systems with WCDMA radio interface servicing PCT1 and PCT2 multirate traffic. IEICE Transactions on Communications E92-B(4), 1156–1165 (2009)
16. Stasiak, M., Wiewióra, J., Zwierzykowski, P.: Analytical modelling of the Iub interface in the UMTS network. In: Proceedings of 6th Symposium on Communication Systems, Networks and Digital Signal Processing, Graz, Austria (2008)
17. Moscholios, I., Logothetis, M., Kokkinakis, G.: Connection-dependent threshold model: a generalization of the Erlang multiple rate loss model. Performance Evaluation 48, 177–200 (2002)
18. Rácz, S., Gerö, B.P., Fodor, G.: Flow level performance analysis of a multi-service system supporting elastic and adaptive services. Performance Evaluation 49(1-4), 451–469 (2002)
19. Roberts, J., Mocci, V., Virtamo, I. (eds.): COST-242 1996. LNCS, vol. 1155. Springer, Heidelberg (1996)
20. Kaufman, J.: Blocking in a shared resource environment. IEEE Transactions on Communications 29(10), 1474–1481 (1981)
21. Roberts, J.: A service system with heterogeneous user requirements – application to multi-service telecommunications systems. In: Pujolle, G. (ed.) Proceedings of Performance of Data Communications Systems and their Applications, pp. 423–431. North Holland, Amsterdam (1981)
22. Głąbowski, M.: Modelling of state-dependent multi-rate systems carrying BPP traffic. Annals of Telecommunications 63(7-8), 393–407 (2008)
23. Bannister, J., Mather, P., Coope, S.: Convergence Technologies for 3G Networks: IP, UMTS, EGPRS and ATM, 1st edn. John Wiley and Sons, Ltd., Chichester (2004)
24. Smith, C.: LMDS: Local Mutipoint Distribution Service. McGraw-Hill, New York (2000)

From DFTs to PEPA: A Model-to-Model Transformation

Leïla Kloul

PRiSM, Université de Versailles, 45, Av. des Etats-Unis, 78000 Versailles
kle@prism.uvsq.fr

Abstract. One of the main attractions of the stochastic process algebra PEPA is its clear compositional structure which allows building a model from elements which reflect the composition of the system to model. Dynamic Fault Trees (DFTs) constitute a simple combinatorial formalism which allows capturing the dynamic behaviour of system failure mechanisms. The resulting model is a combination of component failure events which determines the failure of the complete system. In this paper, we propose to exploit the compositional feature of both PEPA and DFTs in order to develop an automatic translation of a DFT into a PEPA model.

1 Introduction

Standard or static fault trees [11,13] are widespread stochastic models for system reliability analysis. A Fault Tree (FT) model is a combination of component failure events which determines the failure of the complete system. The popularity of this formalism is due to its simplicity and the efficiency of techniques such as BDDs to solve the models. However FTs become inefficient whenever the system to model has strongly-dependent components.

Dynamic fault trees are an extension of standard fault trees to which new primitive gates have been added in order to model functional or temporal dependencies among failure events or component states. Because of the nature of these dependencies, FTs solving techniques are ineffective for solving DFTs.

DFTs are generally solved using automatic conversion to Markov models which may lead to the generation of huge state spaces. Most of the solutions developed in the literature have shortcomings including a lack of formality, a lack of compositional modelling and thus a limitation in modular analysis, and often a lack of generality because the set of possible inputs to some dynamic gates is restricted (see Section 6).

In order to model formally the elements of a DFT, we consider the stochastic process algebra PEPA. Performance Evaluation Process Algebra (PEPA) [9] is a compositional approach which extends classical process algebra by associating a random variable, representing duration, with every action. These random variables are assumed to be exponentially distributed and this leads to a clear mapping from the process algebra model to the continuous time Markov process.

J.T. Bradley (Ed.): EPEW 2009, LNCS 5652, pp. 94–109, 2009.

One of the main attractions of PEPA is its clear compositional structure which allows us to build a model from elements or components which reflect the composition of the system to model. This provides, on one hand, a better understanding of the complexity of the models to build, and on the other hand, a better exploitation of the structural properties when solving the model. PEPA is supported by a wide-range of tools and techniques such as the aggregation [9] and the compact representation of the generator of the Markov chain [10]. The former can result in a significant reduction in the size of the matrix, for example in models which exhibit symmetry due to repeated components, which is generally the case in a DFT. With the latter the matrix representation may be decomposed so that the state space of the model, and its dynamics, are not represented by a single matrix but by a number of smaller matrices. Nevertheless the model is solved as a single entity.

We consider the extended version of PEPA [10] which includes the functional rates. These rates allow functional dependencies to be expressed between the components of the model, which means that the timing behaviour of one component (or indeed the activities it is able to undertake) may depend on the current state of another component.

As a DFT is a combination of system components failure events, we develop an approach which takes advantage of the compositional feature of PEPA in order to obtain a direct mapping from a DFT configuration into a PEPA model. The dynamic dependencies between the components failure events are captured using the functional dependencies between the PEPA components. This approach is a powerful potential solution to modelling and analysing various kind of system components behaviours and interactions. Moreover, unlike previous works, we take into consideration the repair process of both static and dynamic gates, and previous enforced restrictions on some dynamic gates are lifted.

The paper is organised as follows: in Section 2, we present DFTs. A brief overview of the formalism PEPA is presented in Section 3. In Section 4, we develop the formal specifications of both static and dynamic gates. The example of a cardiac system is investigated in Section 5. Section 6 is dedicated to the related work. Finally the conclusions of our work are presented in Section 7.

2 Dynamic Fault Trees

Three types of events define the behaviour of a fault tree: *basic events*, *internal events* and the *top event*. Basic events correspond to the failure events of the physical components of the modelled system. They are the leaves of the tree. Internal events are the output (failure) events of the intermediary gates in the tree. Finally, the top event corresponds to the failure event of the complete system. It is the failure event of the top-gate (top-node) in the tree.

A fault tree becomes a dynamic fault tree as soon as a dynamic gate is introduced in the tree. Therefore a dynamic fault tree may consist of both static and dynamic gates. Static gates include *AND Gate*, *OR Gate* and *K/N Gate*. Dynamic gates include *PAND Gate*, *FDEP Gate*, *Spare Gate* and *SEQ Gate*.

2.1 Static Gates

Static gates were introduced in standard fault trees in order to model Boolean conditions applied on basic events. These gates are the following:

AND Gate: The output event (failure of the gate) occurs if all input events have occurred.

OR Gate: The output event occurs if any of the input events occurs.

K/N Gate: The output event occurs if at least K of the N input events have occurred. It is also known as the voting gate.

2.2 Dynamic Gates

Dynamic gates (Fig. 1) model functional or temporal dependencies among failure events or component states. They take into consideration the order in which the input events occur. This order is important to determine the output event of the gate.

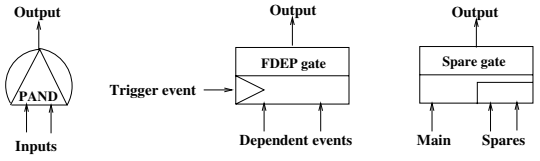


Fig. 1. Dynamic gates

PAND Gate: A Priority-AND gate has been introduced to model failure sequence dependency. The output event of a PAND gate occurs when the input events occur in a specified order (from left to right).

FDEP Gate: A Functional DEPendency gate consists of a trigger event and a set of *dependent basic events*. The occurrence of the trigger event causes the inaccessibility (failure) of the basic events. The FDEP gate outcome is not taken into account when computing the system's failure probability because the gate has no real output (dummy).

Spare Gate: It models heterogeneous spare component management and allocation. A spare gate comprises a main input event and one or more alternate input events (the spares). Initially the main input is powered on whereas the alternate inputs are in the dormant state. When the main input fails, a spare is passed from the dormant state to the active state, replacing the main input in its function. A spare gate fails when the main input fails and all spare components fail or are unavailable. A spare component may fail in both the dormant and the active states. However, during the dormant state, the failure rate is reduced by a factor called the dormancy factor α. According to the value of α, a spare component may be a cold spare ($\alpha = 0$), a hot spare ($\alpha = 1$) or a warm spare

$(0 < \alpha < 1)$. As a spare component can be shared among several spare gates, it can become unavailable because used by one of these gates. Note that the only input events allowed to a spare gate are basic events.

SEQ Gate: The SEQuence enforcing gate forces its inputs to fail in a specific order (from left to right); it never happens that the failure takes place in a different order. It appears that this gate can be emulated using a cold spare gate [12], so it is not considered in the rest of the paper.

3 PEPA

In PEPA a system is described as an interaction of *components* which engage, either singly or multiply, in *activities*. These basic elements of PEPA, components and activities, correspond to *states* and *transitions* in the underlying Markov process. Each activity has an *action type* and its duration is represented by the parameter of the associated exponential distribution: the *activity rate*. This parameter may be any positive real number, or the distinguished symbol $\top$ (read as *unspecified*). Thus each activity, a, is a pair (α, r) consisting of the action type and the activity rate respectively. Activities which are private to a component in which they occur are represented by the distinguished action type, τ. We assume a countable set, $\mathcal{A}$, of all possible action types. We denote by $\mathcal{A}ct \subseteq \mathcal{A} \times \mathbb{R}^*$, the set of activities, where $\mathbb{R}^*$ is the set of non-negative real numbers together with symbol $\top$ [10]. PEPA provides a small set of combinators which allow expressions to be constructed defining the behaviour of components, via the activities they undertake and the interactions between them.

Prefix. $(\alpha, r).P$: This is the basic mechanism for constructing component behaviours. The component carries out activity (α, r), then behaves as component P.

Choice. $P + Q$: This component may behave either as P or as Q: all the current activities of both components are enabled. The first activity to complete, determined by a *race condition*, distinguishes one component, the other is discarded.

Cooperation. $P \underset{L}{\bowtie} Q$: Components proceed independently with any activities whose types do not occur in the *cooperation set* L (*individual activities*). However, activities with action types in set L require the simultaneous involvement of both components (*shared activities*). When the set L is empty, we use the more concise notation $P \parallel Q$ to represent $P \underset{\emptyset}{\bowtie} Q$. In PEPA the shared activity occurs at the rate of the slowest participant. If an activity has an unspecified rate, denoted $\top$, the component is *passive* with respect to that action type.

Hiding. P/L: This behaves as P except that any activities of types within the set L are *hidden*, i.e. they exhibit the unknown type τ and can be regarded as an internal delay by the component. These activities cannot be carried out in cooperation with another component.

Constant. $A \stackrel{def}{=} P$: Constants are components whose meaning is given by a defining equation. $A \stackrel{def}{=} P$ gives the constant A the behaviour of the component P. This is how we assign names to components (behaviours).

The evolution of a model is governed by the structured operational semantics rules of the language. These define the admissible transitions or state changes associated with each combinator. A race condition governs the dynamic behaviour of a model whenever more than one activity is enabled. The underlying process of a PEPA model is a CTMC.

4 Formal Specification of the Gates

We assume that failure event occurrence is an exponentially distributed random variable, the component failure rate being the parameter of the distribution. In the following we show how to model formally both static and dynamic gates using PEPA. Each gate type is modelled in isolation. The individual PEPA models can then be used as basic macro-components to build the model of any DFT configuration. Our modelling approach allows us to lift the restrictions on spare and FDEP gate's input events to be basic events.

4.1 Static Gates

Assuming that the number of input events or entries to a static gate is N, an input event i, $i = 1, \ldots, N$, is modelled using a two states component as follows:

$$input_i \stackrel{def}{=} (failure, a_i).input'_i \qquad\qquad input'_i \stackrel{def}{=} (repair, r_i).input_i$$

Assuming that x_i, $i = 1, \ldots, N$, is the current state of input i, the static gates can be modelled as in the following.

AND Gate: as the output event occurs only if all input events of the gate occur, the rate of action type *failure* depends on the state of these inputs. Moreover, once the gate is in the failure state, it will be repaired if one of its inputs is repaired. This is modelled using action type *repair* as follows:

$$output \stackrel{def}{=} (failure, f_1 \times a).output' \qquad\qquad output' \stackrel{def}{=} (repair, g_1 \times r).output$$

where $f_1(x_1, \ldots, x_N) = \begin{cases} 1 \; if \;\; x_i = input'_i, \;\; \forall i = 1, .., N \\ 0 \; otherwise \end{cases}$

and $g_1(x_1, \ldots, x_N) = \begin{cases} 1 \; if \;\; \exists i, i = 1 \ldots, N \; / \; x_i = input_i \\ 0 \; otherwise \end{cases}$

The whole gate is modelled as: $G\_and \stackrel{def}{=} output||(input_1||input_2||...||input_N)$

OR Gate: as the output of this gate fails if any of its input events fails, it is modelled as follows:

$$output \stackrel{def}{=} (failure, f_2 \times a).output' \qquad\qquad output' \stackrel{def}{=} (repair, g_2 \times r).output$$

where $f_2(x_1, \ldots, x_N) = \begin{cases} 1 \; if \;\; \exists i, i = 1 \ldots, N \; / \; x_i = input'_i \\ 0 \; otherwise \end{cases}$

and $g_2(x_1, \ldots, x_N) = \begin{cases} 1 \; if \;\; x_i = input_i, \;\; \forall i = 1, \ldots, N \\ 0 \; otherwise \end{cases}$

The whole gate equation is: $G\_or \stackrel{def}{=} output||(input_1||input_2||...||input_N)$

Remark: One may assume that the gate fails immediately after the failure of one of its inputs, with no time elapsing. In this case, we need to replace the functional rate in activity $(failure, f_2 \times a)$ with the unspecified rate $(\top)$ and to make each component $input_i$ and $output$ synchronise on action type $failure$.

K/N Gate: as the failure and the repair of this gate depend both on the number of entries down, both corresponding activities have a functional rate as follows:

$$output \stackrel{def}{=} (failure, f_3 \times a).output' \qquad output' \stackrel{def}{=} (repair, g_3 \times r).output$$

where $f_3(x_1, \ldots, x_N) = \begin{cases} 1 \; if \; \exists i_1, \ldots, i_K \; / \; x_{i_1} = input'_{i_1}, \ldots, \; x_{i_K} = input'_{i_K} \\ 0 \; otherwise \end{cases}$

and $g_3(x_1, \ldots, x_N) = \begin{cases} 1 \; if \; \exists i_1, \ldots, i_{N-K+1} / x_i = input_i \;\; \forall i = i_1, \ldots, i_{N-K+1} \\ 0 \; otherwise \end{cases}$

The whole gate equation is: $G\_kn \stackrel{def}{=} output||(input_1||input_2|| \ldots ||input_N)$

4.2 Dynamic Gates

Modelling formally (or informally) dynamic gates, either in isolation or in the context of a DFT configuration is not always obvious because the semantics of these gates is not always clearly defined. Therefore, each time the intended behaviour of a gate is not clear, we discuss the different options we have and the one we adopt when modelling the gate.

PAND Gate: The semantics of PAND gate is not clear about the exact meaning of the term *"in order"*; whether it is a strict ordering or not. We have chosen to implement the strict version. Moreover, we do not consider the case of basic event replicates because the semantics related to the order of their failure remains unclear. It is even not clear if the failure of a replicate should be considered to be in order or out of order. The semantics chosen in [7], one of the rare works discussing this problem, is that the input events fail in order if all the replicates in a position, say i, fail in order with respect to all the replicates in position i. In the literature basic event replicates are in general just ignored. In our model, we distinguish between two cases: the one in which the inputs fail in the right order (from left to right) and the one in which the order is not respected. These are modelled using action types $failure_i$, $i = 1, \ldots, N$, and $failure$ respectively.

$$input_i \stackrel{def}{=} (failure_i, h_i \times a_i).input'_i + (failure, \overline{h}_i \times b_i).input'_i$$
$$input'_i \stackrel{def}{=} (repair, r_i).input_i$$

where $h_i(x_1,\ldots,x_N) = \begin{cases} 1\ if\ \ \forall j < i, \quad 1 \leq i,j \leq N, \ \ x_j = input'_j \\ 0\ otherwise \end{cases}$

and $\overline{h}_i(x_1,\ldots,x_N) = \begin{cases} 1\ if\ \ \exists j < i, \quad 1 \leq i,j \leq N, \ \ x_j = input_j \\ 0\ otherwise \end{cases}$

whereas $h_1 = 1$ and $\overline{h}_1 = 0$, for all states of the other basic items of the gate. Assuming that the increasing numbering of the inputs matches their order from left to right, the component modelling the gate's output event is the following:

$$\begin{array}{ll} output & \stackrel{def}{=} (failure_1, \top).output_1 \\ output_1 & \stackrel{def}{=} (failure_2, \top).output_2 \\ & \vdots \\ output_{N-1} & \stackrel{def}{=} (failure_N, \top).output_N \\ output_N & \stackrel{def}{=} (repair, g_4 \times r).output \end{array}$$

The failure state of the gate is $output_N$. We assume that the gate is repaired only when all its inputs are repaired. This is modelled using function g_4 such as:

$$g_4(x_1,\ldots,x_N) = \begin{cases} 1\ if\ \ x_i = input_i\ \ \forall i = 1,\ldots,N \\ 0\ otherwise \end{cases}$$

Given that $L = \{failure_1,\ldots,failure_N\}$, the whole gate equation is:

$$G\_pand \stackrel{def}{=} output \underset{L}{\bowtie} (input_1||input_2||\ldots||input_N)$$

FDEP Gate: Going through the literature, one can notice that the semantics to apply in some DFT configurations remains unclear when FDEP gate is used. For example, consider the first configuration in Fig. 2 where the FDEP gate triggers two basic input events to gate PAND. Two semantics may be used in this case. In the first one the trigger event does not have a simultaneous effect on the dependent events which thus fail individually. And according to the order of their failure, the PAND gate will fail or not. With the second semantics, the trigger event has a simultaneous effect and the dependent events fail immediately (with no time elapsing) and at the same time. In this case, the PAND gate may be considered as still functional.

In the second configuration of Fig. 2, the FDEP gate triggers two basic primary input events to two spare gates sharing the same spare item. Like in the previous configuration, two interpretations can be used. If we allow the primary inputs to fail individually, following the arrival of the trigger event, then the spare gate whose primary input fails first will have the possibility to use the spare item. However if the primary inputs fail simultaneously, it is not clear which spare gate will use the spare item.

To solve the problem, either the non-determinism is systematically replaced by a probabilistic choice like in [7] or it is allowed and each time detected all combinations of ordering is considered [2]. In our case, we consider that the trigger event has an immediate and simultaneous effect on all dependent inputs,

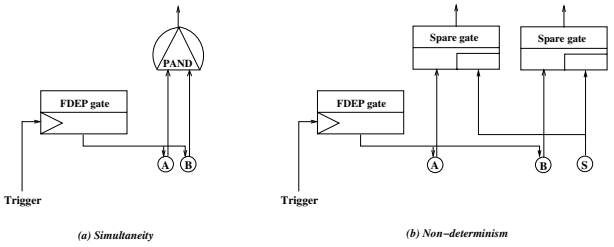


Fig. 2. Simultaneity and non-determinism

and this for all DFT configurations. The non-determinism that may result in configurations like the one in Fig. 2(b) is solved by the race condition governing the dynamic behaviour of a PEPA model whenever more than one activity is enabled.

In our model, the trigger event is represented using component $Trigger$ and its arrival, action type $arrival_{trigger}$. The synchronisation of this component with components $input_i$, $i = 1, \ldots, N$, on $arrival_{trigger}$ puts simultaneously all inputs in the failure state. This is modelled using $failure_{all}$. As previously, the action modelling the case where the inputs fail individually is $failure_i$.

$$Trigger \stackrel{def}{=} (arrival_{trigger}, \mu).Trigger$$

$$\begin{aligned} input_i &\stackrel{def}{=} (arrival_{trigger}, \top).input_i' + (failure_i, a_i).input_i'' \\ input_i' &\stackrel{def}{=} (failure_{all}, b_i).input_i'' \\ input_i'' &\stackrel{def}{=} (repair, r_i).input_i + (arrival_{trigger}, \top).input_i'' + (failure_{all}, b_i).input_i'' \end{aligned}$$

The two last activities of derivative $input_i''$ are required only in the case of the arrival of the trigger event while the input item has already failed. As the output of FDEP is dummy and thus is not taken into consideration during the computation, we do not need to model it. Given that $L = \{arrival_{trigger}\}$ and $L_1 = \{arrival_{trigger}, failure_{all}\}$, the complete gate equation is:

$$G\_fdep \stackrel{def}{=} Trigger \underset{L}{\bowtie} (input_1 \underset{L_1}{\bowtie} \ldots \underset{L_1}{\bowtie} input_N)$$

Spare Gate: in such a gate, there is always a primary input and one or several spare items. Components $input\_p$ and $input\_s_i$, $i = 1, \ldots, N$, model these entries respectively. As a spare input is initially in a dormant state, component $input\_s_i$ needs to be activated by the gate. This is modelled using action type $active_i$. Moreover as the same spare input may fail while it is in the dormant state, we need to distinguish between failing when being active and failing when being dormant. In the latter case the dormancy factor α must be considered in the rate of the corresponding activity.

$$input\_p \stackrel{def}{=} (failure_p, a_p).input\_p' \qquad input\_s_i \stackrel{def}{=} (active_i, \top).input\_s_i'' + (failure_i, \alpha \times a_i).input\_s_i'$$
$$input\_p' \stackrel{def}{=} (repair_p, r_p).input\_p \qquad input\_s_i' \stackrel{def}{=} (repair_i, r_i).input\_s_i$$
$$input\_s_i'' \stackrel{def}{=} (failure_i, a_i).input\_s_i'$$

After the failure of the primary input, the gate activates the spare entries one at once and if the one activated fails or is already in the failure state, the gate considers the next spare. Assuming that the increasing numbering of the spare items matches the order in which they are activated, the component modelling the output event of the gate is the following:

$$\begin{aligned}
output &\stackrel{def}{=} (failure_p, \top).output_1 \\
output_1 &\stackrel{def}{=} (active_1, c_1).output_1 + (failure, \beta_1 \times a).output_2 \\
output_2 &\stackrel{def}{=} (active_2, c_2).output_2 + (failure, \beta_2 \times a).output_3 \\
&\vdots \\
output_N &\stackrel{def}{=} (active_N, c_N).output_N + (failure, \beta_N \times a).output_{N+1} \\
output_{N+1} &\stackrel{def}{=} (repair, g_5 \times r).output
\end{aligned}$$

where $0 \leq \alpha \leq 1$ and $\forall i = 1, 2, \ldots, N$,

$$\beta_i(x_i) = \begin{cases} 1 \; if \;\; x_i = input\_s_i' \; or \; x_i = input\_s_i'' \\ 0 \; otherwise \end{cases}$$

and

$$g_5(x_p, x_1, \ldots, x_N) = \begin{cases} 1 \; if \;\; x_p = input\_p \;\; and \;\; x_i = input\_s_i, \forall i = 1, .., N \\ 0 \; otherwise \end{cases}$$

With function $\beta_i(x_i)$, we specify the case where the spare item fails during its dormancy state ($x_i = input\_s_i'$) and the case where it is unavailable ($x_i = input\_s_i''$). If $L = \{active_1, \ldots, active_N, failure_p\}$, the system equation is:

$$G\_spare \stackrel{def}{=} output \underset{L}{\bowtie} (input\_p || input\_s_1 || input\_s_2 || \ldots || input\_s_N)$$

4.3 The Repair Process

To the best of our knowledge, the repair process of dynamic gates has not been taken into consideration in the models developed in the literature. There are two good reasons for that. Firstly, so far, the objective of modelling DFTs, formally or not, is to assess the chances the underlying systems have to fail. The second reason is related to the lack of clearness in the semantics to apply in each case.

Obviously, the repair of a dynamic gate is related to the repair of its individual inputs. Thus, in our models, we allow a gate's inputs to be repaired at any stage of the failure process of that gate. However, the only stage at which a repaired input is taken into account is when the component modelling the gate's output event (*output*) reaches its defined failure state.

5 Example

We consider a cardiac system which consists of three separate modules: the CPU unit, the motor unit and the pump unit. The CPU unit is composed of a primary CPU (P_CPU) and a warm spare CPU (S_CPU). Both CPUs depend on a cross switch and a system supervisor. The motor unit has two motors, a primary motor (P_Motor) and a shared cold spare one (S_Motor). The switching component (M_Switch) turns on the spare motor when the primary fails. The pump unit consists of three pumps: two primary pumps (Prima_A and Prima_B) and a cold spare pump (S_Pump).

This system can be modelled using the DFT given in Fig. 3. In order to translate automatically the DFT configuration into a PEPA model, and to avoid any ambiguity in the derivation of the model, in particular the synchronisation sets, the following actions are undertaken:

- All action types are indexed with the name of the device or the gate in which the actions occur.
- If L is the number of levels in the tree and K_j is the number of devices at j, we define $x_{j,k}$ as the current state of device k at level j, $0 \leq j \leq L$ and $1 \leq k \leq K_j$. Thus $x_{0,k}$ is the current state of device k at level 0 (tree leaves).

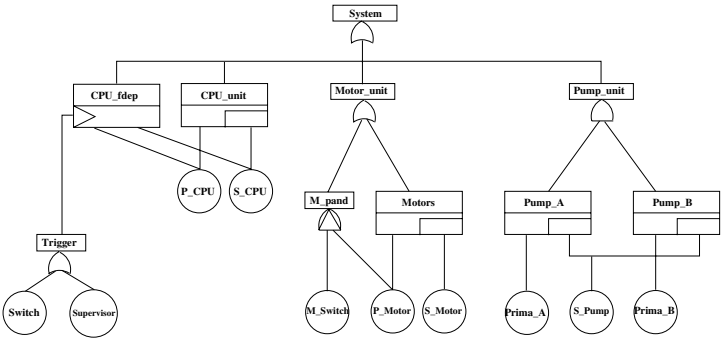


Fig. 3. DFT modelling the cardiac system

The PEPA model of the DFT in Fig. 3 consists of the interaction of three main sub-systems: the CPU unit, the motor unit and the pump unit.

1. The CPU unit: Both the cross switch and the system supervisor are the basic inputs to OR gate and as such have the following basic behaviour

$$\begin{aligned} Switch &\stackrel{def}{=} (failure_{Switch}, a_1).Switch' \\ Switch' &\stackrel{def}{=} (repair_{Switch}, r_1).Switch \end{aligned}$$

$$\begin{aligned} Supervisor &\stackrel{def}{=} (failure_{Supervisor}, a_2).Supervisor' \\ Supervisor' &\stackrel{def}{=} (repair_{Supervisor}, r_2).Supervisor \end{aligned}$$

The output of OR gate is the trigger event for the FDEP gate. This output event is conditioned by the input events as follows:

$$Trigger \stackrel{def}{=} (arrival_{trigger}, f_1 \times a_3).Trigger$$

where $$f_1(x_{0,1}, x_{0,2}) = \begin{cases} 1 \; if \;\; x_{0,1} = Switch' \;\; or \;\; x_{0,2} = Supervisor' \\ 0 \; otherwise \end{cases}$$

The primary CPU is an input item to both the spare gate (CPU_unit) and the FDEP gate. Thus, the component modelling P_CPU takes into consideration both behaviours.

$$\begin{array}{l} P\_CPU \stackrel{def}{=} (arrival_{trigger}, \top).P\_CPU' + (failure_{P\_CPU}, a_4).P\_CPU'' \\ P\_CPU' \stackrel{def}{=} (failure_{all}, b_1).P\_CPU'' \\ P\_CPU'' \stackrel{def}{=} (repair_{P\_CPU}, r_4).P\_CPU + (arrival_{trigger}, \top).P\_CPU'' \\ \qquad + (failure_{all}, b_1).P\_CPU'' \end{array}$$

Similarly the component modelling the spare CPU takes into consideration the behaviour of this item both as an input to the spare and the FDEP gates.

$$\begin{array}{l} S\_CPU \stackrel{def}{=} (arrival_{trigger}, \top).S\_CPU' + (active_{S\_CPU}, \top).S\_CPU''' \\ \qquad + (failure_{S\_CPU}, \alpha \times a_5).S\_CPU'' \\ S\_CPU' \stackrel{def}{=} (failure_{all}, b_2).S\_CPU'' \\ S\_CPU'' \stackrel{def}{=} (repair_{S\_CPU}, r_5).S\_CPU + (arrival_{trigger}, \top).S\_CPU'' \\ \qquad + (failure_{all}, b_3).S\_CPU'' \\ S\_CPU''' \stackrel{def}{=} (failure_{S\_CPU}, a_6).S\_CPU'' + (arrival_{trigger}, \top).S\_CPU' \end{array}$$

Because S_CPU is a warm spare unit, $0 < \alpha < 1$. Finally the PEPA component modelling the output of the CPU unit, that is the spare gate is as follows:

$$\begin{array}{l} CPU\_unit \stackrel{def}{=} (failure_{P\_CPU}, \top).CPU\_unit_1 + (failure_{all}, \top).CPU\_unit_2 \\ CPU\_unit_1 \stackrel{def}{=} (active_{S\_CPU}, c_1).CPU\_unit_1 + (failure, \beta_1 \times a_7).CPU\_unit_2 \\ \qquad + (failure_{all}, \top).CPU\_unit_2 \\ CPU\_unit_2 \stackrel{def}{=} (repair, g_2 \times r_6).CPU\_unit + (failure_{all}, \top).CPU\_unit_2 \end{array}$$

where functions g_2 and β_1 are defined as follows:

$$g_2(x_{0,3}, x_{0,4}) = \begin{cases} 1 \;\; if \;\; x_{0,3} = P\_CPU \\ \quad and \;\; x_{0,4} = S\_CPU \\ 0 \;\; otherwise \end{cases} \qquad \beta_1(x_{0,4}) = \begin{cases} 1 \;\; if \;\; x_{0,4} = S\_CPU'' \\ \quad or \;\; x_{0,4} = S\_CPU''' \\ 0 \;\; otherwise \end{cases}$$

If $L_1 = \{arrival_{trigger}, failure_{all}\}$, $L_2 = \{failure_{P\_CPU}, active_{S\_CPU}, failure_{all}\}$, and $L_3 = \{arrival_{trigger}\}$, the CPU unit sub-system equation is given by:

$$S\_system_1 \stackrel{def}{=} \Big((P\_CPU \underset{L_1}{\bowtie} S\_CPU) \underset{L_2}{\bowtie} CPU\_unit \Big) \underset{L_3}{\bowtie} (Trigger || Switch || Supervisor)$$

The motor unit: It consists of a PAND gate and a spare gate. Using the basic PEPA model for a PAND gate, we have:

$$\begin{aligned}
M_Switch &\stackrel{def}{=} (failure_{M_Switch1}, h_1 \times a_7).M_Switch' \\
&\quad + (failure_{M_Switch2}, \overline{h}_1 \times b_4).M_Switch' \\
M_Switch' &\stackrel{def}{=} (repair_{M_Switch}, r_7).M_Switch \\
\\
P_Motor &\stackrel{def}{=} (failure_{P_Motor1}, h_2 \times a_8).P_Motor' \\
&\quad + (failure_{P_Motor2}, \overline{h}_2 \times b_5).P_Motor' \\
P_Motor' &\stackrel{def}{=} (repair_{P_Motor}, r_8).P_Motor \\
\\
M_pand_1 &\stackrel{def}{=} (failure_{M_Switch_1}, \top).M_pand_2 \\
M_pand_2 &\stackrel{def}{=} (failure_{P_Motor_1}, \top).M_pand_3 \\
M_pand_3 &\stackrel{def}{=} (repair, g_3 \times r_9).M_pand_1
\end{aligned}$$

where $h_1 = 1$, by definition, and thus $\overline{h}_1 = 0$. Moreover we have

$$h_2(x_{0,5}) = \begin{cases} 1 \; if \;\; x_{0,5} = M\_Switch' \\ 0 \; otherwise \end{cases} \quad and \;\; \overline{h}_2(x_{0,5}) = \begin{cases} 1 \; if \;\; x_{0,5} = M\_Switch \\ 0 \; otherwise \end{cases}$$

and $$g_3(x_{0,5}, x_{0,6}) = \begin{cases} 1 \; if \;\; x_{0,5} = M\_Switch \;\; and \;\; x_{0,6} = P\_Motor \\ 0 \; otherwise \end{cases}$$

Now we need to model the remaining component of the spare gate, that is the spare item S_Motor.

$$\begin{aligned}
S_Motor &\stackrel{def}{=} (active_{S_Motor}, \top).S_Motor'' + (failure_{S_Motor}, \alpha \times a_9).S_Motor' \\
S_Motor' &\stackrel{def}{=} (repair_{S_Motor}, r_{10}).S_Motor \\
S_Motor'' &\stackrel{def}{=} (failure_{S_Motor}, a_{10}).S_Motor'
\end{aligned}$$

As the second motor is a cold spare one, that is it cannot fail while it is dormant ($\alpha = 0$), action type $failure_{S\_Motor}$ can be omitted. Finally the output of the spare gate $Motors$ is modelled as follows:

$$\begin{aligned}
Motors &\stackrel{def}{=} (failure_{P_Motor_1}, \top).Motors_1 + (failure_{P_Motor_2}, \top).Motors_1 \\
Motors_1 &\stackrel{def}{=} (active_{S_Motor}, c_2).Motors_1 + (failure, \beta_2 \times a_{11}).Motors_2 \\
Motors_2 &\stackrel{def}{=} (repair, g_4 \times r_{11}).Motors
\end{aligned}$$

where

$$\beta_2(x_{0,7}) = \begin{cases} 1 \; if \;\; x_{0,7} = S\_Motor' \\ \quad or \;\; x_{0,7} = S\_Motor'' \;\; and \;\; g_4(x_{0,6}, x_{0,7}) = \begin{cases} 1 \; if \;\; x_{0,6} = P\_Motor \\ \quad and \;\; x_{0,7} = S\_Motor \end{cases} \\ 0 \; otherwise \end{cases} \quad 0 \; otherwise$$

Finally the output of the whole motor unit is modelled as follows:

$$\begin{aligned}
Motor_unit &\stackrel{def}{=} (failure, f_2 \times a_{12}).Motor_unit_1 \\
Motor_unit_1 &\stackrel{def}{=} (repair, g_5 \times r_{12}).Motor_unit
\end{aligned}$$

where $f_2(x_{1,2}, x_{1,3}) = \begin{cases} 1\ if\ \ x_{1,2} = M\_pand_3\ \ or\ \ x_{1,3} = Motors_2 \\ 0\ otherwise \end{cases}$

and $g_5(x_{1,2}, x_{1,3}) = \begin{cases} 1\ if\ \ x_{1,2} \neq M\_pand_3\ \ and\ \ x_{1,3} \neq Motors_2 \\ 0\ otherwise \end{cases}$

If $M_1 = \{active_{S\_Motor}, failure_{P\_Motor_1}, failure_{P\_Motor_2}\}$,
$M_2 = \{failure_{P\_Motor_1}\}$ and $M_3 = \{failure_{M\_Switch_1}\}$, the equation of the motor unit sub-system is:

$$S\_system_2 \stackrel{def}{=} Motor\_unit \,||\, \Big(\big(Motors \underset{M1}{\bowtie} (P\_Motor||S\_Motor)\big) \underset{M_2}{\bowtie} (M\_pand_1 \underset{M_3}{\bowtie} M\_Switch)\Big)$$

The Pump Unit: It consists of two primary pumps ($Prima\_A$, $Prima\_B$) and a cold spare one (S_pump). S_pump is available to replace both primary pumps.

$$Prima\_A \stackrel{def}{=} (failure_{Prima\_A}, a_{11}).Prima\_A'$$
$$Prima\_A' \stackrel{def}{=} (repair_{Prima\_A}, r_{11}).Prima\_A$$

$$Prima\_B \stackrel{def}{=} (failure_{Prima\_B}, a_{12}).Prima\_B'$$
$$Prima\_B' \stackrel{def}{=} (repair_{Prima\_B}, r_{12}).Prima\_B$$

$$S\_pump \stackrel{def}{=} (active_{S\_pump}, \top).S\_pump'' + (failure_{S\_pump}, \alpha \times a_{13}).S\_pump'$$
$$S\_pump' \stackrel{def}{=} (repair_{S\_pump}, r_{13}).S\_pump$$
$$S\_pump'' \stackrel{def}{=} (failure_{S\_pump}, a_{14}).S\_pump'$$

Since the spare pump is a cold one ($\alpha = 0$), activity $(failure_{S\_pump}, \alpha \times a_{13})$ can be omitted. Both spare gates $Pump\_A$ and $Pump\_B$ are modelled as:

$$Pump\_A \stackrel{def}{=} (failure_{Prima\_A}, \top).Pump\_A_1$$
$$Pump\_A_1 \stackrel{def}{=} (active_{S\_pump}, c_3).Pump\_A_1 + (failure, \beta_3 \times b_6).Pump\_A_2$$
$$Pump\_A_2 \stackrel{def}{=} (repair, g_6 \times r_{14}).Pump\_A$$

$$Pump\_B \stackrel{def}{=} (failure_{Prima\_B}, \top).Pump\_B_1$$
$$Pump\_B_1 \stackrel{def}{=} (active_{S\_pump}, c_4).Pump\_B_1 + (failure, \beta_4 \times b_7).Pump\_B_2$$
$$Pump\_B_2 \stackrel{def}{=} (repair, g_7 \times r_{15}).Pump\_B$$

where $\beta_3(x_{0,9}) = \beta_4(x_{0,9}) = \begin{cases} 1\ if\ \ x_{0,9} = S\_pump'\ \ x_{0,9} = S\_pump'' \\ 0\ otherwise \end{cases}$

$$g_6(x_{0,8}, x_{0,9}) = \begin{cases} 1\ if\ \ x_{0,8} = Prima\_A\ \ and\ \ x_{0,9} = S\_pump \\ 0\ otherwise \end{cases}$$

and

$$g_7(x_{0,9}, x_{0,10}) = \begin{cases} 1\ if\ \ x_{0,9} = S\_pump\ \ and\ \ x_{0,10} = Prima\_B \\ 0\ otherwise \end{cases}$$

The whole pump unit fails if both $Pump\_A$ and $Pump\_B$ are in the failure state. This is represented using gate AND in the tree and is modelled as follows:

$$Pump\_unit \stackrel{def}{=} (failure, f_3 \times a_{15}).Pump\_unit_1$$
$$Pump\_unit_1 \stackrel{def}{=} (repair, g_8 \times r_{16}).Pump\_unit$$

where $f_3(x_{1,4}, x_{1,5}) = \begin{cases} 1\ if\ \ x_{1,4} = Pump\_A_2\ \ and\ \ x_{1,5} = Pump\_B_2 \\ 0\ otherwise \end{cases}$

and $g_8(x_{1,4}, x_{1,5}) = \begin{cases} 1\ if\ \ x_{1,4} \neq Pump\_A_2\ \ or\ \ x_{1,5} \neq Pump\_B_2 \\ 0\ otherwise \end{cases}$

The whole pump unit sub-system equation is given by:

$$S\_system_3 \stackrel{def}{=} (Pump\_unit || S\_pump) \underset{N_1}{\bowtie} \big((Pump\_A \underset{N_2}{\bowtie} Prima\_A) \ ||\ (Pump\_B \underset{N_3}{\bowtie} Prima\_B) \big)$$

where $N_1 = \{active_{S\_pump}\}$, $N_2 = \{failure_{Prima\_A}\}$, $N_3 = \{failure_{Prima\_B}\}$.

The whole cardiac systems behaviour depends on the output of OR gate over the defined sub-systems.

$$System \stackrel{def}{=} (failure, f_4 \times \lambda).System_1 \qquad System_1 \stackrel{def}{=} (repair, g_9 \times \mu).System$$

where $f_4(x_{2,2}, x_{2,3}, x_{2,4}) = \begin{cases} 1\ if\ \ x_{2,2} = CPU\_unit_2,\ or\ x_{2,3} = Motor\_unit_2 \\ \quad or\ x_{2,4} = Pump\_unit_1 \\ 0\ otherwise \end{cases}$

and $g_9(x_{2,2}, x_{2,3}, x_{2,4}) = \begin{cases} 1\ if\ \ x_{2,2} \neq CPU\_unit_2,\ and\ x_{2,3} = Motor\_unit \\ \quad and\ x_{2,4} = Pump\_unit \\ 0\ otherwise \end{cases}$

The cardiac system equation is:

$$Cardiac\_sys \stackrel{def}{=} System || S\_system_1 || S\_system_2 || S\_system_3$$

6 Related Work

Several works are dedicated to the mapping of DFTs into Bayesian networks, among which [3,4,5]. In [4], DFTs are mapped into discrete-time Bayesian networks. Such an approach has high memory needs as multi-dimensional tables have to be used for (conditional) probabilities. Moreover, the derived solution is approximate. A continuous-time Bayesian network framework for dynamic systems reliability analysis is presented in [5]. A closed-form solution is derived.

Only few works investigate the mapping of DFTs into high-level modelling formalisms. In [6] a DFT is mapped into a GSPN by means of graph transformation rules. No proof on the correctness of the transformation is derived. In [2], I/O interactive Markov chain formalism is used to analyse DFT. The restriction on spare and FDEP gates' dependent events to be basic events is lifted. Unlike prior works, the repair process is discussed. However the discussion is restricted to the case of basic events and AND gate.

DIFTree, a fault tree methodology for the analysis of DFTs is presented in [8,12]. It combines solution techniques based on BDD and Markov chains. In both papers, the authors do not present the type of gates considered, nor the semantics applied in each case.

In [7], the authors use an approach based on formal specification (Z notation) and denotational semantics techniques from software engineering and programming language design. Several constraints on the input events are considered.

7 Summary and Conclusion

In this paper, we propose to exploit the compositional feature of both PEPA and DFTs in order to develop an automatic translation of a dynamic fault tree into a PEPA model. The purpose of such model-to-model transformation is to take advantage of a wide-range of tools and techniques, such as the aggregation and the compact representation of the generator of the underlying Markov chain, that support PEPA. Clearly DFTs may exhibit symmetries because of the use of several copies of the same gates. As each gate is modelled using a PEPA component, the aggregation technique can be used to reduce the size of the underlying Markov chain. In the future we seek to investigate the relation between a DFT and the tensorial representation of the underlying Markov chain. We believe that such investigations may lead to interesting results.

References

1. Amari, S., Dill, G., Howard, E.: A New Approach to Solve Dynamic Fault Trees. In: Annual Reliability and Maintainability Symposium, Tampa Florida, January 27–30 (2003)
2. Boudali, H., Crouzen, P., Stoelinga, M.: Dynamic Fault Tree analysis using Input/Output Interactive Markov Chains. In: DSN 2007 (2007)
3. Boudali, H., Dugan, J.B.: A New Bayesian Network Approach to Solve Dynamic Fault Tree. In: Annual Reliability and Maintainability Symposium, January 24-27 (2005)
4. Boudali, H., Dugan, J.B.: A discrete-time Bayesian network reliability modeling and analysis framework. Reliability Engineering & System Safety 87, 337–349 (2005)
5. Boudali, H., Dugan, J.B.: A Continuous-Time Bayesian Network Reliability Modeling and Analysis Framework. IEEE Transactions on Reliability 55(1) (March 2006)
6. Codetta-Raiteri, D.: The Conversion of Dynamic Fault Trees to Stochastic Petri Nets, as a case of Graph Transformation. ENTCS 127, 45–60 (2005)
7. Coppit, D., Sullivan, K.J., Dugan, J.B.: Formal Semantics of Models for Computational Enginnering: A Case Study on Dynamic Fault Trees. In: Proc. of the 11th IEEE International Symposium on Software Reliability Engineering, San Jose, October 8–11 (2000)
8. Dugan, J.B., Venkataraman, B., Gulati, R.: DIFtree: A software package for the analysis of dynamic fault tree models. In: Annual Reliability and Maintainability Symposium, Philadelphia (January 1997)

9. Hillston, J.: A Compositional Approach to Performance Modelling, PhD. Thesis, University of Edinburgh (1994)
10. Hillston, J., Kloul, L.: An Efficient Kronecker Representation for PEPA Models. In: de Luca, L., Gilmore, S. (eds.) PROBMIV 2001, PAPM-PROBMIV 2001, and PAPM 2001. LNCS, vol. 2165, pp. 120–135. Springer, Heidelberg (2001)
11. Limnios, N.: Arbres de Defaillance. Hermes, Paris (1991)
12. Manian, R., Dugan, J.B., Coppit, D., Sullivan, K.J.: Combining Various Solution Techniques for Dynamic Fault Tree Analysis of Computer Systems. In: Proc. of the 3rd IEEE High-Assurance Systems Engineering Symposium, Washington, DC, November 13–14 (1998)
13. Schneeweiss, W.G.: The Fault Tree Method. LiLoLe, Hagen (1999)

Passage-End Analysis

Allan Clark, Adam Duguid, and Stephen Gilmore

The University of Edinburgh, Scotland

Abstract. Passage-end calculations are a new style of passage measurement for eXtended Stochastic Probes (XSP) [1] which add the ability to split the analysis into several cases depending on conditions which hold at the end of a passage. This makes it possible to separate successful responses to a request from negative responses, timeouts or other failures. This allows the expression of service level agreements such as: "At least 90 percent of all requests receive a response within 10 seconds and at least 60 percent of all such requests are successful."

1 Introduction

Many systems which are analysed for response-time profiles have more than one way in which the passage in question may terminate. Commonly a request from a client may end in a successful completion such that the client received the desired service but may also end in failure due to a timeout or rejection. There may also be two or more ways in which the request can be satisfied, for example a data retrieval request may be serviced by accessing the cache or the disk. In these situations we wish to analyse the passage-time profiles for the separate cases of success or failure, and cache or disk retrieval. It may be that the general response-time analysis indicates acceptable performance but that successful requests are serviced too slowly.

A stochastic probe [2,3] is a component added to a model in order to make reasoning about the model more convenient. In the context of PEPA [4] a stochastic probe is a single sequential component which observes the activities of a PEPA model.

For example we may analyse the response-time of a service as observed by a single client. Once this is known, in order to provide more information on how this may be improved, we then analyse the response-time of all requests which are made when the service is definitely not broken. We would expect this to be better than the more general case of all requests. Conversely we may also analyse the response-time for all requests made when the service certainly is broken and expect this to be worse than for the general case. From this we may determine whether, in order to improve response-time, it is better to repair the server more quickly or make the server more reliable such that it breaks less often.

Although with eXtended Stochastic Probes splitting the measurement of a passage with respect to the starting conditions is convenient it is not clear how one may split a passage based on how the passage completes — or based on some event during the passage. The novelty in this paper is the use of several

J.T. Bradley (Ed.): EPEW 2009, LNCS 5652, pp. 110–115, 2009.

absorbing states, one for each target event. For a passage-end calculation the user must specify a list of target actions. These may be actions performed by the model itself or communication signals sent by user defined probes. During transient analysis each target state is modified by adding a transition to a distinct absorbing state based on the causal event.

As an example, consider analysing the response-time of a service which begins with a request but may be concluded with either a cached response or a networked response. It is not possible to simply measure these two passages with separate runs using a probe such as: *request*:start,*cached*:stop. The reason is that this will compute the probability of completing the passage at (or within) time t via a cached response plus the probability of completing the passage via a networked response and then restarting and completing the passage via the cached response all at (or within) time t. Indeed you may complete the passage twice, three times or any number of times via the networked response before finally completing via the cached response.

Using the same algorithm we can calculate the raw pdf and cdf of the passage from the request to the cached response. In this case the cdf will not tend to one but to the fraction of requests which are ultimately serviced by the cache. Similarly for the area under the pdf and for both functions of the request to networked passage. We can normalise these graphs based on the probability of completing at or within the given time t via any target event.

We can instead normalise the raw pdf and cdf by dividing through the probability of completing at (or within) time t by the percentage of all requests which are ultimately serviced by the target event in question. We can know this percentage by calculating enough hops such that sufficiently close to all of the probability mass at π^N is in one of the absorbing states. We may then take the probability of being in the appropriate absorbing state at π^N.

Hence using a passage-end calculation it is possible to calculate:

- The probability of completing a passage by a cached response at or within a given time.
- The probability that, assuming the passage completes at or within a given time in some way, that it does so via the cached response. This answers such questions as: "What percentage of responses received within 5 seconds are received via the cache/network?"
- The cdf and pdf profiles for all requests which are serviced by the cache.

In the above "cached" may be replaced by "networked" or any other target event of the given passage, including probe communication signals. The second kind of question is helpful in evaluating some SLAs (Service-Level Agreements), particularly for services which may end in success, failure or cancellation. For example the SLA may say that ninety percent of requests receive a response within 10 seconds. We may analyse the model and find that this is indeed the case but a passage-end analysis reveals that eighty-nine percent of such requests are rejection/failure responses. Hence we may wish to modify our SLA to state that ninety percent of all successful requests receive a response within 10 seconds.

2 Example of Passage-End Analysis

We consider an example of passage-end analysis applied to emergency response service quality [5]. The scenario centres on the function of an automatic crash response subscription service such as OnStar [6] from General Motors. These utilise multiple built-in sensors which capture critical details in the event of a car crash. The built-in communications module automatically contacts the OnStar service and relays the information obtained from the sensors, including location information from the on-board GPS. The service attempts to contact the driver to ask if they need help. If the service cannot contact the driver then they send medical assistance to the car's location.

The PEPA model in Figure 1 represents the scenario where the protocol for attempting to call the driver is to try three times. After three unsuccessful attempts to contact the driver the system assumes that the driver is unavailable and at this point must decide on the basis of the car telemetry whether to dispatch an ambulance. We are interested then in the time between the airbag deploying and either a successful or failed attempt to contact the driver. The start event of our passage is then the *airbag* activity and the two ways in which a passage may terminate is with an *answer* or a $timeout_3$ activity. Note that the timeout activities are distinguished such that only the third timeout will end the passage.

$$
\begin{aligned}
Car_1 &\stackrel{def}{=} (airbag, r_1).Car_2 \\
Car_2 &\stackrel{def}{=} (reportToService, r_2).Car_3 \\
Car_3 &\stackrel{def}{=} (processReport, r_3).Car_4 \\
Car_4 &\stackrel{def}{=} (callDriver, r_4).Car_5 \\
Car_5 &\stackrel{def}{=} (timeout_1, p * r_5).Car_6 \\
&\quad + (answer, (1-p) * r_5).OK \\
Car_6 &\stackrel{def}{=} (timeout_2, p * r_5).Car_7 \\
&\quad + (answer, (1-p) * r_5).OK \\
Car_7 &\stackrel{def}{=} (timeout_3, p * r_5).Rescue \\
&\quad + (answer, (1-p) * r_5).OK \\
OK &\stackrel{def}{=} (finish, r_1).Car_1 \\
Rescue &\stackrel{def}{=} (rescue, r_6).Car_1
\end{aligned}
$$

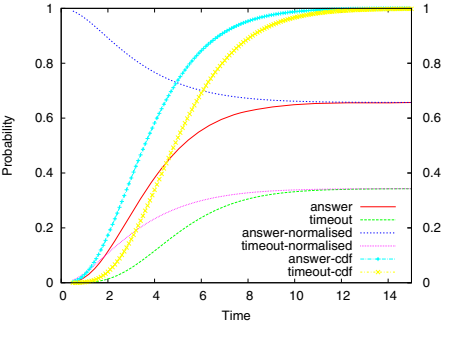


Fig. 1. The model of the driver contact protocol within the airbag crash assistance case scenario. The initial state of this process is Car_1. The graph shows results of passage-end analysis of driver contact retries.

The results from our passage-end analysis of the model are shown in the graph in Figure 1. The first two lines to consider are the 'answer' and 'timeout' lines, these plot the unmodified cumulative distribution functions of completing the passage via the driver answering the phone or a third timeout occurring respectively. Note that neither of these two lines tend to one as is usual for a cdf of a passage. This is because the passage may never end in the prescribed

ways. By looking at the long-term probabilities, or the limits of these two lines (which sum together to one) we can say what percentage of airbag initiations end in the driver answering the phone (and conversely a third timeout occurring). From these two lines we can also answer the question: "What is the probability that the driver answers the phone a given time after the airbag was deployed, regardless of whether the driver is capable of answering at any time".

The next two lines to consider are the above cdf functions normalised by dividing the probability (in each case) at each time by the probability of completing the passage within the given time in any way. These lines tend to the same value in the limit as before because eventually there is a probability of very close to one that we will have completed the passage in some way within that time. These lines allow us to answer the question: "If the passage is completed within a given time t, what is the probability that the passage was completed via the driver answering the phone" (or conversely by a third timeout occurring).

The final two lines to consider are the 'answer-cdf' and 'timeout-cdf'. Here we have normalised the original two lines by dividing the value at each time by the percentage of all requests which are eventually serviced in the given way. By doing this we achieve the cumulative distribution function of all requests initiated by an airbag deployment which then result in the driver answering the phone or the call timing out for the third time respectively. This can then answer our question: "What percentage of airbag deployments whose driver cannot be contacted are serviced within a given amount of time". This figure is often more interesting than just the question "What percentage of airbag deployments are resolved within a given time in some way" since if the driver is unhurt the response-time is of less importance.

3 Implementation

Passage-end analysis for XSP is fully implemented in the International PEPA Compiler (IPC), a formal analysis tool for steady-state and transient evaluation of PEPA models. The IPC compiler is part of the `ipclib` [7] suite, a collection of tools for the specification and evaluation of complex performance measures over Markovian process algebra models. The `ipclib` suite is an extension of the IPC tool previously used for computing response-time quantiles from PEPA models [8,9]. The IPC tool has been applied to a number of modelling problems such as performance of personal-area networks [10] and compiler optimisations [11]. Although we have concentrated here mostly on passage-time computation, IPC also supports the computation of steady-state, transient and counting measures as described in [12].

4 Conclusions

We have modified passage-time analysis to allow for distinct passage results. We have done so within the framework of eXtended Stochastic Probes to ensure that our passage-end queries remain robust over model modifications and do not

require that the modeller modify their original model. The set of queries which can be specified with XSP is therefore extended.

Another bonus which we obtain almost for free is the ability to analyse passages which may never complete at all. This only works for passages in which there is only one source state, because if the model deadlocks we cannot analyse the embedded Markov chain to obtain the distribution of probability to the source states at the beginning of the passage. This allows the modeller to provide a concrete answer to the question: "How long should I wait for my response?" because we can now say that a given percentage of all requests which ultimately are successfully serviced are serviced within 10 seconds, hence if you have waited longer than 10 seconds it is likely you will never receive a response and hence can cancel the request yourself.

Acknowledgements

The authors are supported by the EU FET-IST Global Computing 2 project SENSORIA ("Software Engineering for Service-Oriented Overlay Computers" (IST-3-016004-IP-09)).

References

1. Clark, A., Gilmore, S.: State-aware performance analysis with eXtended Stochastic Probes. In: Thomas, N., Juiz, C. (eds.) EPEW 2008. LNCS, vol. 5261, pp. 125–140. Springer, Heidelberg (2008)
2. Argent-Katwala, A., Bradley, J., Dingle, N.: Expressing performance requirements using regular expressions to specify stochastic probes over process algebra models. In: Proceedings of the Fourth International Workshop on Software and Performance, Redwood Shores, California, USA, pp. 49–58. ACM Press, New York (2004)
3. Argent-Katwala, A., Bradley, J., Clark, A., Gilmore, S.: Location-aware quality of service measurements for service-level agreements. In: Barthe, G., Fournet, C. (eds.) TGC 2007 and FODO 2008. LNCS, vol. 4912, pp. 222–239. Springer, Heidelberg (2008)
4. Hillston, J.: A Compositional Approach to Performance Modelling. Cambridge University Press, Cambridge (1996)
5. Clark, A., Gilmore, S.: Evaluating quality of service for service level agreements. In: Brim, L., Leucker, M. (eds.) Proceedings of the 11th International Workshop on Formal Methods for Industrial Critical Systems, Bonn, Germany, August 2006, pp. 172–185 (2006)
6. OnStar: OnStar by General Motors, car safety device and vehicle security system (April 2009), http://www.onstar.com/
7. Clark, A.: The ipclib PEPA Library. In: Harchol-Balter, M., Kwiatkowska, M., Telek, M. (eds.) Proceedings of the 4th International Conference on the Quantitative Evaluation of SysTems (QEST), pp. 55–56. IEEE, Los Alamitos (2007)
8. Bradley, J., Dingle, N., Gilmore, S., Knottenbelt, W.: Extracting passage times from PEPA models with the HYDRA tool: A case study. In: Jarvis, S. (ed.) Proceedings of the Nineteenth annual UK Performance Engineering Workshop, University of Warwick, July 2003, pp. 79–90 (2003)

9. Bradley, J., Dingle, N., Gilmore, S., Knottenbelt, W.: Derivation of passage-time densities in PEPA models using IPC: The Imperial PEPA Compiler. In: Kotsis, G. (ed.) Proceedings of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems, University of Central Florida, pp. 344–351. IEEE Computer Society Press, Los Alamitos (2003)
10. Ding, J., Hillston, J., Laurenson, D.: Performance modelling of content adaptation for a personal distributed environment. Personal Wireless Communication (November 2007)
11. Djoudi, L., Kloul, L.: Assembly code analysis using stochastic process algebra. In: Thomas, N., Juiz, C. (eds.) EPEW 2008. LNCS, vol. 5261, pp. 95–109. Springer, Heidelberg (2008)
12. Argent-Katwala, A., Bradley, J.T.: Functional performance specification with stochastic probes. In: Horváth, A., Telek, M. (eds.) EPEW 2006. LNCS, vol. 4054, pp. 31–46. Springer, Heidelberg (2006)

Stochastic Monotonicity in Queueing Networks*

H. Castel-Taleb[1] and N. Pekergin[2]

[1] Institut Telecom
Telecom et Management SudParis
9,rue Charles Fourier
91011 Evry Cedex, France
hind.castel@int-evry.fr
[2] LACL
Université Paris-Est Val de Marne
61, av. du Général de Gaulle,
94010 Créteil Cedex, France
nihal.pekergin@univ-paris12.fr

Abstract. Stochastic monotonicity is one of the sufficient conditions for stochastic comparisons of Markov chains. On a partially ordered state space, several stochastic orderings can be defined by means of increasing sets. The most known is the strong stochastic (sample-path) ordering, but weaker orderings (weak and weak*) could be defined by restricting the considered increasing sets. When the strong ordering could not be defined, weaker orderings represent an alternative as they generate less constraints. Also, they may provide more accurate bounds.

The main goal of this paper is to provide an intuitive event formalism added to stochastic comparisons methods in order to prove the stochastic monotonicity for multidimensional Continuous Time Markov Chains (CTMC). We use the coupling by events for the strong monotonicity. For weaker monotonicity, we give a theorem based on generator inequalities using increasing sets. We prove this theorem, and we present the event formalism for the definition of the increasing sets. We apply our formalism on queueing networks, in order to establish monotonicity properties.

1 Introduction

Stochastic comparison is an efficient method for the performance evaluation of computer and telecommunication systems. We study systems represented by multidimensional Markov chains. In this case, the quantitative analysis could be very hard if there is not a specific form for the probability distribution. Stochastic comparison provides the comparison of the considered system with other systems easier to analyze, in order to derive performance measure bounds. Stochastic comparison methods are based on the stochastic ordering theory which is complex on multidimensional states spaces. Several stochastic orderings can be defined, corresponding to different comparison relations of probability distributions.

* partially supported by french research project ANR-SETI06-02

J.T. Bradley (Ed.): EPEW 2009, LNCS 5652, pp. 116–130, 2009.

Different stochastic comparison methods can be used: increasing sets, and the coupling. Increasing sets method [10], [11] is a general formalism, allowing the definition of the strong stochastic ordering ($\preceq_{st}$) and weaker orderings ($\preceq_{wk}$, and $\preceq_{wk^*}$).

The coupling method consists in the comparison of process realizations under the same event occurrences. Both increasing set and coupling methods lead to the comparison of transition rates but the approach used is different: one with the coupling of sample paths, and the other from the definition of increasing set families. However, the coupling method could be applied only for the strong ordering, while the method with increasing sets is more general, allowing the definition of the strong and weaker orderings.

Weaker orderings could be interesting for the stochastic comparisons, as strong ordering generates hard conditions. For weaker stochastic comparisons we need to use the increasing set formalism, and in this case, the stochastic monotonicity is one of the sufficient conditions additional to the comparison of generators.

Related works

The stochastic monotonicity can be defined as the increasing (decreasing) in time of the considered process. Stochastic comparison methods as the coupling [8] can be applied in order to prove this property on a Markov process. The coupling [7] used for the stochastic comparison of different processes is equivalent to the definition of a coupled version of the processes with realizations staying in a set K [3], [9]. This means that the coupling is equivalent to both a sample path ordering and inequalities between transition rates. In [7] the coupling method has been presented first for birth and death processes representing the M/M/1 queues, and after for general birth and death processes on multidimensional state spaces. The strong monotonicity can be proved by the coupling of the process with itself. In [8], the coupling is applied on Jackson networks in order to prove the strong monotonicity. The approach proposed is based on the computation of transition rates of the coupled process in order to verify an increasing in time. Strong monotonicity is a very interesting property because it simplifies the comparison of the processes in the case of strong ordering [10]. For weaker orderings, there is not many studies. In [12], an operator theory approach is presented for weak stochastic comparisons of multidimensional CTMC. The comparison of a Jackson network with independent M/M/1 queues is presented in order to bound the transient behaviour. In [13], the weak comparison of multidimensional marginal distributions of discrete-time Markov chains is provided from monotonicity and transition probability inequalities. However, in our best knowledge there is no study about proving the weak monotonicity of a Markov process. The goal of the present paper is to develop a formalism based on events for the stochastic monotonicity. For the strong monotonicity, we present the coupling of the process with itself after the happening of the same event. For weaker monotonicity, we provide a theorem based on generator inequalities and increasing sets. We provide the proof of this theorem using associated Markov chains. As the number of increasing sets could be important, then we use the event formalism in order to

reduce it. This paper is organized as follows: first we present basic concepts about stochastic comparisons on multidimensional and partially ordered state space. Section 3 is devoted to the stochastic monotonicity and the methods applied to prove it. At the end, we focus on the stochastic monotonicity in queueing networks, in order to prove the monotonicity using the proposed formalism.

2 Stochastic Comparison Method

Quantitative analysis of multidimensional Markov processes is a hard problem due to the state space explosion since the state space size is generally the product of state space sizes of components. As a solution, one can bound the original Markov process by another Markov process which is easier (taking values on smaller state spaces or having special structures) to analyze, in order to compute bounds on performance measures [2], [14], [1]. The stochastic comparisons are established by means of two methods on partially ordered state spaces: increasing sets, and the coupling. The goal of stochastic comparisons is to generate stochastic orderings, which can be defined as a relation order between random variables, probabilities distributions, or stochastic processes. First, we define what is a stochastic ordering between random variables.

2.1 Stochastic Ordering Theory

Let E be a discrete, and countable state space, and $\preceq$ be at least a preorder (reflexive, transitive but not necessarily an anti-symmetric binary relation) on E. We suppose that E is a multidimensional state space, where each component is discrete, as it is the case in the queueing models. Several stochastic orderings can be defined, the most known is the strong stochastic ordering $\preceq_{st}$, but also weaker orderings can be defined: $\preceq_{wk}$, and $\preceq_{wk^*}$ [10]. The strong stochastic ordering is equivalent to a sample path ordering, the $\preceq_{wk}$ ordering to a tail distributions comparison, and $\preceq_{wk^*}$ serve the same role for cumulative distribution functions. Different formalisms can be used to define a stochastic ordering: increasing functions, and increasing sets [15]. We focus on the increasing set formalism in this paper, as we will use it for the comparison of the processes in this paper. Let $\Gamma \subseteq E$, we denote by

$$\Gamma \uparrow = \{y \in E \mid y \succeq x, x \in \Gamma\} \tag{1}$$

Definition 1. *Γ is called an increasing set if and only if $\Gamma = \Gamma \uparrow$*

From the general definition of an increasing set, three stochastic orderings have been defined from families of increasing sets [10]. The first one is $\Phi_{st}(E)$ which is defined from all the increasing sets of E:

$$\Phi_{st}(E) = \{\text{all increasing sets on } E\} \tag{2}$$

Other families $\Phi_{wk}(E)$ and $\Phi_{wk^*}(E)$ are defined from particular kinds of increasing sets.

$$\Phi_{wk}(E) = \{\{x\}\uparrow,\ x \in E\},\ \{x\}\uparrow = \{y \in E,\ y \succeq x\} \tag{3}$$

and

$$\Phi_{wk^*}(E) = \{E - \{x\}\downarrow,\ x \in E\}, \{x\}\downarrow = \{y \in E,\ y \preceq x\} \tag{4}$$

$\Phi_{st}(E)$ induces the $\preceq_{st}$ ordering, and $\Phi_{wk}(E)$ and $\Phi_{wk^*}(E)$ generate respectively $\preceq_{wk}$ and $\preceq_{wk^*}$ orderings [10].

Let X and Y be two random variables defined on E, and their probability measures given respectively by the probability vectors p and q where $p[i] = Prob(X = i),\ \forall i \in E$ (resp. $q[i] = Prob(Y = i),\ \forall i \in E$). If $\Phi(E)$ represents one of these families ($\Phi_{st}(E)$, $\Phi_{wk}(E)$, or $\Phi_{wk^*}(E)$), then a stochastic ordering $\preceq_\Phi$ representing ($\preceq_{st}$, $\preceq_{wk}$, or $\preceq_{wk^*}$) can be defined as follows [10]:

Definition 2

$$X \preceq_\Phi Y \Leftrightarrow \sum_{x\in\Gamma} p[x] \leq \sum_{x\in\Gamma} q[x], \forall \Gamma \in \Phi(E) \tag{5}$$

Let 1_Γ the indicator function of $\Gamma \subset E$, if we apply 1_Γ to p, then $p\ 1_\Gamma = \sum_{x\in\Gamma} p[x]$, and so the precedent definition can be also written:

$$X \preceq_\Phi Y \Leftrightarrow p\ 1_\Gamma \leq q\ 1_\Gamma,\ \forall \Gamma \in \Phi(E) \tag{6}$$

Next, we present the stochastic comparison of Continuous Time Markov Chains (CTMC). Let $\{X(t), t \geq 0\}$ (resp. $\{Y(t), t \geq 0\}$) be a CTMC defined on E. We give the definition [10] of the $\preceq_\Phi$-stochastic comparison .

Definition 3. *We say that* $\{X(t), t \geq 0\} \preceq_\Phi \{Y(t), t \geq 0\}$

if :

$$X(0) \preceq_\Phi Y(0) \Longrightarrow X(t) \preceq_\Phi Y(t), \forall t > 0 \tag{7}$$

In the case of the $\preceq_{st}$ ordering, the coupling method can be used for the stochastic comparison of CTMC. As presented in [7], [8], it remains to define two CTMC: $\{\widehat{X}(t),\ t \geq 0\}$ and $\{\widehat{Y}(t),\ t \geq 0\}$ governed by the same infinitesimal generator matrix as respectively $\{X(t), t \geq 0\}$, and $\{Y(t), t \geq 0\}$, representing different realizations of these processes with different initial conditions. The theorem of the $\preceq_{st}$-comparison using the coupling states as follows [7]:

Theorem 1. *We say that:* $\{X(t), t \geq 0\} \preceq_{st} \{Y(t), t \geq 0\}$ *if and only if there exists the coupling* $\{(\widehat{X}(t), \widehat{Y}(t)),\ t \geq 0\}$ *such that:*

$$\widehat{X}(0) \preceq \widehat{Y}(0) \Rightarrow \widehat{X}(t) \preceq \widehat{Y}(t),\ \forall t > 0 \tag{8}$$

If we suppose that $\{X(t),\ t \geq 0\}$ (resp. $\{Y(t),\ t \geq 0\}$) is a CTMC with infinitesimal generator matrix A (resp. B), then we present the theorem of the $\preceq_\Phi$-stochastic comparison of CTMC using increasing set formalism [10], [15].

Theorem 2. *We say that:* $\{X(t), t \geq 0\} \preceq_{\Phi} \{Y(t), t \geq 0\}$ *if and only if the following conditions are verified:*

1. $X(0) \preceq_{\Phi} Y(0)$
2. $\{X(t), t \geq 0\}$ *or* $\{Y(t), t \geq 0\}$ *is* $\preceq_{\Phi}$*-monotone*
3. $$\forall x \in E,\ \sum_{z \in \Gamma} A(x,z) \leq \sum_{z \in \Gamma} B(x,z),\ \forall \Gamma \in \Phi_{(}E) \tag{9}$$

Where $A(x,z)$ (resp. $B(x,z)$) denotes the transition rate from the state x to z for the process $\{X(t), t \geq 0\}$ (resp. $\{Y(t), t \geq 0\}$). The monotonicity is a property used in this theorem, corresponding to an increasing (decreasing) in time of a process. Next we give the definition of the $\preceq_{\Phi}$-monotonicity.

Definition 4. *We say that* $\{X(t), t \geq 0\}$ *is* $\preceq_{\Phi}$ *−monotone increasing (resp. decreasing) if*

$$X(t) \preceq_{\Phi} (\succeq_{\Phi}) X(t+\tau), \forall t \geq 0, \forall \tau \geq 0 \tag{10}$$

Next, we focus on how to use stochastic comparisons methods in order to define a formalism for the stochastic monotonicity. We study only the case of monotone increasing processes, called in this paper monotone processes. First, we present the coupling for the strong monotonicity. Secondly, for the general $\preceq_{\Phi}$ monotonicity, we propose a transition rates inequalities using increasing sets method.

3 Stochastic Monotonicity

We propose to apply two methods to prove the monotonicity: the coupling for the $\preceq_{st}$-monotonicity, and increasing sets for the $\preceq_{\Phi}$-monotonicity.

3.1 The Coupling

We present in this section the coupling method in order to prove the strong monotonicity of a process. The strong monotonicity can be proved by the coupling of the process with itself [7]. The $\preceq_{st}$-monotonicity of a CTMC $\{X(t), t \geq 0\}$, can be defined using the coupling of the processes [7], [8]. As presented in [7], [8], it remains to define two processes: $\{\widehat{X}(t),\ t \geq 0\}$ and $\{\widehat{X'}(t),\ t \geq 0\}$ governed by the same infinitesimal generator matrix as $\{X(t), t \geq 0\}$, representing different realizations of $\{X(t), t \geq 0\}$ with different initial conditions. The theorem of the $\preceq_{st}$-monotonicity using the coupling states as follows [7]:

Theorem 3. *We say that* $\{X(t), t \geq 0\}$ *is* $\preceq_{st}$ *-monotone if and only if there exists the coupling* $\{(\widehat{X}(t), \widehat{X'}(t)),\ t \geq 0\}$ *such that:*

$$\widehat{X}(0) \preceq \widehat{X'}(0) \Rightarrow \widehat{X}(t) \preceq \widehat{X'}(t),\ \forall t > 0 \tag{11}$$

In section 4 we will present an example on queueing networks using the coupling for the strong monotonicity. Next, we generalize to the $\preceq_{\Phi}$-monotonicity, we use the increasing set theory, and we provide generator inequalities.

3.2 Increasing Sets

In the case of discrete time Markov chains (DTMC), the monotonicity can be expressed using the probability transition matrix. For the time-homogeneous DTMC $\{X(n), n \geq 0\}$ with probability transition matrix P, the $\preceq_\Phi$-monotonicity is defined as follows [15]:

Theorem 4. *$\{X(n), n \geq 0\}$ is $\preceq_\Phi$-monotone if and only if:*

$$p \preceq_\Phi q \Rightarrow pP \preceq_\Phi qP \tag{12}$$

For a CTMC $\{X(t), t \geq 0\}$ we have the following theorem [15]:

Theorem 5. *$\{X(t), n \geq 0\}$ is $\preceq_\Phi$-monotone if and only if:*

$$p \preceq_\Phi q \Rightarrow p\ exp(tA) \preceq_\Phi q\ exp(tA) \tag{13}$$

Since these inequalities are difficult to use we need to define easier inequalities on the transition rates. The $\preceq_{st}$-monotonicity is equivalent to the following inequality [10]:

Theorem 6. *$\{X(t), t \geq 0\}$ is $\preceq_{st}$-monotone if and only if the following condition is verified:*

$$\forall \Gamma \in \Phi_{st}(E),\ \sum_{z \in \Gamma} A(x,z) \leq \sum_{z \in \Gamma} A(y,z),\ x \preceq y \in E,\ x,y \in \Gamma \text{ or } x,y \notin \Gamma \tag{14}$$

There is no similar inequality for the $\preceq_\Phi$-monotonicity. We propose the following theorem:

Theorem 7. *$\{X(t), t \geq 0\}$ is $\preceq_\Phi$-monotone if the following condition is verified:*

$$\forall \Gamma \in \Phi(E),\ \sum_{z \in \Gamma} A(x,z) \leq \sum_{z \in \Gamma} A(y,z),\ x \preceq y \in E,\ x,y \in \Gamma \text{ or } x,y \notin \Gamma \tag{15}$$

Proof. In order to prove that: $\{X(t), t \geq 0\}$ is $\preceq_\Phi$-monotone, we will prove that the associated Markov chain $\{X^\lambda(n), n \geq 0\}$ is $\preceq_\Phi$ monotone [15], [10]. $\{X^\lambda(n), n \geq 0\}$ is defined by the probability transition matrix P^λ given by:

$$P^\lambda = I + \frac{1}{\lambda} A$$

where $\lambda \geq sup_{i \in E} ||A_{i,i}||$. We have two main steps for the proof of this theorem:

1. First, we have to prove that if:

$$\forall \Gamma \in \Phi(E),\ \sum_{z \in \Gamma} A(x,z) \leq \sum_{z \in \Gamma} A(y,z), \forall x \preceq y\ \in E,\ x,y \in \Gamma \text{ or } x,y \notin \Gamma \tag{16}$$

Then:

$$\forall \Gamma \in \Phi(E),\ \sum_{z \in \Gamma} P^\lambda(x,z) \leq \sum_{z \in \Gamma} P^\lambda(y,z), \forall\ x \preceq y\ \in E \tag{17}$$

2. Secondly, we have to prove that if:

$$\forall \Gamma \in \Phi(E),\ \sum_{z\in\Gamma} P^{\lambda}(x,z) \leq \sum_{z\in\Gamma} P^{\lambda}(y,z), \forall\ x \preceq y \in E \tag{18}$$

Then $\{X^{\lambda}(n), n \geq 0\}$ is $\preceq_{\Phi}$-monotone. We begin with the first step.

First step. Let $\delta_x(\Gamma)$ be the one point distribution with mass in x (so it is such that: $\delta_x(\Gamma) = 1$ if $x \in \Gamma$, otherwise 0). So we deduce that: $\forall \Gamma \in \Phi(E)$

$$\sum_{z\in\Gamma} P^{\lambda}(x,z) = \delta_x(\Gamma) + \frac{1}{\lambda}\sum_{z\in\Gamma} A(x,z)$$

In the case of $x \preceq y$, we have: $\delta_x(\Gamma) \leq \delta_y(\Gamma)$. Moreover, for both cases i. $x, y \in \Gamma$ and ii. $x, y \notin \Gamma$, $\delta_x(\Gamma) = \delta_y(\Gamma)$. So if:

$$\forall \Gamma \in \Phi(E),\ \sum_{z\in\Gamma} A(x,z) \leq \sum_{z\in\Gamma} A(y,z), \forall x \preceq y\ \in E,\ x, y \in \Gamma\ or\ x, y \notin \Gamma \tag{19}$$

then we have:

$$\forall \Gamma \in \Phi(E),\ \sum_{z\in\Gamma} P^{\lambda}(x,z) \leq \sum_{z\in\Gamma} P^{\lambda}(y,z),\ \forall x \preceq y\ \in E,\ x, y \in \Gamma\ or\ x, y \notin \Gamma \tag{20}$$

For the case where $x \notin \Gamma$, and $y \in \Gamma$, then we have $\delta_x(\Gamma) = 0$, and $\delta_y(\Gamma) = 1$. So we have:

$$\sum_{z\in\Gamma} P^{\lambda}(x,z) - \sum_{z\in\Gamma} P^{\lambda}(y,z) = \frac{1}{\lambda}\sum_{z\in\Gamma} A(x,z) - \frac{1}{\lambda}\sum_{z\in\Gamma} A(y,z) - 1 \tag{21}$$

as

$$\sum_{z\in\Gamma} A(x,z) - \sum_{z\in\Gamma} A(y,z) \leq |\sum_{z\in\Gamma} A(x,z)| + |\sum_{z\in\Gamma} A(y,z)| \tag{22}$$

then:

$$\sum_{z\in\Gamma} P^{\lambda}(x,z) - \sum_{z\in\Gamma} P^{\lambda}(y,z) \leq \frac{1}{\lambda}(||A|| + ||A||) - 1 \tag{23}$$

where

$$||A|| = Sup_{x\in E}|A(x,x)|$$

We suppose that: $\lambda \geq 2||A||$, then we obtain:

$$\forall \Gamma \in \Phi(E),\ \sum_{z\in\Gamma} P^{\lambda}(x,z) \leq \sum_{z\in\Gamma} P^{\lambda}(y,z), \forall x \preceq y\ \in E,\ x \notin \Gamma, y \in \Gamma \tag{24}$$

so we deduce that:

$$\forall \Gamma \in \Phi(E),\ \sum_{z\in\Gamma} P^{\lambda}(x,z) \leq \sum_{z\in\Gamma} P^{\lambda}(y,z),\ x \preceq y \tag{25}$$

Second step. We focus now on the second point, which means that if:

$$\forall \Gamma \in \Phi(E),\ \sum_{z\in\Gamma} P^{\lambda}(x,z) \leq \sum_{z\in\Gamma} P^{\lambda}(y,z), \forall x \preceq y\ \in E \tag{26}$$

then $\{X^{\lambda}(n), n \geq 0\}$ is $\preceq_{\Phi}$ monotone, which remains to verify that from theorem 4:

$$p \preceq_{\Phi} q \Rightarrow pP^{\lambda} \preceq_{\Phi} qP^{\lambda} \tag{27}$$

We suppose that: $p \preceq_{\Phi} q$, which is equivalent to: $\forall \Gamma \in \Phi(E),\ \sum_{x\in\Gamma} p[x] \leq \sum_{x\in\Gamma} q[x]$ and we have to prove that:

$$p\ P^{\lambda} \preceq_{\Phi} q\ P^{\lambda} \tag{28}$$

or equivalently:

$$\Gamma \in \Phi(E),\ p\ P^{\lambda} 1_{\Gamma} \leq q\ P^{\lambda} 1_{\Gamma}, \tag{29}$$

where 1_{Γ} is the indicator function of $\Gamma \subset E$. We denote by u_{Γ} a vector of E, defined by: $P^{\lambda}\ 1_{\Gamma} = u_{\Gamma}$, which means that: $u_{\Gamma}(x) = \sum_{z\in\Gamma} P(x,z)$ As we have supposed that:

$$\forall \Gamma \in \Phi(E),\ \sum_{z\in\Gamma} P^{\lambda}(x,z) \leq \sum_{z\in\Gamma} P^{\lambda}(y,z) \tag{30}$$

Then we can deduce that:

$$\forall \Gamma \in \Phi(E),\ u_{\Gamma}[x] \leq u_{\Gamma}[y], \forall x \preceq y\ \in E \tag{31}$$

Which means that u_{Γ} increases with any state $x \in E$ (or is an increasing function $E \to R), \forall \Gamma \in \Phi(E)$. As we have supposed that : $\forall \Gamma \in \Phi(E),\ \sum_{x\in\Gamma} u_{\Gamma}[x] \leq \sum_{x\in\Gamma} u_{\Gamma}[x]$, which means that u_{Γ} increases with x, then:

$$\sum_{x\in E} p[x]u_{\Gamma}[x] \leq \sum_{x\in E} q[x]u_{\Gamma}[x] \tag{32}$$

so equation (29) is verified, and the associated Markov chain is monotone, and also the Markov process.

4 The Monotonicity in Queueing Systems

We apply the coupling and the increasing set formalism in order to prove the monotonicity property in queueing systems. First, we present the event formalism on the coupling for the strong monotonicity in order to provide an intuitive approach.

4.1 The Monotonicity by Coupling

The strong monotonicity of Jackson networks has been proved in [8], [7], by computing the transition rates of the coupled process. Next, we present an intuitive approach based on the coupling using events happening on the system. We propose to study a queueing system similar to a Jackson network except that queues have finite capacity (see an example in Fig1 for n=6).

The arrival rate in each queue i is Poisson with rate λ_i, the service time is exponential with parameter μ_i. Each queue i has a finite capacity B_i. After a service from a queue i, a customer can go outside with a probability d_i or transit to a queue j with a probability p_{ij} if queue j is not full. An arrival in a full queue is lost.

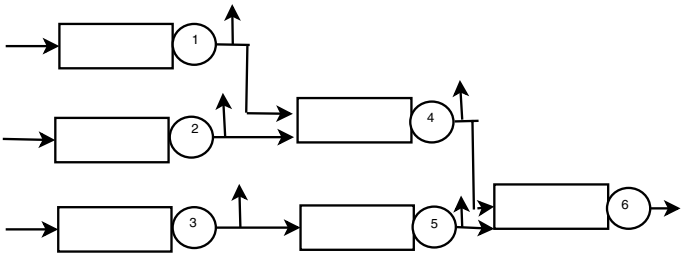


Fig. 1. An example of the system under study for n=6

We denote by $\{X(t), t \geq 0\}$ the Continuous Time Markov Chain (CTMC) representing the evolution of this system. This process is defined on $E = \mathbb{N}^n$. We propose to use the known partial ordering component by component denoted $\preceq$ on this state space:

$$\forall x, y \in \mathbb{N}^n,\ x \preceq y \Leftrightarrow x_i \leq y_i, \forall i = 1, \ldots, n \tag{33}$$

This order is widely used for multidimensional state spaces as it allows to compare queues by queues the behaviour of queueing networks. We have the following proposition:

Proposition 1. *$\{X(t), t \geq 0\}$ is $\preceq_{st}$-monotone.*

Proof. We use Theorem 3, so we prove that there exists two processes $\{\widehat{X}(t),\ t \geq 0\}$ and $\{\widehat{X'}(t), t \geq 0\}$ with the same infinitesimal generator matrix than $\{X(t),\ t \geq 0\}$ representing two different realizations with different initial conditions, and we prove that:

$$\widehat{X}(0) \preceq \widehat{X'}(0) \Rightarrow \widehat{X}(t) \preceq \widehat{X'}(t),\ t > 0 \tag{34}$$

Remember that $\{X(t), t \geq 0\}$ is a multidimensional process on E, it is represented by the vector: $X(t) = (X_1(t), \ldots, X_i(t), \ldots, X_n(t))$ also for $\{\widehat{X}(t),\ t \geq 0\}$ and $\{\widehat{X'}(t),\ t \geq 0\}$ which are represented by n components.

Let suppose that: $\widehat{X}(t) \preceq \widehat{X'}(t)$, we show that: $\widehat{X}(t+\Delta t) \preceq \widehat{X'}(t+\Delta t)$, by considering the evolution in all states even boundary states. We consider all events occurring during the time interval Δt:

1. an arrival in queue i: we can see easily that the arrival rate in queue i is λ_i from states $\widehat{X}(t)$ and $\widehat{X'}(t)$. So if $\widehat{X}(t)$ increases with an arrival in queue i, than $\widehat{X'}(t)$ will increase also. From the component $\widehat{X}_i(t)$, we obtain $\widehat{X_i}(t+\Delta t) = min\{B_i, \widehat{X_i}(t)+1\}$, and from $\widehat{X'_i}(t)$, we obtain $\widehat{X'_i}(t+\Delta t) = min\{B_i, \widehat{X'_i}(t)+1\}$. Since others components do not change, and $\widehat{X}(t) \preceq \widehat{X'}(t)$ then $\widehat{X}(t+\Delta t) \preceq \widehat{X'}(t+\Delta t)$.
2. a transit from queue i to queue j: as the transition rate is $\mu_i p_{ij}$ for $\widehat{X}(t)$ and $\widehat{X'}(t)$ then the evolutions are the same, and a transit with this event of one of the process can be compensated by the other. The transit occurs if $\widehat{X_i}(t) > 0$ and the customer is accepted in queue j if $\widehat{X_j}(t) < B_j$, otherwise it is lost. From $\widehat{X}(t)$, we obtain $\widehat{X_i}(t+\Delta t) = max\{0, \widehat{X_i}(t)-1\}$, and $\widehat{X_j}(t+\Delta t) = min\{B_j, \widehat{X_j}(t)+1\}$. From $\widehat{X'}(t)$, similarly, $\widehat{X'_i}(t+\Delta t) = max\{0, \widehat{X'_i}(t)-1\}$, and $\widehat{X'_j}(t+\Delta t) = min\{B_j, \widehat{X'_j}(t)+1\}$. Since others components do not change, and $\widehat{X}(t) \preceq \widehat{X'}(t)$ then $\widehat{X}(t+\Delta t) \preceq \widehat{X'}(t+\Delta t)$.
3. a service from queue i to the outside: as the service rate is $\mu_i d_i$ for the two processes, then if we have a service in queue i for $\widehat{X'}(t)$, we have also a service for $\widehat{X}(t)$. So $\widehat{X'_i}(t+\Delta t) = max\{0, \widehat{X'_i}(t)-1\}$, and $\widehat{X_i}(t+\Delta t) = max\{0, \widehat{X_i}(t)-1\}$, then $\widehat{X}(t+\Delta t) \preceq \widehat{X'}(t+\Delta t)$.

So we deduce that $\{X(t), t \geq 0\}$ is "$\preceq_{st}$"-monotone.

Using the coupling based on events, we can easily study others networks as G-Networks [5]. In such networks, negative customers which delete a (positive) customer are introduced. The strong monotonicity is not verified on G-Networks due to the transit event of a negative customer:

2^- a transit of negative customer from queue i to queue j with transition rate $\mu_i p^-_{ij}$ both for $\widehat{X}(t)$ and $\widehat{X'}(t)$. The transit occurs if $\widehat{X_i}(t) > 0$ and the negative customer deletes a customer if queue j is not empty and it disappears anyway. From $\widehat{X}(t)$, we obtain $\widehat{X_i}(t+\Delta t) = max\{0, \widehat{X_i}(t)-1\}$, and $\widehat{X_j}(t+\Delta t) = max\{0, \widehat{X_j}(t)-1\}$. From $\widehat{X'}(t)$, similarly, $\widehat{X'_i}(t+\Delta t) = max\{0, \widehat{X'_i}(t)-1\}$, and $\widehat{X'_j}(t+\Delta t) = max\{0, \widehat{X'_j}(t)-1\}$. In the case where for queue i: $\widehat{X_i}(t) = 0$ and $\widehat{X'_i}(t) \neq 0$ and for queue j: $\widehat{X_j}(t) = \widehat{X'_j}(t) \neq 0$, then $\widehat{X}(t+\Delta t) \not\preceq \widehat{X'}(t+\Delta t)$ although $\widehat{X}(t) \preceq \widehat{X'}(t)$.

The strong monotonicity is a very interesting property for the strong comparison of processes. In [1], bounding aggregations are used to reduce the state space size of multidimensional Markov processes. Bounding aggregations are based on the stochastic comparison by mapping functions into a smaller state space. An algorithm has been presented which can be applied only on strong monotone

Markov processes. Next, we focus on the weak monotonicity. The strong monotonicity does not imply the weak monotonicity. From theorem 5 we can see that if the implication is true for $\preceq_{st}$ ordering, then for two vectors p and q such that $p \preceq_{wk} q$, but $p \not\preceq_{st} q$, we cannot conclude. So we need to develop a formalism for the weak monotonicity. We apply theorem 7 which we have proved above. This theorem is not easy to apply as we have many increasing sets to define. We propose to use the event formalism in order to solve this problem. Next, we apply the increasing set formalism with events in order to prove the weak monotonicity of independent M/M/1 queues.

4.2 The Monotonicity by Increasing Sets

The weak monotonicity of independent M/M/1 queues is an interesting result since this process could represent an interesting bounding system as the transient behaviour is known [11]. Independent queues are defined by deleting links between queues, and by adding transit flows to the arrivals in the queues. As it has been explained in [10], the strong ordering could not exist. Authors prove using an operator approach that the weak ordering could be defined.

In the present paper, we study similar systems but with finite queue capacities. We use the increasing set formalism based on events for the stochastic comparison. As the weak monotonicity is a sufficient condition, then we need to prove this property. Let $\{Y(t), t \geq 0\}$ be a Markov process defined by n independent M/M/1 queues with finite capacity. Each queue i is defined by: arrival rates $\lambda_i + \sum_{j \neq i} \mu_j p_{ji}$, a service rate μ_i, and a capacity B_i. We denote by B the infinitesimal generator. We use the same assumptions than previously: state space $E = \mathbb{N}^n$, and the preorder $\preceq$ component by component. We have the following proposition.

Proposition 2. *$\{Y(t), t \geq 0\}$ is $\preceq_{wk}$-monotone*

Proof. As E is multidimensional, then $\Phi_{wk}(E)$ could be very large. So we need to define the increasing sets which are used for the $\preceq_{wk}$-monotonicity. It follows from inequality (14) that we have to prove that transitions rates from states x and y to states of the increasing sets are verified. Since these transitions are triggered by events, we define the increasing sets from states x and y and these events. From a state x, in a queue i, we can have an arrival, or a service or nothing. Let e_i be a binary vector on $\{1, \ldots, n\}$, where all the components are null except the component i which equals 1. This vector will be used to represent the evolution of the process from a state x after an event. For example, with an arrival in queue i, we have a transition from state x to state $x + e_i$. So the increasing sets used for the monotonicity are:

$$\{x\}\uparrow, \{x + e_i\}\uparrow, \{x - e_i\}\uparrow, \{y\}\uparrow, \{y + e_i\}\uparrow, \{y - e_i\}\uparrow \tag{35}$$

As we must also take the condition:

$$x, y \in \Gamma \text{ or } x, y \notin \Gamma \tag{36}$$

then we do not have to consider the increasing set: $\{y\}\uparrow$ as x is not in this increasing set. We denote by $S_{wk}(E)$ the set of increasing states which are sufficient for the comparison. It is defined by:

$$S_{wk}(E) = \{\{x\}\uparrow, \{x+e_i\}\uparrow, \{y+e_i\}\uparrow, \{x-e_i\}\uparrow, \{y-e_i\}\uparrow\} \tag{37}$$

Next, we define each increasing set.

Increasing sets definition. We have three constraints to use: the condition $x \preceq y$, the events, and the condition $x, y \in \Gamma$ or $x, y \notin \Gamma$.

We give for each increasing set the list of states to which the transitions are not null. The "..." in the sets means that there are other states, but we do not need to give them as transitions are null.

- For $\{x+e_i\}\uparrow$, we need to define states which are upper than $x+e_i$. In this case, if $x_i < y_i$ we have the following states: $x+e_i$, y, $y+e_i$ (as $y \succeq x+e_i$, and $y+e_i \succeq x+e_i$). As the condition in the equation (36) will not be verified, then we don't take this case. If $x_i = y_i$, then we have the states: $x+e_i$, $y+e_i$, and so the condition (36) will be verified. So if $x_i < B_i$ and $y_i < B_i$:

$$\{x+e_i\}\uparrow = \{x+e_i, \ldots, y+e_i, \ldots\} \tag{38}$$

- For $\{y+e_i\}\uparrow$, we need to define states which are upper than $y+e_i$, so we have only $y+e_i$. If $y_i < B_i$, then:

$$\{y+e_i\}\uparrow = \{y+e_i, ...\} \tag{39}$$

- For $\{x\}\uparrow$, we need to define states which are upper than x. So we have x, we have also y as $x \preceq y$, and we can have also some states $y-e_k$, $k=1,\ldots,n$ such that $y-e_k \succeq x$, we have also $x+e_k (k=1\ldots n$, and $y+e_k (k=1,\ \ldots n)$. So we have:

$$\{x\}\uparrow = \{x, \ldots y-e_k\ (k=1,\ldots,n\ if\ y_k > 0\ and\ y-e_k \succeq x)\ldots, y, \ldots,$$

$$x+e_k(k=1\ldots n,\ if\ x_k < B_k), \ldots, y+e_k(k=1,\ \ldots n\ if\ y_k < B_k)\} \tag{40}$$

- For $\{x-e_i\}\uparrow$, we obtain if $x_i > 0$

$$\{x-e_i\}\uparrow = \{x-e_i, \ldots, y-e_k(k=1\ldots n,\ y-e_k \geq x-e_i), \ldots,$$

$$x, \ldots y, \ldots, x+e_k(k=1\ldots n,\ x_k < B_k), \ldots y+e_k(k=1\ldots n,\ y_k < B_k)\} \tag{41}$$

- For $\{y-e_i\}\uparrow$, we have y in the set, but we are not sure to have x. In order to have the condition 36 verified, then we take the case where x is also in the increasing set which could be true if $y-e_i = x$ (so in the set we write only one of them : $y-e_i$) . If $y_i > 0$, then

$$\{y-e_i\}\uparrow = \{y-e_i, \ldots, y, \ldots x+e_k(k=1\ldots n,\ x+e_k < B_k), \ldots,$$

$$y+e_k(k=1\ldots n,\ y_k < B_k)\}\ if\ y_i > 0 \tag{42}$$

We compute now the transition rates in these increasing sets. In order to simplify the notation, we denote as follows the increasing sets: $\Gamma_{x+e_i} = \{x+e_i\}\uparrow$, $\Gamma_x = \{x\}\uparrow$, $\Gamma_{x-e_i} = \{x-e_i\}\uparrow$, $\Gamma_{y+e_i} = \{y+e_i\}\uparrow$, $\Gamma_{y-e_i} = \{x-e_i\}\uparrow$.

Now, as we have defined the increasing sets, we can compute the transition rates of the processes in order to compare them.

Transition rates comparison. For each increasing set, we compute the transition rates $\sum_{z\in\Gamma} B(x,z)$ and $\sum_{z\in\Gamma} B(y,z)$, $\forall\Gamma \in S_{wk}(E)$, in order to compare them.

Γ	$\sum_{z\in\Gamma} B(x,z)$	$\sum_{z\in\Gamma} B(y,z)$
Γ_{x+e_i}	$\lambda_i + \sum_{j\neq i}\mu_j p_{ji}$	$\lambda_i + \sum_{j\neq i}\mu_j p_{ji}$
Γ_{y+e_i}	0	$\lambda_i + \sum_{j\neq i}\mu_j p_{ji}$
Γ_x	$-\sum_{k=1}^{n}\mu_k 1_{x_k>0}$	$-\sum_{k=1}^{n}\mu_k 1_{y_k>0}1_{y_k=x_k}$
Γ_{x-e_i}	$-\sum_{k\neq i}\mu_k 1_{x_k>0}$	$-\sum_{k\neq i}\mu_k 1_{y_k>0}1_{y_k=x_k}$
Γ_{y-e_i}	$-\sum_{k}\mu_k 1_{x_k>0}$	$-\sum_{k\neq i}\mu_k 1_{y_k>0}$

The comparison of the transition rates is easy for increasing sets Γ_{x+e_i} and Γ_{y+e_i}. For other increasing sets, we need to explain how to compare transition rates. We need to compare the term:

$$\mu_k 1_{x_k>0} \qquad \text{with} \qquad \mu_k 1_{y_k>0}1_{y_k=x_k}$$

As : $1_{y_k>0}1_{y_k=x_k} = 1_{x_k>0}1_{y_k=x_k}$ and : $1_{x_k>0}1_{y_k=x_k} \leq 1_{x_k>0}$, then we have the following inequality:

$$-\sum_{k=1}^{n}\mu_k 1_{x_k>0} \leq -\sum_{k=1}^{n}\mu_k 1_{y_k>0}1_{y_k=x_k} \tag{43}$$

For increasing sets Γ_{x-e_i} and Γ_{y-e_i} the comparison is similar.

It is easy to see that:

$$\forall\Gamma \in S_{wk}(E),\ \forall x \preceq y \mid x,y \in \Gamma,\ or\ x,y \notin \Gamma$$

the following inequality is verified:

$$\sum_{z\in\Gamma} B(x,z) \leq \sum_{z\in\Gamma} B(y,z)$$

So from theorem 7 we deduce that $Y(t)$ is $\preceq_{wk}$-monotone.

This result will be very useful for the comparison of $\{X(t), t \geq 0\}$ and $\{Y(t), t \geq 0\}$. $\{Y(t), t \geq 0\}$ is an interesting process as both stationary and transient probability distribution can be computed as the product of M/M/1 queues probabilities distributions. The $\preceq_{wk}$-comparison of these processes could be interesting as the $\preceq_{st}$ ordering could not exist between the processes. Using the coupling of the processes, if we have a service in queue i in the process $Y(t)$

with a transition rate μ_i, then this event could be compensated only by a service in queue i in the process $X(t)$. But it is not possible as the service rate in $X(t)$ is equal to $\mu_i d_i$, so lower than μ_i. We give briefly the proof of the $\preceq_{wk}$-comparison of $X(t)$ and $Y(t)$. We apply theorem 2: so we compare $\sum_{z\in\Gamma} A(x,z)$ and $\sum_{z\in\Gamma} B(x,z)$ for increasing sets Γ_{x+e_i}, $\Gamma_{x-e_j+e_i}$, Γ_x, Γ_{x-e_i}.

Γ	$\sum_{z\in\Gamma} A(x,z)$	$\sum_{z\in\Gamma} B(x,z)$
Γ_{x+e_i}	λ_i	$\lambda_i + \sum_{j\neq i} \mu_j p_{ji}$
$\Gamma_{x-e_j+e_i}$	$\mu_j p_{ji} + \lambda_i$	$\lambda_i + \sum_{j\neq i} \mu_j p_{ji}$
Γ_x	$-\sum_{k=1}^{n} \mu_k 1_{x_k>0}$	$-\sum_{k=1}^{n} \mu_k 1_{x_k>0}$
Γ_{x-e_i}	$-\sum_{k\neq i} \mu_k 1_{x_k>0}$	$-\sum_{k\neq i} \mu_k 1_{x_k>0}$

So we can deduce from the theorem 2 that: $\{X(t), t \geq 0\} \preceq_{wk} \{Y(t), t \geq 0\}$, which means that:

$$P(X(t) \in \Gamma) \leq P(Y(t) \in \Gamma), \forall \Gamma \in \Phi_{wk}(E) \tag{44}$$

The relevance of this comparison is that stationary and transient probability distribution of $Y(t)$ can be computed easily as the product of probability distributions of each M/M/1 queue. We deduce that $\forall x = (x_1, \ldots, x_n)$:

$$P(X(t) \succeq x) \leq \Pi_{i=1}^{n} P(Y_i(t) \succeq x_i) \tag{45}$$

Note that we can also define a lower bound represented by n independent M/M/1 queues with arrival rate λ_i, and service rate μ_i.

5 Conclusion

We present a formalism based on events to establish stochastic monotonicity of Markov processes. We propose the coupling by event in order to provide an intuitive approach for the strong stochastic monotonicity. For weaker monotonicity, we provide a new theorem based on generator inequalities and increasing sets. The considered increasing sets are constituted by considering events in order to limit the number of increasing sets. Furthermore, as a future work, it will be interesting to develop an algorithm based on events for the stochastic monotonicity of Markov processes.

References

1. Castel, H., Mokdad, L.: Performance measure bounds in wireless networks by state space reduction. In: 13th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2005), Atlanta Georgia, September 27–29 (2005)

2. Castel, H., Fourneau, J.M., Pekergin, N.: Stochastic bounds on partial ordering: application to memory overflows due to bursty arrivals. In: Yolum, p., Güngör, T., Gürgen, F., Özturan, C. (eds.) ISCIS 2005. LNCS, vol. 3733, pp. 244–253. Springer, Heidelberg (2005)
3. Doisy, M.: Comparaison de processus Markoviens. PhD. thesis, Univ. de Pau et des pays de l'Adour 92
4. Economou, A.: Necessary and sufficient condition for the stochastic comparison of Jackson Networks. Prob. in the Engineering and Informational Sciences 17, 143–151 (2003)
5. Gelenbe, E.: Product Form Queuing Networks with Negative and Positive Customers. Journal Of Applied Probability 28, 656–663 (1991)
6. Fourneau, J.M., Pekergin, N.: An algorithmic approach to stochastic bounds. In: Calzarossa, M.C., Tucci, S. (eds.) Performance 2002. LNCS, vol. 2459, p. 64. Springer, Heidelberg (2002)
7. Lindvall, T.: Lectures on the coupling method. Wiley Series in Probability and Mathematical statistics (1992)
8. Lindvall, T.: Stochastic monotonicities in Jackson queueing networks. Prob. in the Engineering and Informational Sciences 11, 1–9 (1997)
9. Lopez, F.J., Martinez, S., Sanz, G.: Stochastic domination and Markovian couplings. Adv. Appl. Prob. 32, 1064–1076 (2000)
10. Massey, W.: Stochastic orderings for Markov processes on partially ordered spaces. Mathematics of Operations Research 12(2) (May 1987)
11. Massey, W.: A family of bounds for the transient behavior of a Jackson Network. J. App. Prob. 23, 543–549 (1986)
12. Massey, W.: Open networks of queues: their algebraic structure and estimating their transient behavior. Adv. App. Prob. 16, 176–201 (1984)
13. Ridder, A.: Weak stochastic ordering for multidimensionnal Markov chains. Oper. Research Letters 18, 121–126 (1995)
14. Pekergin, N.: Stochastic performance bounds by state space reduction. Performance Evaluation 36-37, 1–17 (1999)
15. Stoyan, D.: Comparison methods for queues and other stochastics models. J. Wiley and Son, Chichester (1976)
16. Truffet, L.: Reduction technique for discrete time Markov chains on totally ordered space using stochastic comparisons. J. App. Prob. 37(3) (2000)

Fast Generation of Scale Free Networks with Directed Arcs

Huqiu Zhang and Aad van Moorsel*

School of Computing Science
Newcastle University
Newcastle upon Tyne, UK
{huqiu.zhang,aad.vanmoorsel}@newcastle.ac.uk

Abstract. To evaluate peer-to-peer systems through discrete-event simulation, one needs to be able to generate sufficiently large networks of nodes that exhibit the desired properties, such as the scale-free nature of the connectivity graph. In applications such as the web of trust or analysis of hyperlink structures, the direction of the arcs between two nodes is relevant and one therefore generates directed graphs. In this paper we introduce a model to generate directed scale free graphs without multiple arcs between the same pair of nodes and loops. This model is based on existing models that allows multiple arcs and loops, but considerably more challenging to implement in an efficient manner. We therefore design and implement a set of algorithms and compare them with respect to CPU and memory use, in terms of both theoretical complexity analysis and experimental results. We will show through experiments that with the fastest algorithms networks with a million or more nodes can be generated in mere seconds.

1 Introduction

To evaluate novel peer-to-peer algorithms through discrete-event simulation one first needs to generate networks of nodes and relationships between nodes [1,2,3]. To obtain reliable results with small enough confidence intervals, one needs to generate many of these networks and it is therefore of importance that one is able to generate networks with the desired characteristics in reasonable time.

Very often one is interested in generating networks that are *scale-free*. In scale-free networks, the distribution over the node degrees confirms to a power law distribution, and it turns out that the scale-free phenomenon occurs frequently in real-life networks [4]. The power law node degree distributions also indicate that the resulting network exhibits the small-world phenomenon [5], that is, each pair of nodes are likely to have short paths connecting them.

There exist a large number of models to generate scale-free networks with undirected links (see [6] and references in [7]). For networks with directed arcs,

* The authors are supported in part by: EU coordination action 216295 ('AMBER: Assessing, Measuring, and Benchmarking Resilience') and UK Department of Trade and Industry, grant nr. P0007E ('Trust Economics').

J.T. Bradley (Ed.): EPEW 2009, LNCS 5652, pp. 131–148, 2009.

however, we are aware of only two models, which are almost identical [7] and [8]. Both models generate networks with both indegree and outdegree distributions satisfying a power law distribution. They both belong to the class of preferential attachment models, which are attractive since they provide an intuitively natural constructive approach to network generation. Moreover, the models can be implemented straightforwardly using 'Reversed Look-up' detailed in Section 3.5, as we will comment on later. A disadvantage is that the models allow multiple arcs with the same origin and destination ('multiple arcs') and with the destination to be equal to the origin ('loops').

For several applications or experiments, among which our work in trust path discovery [2,3], one can argue that it would be more natural or desirable to run simulations using networks that do not contain multiple arcs and loops. In this paper we therefore modify the models in [7,8] to create a network with directed arcs, but without multiple arcs or loops. Our model generates a network for a given N, the number of nodes, and a given probability p such that the resulting expected number of arcs equals $\frac{N}{p}$.

The model we propose poses theoretical as well as computational challenges. From a theoretical perspective, a proof that the degree distribution is power law is not available (and, if possible, far from straightforward). We therefore argue through experimental results that there is sufficient remaining structure in our model to claim that the generated network's degree distributions are (close to) power law, and the remaining networks can therefore be reasonably assumed to be scale-free.

In addition, the algorithmic implementation of our model poses implementation challenges. Firstly, in the formal representation of our model, all potential arcs get assigned a weight that may change with every added arc. As a consequence, a straightforward implementation (termed the Base algorithm in Section 3.1) requires considerable effort in updating and storing weights. Secondly, once the weights are determined, selecting the next arc corresponds to weighted random sampling (a specific case of [9]), which in general requires computational effort similar to a linear search.

In this paper, we derive an algorithm that resolves both the problem of updating weights and of the linear search. In fact, the method does not update weights, but instead allows 'pseudo arcs', which are ignored. As a consequence, multiple samples may be required until an arc is successfully added, and we therefore call our method the Multi-sampling algorithm. It will turn out that in none of the situations we encountered the need for resampling negates the gain achieved by not updating. Moreover, because we do not update, we are able to do Reversed Look-up to establish a constant time algorithm for weighted random sampling. Together, these two aspects dramatically speed up the generation of scale-free directed graphs: networks with one million nodes can be generated in seconds.

In what follows we first present our model in Section 2 and demonstrate experimentally its scale free nature in Section 2.1. In Section 3 we then introduce algorithms to generate networks according to our model. Section 4 analyses the complexity of the algorithms, theoretically as well as experimentally.

2 Generation of Scale-Free Networks with Directed Arcs

Preferential attachment refers to a class of network models in which a network is modelled through a process of growth [6,7,8,10]. Starting from a single node, in each iteration a node is added and some links are added according to a specific algorithm. Depending on the precise model, it may then be possible to show that the resulting network has certain properties, such as a power law degree distribution. Bollobás *et al.* [7], for instance, provide a clear articulation of the benefits of using such models to explain properties of networks. By construction, the networks represent an intuitive mapping on 'real-life' processes and networks, explaining why the real-life network may have become scale free. That is, it is not only the fact that mathematically the network can be shown to exhibit scale-free and other properties, it is also the construction that lends credit to its use as a representation of a real system. It is therefore natural, and has been common practice [11] to also use models based on preferential attachment in simulation studies.

For scale-free networks with directed arcs, the literature provides essentially one model based on preferential attachment, developed in [7] and [8]. Both [7] and [8] show that the resulting indegree as well as outdegree node distribution follows a power law distribution. In this model multiple arcs may share identical origin and destination pairs, and arcs may 'loop', that is, arcs may have the same destination as the origin. We will put more restrictions on this model in this paper, but first explain the main idea behind the model of [7] and [8]–we will take the specific model from [8], which is a natural and generally applicable case of the model presented in [7].

We introduce the following notation to represent the growing network generated using these models. In the nth iteration, $n \geq 1$, there are n nodes and a set of arcs. We number the nodes 1 through n, and write $i \to j$ to denote an arc from node i to node j, $i, j \leq n$ and $i \not\to j$ if no arc $i \to j$ exists. The algorithm terminates when there are N nodes, with $N > 0$ some predefined target size of the generated network.

The main idea behind the model in [8] is as follows. Starting from a single node, the algorithm adds in each iteration one node and one or more arcs. Each iteration starts with adding a node and one arc from the new node to an existing node. Then, with probability $1-p$ an additional arc is added between two existing nodes, while with probability p we start the next iteration with a new node. For every node, the model thus adds on the average $\frac{1}{p}$ arcs to the network.

If a new arc is added, it will be from node i to node j depending on the existing indegree and outdegree of each node. The reasoning behind this follows from characteristics of scale-free and small-world properties: nodes with many arcs are more likely to get additional arcs. At this point we modify the model in [7] and [8], by not allowing multiple arcs and loops. In particular, for any node i, let I_i be the in-degree of node i and O_i the out-degree of node i, then weights $w_{i,j}$ are defined as follows:

$$w_{i,j} = \begin{cases} (\mu + O_i)(\lambda + I_j) & \text{if} \quad i \not\to j \text{ and } \ i \neq j \\ 0 & \text{if} \quad i \to j \text{ or } \ i = j \end{cases} \tag{1}$$

The probability $p_{i,j}$ that, given an arc is added to the network, it is between i and j is then given by

$$p_{i,j} = \frac{w_{i,j}}{\sum_{i=1}^{n} \sum_{j=1}^{n} w_{i,j}}. \tag{2}$$

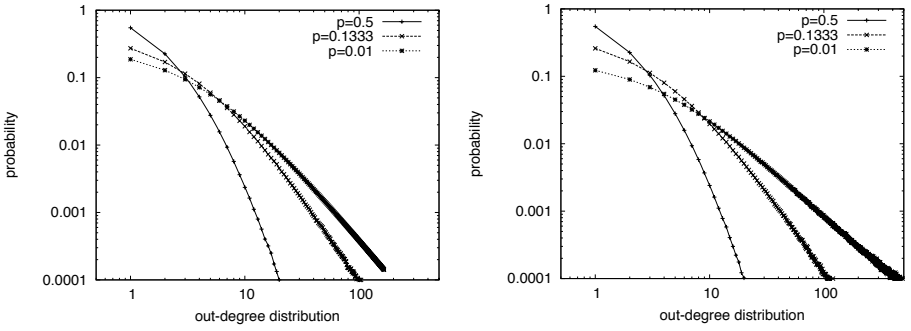
The above implies that arcs between nodes with a high number of incoming or outgoing arcs are more likely than between nodes with low number of arcs. Note that if an arc is added from a new node i to an existing node, that the outdegree O_i of the new node is zero, thus simplifying the weights to $w_{i,j} = \lambda + I_j$ (the constant μ can be omitted).

The main difference between the model based on Equation (1) and [8] is that $w_{i,j}$ is set to 0 in Equation (1) if an arc already exists or if it is a loop. An alternative approach would be to generate a network using the model in [8] and then remove multiple arcs and loops. However, that generates networks with an unpredictable number of arcs, since it is unknown how many arcs are multiple arcs or loops. In our model, we input parameters N and p to generate networks with any given number of nodes N and number of arcs $\frac{N}{p}$.

2.1 Power Law Node Degree Distribution

The first issue to address is whether our model results in power law distributions for indegrees as well as outdegrees. From [7,8] it is known that if multiple arcs and loops are allowed, that then the outdegree and indegree node distributions are power law with parameter $2 + \frac{p(\mu+1)}{1-p}$ and $2 + p\lambda$, respectively. To illustrate this, Figure 1(a) provides the outdegree node distribution for the algorithm in [8], where we use parameters $N = 100,000$, $\lambda = 0.75$ and $\mu = 3.55$, as representative for web hyperlinks [12]. We vary p, where $p = 0.1333$ is the value representative for the network of web hyperlinks. For the log-log scale of Figure 1(a) a power law distribution results in a straight line. The theoretical results in [7,8] require both the total number of nodes and the indegree and outdegree (the outdegree is given on the x-axis of Figure 1(a)) to go to infinity for the Power law distribution to be guaranteed. That is, small values of the indegree and outdegree do not follow the power law distribution, as one can see from Figure 1(a). Note that in particular for $p = 0.5$, which is a very lightly connected network, the convergence to a power law distribution is slow.

Unfortunately, the elegant and straightforward mathematical proof for power law node degree distributions in [7,8] does not extend to our case because we have to account for the cases in which weights are set to 0. This makes that we have not been able to prove the scale free properties of our model. Heuristically, one can argue that the weights that are set to 0 are distributed over the nodes proportional to the number of incoming and outgoing arcs. As a consequence, one may hope that these proportions are such that the remaining likelihood for nodes to be selected as origin or destination is modified in regular manner, thus preserving the power law nature of the node distribution. However, we note that the dependence between indegree and outdegrees of nodes that was



(a) For [8] (b) Using Equation (1)

Fig. 1. Outdegree distributions for the model in [8] and using Equation (1)

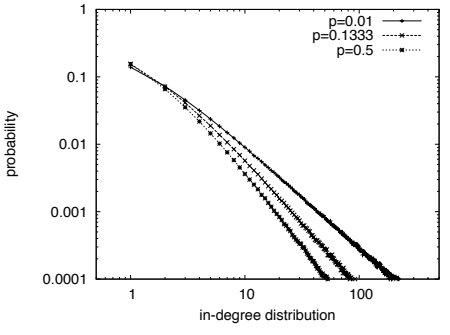


Fig. 2. Indegree distribution using Equation (1)

demonstrated in [8] makes proving this difficult, or may prove the heuristic incorrect. Nevertheless, our experimental results seem to indicate the scale-free nature of the node indegree as well as outdegree distribution is preserved (or at least close to preserved).

We illustrate the resulting node distributions in Figure 1(b) and Figure 2. Of particular importance is the parameter p, since if p gets smaller, the number of arcs increases and hence more weights are set to 0 in Equation (1). As a consequence, one may expect larger difference between Figure 1(b) and Figure 1(a). Indeed, when comparing the two models, we see almost identical results for $p = 0.5$, and more substantial differences for smaller values of p. However, for smaller values of p, the power law characteristic of the node degree distribution is more pronounced, leading us to believe that our model creates node degree distributions that are (not far from) power law distributions.

We will now turn our attention to the algorithmic implementation of our model. To realistically represent modern-day systems, it is not unreasonable to run simulations or do analysis of systems with a million nodes. We will see that the fact that we disallow multiple arcs and loops introduces several aspects that

make it difficult to scale algorithms to such large networks. In this paper we derive several algorithms, using various ideas to reduce computation time as well as memory consumption compared to a 'Base' algorithm–we term these ideas 'Node Weights', 'Node Weights with Subtraction', 'Multi-sampling' and 'Reversed Look-up', respectively, and introduce these in the next section.

3 Generator Algorithms

We will see that a main challenge in making network generation algorithms based on our model scalable lies in the need to update weights $w_{i,j}$ in Equation (1) after every added arc. This issue we address first. A second challenge is the selection of the arc once the weight are updated (weighted random sampling)–this we can only resolve in one specific algorithm, namely Multi-sampling, as we discuss in Section 3.5 when we introduce Reversed Look-up.

For all algorithms we will compute the time and memory complexity. In the complexity analysis we ignore updating of the indegree and outdegree of a node with every added arc–this has to be done in all algorithms. Similarly, we do not consider the storage in memory of the actual networks with all its nodes and arcs. This also has to be done in all cases, and it should be noted that storage of the N nodes and $\frac{N}{p}$ arcs is of dominant order in all but the Base algorithm.

3.1 Base Algorithm

The Base algorithm is straightforward: store all elements $w_{i,j}$ in a matrix of size $N \times N$ and update these weights after every addition of an arc. In particular, if arc $x \to y$ is added, then $w_{x,j}$ needs to be updated for $j = 1, \ldots, n$, and $w_{i,y}$ needs to be updated for $i = 1, \ldots, n$. That is, a complete row and a complete column in the matrix needs to be updated. In addition, $w_{x,y}$ needs to be set to 0.

To make this precise, we introduce the superscript $+$ to denote the updated weights when an arc is added. Similarly, we represent the increase of the outdegree of a node i as O_i^+ and the increase of the indegree of node j as I_j^+. Assume that $x \to y$ is the last arc added, then $O_i^+ = O_i + 1$ if and only if $i = x$ and $O_i^+ = O_i$ otherwise. Similarly, $I_j^+ = I_j + 1$ if and only if $j = y$ and $I_j^+ = I_j$ otherwise. Hence, when $x \to y$ is the added arc, the weights in Equation (1) need to be updated as follows:

$$w_{i,j}^+ = \begin{cases} w_{i,j} + \lambda + I_j & \text{if} \quad i = x \text{ and} \quad i \not\to j \\ w_{i,j} + \mu + O_i & \text{if} \quad j = y \text{ and} \quad i \not\to j \\ 0 & \text{if} \quad i = x \text{ and} \quad j = y \\ w_{i,j} & \text{otherwise} \end{cases} \tag{3}$$

This process of updating weights has the following time complexity. If an arc is added at iteration n, there are up to $2n - 1$ (a row and a column) weights

updated. Since at each iteration $\frac{1}{p}$ arcs are added, the total time complexity is $\frac{1}{p}\sum_{n=1}^{N}(2n-1) = O(\frac{N^2}{p})$ updates. A matrix is used to store the weights, thus requiring $O(N^2)$ storage.

The second aspect to be considered is the time it takes to draw a weighted random number according to the probabilities in Equation (2). The base algorithm for weighted random sampling is to draw a random number r between 0 and 1 and add up probabilities $p_{i,j}$ in Equation (2) in order until the sum exceeds r. ('In order' may for instance be implemented through a double `for` loop: `for (i=1) to n do { for (j=1) to n do{...}}`)

This way of weighted random number generation has complexity similar to a linear search through a list of size $n \times n$: it requires on average $\frac{1}{p}\frac{n^2}{2}$ operations at iteration n, and summing this leads to the results for pyramid numbers, which implies that the resulting time complexity is $O(\frac{N^3}{p})$ operations in total. There is no additional required memory for this way of drawing weighted random numbers.

We note that some algorithmic tricks can be thought off to speed up the drawing of weighted random numbers, such as traversing the matrix backward when the random number is larger than 0.5 (for instance). This, however, does not change the order of the algorithm. Similarly, as we already remarked, if a new node is added, the outdegree of the new node is 0, and hence the weights in Equation (1) simplify. We exploit this in our implementations to make the algorithms more efficient when adding a node, but the order of the algorithm does not change because of it. We therefore will not discuss such issues in more detail.

3.2 Node Weights

Since the Base algorithm stores the complete matrix of weights its memory requirement of $O(N^2)$ makes it less attractive for a large network size. Roughly speaking, modern day personal computers may be expected to hold up to 10^9 doubles in memory, thus limiting N to about 30,000.

The Node Weights method resolves this issue, by storing and updating weights per node, instead of per arc. This immediately implies that storage requirements will go down to $O(N)$. In particular, at iteration n, for each node i we store

$$w_i = \sum_{j=1}^{n} w_{i,j}. \tag{4}$$

The probability p_i that, given an arc is added to the network, it has i as the origin is then given by:

$$p_i = \frac{w_i}{\sum_{i=1}^{n} w_i}. \tag{5}$$

Furthermore, once the origin is determined, the destination is determined by computing $w_{i,j}$ on the fly for given origin node i.

Important is that the node weights w_i can be updated without knowing the individual values $w_{i,j}$, because otherwise there would be no gain from maintaining node weights. This works in a similar manner as for the Base algorithm: if

arc $x \to y$ is added we have again that $O_x^+ = O_x + 1$ and $I_y^+ = I_y + 1$, and we derive that the updated node weights w_i^+ obey:

$$w_i^+ = \begin{cases} w_x + \sum_{j|x \not\to j}(\lambda + I_j) - w_{x,y} & \text{if} \quad i = x \\ w_i + \mu + O_i & \text{if} \quad i \not\to y \text{ and } i \neq x, y \\ w_i & \text{otherwise} \end{cases} \tag{6}$$

Updating the weights by above equations takes order n operations for each arc in the nth iteration, thus giving $O(\frac{N^2}{p})$ overall complexity for updating, as in the base case. However, selection of an arc through weighted random sampling is an order less expensive than in the base case. A sequential search is used to find the origin node, and for this origin node all possible destination nodes are considered. (Again, we sum up probabilities p_i and then probabilities $p_{i,j}$ until they sum to r. To make this more precise would lead to cumbersome explanation not necessary for the thrust of this paper.) As we remarked, for the chosen node i, we generate the weights $w_{i,j}$ on the fly from Equation (1) since we do not store the individual weights. The time complexity for the weighted random sampling in the Node Weights algorithm is thus $O(\frac{N^2}{p})$.

3.3 Node Weights with Subtraction

Node Weights with Subtraction is a variation of Node Weights in which we decrease the number of updates. From Equation (6) one sees that weights w_i are updated for every node i that is not connected to y ($i \not\to y$). In Node Weights with Subtraction we do the opposite, and update w_i if and only if $i \to y$. The main observation behind the method is that the node weights in Equation (4) can be rewritten as:

$$w_i = \sum_{j=1}^{n} w_{i,j} = \sum_{j=1|i \not\to j, i \neq j}^{n} (\mu + O_i)(\lambda + I_j)$$

$$= \sum_{j=1}^{n} (\mu + O_i)(\lambda + I_j) - \sum_{j=1|i \to j \vee i = j}^{n} (\mu + O_i)(\lambda + I_j)$$

$$= (\mu + O_i)(n\lambda + A_n) - \sum_{j=1|i \to j \vee i = j}^{n} (\mu + O_i)(\lambda + I_j),$$

where A_n is the total number of arcs at iteration n. For each node, we then keep track of the term $(\mu + O_i)(n\lambda + A_n)$ (which we can easily track and update) as well as of $\sum_{j=1|i \to j \vee i = j}^{n} (\mu + O_i)(\lambda + I_j)$. Since the latter term has fewer elements in the sum it is less effort to update that term than it is to update the actual values w_i (as in the Node Weights method).

We will not write down the equivalent of Equation (6) to update the elements

$$\sum_{j=1|i \to j \vee i = j}^{n} (\mu + O_i)(\lambda + I_j)$$

Note that although it can be expected that the Node Weights with Subtraction method is more efficient than Node Weights, the time and memory complexity orders do not change.

3.4 Multi-sampling

The idea behind Multi-sampling is radically different from the previous approaches in that updates are no longer carried out. Instead of enforcing that no multiple arcs or loops will be generated by setting weights to 0, Multi-sampling allows an arc to be selected that leads to multiple arcs or a loop, but then ignores it and retries the sampling for an arc. We will see that this implies a constant computational complexity for updates in each iteration, but that it increases the number of samples, thus resulting in $O(\frac{N}{fp})$ computational complexity, where $\frac{1}{f}$ is the average number of samples per iteration.

The algorithm works as follows. Introduce, for $i = 1, \ldots, n$ the following weights:

$$\begin{aligned} v_i &= \mu + O_i, \\ w_j &= \lambda + I_j. \end{aligned} \tag{7}$$

Then generate an arc $x \to y$ by conducting weighted random sampling using the weights v_i to determine x and by conducting weighted random sampling using the weights w_i to determine y. If $x \to y$ already exists or if $x = y$, then repeat the procedure until a new arc is added.

We now have to show that this procedure correctly implements our model, i.e., that it generates probabilistically equivalent networks as when the probability of adding an arc $x \to y$ is given by Equation (2). This follows directly from considering, for the Multi-sampling method, the conditional probability $p^{\text{MS}}_{x,y}$:

$$\begin{aligned} p^{\text{MS}}_{x,y} &= Prob\{\text{arc } x \to y \text{ is added } | \text{an arc is added}\} \\ &= \frac{Prob\{\text{arc } x \to y \text{ is added } \wedge \text{an arc is added}\}}{Prob\{\text{an arc is added}\}} \end{aligned}$$

If $x \to y$ already exist, then an arc is not added, and so the numerator evaluates to false, resulting in $p^{\text{MS}}_{x,y} = 0$ if $x \to y$ already exists. Similar, if $x = y$, an arc is not added and $p^{\text{MS}}_{x,y} = 0$. If $x \to y$ does not yet exist, then an arc is added and the numerator becomes

$$Prob\{\text{arc } x \to y \text{ is added}\} = \frac{v_i w_j}{\sum_{i=1}^{n} \sum_{j=1}^{n} v_i w_j}.$$

The denominator equals:

$$Prob\{\text{an arc is added}\} = \frac{\sum_{i=1}^{n} \sum_{j=1|i\neq j, i\neq j}^{n} v_i w_j}{\sum_{i=1}^{n} \sum_{j=1}^{n} v_i w_j}.$$

As a result we obtain for the conditional probability that given an arc is added, it is arc $x \to y$:

$$p^{\text{MS}}_{x,y} = \frac{v_i w_j}{\sum_{i=1}^{n} \sum_{j=1|i\not\to j, i\neq j}^{n} v_i w_j}. \tag{8}$$

Hence, filling in $w_{i,j}$ in Equation (2) and v_i and w_j in Equation (8) we find that probabilistically Multi-sampling generates networks consistent with our model.

For the performance of the Multi-sampling method it will be important to determine the amount of resampling that is required. Every time an arc $x \to y$ is selected that already exists or $x = y$, a new attempt must be made (which involves drawing a new random number, and selecting an arc according to the procedure under Equation (7)). To determine the average number of resamples, let f be the probability the sample is successful:

$$f = \frac{\sum_{i=1}^{n} \sum_{j=1|i\not\to j, i\neq j}^{n} v_i w_j}{\sum_{i=1}^{n} \sum_{j=1}^{n} v_i w_j}. \tag{9}$$

Then the average number of tries until a sample is successful equals $\sum_{k=1}^{\infty} k(1-f)^{k-1} f = \frac{1}{f}$. Obviously, the required number of samples gets high if the success probability f is small. Since f is a potential bottleneck for the Multi-sampling methods Section 4.3 shows experimental results for f.

Finally, we note that the storage requirements for Multi-sampling only involve maintaining weights v_i and w_j in Equation (7). These weights can easily be computed from the already stored in- and out-degrees, and therefore there is no specific storage needed for the weights.

3.5 Reversed Look-Up

Thus far the algorithms have dealt with the issue of updating weights, either decreasing the storage needed for the weights or the time required for updates. However, considerable computational effort is also required to sample weighted random numbers once the weights are established. For the Multi-sampling approach, however, the weights are such that one can store the weights in such a way that, given a random number, the appropriate weighted random number can directly be read from the data structure.

We call this idea 'Reversed Look-up', and it has its origin in a commonly proposed algorithm for weighted random sampling if all weights have integer values (see for instance [13] and also the BA algorithm implementation in Peersim [11] for examples of this and related ideas). In our case, following Equation (7), in iteration n weighted random sampling selects node i with probability p_i defined as:

$$p_i = \frac{v_i}{\sum_{i=1}^{n} v_i} = \frac{\mu + O_i}{\sum_{i=1}^{n} (\mu + O_i)} = \frac{\mu + O_i}{n\mu + A_n}, \tag{10}$$

where A_n is the total number of arcs at iteration n. We will now first deal with the constants μ, before applying Reversed Look-up to the integer-valued outdegrees O_i.

Let r be a random number between 0 and 1. To deal with the constant μ, we select node $x = 1, \ldots, n$, if $\frac{\mu(x-1)}{n\mu+A_n} \leq r < \frac{\mu x}{n\mu+A_n}$. This means that if $r < \frac{n\mu}{n\mu+A_n}$ the origin node for the new arc is selected uniformly from all n nodes, since μ contributes the same constant value to any probability p_i. If, on the other hand, $r \geq \frac{n\mu}{n\mu+A_n}$, the outgoing arc is not yet decided and the Reversed Look-up comes into effect, as follows.

We maintain an array of size A_n with integer values, such that in each array element a node number is stored. More precisely, the array has O_i elements with value i. We then select any of the array elements with equal probability–since there are O_i array elements for node i the likelihood that a node is chosen is proportional to O_i (we make the correctness argument precise below).

To select an array element according to a uniform distribution, we first scale up the random number $r \geq \frac{n\mu}{n\mu+A_n}$ so it is a uniformly distributed number between 0 and A_n: $r_{\text{new}} = r * (n\mu + A_n) - n\mu$. Then we select array index k as $k = \lfloor r_{\text{new}} \rfloor$ (the floor operator $\lfloor \rfloor$ indicating rounding to the nearest lower integer). The origin node is then the value of the array element at index k.

Once the origin node is determined, the destination node is determined similarly by maintaining an array of $A_N = \frac{N}{p}$ elements with nodes based on indegrees I_j.

To demonstrate the correctness of the approach, node x is selected according to a uniform distribution (that is, with probability $\frac{1}{n}$) if $r < \frac{n\mu}{n\mu+A_n}$ and with probability $\frac{O_x}{A_n}$ if $r \geq \frac{n\mu}{n\mu+A_n}$. Together, this means that node x is the origin with probability $\frac{1}{n}\frac{n\mu}{n\mu+A_n} + \frac{O_x}{A_n}(1 - \frac{n\mu}{n\mu+A_n}) = \frac{\mu+O_x}{n\mu+A_n}$, which confirms to Equation (10).

This idea of Reversed Look-up works because of the integer value of the weights and because updates can be implemented very simply. For instance, it is not easy (if at all possible) to efficiently implement Reversed Look-up when weights may decrease, such as in the Node Weights method. When using Multi-sampling updating the array is straightforward. Assume arc $x \rightarrow y$ is last added, then we add to the array for indegrees one element with value y and to the array with outdegrees one element with value x.

The time complexity of the Reversed Look-up method is minimal. To update the array and read the right element from the array there are some operations each time an arc is added, resulting in time complexity $O(\frac{N}{p})$. Importantly, memory use is increased through the use of two arrays of $O(\frac{N}{p})$ elements.

We will see in the results section that the ability to use Reversed Look-up in the Multi-sampling approach dramatically reduces the computational effort to generate scale-free directed networks. We will see that networks with a million or more nodes are feasible.

4 Evaluation of Generator Algorithms

Before discussing our experimental results, let us recap the theoretical complexity results we derived in Section 3, by comparing the order of all algorithms in Table 1, for CPU time and memory consumption, respectively. In Table 1

Table 1. Orders of CPU and memory use for various algorithms

	CPU Time		Memory Use		
Algorithm	updating weights	selecting the arc	weights	selecting the arc	network
Base	$\frac{N^2}{p}$	$\frac{N^3}{p}$	N^2	0	$\frac{N}{p}$
Node Weights	$\frac{N^2}{p}$	$\frac{N^2}{p}$	N	0	$\frac{N}{p}$
Node Weights with Subtraction	$\frac{N^2}{p}$	$\frac{N^2}{p}$	N	0	$\frac{N}{p}$
Multi-sampling	0	$\frac{N}{fp}$	0	0	$\frac{N}{p}$
Multi-sampling with Reversed Look-up	0	$\frac{N}{fp}$	0	$\frac{N}{p}$	$\frac{N}{p}$

the time complexity is divided in two: the time an algorithm takes in updating weights, and the time an algorithm takes in determining the correct arc to add based on weighted random sampling. Note that the CPU time complexity does not include aspects that are required in all algorithms, such as updates of indegrees and outdegrees. The memory use is given for the same two aspects: storing weights and memory used for selecting the arc through weighted random sampling. The table also shows the memory requirement for the generated network itself, since this is a dominant factor in the memory use of all algorithms.

We see from Table 1 in the first two columns on CPU time that we may expect that Multi-sampling will outperform the Base and Node Weight methods, unless the probability f becomes too small (f is the probability resampling is not needed, and thus $\frac{1}{f}$ is the expected number of samples for each added arc). We will show in our experiments that Multi-sampling is indeed the preferred method, and we will also experimentally show that the number of retries $\frac{1}{f}$ decreases with the number of iterations to a small, almost constant, value. The latter is important, since otherwise the method would break down because of excessive resampling.

We also see from the table that a potential bottleneck exists for Multi-sampling with Reversed Look-up in the use of memory to select the arc using weighted random sampling. After all, the point of the Reversed Look-up method was to trade memory for CPU. However, we will see that this use of memory is of the same order as that for storing the generated network itself, something all algorithms need to do. This implies that the base and Node Weight methods never really are competitive compared to Multi-sampling with Reversed Look-up: even if memory use increases in Multi-sampling with Reversed Look-up, the time the other algorithms take leaves them unattractive.

In what follows we analyse the results of our experiments. All the methods are implemented and executed within the Java Peersim simulation environment for peer-to-peer networks [11]. As a general-purpose p2p simulation environment Peersim is concerned with more than efficiency alone and one may expect some performance or memory usage overhead compared to bespoke implementations. Nevertheless, we are convinced that our implementation and experiments are indicative for typical use of the algorithms we developed.

To achieve fair results, all the experiments were conducted on the same machine. It has a Pentium(R) processor, with CPU 3GHz and 2GB of RAM. Effectively, we were able to use up to about 1.4GB of memory in our experiments. All experiments were repeated up to fifty times to create tight confidence intervals. In general, results did not show much variance. Even in cases that computation time for each data point was too high (up to one day) to do many experiments, we still achieved relatively stable results. We added time stamps to the code to measure the time the algorithm needs to generate the networks. To measure the memory usage, we use common Java methods–because memory allocation is managed by the Java Virtual Machine, the results may be influenced by the working of the JVM. However, we will see that the results can be satisfactory explained from our understanding of the working of the algorithms and implementation.

4.1 Baseline Performance Comparison

We first compare the performance of the various algorithms for typical settings. Since discrete-event simulation studies of peer-to-peer algorithms often concern networks of some ten thousands of nodes (e.g., [1,3]), we vary the network size N from $10,000$ to $50,000$ nodes. In addition, we use the parameter values derived in [8] for the world-wide web (in turn attributed to data from [12]): $p = 0.1333$, $\lambda = 0.75$ and $\mu = 3.55$. Note that the values of λ and μ do only influence the CPU or memory use through the probability f in (9), but that p is a very important factor for all metrics and methods.

Figure 3 shows CPU time for the various methods. To simplify the understanding of the figures, we note that we always label the curves in the order they appear in the graph (from top to bottom). In this case, the Base method only generated a single point for $N = 10,000$. The Base method does not complete for larger values of N because it runs out of memory. The Base method thus clearly performs worst. The two increasing curves in Figure 3 are for Node Weights without and with Subtraction, respectively. In fact, the increase is roughly quadratic in the number of nodes, as we expect from the complexity results in Table 1. Finally, the two Multi-sampling approaches easily outperform the others, both demonstrating an almost flat line.

The memory consumption for the different methods is displayed in Figure 4(a) (notice the logarithmic scale) and Figure 4(b) (in linear scale). One sees that the Base method indeed runs out of memory. As we remarked, we can effectively use up to 1.4GB of memory, and for $N = 10,000$ the Base method uses close to 1.0GB already. The other approaches all exhibit similar memory consumption–this can explained from the fact that memory use in all cases is of order $\frac{N}{p}$,

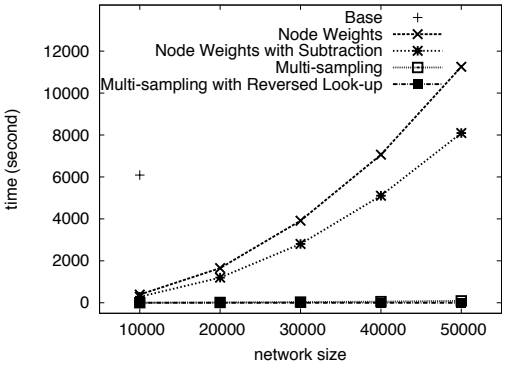
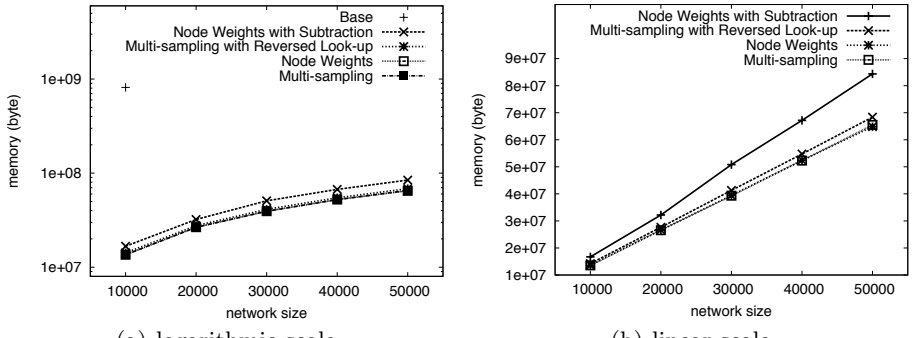


Fig. 3. CPU time for average networks ($p = 0.1333$)



(a) logarithmic scale (b) linear scale

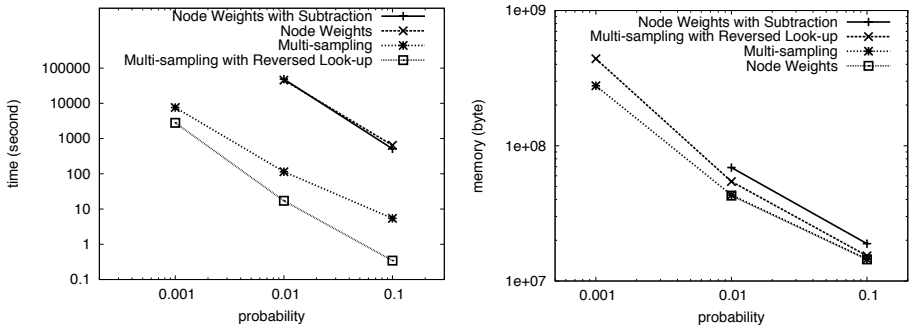
Fig. 4. Memory use for average networks($p = 0.1333$, logarithmic scale and linear scale)

dominated by the storage of the network itself. Note that Multi-sampling with Reversed Look-up consumes slightly more memory than either Multi-sampling or Node Weights, but in essence all four methods are comparable.

In conclusion, we see that for a common range of parameter values, Multi-sampling with or without Reversed Look-up clearly outperform the other methods with respect to the required computation time. Although the Node Weights method is competitive when considering memory use, it does not dramatically improve over either Multi-sampling method. Therefore, the results suggest that the choice is be between the two variations of Multi-sampling. When generating networks with a million or more nodes in Section 4.4, we will discuss in more detail the CPU and memory implications of using Reversed Look-up or not (see Figure 8(a) and 8(b)).

4.2 Highly Connected Networks (p Small)

Since the complexity numbers in Table 1 all tend to infinity when the probability $p \downarrow 0$ we now research the performance of the various approaches when p gets



(a) CPU time (b) Memory use

Fig. 5. CPU time and memory use for highly connected networks ($N = 10,000$)

small. For $N = 10,000$ Figures 5(a) and 5(b) show the CPU and memory results, respectively, for small values of p (note the logarithmic scales of the figures). For a network with N nodes, the average number of arcs is $\frac{N}{p}$ and the maximum number of arcs is $N(N-1)$. We thus see that realistically one should have p considerable larger than $\frac{1}{N-1}$. In our network of size $N = 10,000$ we let p get as small as 0.001.

Figure 5(a) shows that only Multi-sampling (with or without Reversed Look-up) can generate networks with $p = 0.001$. For instance, the Node Weight methods require more than 10 hours to generate networks for $p = 0.01$, and networks for smaller p can therefore not be generated in practice.

Even though Figure 5(b) demonstrates that the Node Weights method uses as little memory as Multi-sampling, we see that the conclusions from Section 4.1 do not change significantly compared to the results for small p values in this section. Multi-sampling is so fast that the Node Weight methods cannot compete, even though Node Weight is memory efficient. Note that based on Table 1 this was not a foregone conclusion because of the unknown implications of the probability f (which relates to the required amount of resampling) in the Multi-sampling methods. We study this further in the next section.

4.3 Amount of Resampling in Multi-sampling Methods

Table 1 shows the dependence of the CPU time needed for Multi-sampling methods on the probability f, which is the probability an arc is successfully added, as given in Equation (9). Unfortunately, we do not have expressions or convergence results for f. We therefore experimentally investigate f as a function of the iteration number.

We generate networks of up to 180,000 nodes, count the number of resamples over intervals of 20,000 nodes and divide it by the number of arcs added in these intervals. In so doing, we obtain $\frac{1}{f} - 1$, the number of 'wasted' resamples for each successfully added arc. (We note that using the Node Weight methods we can numerically compute f precisely for any network, but the Node Weight method

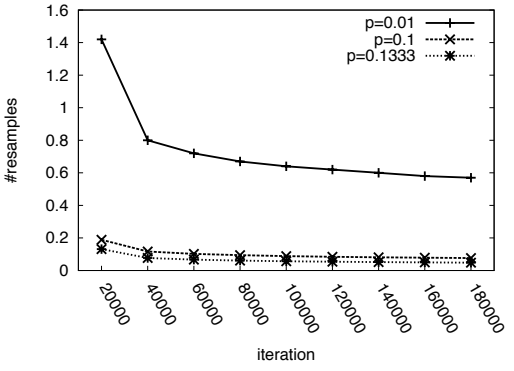


Fig. 6. Number of resamples

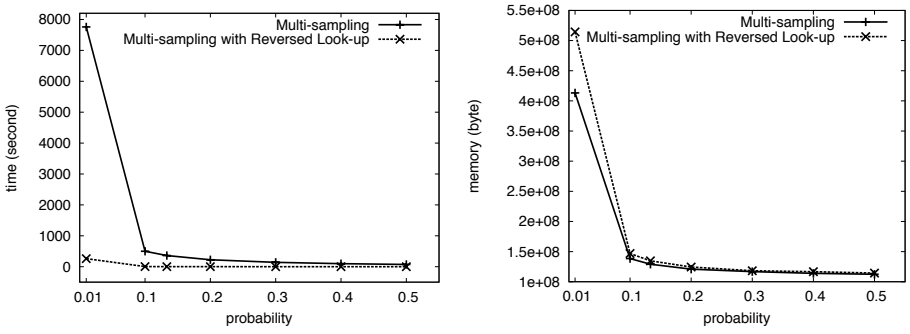
is prohibitively slow for the small values of p and large N considered. Hence, we used Multi-sampling with Reversed Look-up and sampled $\frac{1}{f}$, repeating the experiment sufficiently often to gain tight enough confidence intervals.)

Figure 6 presents the average number of resamples as a function of the iteration count. We show a curve for $p = 0.01$, $p = 0.1$ and $p = 0.1333$ (as in Subsection 4.1). Clearly, f depends very much on p, but in all cases the number of wasted samples is small. For instance, for $p = 0.01$ the number of resamples is less than 1 per successfully added arc, while for more regular values such as $p = 0.1333$ the number of resamples is even less.

Importantly, however, all curves decrease as a function of the iteration count. This implies that for larger network the danger decreases that f becomes a bottleneck. This probably can be explained from the fact that with increasing iteration count n the number of existing arcs $\frac{n}{p}$ becomes less and less significant compared to the number $n(n-1) - \frac{n}{p}$ of not yet existing arcs. However, this depends on the actual values of the weights corresponding to existing arcs, so we only suggest it as a possible explanation that remains to be proven. Clearly, it would be of great interest to derive theoretical results for f or its convergence, but this is beyond the scope of our current presentation.

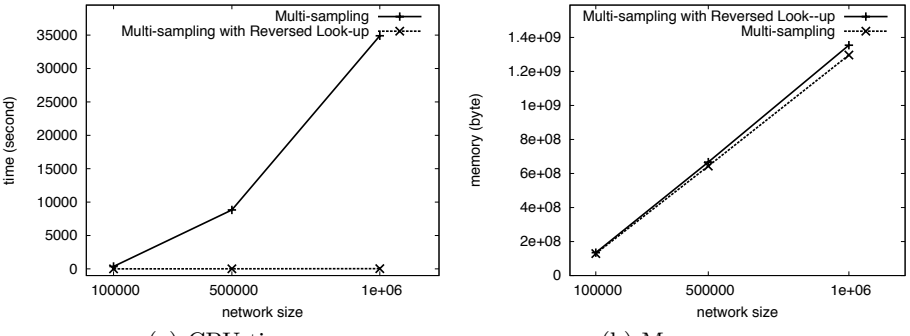
4.4 Networks with a Million Nodes

Finally, we want to push the algorithms and generate as large a network as we can using our current implementation in Peersim. We have already seen that only the Multi-sampling methods should be considered for average sized networks, and have identified that Multi-sampling with Reversed Look-up is superior in time, while plain Multi-sampling is more memory efficient. Figure 7(a) and Figure 7(b) confirm this for $N = 100,000$, and different values of p. More precisely, Reversed Look-up takes about 20% more memory for the whole range of p values (200MB for $p = 0.01$). On the other hand, computation time for Multi-sampling gets considerably worse, and can differ more than a factor 10. So, an early conclusion would be to exclusively use Multi-sampling with Reversed Look-up.



(a) CPU time (b) Memory use

Fig. 7. CPU time and memory use for the Multi-sampling variants ($N = 100,000$)



(a) CPU time (b) Memory use

Fig. 8. CPU time and memory use for large networks ($p = 0.1333$)

We see in Figure 8(a) and 8(b) how far we can push the two Multi-sampling methods. As far as CPU use is concerned Reversed Look-up is clearly beneficial, since Multi-sampling without Reversed Look-up takes close to 10 hours (Figure 8(a)). Considering memory usage, Figure 8(b) shows that the two methods do not differ too much. Moreover, with 1 million nodes we are exactly at the limit of the available memory of about 1.4GB. In other words, Multi-sampling with Reversed Look-up in the preferred approach, allowing us to generate networks with up to one million nodes in seconds (to be precise, 32 seconds for 1 million nodes).

5 Conclusion

This paper proposes a network model for scale-free directed networks without multiple arcs and loops, for any given number of nodes and average number of arcs. If generated efficiently, such networks are useful for analysis of peer-to-peer algorithms using discrete-event simulation. We experimentally demonstrated that the networks resulting from our model have node degree distributions that

are (close to) power law, and thus that the resulting networks are scale-free. Further formal analysis would be of interest to analyse the extent to which certain properties hold for the proposed model and to determine aspects such as distribution parameters. The paper is particularly focused on the development of fast algorithms that allow the model to be effectively used in discrete-event simulation studies. We have derived an approach termed Multi-sampling with Reversed Look-up that under almost all circumstances outperforms other methods. Experimentally, we have shown that the amount of resampling required in the method is bounded and does not significantly reduce the applicability of the method across a broad parameter range. In addition, although the method requires additional memory to speed up the process of weighted random sampling, memory use does not much exceed that for maintaining the network itself (necessary for all algorithms). As a consequence, using Multi-sampling with Reversed Look-up one can generate networks with a million or more nodes within seconds on current-day desktops.

References

1. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks. In: ICS 2002: Proceedings of the 16th International Conference on Supercomputing, pp. 84–95. ACM Press, New York (2002)
2. Ribeiro de Mello, E., van Moorsel, A.P.A., da Silva Fraga, J.: Evaluation of P2P search algorithms for discovering trust paths. In: European Performance Engineering Workshop, pp. 112–124 (2007)
3. Zhang, H., van Moorsel, A.P.A.: Evaluation of P2P algorithms for probabilistic trust inference in a web of trust. In: Thomas, N., Juiz, C. (eds.) EPEW 2008. LNCS, vol. 5261, pp. 242–256. Springer, Heidelberg (2008)
4. Newman, M.E.J.: Power laws, Pareto+ distributions and Zipf's law. Contemporary Physics 46(323) (2005)
5. Caldarelli, G.: Scale-Free Networks: Complex Webs in Nature and Technology. Oxford University Press, Oxford (2007)
6. Albert, R., Barabási, A.: Statistical mechanics of complex networks. Reviews of Modern Physics 74 (2002)
7. Bollobás, B., Borgs, C., Chayes, J., Riordan, O.: Directed scale-free graphs. In: SODA 2003: Proceedings of the Fourteenth Annual ACM–SIAM Symposium on Discrete Algorithms, pp. 132–139 (2003)
8. Krapivsky, P.L., Rodgers, G.J., Redner, S.: Degree distributions of growing networks. Physical Review Letters 86, 5401 (2001)
9. Efraimidis, P.S., Spirakis, P.G.: Weighted random sampling with a reservoir. Inf. Process. Lett., 181–185 (2006)
10. Barabasi, A.L., Albert, R.: Emergence of scaling in random networks. Science 286, 509 (1999)
11. Jesi, G.P.: Peersim: A peer-to-peer simulator (2004), http://peersim.sourceforge.net
12. Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., Wiener, J.: Graph structure in the web. Comput. Netw. 33(1-6), 309–320 (2000)
13. jimt: Efficiently selecting a random, weighted element, http://www.perlmonks.org/?node_id=577433

A More Realistic Peer-to-Peer Grid Market Model

Uli Harder and Fernando Martínez Ortuño*

Department of Computing, Imperial College London
Huxley Building, 180 Queens Gate, London SW7 2RH, UK
{uh,fermaror}@doc.ic.ac.uk
http://aesop.doc.ic.ac.uk/

Abstract. In this paper we investigate the behaviour of a peer to peer driven Grid computing network. We present mean field approximation of the simulation model and show that it captures the essence of the model. In contrast to earlier work we limit the budget of the nodes and observe the consequences for the price development. We also show that the proposed peer to peer network scales much better than a central server approach. By allowing the agents in the simulation to hibernate and change the price individually only using local information the price development of the model becomes stable. Lastly we investigate the distribution of the times the agents wait to sell or buy resources.

1 Introduction

Using economic ideas for the management of computer systems has a long history. Some of the earliest work usually cited is that by Greenberger [1], Sutherland [2] and Nielsen [3]. In recent years distributed computing has embraced the idea of Grid computing [4]. Micro and macro-economic ideas have been proposed for the Grid as management and scheduling framework by for example POPCORN [5], TYCOON [6], Wolski et al. [7] and Buyya's *Grid Economy* [8]. In this paper we focus on MaGoG which has been proposed by Richardson et al. [9]. MaGoG is a peer to peer based Grid Computing scheme where real money is exchanged between parties. Peer to peer technology is used in Grid technologies like P-Grid [10] but without use of economic incentives. We explore under which conditions a stable market develops for Computing Power using the MaGoG scheme using a multi-agent simulation. In the past multi agent simulations have been used to model for instance privatised electricity markets [11]. We also show that the MaGoG system scales better than a centralised system and we investigate the waiting time distribution for its agents.

2 The Simulation Model

We try to keep our model as close as possible to the actual MaGoG system [9]. As in our publication [12] we use *igraph* [13] to create an overlay network. These

* This work is partially supported by an EPSRC grant (EP/D061717/1).

J.T. Bradley (Ed.): EPEW 2009, LNCS 5652, pp. 149–154, 2009.

networks are either of Erdős-Rényi [14] or Barabási-Albert [15] type. The latter is meant to be a good representation of social networks [16]. For a chosen network we allocate each node to be either a buyer or seller of a resource. Nodes use the network links bi-directionally to receive and transmit messages. In our previous work [12] we found that price increase was in some cases exponential which is unrealistic in a closed system. Here, we limit the amount of money buyers can bid for. Buyers have a budget that gets replenished after B time epochs, if they have used all their funds before then they cannot bid anymore until their funds are filled up again. This way we limit the amount of money available to buyers in the system. In addition we use *integer* money which has a smallest unit, so that buyers cannot make infinitesimally small bids. To match up deal partners we use the algorithm applied by the Australian Securities Exchange to decide the opening price at the beginning of a trading session[1]. Aspects like the time to live for messages (TTL), bidirectional message flooding, the concept of epochs, *pubs* and price changes after deals have expired have not been changed from [12].

Motivated by the results of the mean field approximation we introduce two new features to the agents in the simulation: hibernation and the size of the price change Δ. The nodes estimate the current proportion of active buyers and sellers in the network, by monitoring the messages they forward. A node goes into hibernation if there is a surplus of nodes of its own kind. In hibernation the node is in a virtual satisfied state to change the buyer/seller ratio dynamically. The node also changes the size of Δ of the bid/ask price $p' = (1 \pm \Delta)p$ using local information from the prices of the deals that are closed in its pub. Following the results of the analytic model, the node increases its Δ when prices increase (the buyer/seller ratio increases) and reduces its Δ when price decreases (the buyer/seller ratio decreases), trying to reach a condition for stable prices.

3 A Mean Field Approximation

Assuming the network is fully connected, one can make arguments to find that the price of a deal after n steps is given by

$$P(n) = P_0(\delta_+^b \delta_-^{1-b})^n \tag{1}$$

where P_0 is the initial price and $\delta_\pm = 1 \pm \Delta$. The price change Δ ranges between 0 and 1. The buyer to seller ratio b ranges from 0 to 1 inclusively.

[1] The algorithm is based on the application of four principles: 1) determine the maximum executable volume and establish the price at which maximum volume will be executed. 2) If there are multiple prices that accomplish 1) establish the minimum surplus, which determines, from the pre-selected prices by 1), which one will leave the minimum quantity of unmatched orders. 3) If there are still several possible prices, ascertain where market pressure exists. 4) If this is not enough to clear the orders use a past reference price.
http://www.asx.com.au/resources/education/basics/open_Close.htm

This expression is found by considering the following. Time is synchronous for all the network. At the beginning ($n = 0$), all sellers in the network start asking an initial price of P_0, and all buyers start bidding P_0. In every time step, nodes try to find a deal. A node finds a deal when, at the requested price, there is enough demand (or supply) provided by the other nodes. If a node is successful in a time step, it will change its ask/bid price in its own interest for the next time step, i.e., the new ask price of a seller will be its former ask price multiplied by δ_+ and the new bid price of a buyer will be its former one multiplied by δ_-. If a node does not manage to find a deal in a time step, it will change its ask/bid price against its own interest for the next time step, i.e., the new ask price of a seller will be its former ask price multiplied by δ_- and the new bid price of a buyer will be its former bid price multiplied by δ_+. Following this pattern in every time step, one can deduce that the price of the deals that are made in the network is given by expression 1.

So, obviously the system's development with respect to the price depends on the expression in the bracket of 1, which from now on we define as $F = F(\Delta, b) = \delta_+^b \delta_-^{1-b}$. The price $P(n)$ tends to the initial price P_0 for $F = 1$ and to either zero or infinity otherwise. We can calculate the critical value of $b(\Delta)$ for which F is one.

$$b(\Delta) = \log(1 - \Delta)/(\log(1 - \Delta) - \log(1 + \Delta)) \tag{2}$$

This is shown in figure 1. The buyer/seller ratio has to be bigger than 0.5 to achieve stable prices. This insight will be used in the next section to change the ratio dynamically with hibernation and individual price changes. We have run simulations to validate the mean field approximation for different ratios of buyers and sellers, as well as for a wide range of Δ values. One typical example is a situation with 60% buyers and 40% sellers, in this case the analytic price is given by $P(n) = P_0[F(\Delta, 0.6)]^n$. In figure 1 we plot $F^{10}(\Delta, 0.6)$ and one can see that for $0 < \Delta < 0.3894$ the price will tend to zero and for $0.3894 < \Delta < 1$ the price will tend to infinity. For $\Delta = 0.3894$, the price will remain at the initial price. In a simulation, for a Barabási Albert network with 16,384 nodes the final price of the simulations for $\Delta > 0.5$ tends to one, which is lowest possible in the simulation. And for $\Delta \leq 0.4$ the price tends to maximum possible, due to budget restraints. So, the simulation has its critical point for the final price at a similar value of Δ as the mean field approximation.

4 Simulations with Adaptive Agents and a Comparison with a Centralised Model

As described in section 2 we change the model presented in [12] to include agent features that allow hibernation and a change of the price change for each agent. With these two new features, each node in the network tries to push the system towards a stable price by using only its local information.With this new

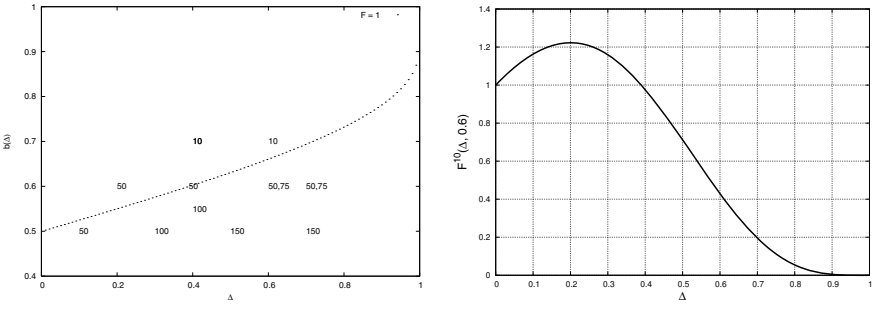


Fig. 1. Left: Hibernation time that makes the dynamic simulation go to a final stable deal price, for several initial conditions and a BA network of 512 nodes. Above the dotted line $F > 1$ and below $F < 1$. Right: Plot of $F^{10}(\Delta, 0.6)$.

simulation model we find that price oscillates around a stable value[2]. Comparing several simulation runs with different initial ratio b, we find that the final value for b is around 80% and the average Δ of all nodes has always a value around 0.8. It appears that the system is pushed towards the curve predicted by the mean field approximation. In our experiments we found that any simulation starting with values close to the analytic curve in Fig. 1 (left) is attracted to it. If the simulations are started further away form the curve, the hibernation times need to be larger below and smaller above the curve to achieve a non-zero finite price.

The explanation for these values of the hibernation time is as follows. When the simulation reaches stability, the ratio of buyers/sellers in the network is around $b = 0.8$. This means that the hibernation mainly affects the sellers. Therefore a higher hibernation time causes sellers to be out of the market for longer, which makes buyers continue increasing prices. In other words, a higher hibernation time causes a higher increase of prices. Consequently, for initial points that would cause a deeper drop of price according to the analytic model (those in the lower right part of Fig. 1(left)), the hibernation time must be higher in order to compensate it and produce a deeper increase of price, achieving in this way an equilibrium. The introduction of hibernation and local Δs is more important for price stability than the a budget limit and a minimum price.

We also compare the performance of the distributed version of the grid computing market on a Barabási-Albert graph that we propose with the one of an hypothetical central version of the system. Since the load of the system will be given by the amount of requests the central server will have to handle, we measure the average amount of messages that are present in the buffers of the

[2] We verified this for a simulation with a TTL of seven, a pub size of 100 messages, initial price of 1200, a budget of 10,000 refilled after 10 Epochs, nodes re-enter the market after being "picked" four times, and the network is a Barabási-Albert graph with 512 nodes. The initial buyer/seller ration is $b = 0.5$, the initial $\Delta = 0.7$, nodes hibernate for 150 Epochs if the measured $b \neq 0.5$, the Δ is changed based on the last 10 Epochs.

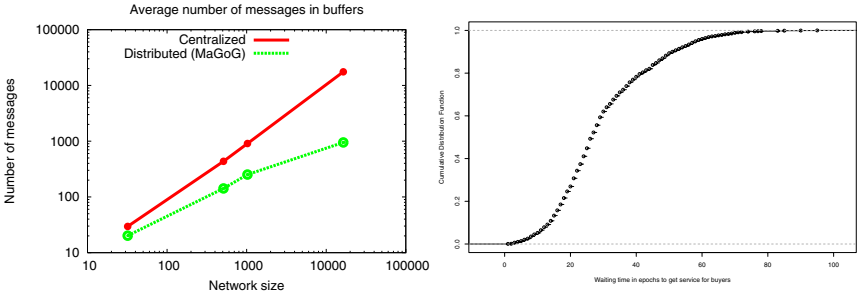


Fig. 2. Left: Comparison of the average number of messages in buffers for a centralised grid computing market and MaGoG (completely distributed grid computing market). Right: Cumulative Distribution Function for the time it takes buyers to find a match.

system. In the case of the central system, the only buffer is the one of the central server; whereas in the case of MaGoG, all nodes contribute with their respective buffers to handle the requests. For the comparison we increase the capacity of the buffers in the system until there are no losses, i.e., no messages are dropped from the buffers due to a lack of space. For the central system, Fig. 2 shows the average number of messages (in the central buffer) per epoch. For the distributed system, Fig. 2 shows the average number of messages per buffer (the buffers of all agents in the network) per epoch. The centralised system has 47% more messages than MaGoG in a network of 32 nodes; 206% more messages in a network of 512 nodes; 260% more messages in a network of 1024 nodes, and 1748% more messages in a network of 16384 nodes. The data shows that MaGoG scales better than a central system.

In previous work [12] we have shown that the system reaches an equilibrium state for the overall utilisation. Whilst this is still true with the modifications made in we now want to investigate how long nodes wait for to sell or buy resources, the "response time". Using long-running simulations we determine the response time distribution for various different nodes in the system. We look at loosely connected and highly connected nodes. We find that the waiting time for a node depends on its type (buyer or seller), and not on the degree of connectivity of the node. The cumulative distribution function for the waiting time of buyers is shown in Fig. 2 (right). As Fig. 2 (right) shows, 90% of the time a buyer gets service in less than 50 epochs, whereas a buyer gets service always in less than 100 epochs. The waiting distribution for sellers is similar but the average waiting time is shorter due to more buyers than sellers in the system.

5 Conclusions

In this paper, we have given numerical reasons of why a distributed Grid Computing market like MaGoG outperforms a centralised system. The increase of computers interconnection and the eventual deployment of a global Grid

Computing market will only be possible by using a decentralised system, since numerical results show that a centralised one will not scale.

We have also derived an analytic approximation to gain a better understanding of our simulation program that models the MaGoG system. By using the results of this analytic approximation, we have improved our simulation program in order to make the system more stable in relation to the price evolution. This global stability in the system is obtained by locally implementing mechanisms in the nodes following the analytic results. We have found that the contributions of individual nodes in the network make the whole system achieve equilibrium.

References

1. Greenberger, M.: The priority problem and computer time sharing. Management Science 12(11), 888–906 (1966)
2. Sutherland, I.E.: A futures market in computer time. Commun. ACM 11(6), 449–451 (1968)
3. Nielsen, N.R.: The Allocation of Computing Resources—Is Pricing the Answer? Communications of the ACM 13(8), 467–474 (1970)
4. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. In: Computational Grids. Morgan-Kaufman, San Francisco (1999)
5. Regev, O., Nisan, N.: The POPCORN market—An online market for computational resources. In: ICE 1998, Proceedings of the first international conference on Information and computation economies, pp. 148–157. ACM, New York (1998)
6. Lai, K., Rasmusson, L., Adar, E., Zhang, L., Huberman, B.A.: Tycoon: An implementation of a distributed, market-based resource allocation system. Multiagent Grid Syst. 1(3), 169–182 (2005)
7. Wolski, R., Plank, J.S., Brevik, J., Bryan, T.: Analyzing Market-based Resource Allocation Strategies for the Computational Grid. The Int. Journal of High Performance Computing Applications 15(3), 258–281 (2001)
8. Buyya, R., Abramson, D., Venugopal, S.: The Grid Economy. Proceedings of the IEEE 93(3), 698–714 (2005)
9. Cohen, J., Richardson, C., Harder, U., Martínez Ortuño, F., Darlington, J.: Node-level Architecture Design and Simulation of the MAGOG Grid Middleware. In: AusGrid 2009, vol. 99, pp. 57–66 (2009)
10. Aberer, K., Cudré-Mauroux, P., Datta, A., Despotovic, Z., Hauswirth, M., Punceva, M., Schmidt, R.: P-grid: a self-organizing structured p2p system. SIGMOD Rec. 32(3), 29–33 (2003)
11. Bagnall, A.J., Smith, G.D.: A multiagent model of the UK market in electricity generation. IEEE Trans. on Evolutionary Computation 9(5), 522–536 (2005)
12. Harder, U., Martínez Ortuño, F.: Simulation of a peer to peer market for Grid Computing. In: Al-Begain, K., Heindl, A., Telek, M. (eds.) ASMTA 2008. LNCS, vol. 5055, pp. 234–248. Springer, Heidelberg (2008)
13. Csàrdi, G., Nepusz, T.: The igraph software package for complex network research. InterJournal Complex Systems, 1695 (2006)
14. Erdős, P., Rényi, A.: On random graphs I. Publ. Math (Debrecen) 6, 290–297 (1959)
15. Barabasi, A.-L., Albert, R.: Emergence of scaling in random networks. Science 286, 173 (1999)
16. Evans, T.: Complex networks. Contemporary Physics 45, 455–474 (2004)

Migrating Auctioneers on Internet Auctions for Improved Utility and Performance

Ricardo Lent

Intelligent Systems and Networks,
Department of Electrical and Electronic Engineering,
Imperial College London
London SW7 2AZ, UK
r.lent@imperial.ac.uk

Abstract. The paper studies a technique to improve the utility and performance of an automated auction application where the auctioneer and bidders communicate through the Internet. The lack of quality-of-service guarantees from site to site can severely influence the results of an auction by affecting the seller's income rate, auction fairness, and overall protocol efficiency. By properly identifying and migrating auctioneers to suitable hosts on the network, it is shown that the expected seller's utility can be improved along with other metrics of interest. The paper proposes a scalable approach to conduct the host search on a large network by applying the Simulated Annealing metaheuristic in conjunction with network probing. A detailed simulation study assess the performance of the system and quantifies the benefits of the proposal.

1 Introduction

Electronic trading with automated auctions is an activity that is being adopted quickly because of the low costs involved while providing good speed and simplicity to users. Automated auctions are increasingly being employed in security markets to trade various financial instruments, such as equities, bonds, foreign exchange and derivatives. On the other hand, web-based auctions have become very popular as they offer a convenient way for users to buy and sell a plethora of goods and services, in some cases with partial or full automated support.

In general, commercial auction systems are being supported by distributed infrastructures for load balancing and availability reasons (e.g., grids or clouds), but mainly operate on a centralized scheme with a predefined (central) site that serves as the communication bridge between seller and bidders (e.g., E-bay and Amazon.com auctions). This central site assumes a partial or full role of an auctioneer for the auction and is responsible for handling bid reception and processing.

The paper considers a slightly different electronic trading scenario, where bidders can initially submit their bids directly to the seller's computer and without an intermediary site (i.e., both auctioneer and seller reside on the same computer) and explore the possibility of migrating the auctioneer to a different

J.T. Bradley (Ed.): EPEW 2009, LNCS 5652, pp. 155–169, 2009.

host for utility and performance reasons. A central site or auction centre (AC) is present in this scenario as an information board to help sellers and bidders to find each other and according to their interests. The AC may also serve to support other tasks, such as user authentication, authorization, access, and supervision that would ensure consistency, fairness, and security to all auction participants.

The distributed structure of the auctions brings a number of advantages over centralized structures:

- It easily allows the system to scale to a very large number of participants as bids are not addressed to a single node (or set of nodes) in the network. Therefore, the auction traffic load is expected to be reasonably distributed across the network.
- It simplifies the deployment of the system by reducing the equipment and network requirements to system administrators.
- It can be highly available at low cost.
- It prevents participants to expose their trading strategy to a third party, which could deter their profit if misused.
- It makes the system easier to defend to security attacks as there is no single target that can be compromised.
- It can offer utility and performance advantages as elaborated in this paper.

Recent papers have addressed the applicability and design of distributed auctions in the context of computer and communication networks. Esteva [1] suggested a distributed architecture for electronic auctions and specified protocols for a first-price, second-price, Dutch and English auctions by means of π-calculus. Ezhilchelvan and Morgan [2] developed an alternative distributed system for Internet-based auctions. Hausheer and Stiller have introduced "PeerMart" [3], a system that uses a double auction on a structured P2P overlay network to distribute brokering load for pricing various goods and services (e.g. bandwidth pricing [4]). Franklin and Reiter [5] developed a distributed system for sealed-bid auctions. Auctions in mobile networks have been studied by Frey et al. [6], Wu et. al in the form of a continuous double auction part of a routing algorithm [7] and Fourati and Al Agha [8]. The application of auctions for bandwidth allocation have been considered by Dramitinos et al. [9] for 3G networks with generalized automated Vickrey auctions. Li and Mahanti [10] have suggested the use of auctions to coordinate streaming sessions. The auction model used in this paper follows the English auction model that was analyzed by Gelenbe [11]. Bidding rates could be very high in automated auctions, so that both network and computing responsiveness (trading software) could be critical to achieve high profits. Lent [12] studied the impact of various network parameters on Gelenbe's auction model.

The Internet offers a best-effort service with no quality-of-service guarantees. Given the heterogeneity of computing and network resources that compose the network, and the continuous development of new technologies that are asymmetrically deployed, the Internet remains highly unbalanced. The unbalance can be experienced not only in terms of the variety of these resources that change from site to site, but also in terms of network usage. As a result, bidders operating

from different parts of the network could experience very different communication conditions to the seller (different message latencies, variable message losses, etc.) A strong difference in network delays would produce unfair advantage to some participants, whose bids could get consideration from the auctioneer even before earlier generated (but delayed) ones. On the other hand, excessive delays could also produce a high waste of resources not only because participants must be idle until messages arrive, but also because a number of incorrect messages (e.g., short bids, unnecessary retransmissions, etc.) could be produced as a result of the lack of timely auction status information.

This paper offers the following contributions:

- It proposes a migration technique for network auctions to allow sellers to improve their utility (income rate) by opportunistically migrating them into better hosts on a network. In addition to achieving improved utility, other relevant performance metrics, such as auction fairness and efficiency, are also improved.
- It applies a local search metaheuristic (Simulated Annealing) to conduct a scalable search that is suitable for large networks as it produces a low overhead, to dynamically discover better hosts. To discern good from bad hosts, the paper defines a cost metric that captures the suitability of a node for communications with a group (the set of bidders) from a quality-of-service point of view.
- It evaluates the approach in a detailed simulation environment that approximates the operation of network protocols (including the auction protocol) with an accurate modelling of packet delays (transmission, propagation and queueing) and considering the structure of the underlying network.

The rest of the paper is organized as follows. Section 2 describes the search mechanism that is used to discover good hosts along with practical considerations for probing the network. Section 3 describes the auction protocol considered in the study. Section 4 explains the simulation environment and the results of a Monte Carlo evaluation of the approach. Finally, Section 5 provides concluding comments on this work.

2 Host Search

The objective is to identify a suitable node to host the auctioneer of a network auction. On large networks, there exist prohibitive costs and temporal issues on collecting global network information. Therefore, we approach the problem with a local search rather than with a centralized computation for the best host.

Consider an auction operating on a network that is represented by the graph $G = (V, E)$, where V is a set of vertices (nodes) and E is the set of edges (links). An auctioneer s ($s \in V$) communicates with a set of bidders b ($b \in B \subseteq V$). Consider a function $\phi : \{s, B\} \rightarrow \Re$ (a cost function). The problem is to identify a node $m^* \in V$ such that $\phi(m^*, B) \leq \phi(n, B)$ for all $n \in V$ (i.e., the group communication cost is minimized). Graph G depends on network characteristics

and is in general directed. However, by limiting the problem to work with certain metrics, such as round-trip delays, G can be approximated as undirected, which greatly simplifies the evaluation of costs. A possible formulation for ϕ is:

$$\phi(s, N) = \sqrt{c_0 P_{s,m}^2 + \sum_{n \in N} c_n P_{m,n}^2} \tag{1}$$

which calculates the Euclidean metric (length or distance) of the individual path costs $P_{m,n}$. $P_{m,n}$ is the measured metric of interest (i.e., average round-trip delay) between nodes m and n. The positive or zero constants c_i are introduced to weight the contribution of each participant in the group communication. Equation 1 is a general expression for the cost of a group communication and includes the cost from the origin s to the new host m (term $P_{s,m}$). The paper assumes that a migration implies a transfer of all auction tasks, so that the communication between s and m is of small relevance ($c_0 = 0$). The other constants would be equal in most cases (e.g., $c_i = 1$, $i \in N$), so that the search will tend to find the centroid node for the group (node with the same or similar path cost to all members of the group). However, the expression is flexible enough to direct the search if needed, to find hosts closer to certain participants (by increasing their weights).

To implement the auctioneer migration, the following three basic functions (modules) are required: probe, search, and migrate. The search is conducted by a Search Agent. An execution environment (daemon) runs at each participating node to let search agents operate and execute the auctioneer migration when needed.

2.1 Network Probing Module

The purpose of the network probing module is to measure the metrics of interest from a given node to a set of destinations. These measurements can be later used to calculate the cost function from the given node to determine the next steps of the search process.

The basic element of the probing module is an end-to-end tester that has the ability of measuring message latency, jitter, loss, hop count, or effective bandwidth (throughput) between any two nodes. For a network auction, the main interest is in reducing network latency. However, this module can be general enough to accommodate other metrics (for other applications for example). To test a connection, the search module sends one or more messages to a pre-defined port(s) of the remote host, which is controlled by the execution environment. Standard tests can only be conducted (e.g., ICMP).

Message latency can be measured from its sending and receiving times. The sending time can be inserted by the origin into the message so that the destination can calculate message latency. However, both the origin and destination systems require to have a clock synchronization, which can be achieved for example by requiring both systems to run the Network Time Protocol (NTP). Similarly, round-trip latencies can be calculated with a timestamp at the origin,

but not time synchronization is required. In most cases, it is possible to reuse the Ping servers that are available in standard TCP/IP implementations to measure round-trip message latencies (therefore, the module only needs to implement an ICMP echo client.

Similarly, hop count can be measured by implementing an ICMP echo client by increasing the time-to-live field in the packet to obtain ICMP time exceeded replies from the intermediate routes along the path (exactly in the same way the program *traceroute* works). When the underlying network is provided by a CPN, each origin node may obtain the hop-count information directly from the CPN [13].

End-to-end throughput and message loss can be measured by making the probing module to generate a pre-determine packet flow between the probing nodes and comparing it with the arriving packet flow.

Most network probes return average values (either absolute or exponential averages), so that more than one test message will be sent to check the network status. The exception is hop count for TCP/IP networks given that it rarely changes.

2.2 Search Module

The purpose of the Search Module is to determine a suitable host for the auctioneer. To search for this host, we apply a Simulated Annealing (SA) metaheuristic [14]. The basic idea is to iteratively compare the cost function as evaluated at the current hosting node to the cost evaluated at a neighbour host. If a better cost is found, the Search Agent moves to that neighbour and repeats the process iteratively until no further improvement can be found (which gives the solution to the search). An stochastic behaviour is introduced to avoid local minima. With probability p, the Search Agent may move to a neighbour with a worse than the current cost (see Algorithm 1). Probability p is a function of the two cost difference (δ) and the elapsed search time (t_s): $\exp\{-\delta/T_e\}$. T_e is the temperature parameter that gradually decreases in inverse proportion to t_s (we have used $T_e = 100/t_s$ in the simulation study that is described later).

In addition, the module keeps track of the nodes most recently probed in a *memento list* (similar to the tabu list in Tabu Search), which is used to avoid repeating probes from previously tested nodes. The memento helps to keep a low search overhead by avoiding unnecessary repetitions. Also, caching test results with a *time-to-live* mark allows to reuse information by the same or later search process.

The use of the memento is useful in those networks whose structure allows to gradually decrease Φ with each Search Agent move. However, it is less useful in networks with a more stochastic structure, such as those in generic peer-to-peer overlays. In those cases, a more stochastic search is adequate (SA with values of p close to 1) and without memento to allow the exploration of options from previously visited nodes. Caching test results can improve the performance of the process as well in this case.

Algorithm 1. SA Agent

```
1: currentnode = sourcenode
2: finished = false
3: while not finished do
4:    Create moveset with random neighbour
5:    Probe moveset
6:    (cost, node) ⇐ mincost(moveset)
7:    if currentnode != node OR with probability P then
8:       currentnode ⇐ migrate(node)
9:    else
10:      finished=true
11:   end if
12: end while
13: Report results to sourcenode
```

2.3 Migration Module

Each agent operates within the execution environment, which provides access to the communication ports of the hosting node. As a result of a search decision, a Search Agent may migrate to a different host to continue the probe-and-move process. At the end of the process, the Search Agent returns to the origin and informs the application of the best location that was found by the agent and the cost. The application (i.e., auctioneer) then may decide to migrate depending on the cost advantage. The auctioneer migration is described in the next section.

3 Auction System

The basic software elements are the auctioneer, a number of bidders and an auction centre, all of which run on the execution infrastructure of a host machine. The auctioneer is a software agent that is responsible for selling a good or service for the highest possible price on behalf of its owner. Similarly, a bidder is a software agent that works on behalf of its owner to buy a good or service by offering bids to the auctioneer. The auction centre serves to match buyers and sellers. Sellers can use the auction centre to advertise auctions and bidders can find the sellers offering the goods or services that they are interested in. The auction centre plays an active role in the progress of an auction by allowing bidders to discover suitable sellers, not only at the beginning of an auction, but also during its execution to allow extra bidders to join on-going auctions. The implementation of the auction centre could be done in a centralized or distributed way. The latter being preferred because it is more resilient to network attacks. The auction centre can fulfil other functions as well, such as registration, supervision and trade settlement (e.g. transfer of goods, payments, etc.) However, this paper does not focus on the design of the auction centre and leaves the issue open for a future research.

The paper assumes that the auctioneer and bidders engage in an English auction. In this type of auction, the auctioneer announces the initial price of

the good or service and bidders make incremental offers until one of them is accepted. At this point, the auction is terminated and the selling price is the value of the accepted bid. In agent-based auctions, whether to accept an offer or to submit a bid would depend on the particular agent design and goals (i.e., the strategy that they use to maximize their utility), which could involve verifying the price or similar items in the market at decision time.

3.1 Operation

Consider the following protocol, which supports the main aspects of a network auction. Specialized functions, such as authentication, authorization and access control have been excluded as they indirectly relate to this study.

Sellers advertise items with an ADV message, which contains the seller's details, item description, auction model and initial price (equal to the current value). As mentioned previously, the study is limited to English auctions (incremental bids, no time limit), so that the auction model field in the ADV message will be more meaningful in the future when other models are introduced. After receiving an ADV, the auction centre adds the item to its database along with the seller details. The AC allows bidders to search for items (who will send a QRY message for this). The AC will answer to queries with a RPY message that will list the matching items (zero or more) to the request. From the list, a bidder may select the most appropriate offer based on its trading criteria, which could include the price, the reputation of the seller, the location of the item, etc. If a seller is selected, the bidder will send a bid (BID message) to the corresponding auctioneer with an offer greater than the current price. Bidders may join an auction at any time before its termination.

On reception of a bid, the auctioneer may decide to conclude the auction based on its particular trading criteria only if the offer is greater than the current value. If the offer is equal or less than the current value, then the offer is disregarded. The auctioneer will announce the value of the newest offer to all registered bidders (the ones from whom the auctioneer have received an offer from the beginning of the auction) by using an HBID message. English auctions require each bidder to "hear" all other bids. The AC is also informed of the new current value, so that its database can be kept up-to-date with the progress of the auction (with an ADVUPD message). As with bidders, the auctioneer trading criteria could include a number of factors, such as the offer value, trust and reputation issues, etc. While deciding whether to accept the highest offer, the auctioneer may receive offers from other bidders. If an arriving offer is greater than the current offer, then the auctioneer will disregard the previous to start considering the new one.

A bidder that is not the highest bidder may send a new bid some time after receiving a HBID message and according to its trading strategy. The trading strategy may dictate that new bids can be generated only if the current value of the auction is below a pre-calculated threshold for example. Bidders may withdraw from the auction as long as they are not the highest bidder (by stop bidding and sending a WDRW message to the auctioneer) at any time.

The process continues iteratively until the auctioneer decides to terminate the auction. To end the auction, the auctioneer sends a WBID message to all bidders (including the highest bidder) and an ADVUPD message to the auction centre to close the auction in its database.

To ensure consistency, the protocol, as a finite state machine, includes a number of timeout mechanisms that can return to the auction state to its initial state. Such mechanisms are needed to deal with unexpected situations, such as the loss of messages or computer or network malfunction. They also deal with the lack of demand (to return an auctioneer to its init state when it has never received a bid).

Because of the time-critical constraints, the protocol is assumed to operate over a connectionless transport protocol (i.e. UDP) Therefore, network dynamics may cause reception of unexpected messages. Auctioneers may receive delayed bids with a lower or equal price to the current offer or they may receive offers after termination of the auction. Similarly, bidders may receive out-of-sequence price updates (HBID). The reception of these unexpected messages are discarded by the auction agents.

3.2 Auctioneer Migration

The auctioneer migration involves two-phases. During the first phase, the auctioneer identifies a suitable host (surrogate) to handle its execution according to the specific goal defined in the auctioneer agent and implemented as described in Section 2. Surrogates may receive in exchange certain benefits for allowing others to use their network and computing resources, which would involve some form of currency. The design of the system of incentives is beyond the scope of this paper. During the second phase of the process, the auctioneer contacts the selected host with a MREQ message, which contains the basic details of the auction. The MREQ is received by the control software of the execution infrastructure running at the destination machine. The selected host may then accept to serve as migration host depending on observations of its own state and evaluation of other criteria, such as the incentive offered by the auctioneer.

If the host accepts the migration, it will send a MACK message to the auctioneer. Otherwise, it will reject the task with a MREJ message. If the host rejects the request, the auctioneer will try again with another selection. If there is an agreement, the control software on the host will instantiate a surrogate after reception of a MADV message from the auctioneer. The MADV message confirms the migration and transfers the current state of the auction: AC address, bidders, auction model, current value, etc. Any bid arriving at the auctioneer after the migration has occurred will be forwarded to the surrogate. The surrogate also keeps the seller informed of the current value with MUPD messages. Note that additional migrations are also possible. AC list surrogates when a migration has occurred in an auction, so that new bidders can join the auction by directly sending their messages to the surrogate.

The surrogate acts as on behalf of the auctioneer, so it will be responsible for accepting offers . Once the surrogate accepts an offer (MWBID message), it

will transfer the state back to the original auctioneer, which will take care of post-auction operations (handle payments, arrange delivery of goods, etc.)

4 Evaluation

The simulation setup consists of a network connecting the auction elements (bidders, sellers and auction centre). A simulation module of the auction protocol described in Section 3.1 was developed and integrated into the packet-level simulator *INES* [15] to evaluate relevant performance metrics of the system. The Search Agent was also implemented. The network consists of 1000 nodes organized in a topology of hierarchical structure. The nodes are connected with full-duplex links with a random link bandwidth from a discrete exponential distribution (with possible values: 32, 64, 128, 512, 768 and 1000 Kbps) and a propagation latency of 1 ms (Figure 1).

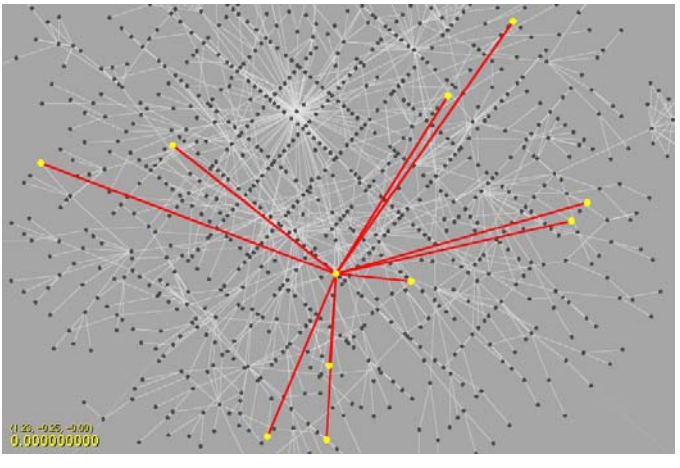


Fig. 1. Auction running on a network of 1000 nodes and 1165 links of random transmission rates. Random location for seller (at the centre) and bidders.

The simulated auction considers the case of a single AC with bidders and auctioneer with prior knowledge of the AC address. The location of the auction elements is randomly selected at be beginning of each simulation run from the set of leaf nodes of the network. The seller announces the selected good to sell to the AC with an initial price of zero (asking price of one). Immediately after starting, bidders send a QRY packet to the AC enquiring for the good. Bidders start with a random estimation of the value of the good, which lies uniformly between 80 and 100. If the price surpasses their estimation during the auction, the bidder withdraws from the auction. A bid is placed at the expiration of a timer with an exponentially generated interval (parameter b) that is started upon arrival of a reply from the AC for a good or an HBID message (outbid message) from the auctioneer. A bid always increments in one the current

(known) value of the good. Parameter b was kept equal to all bidders in the simulation. Similarly, the auctioneer may decide to accept a bid and terminate the auction upon expiration of a timer started after the reception of a bid (again, with an exponentially distributed time with parameter d).

Both b and d can model the strategy and responsiveness of bidders and auctioneer. For instance, they could account among other things, for time needed to valuate and to verify the price of the item in the market. Two values for b were used in the simulations (10 and 100 bids/sec) and various values for d from 0.1 to 1. The auction model follows closely the one developed by Gelenbe [11].

4.1 Search

At the beginning of the simulation, all auction elements are randomly located. To optimize the host selection for the seller, a search process as described in section 2 is conducted prior to the start of the auction from the (random) location of the seller. If a better cost is found, the auctioneer migrates to the optimal host to take advantage of the network structure. In the proposed system, a new search may be started if there are variations in the set of bidders as some may leave the auction, while new ones may join. The simulation study focuses on the static case where bidders do not change, so that the search is only conducted at the beginning of the auction. After it migrates, the auctioneer may conduct one or more auctions. The case of dynamic migration (while an auction is taking place) is a subject of on-going research by the author. The average search time is shown in Figure 2 as a function of the number of participating bidders. The probability of finding a better host for the seller was observed to be high (Figure 3). The points on the curves represent the average of approximately 3000 simulation runs. The 95% confidence interval of the samples is also indicated.

Is is interesting to notice that as more bidders participate in the auction, the search path decrease given that a randomly located seller on a finite network tends to be optimal for a large number of bidders (Figure 4). This effect is also illustrated by the cost ratio (new cost to original cost ratio), which tends to decrease with large number of bidders (Figure 5).

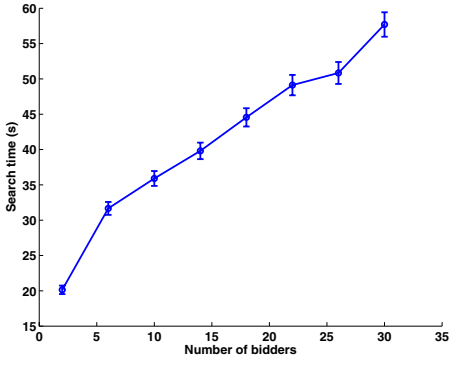


Fig. 2. Search time

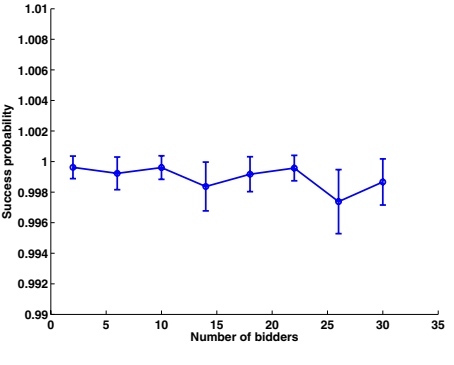


Fig. 3. Search success probability

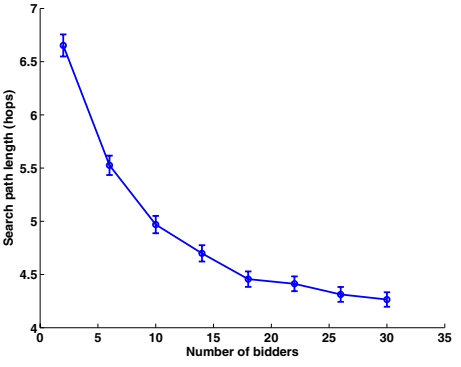


Fig. 4. Search path length

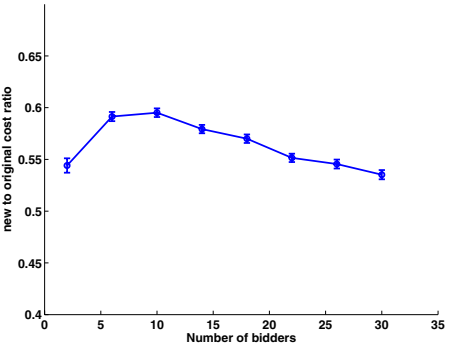


Fig. 5. New to original cost ratio

4.2 Income Rate

A main interest is in determining the expected seller's utility, which is defined by its income rate (Φ). Φ is calculated by dividing the accepted offer over the auction duration. We compare the results for Φ obtained with the two types of systems (original and optimized). The results are depicted in Figure 6. It can be observed that the optimal migration of a seller with respect to its bidders can produce a superior income rate.

The reason for the improved Φ is that from the optimal host, messages can be delivered faster from seller to bidders and vice versa as depicted in Figure 7. By optimally moving the seller on the given network topology it was possible to observe reductions on bid transmission times of about 40% on average.

The better host also allowed for improvements in the message loss ratio. Because of the near real-time requirements of the application, UDP was used as the transport protocol to transmit messages. UDP does not implement a retransmission mechanism. The observed bid loss ratio during the simulations is depicted in Figure 8 and shows improvements of around 20%–25% in the optimized system.

4.3 Fairness

In addition to reducing average transmission times and bid loss ratios, the use of an Euclidean metric for determining a suitable host for the auctioneer also brought the additional benefit of reducing message jitter (variation from the average delay), which helps in improving auction fairness. Because delays can be different to each seller-bidder pair as a result of the network structure and location of auction participants, a bidder may be out-bid by another (with an equal offer) even if the latter sends the bid at a later time but arrives before the former bid. Figure 9 measures auction fairness by the probability of ending an auction with an offer that was the first and highest one submitted. It can be observed that fairness decreases with smaller selling rates (i.e., larger number

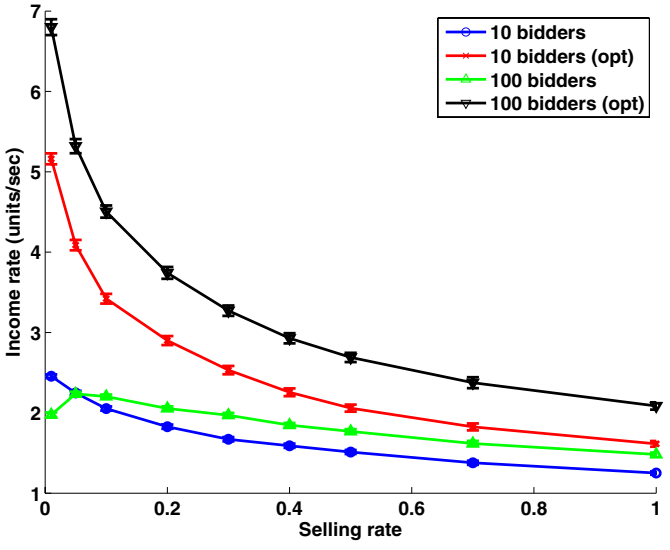


Fig. 6. Income rate vs. auctioneer's decision rate

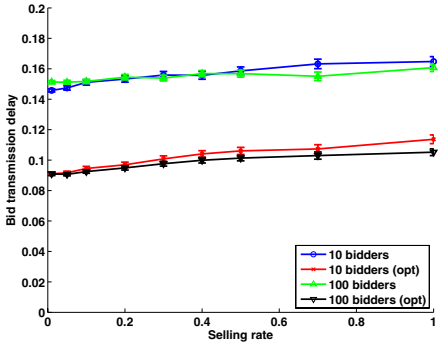


Fig. 7. Bid transmission latency

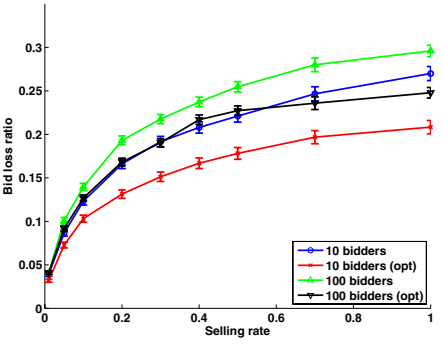


Fig. 8. Bid loss ratio

of auction rounds, as the auction takes longer to finish). The optimized system offers again a clear advantage over the original one by improving auction fairness by a wide factor.

4.4 Efficiency

In addition to decreasing the expected seller's income, longer network delays produce larger numbers of "short" bids, that is, bids not sufficiently high. These short bids are generated because network delays cause an inevitable interval between the arrival of a valid bid at the seller and the arrival of the HBID at bidders which inform them of the current price. Therefore, during that interval,

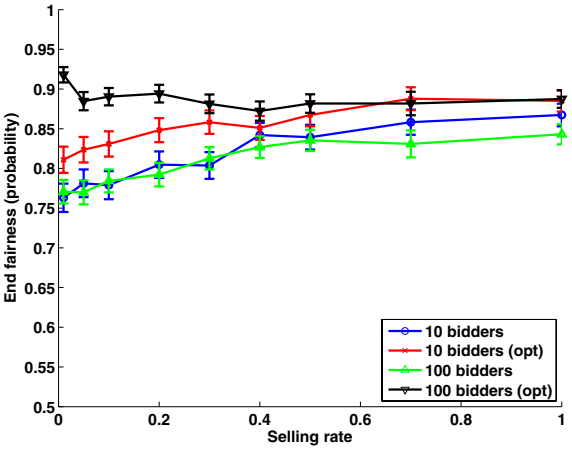


Fig. 9. Auction fairness: probability of accepting the highest and earliest bid

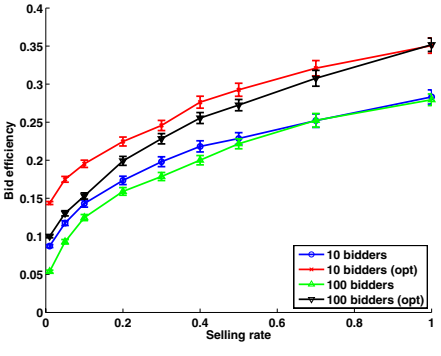


Fig. 10. Bid efficiency

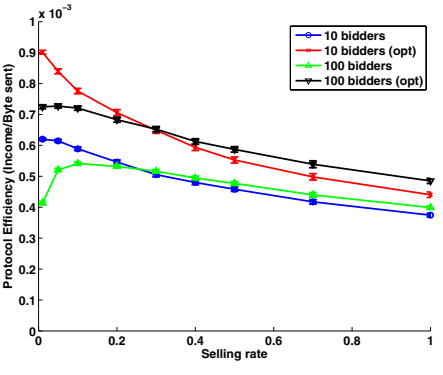


Fig. 11. Protocol efficiency

bids will be generated with an incorrect assumption of the current price and creating the consequent short bid (given that we assume unit increments). We differentiate short bids from "late bids". The latter referring to bids received after termination of the auction. Both short and late bids waste bandwidth and other resources and should be avoided by the network and protocol design as possible. We define as "bid efficiency" as the ratio of valid bids (bids arriving at the seller with a correct price) over the total number of bids received. Figure 10 shows the observed bid efficiency vs. seller's decision rate.

To study the efficiency of the protocol described in Section 3.1, we have looked at the total traffic generated by an auction and compared it to the income generated to the seller. The results are reported in Figure 11 in the form of the income rate to byte sent. In both cases, it can be observed a clear advantage to the seller when operating on the optimized system.

5 Conclusion

The paper has studied a technique to improve the utility and performance of automated auctions that operate on the Internet. The Internet as a best effort network offers no guarantees on the quality of communications between seller and bidders and as a result, bids can be affected by variable latency and loss. Those factors can greatly influence the expected results of an auction. In brief, the proposed idea is to implement auctioneer migration to a host that can offer an improved group communication from a quality-of-service point of view. To select the best hosts for auctioneers, the paper proposed a search mechanism that can scale to large networks and that applies Simulated Annealing and low-overhead network probing. A detailed simulation study of the system allowed to quantify the improvements that can be achieved by the proposal in terms of the seller's income rate, auction fairness, protocol effectiveness and other relevant metrics.

References

1. Esteva, M., Padget, J.A.: Auctions without auctioneers: Distributed auction protocols. In: Agent Mediated Electronic Commerce (IJCAI Workshop), pp. 220–238 (1999)
2. Ezhilchelvan, P., Morgan, G.: A dependable distributed auction system: Architecture and an implementation framework. In: Proceedings of the Fifth International Symposium on Autonomous Decentralized Systems (2001)
3. Hausheer, D., Stiller, B.: Decentralized auction-based pricing with PeerMart. In: 9th IFIP/IEEE International Symposium on Integrated Network Management (IM 2005) (May 2005)
4. Hausheer, D., Stiller, B.: PeerMart: Decentralized auctions for bandwidth trading on demand. ERCIM News 68, 42–43 (2007)
5. Franklin, M.K., Reiter, M.K.: The design and implementation of a secure auction service. IEEE Trans. Softw. Eng. 22(5), 302–312 (1996)
6. Frey, H., Lehnert, J., Sturm, P.: Ubibay: An auction system for mobile multihop ad-hoc networks. In: Workshop on Ad hoc Communications and Collaboration in Ubiquitous Computing Environments (AdHocCCUCE 2002) (2002)
7. Wu, Z., Chen, Z., Xu, L., Guo, F.: Routing protocols based on double auction for ad hoc networks. In: International Conference on Networking, Architecture, and Storage (NAS 2007), pp. 55–61 (2007)
8. Fourati, A., Agha, K.A.: Deploying auctions over ad hoc networks. In: ICEBE 2006: Proceedings of the IEEE International Conference on e-Business Engineering, pp. 9–16 (2006)
9. Dramitinos, M., Stamoulis, G., Courcoubetis, C.: Auction-based resource allocation in UMTS high speed downlink packet access (HSDPA). In: Next Generation Internet Networks, April 2005, pp. 434–441 (2005)
10. Li, Z., Mahanti, A.: A progressive flow auction approach for low-cost on-demand P2P media streaming. In: QShine 2006: Proceedings of the 3rd International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks, p. 42 (2006)

11. Gelenbe, E.: Analysis of automated auctions. In: Levi, A., Savaş, E., Yenigün, H., Balcısoy, S., Saygın, Y. (eds.) ISCIS 2006. LNCS, vol. 4263, pp. 1–12. Springer, Heidelberg (2006)
12. Lent, R.: On the impact of network QoS on automated distributed auctions. In: Proc. of the 2nd International Conference on Bio-Inspired Models of Network, Information, and Computing Systems, Workshop on Technologies for Situated and Autonomic Communications (2007)
13. Gelenbe, E., Lent, R., Xu, Z.: Measurement and performance of cognitive packet networks. J. Computer Networks 37, 691–701 (2001)
14. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220, 671–680 (1983)
15. Lent, R.: INES: Network simulations on virtual environments. In: Proc. of 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (2008)

Analytical Model of the Soft Handoff Mechanism in the UMTS Network

Maciej Stasiak, Piotr Zwierzykowski, and Damian Parniewicz

Poznan University of Technology
Chair of Communications and Computer Networks
ul. Polanka 3, Poznań 60965, Poland
piotr.zwierzykowski@put.poznan.pl

Abstract. The paper proposes an analytical approach to blocking probability calculation in the UMTS network. In the proposed method, handoff connections are modelled on the basis of the fixed point methodology. The results of analytical calculation of the blocking probability in the group of cells carrying a mixture of different multi-rate traffic streams are compared with the results of simulation experiments, which confirms a good accuracy of the method. The described calculation scheme is applicable for cost-effective resource management in 3G mobile networks and can be easily applied to network capacity calculations.

Keywords: analytical model, soft handoff, UMTS.

1 Introduction

The third-generation (3G) system, known as the Universal Mobile Telecommunications System (UMTS) using Wideband Code Division Multiple Access (WCDMA), introduces very variable data rates on the air interface, as well as the independence of the radio access infrastructure and the service platform. For users, UMTS makes available a wide spectrum of circuit-switched or packet data services.

The variable bit rate and the variety of traffic on the air interface have presented completely new possibilities for operators and users alike, but also new challenges to network planning and optimization. In contrast to second-generations (2G) systems, the 3G system performance depends heavily on the traffic conditions, making many performance measures soft (e.g. soft handoff, soft capacity). Consequently, the network planning engineer cannot plan in advance the network in order to find a most favourable general solution, because any solution is valid only for a given traffic condition; when traffic changes, the optimum plan also changes [1]. Consequently, this makes the network modelling and capacity calculations for systems with the WCDMA radio interface so crucial.

J.T. Bradley (Ed.): EPEW 2009, LNCS 5652, pp. 170–182, 2009.

In WCDMA, all users in the same cell share the same frequency channel simultaneously. Moreover, the same frequency can be used in other cells of the same network and users can be distinguished by means of codes. This solution allows a mobile station to decode a signal from more than one base station simultaneously by despreading the single received radio signal using a number of different scrambling and channelization codes. This technique is exploited in WCDMA to support a feature known as the soft handoff, whereby a mobile station can communicate with more than one base station simultaneously as it moves between cells in the network.

The soft handoff has a number of advantages as compared to the hard handoff used in Frequency-Division Multiple Access (FDMA) and Time Division Multiple Access (TDMA) systems such as the Global System for Mobile Communications (GSM), in which simultaneous communication with more than one base station is not allowed [3,1]. Soft handoff is a form of diversity, increasing the signal-to-noise ratio. By combining the information received on each individual link, we can reduce the required signal-to-noise ratio per link, when compared with the situation with one radio link only. Compared with the conventional hard handoff, soft handoff has the advantages of smoother transmission and less ping-pong effects. Soft handoff has also the disadvantages of complexity and extra resource consumption.

Soft handoff can appear in different forms. We can distinguish soft handoff, when a mobile station is connected with two or more base stations; softer handoff, when a mobile station is connected with two or more cells of the same base station; and soft - softer handoff, which is a combination of soft and softer handoff.

In the literature, there are papers which concern the soft handoff mechanism in analytical models of UMTS systems [4,5,6,7]. In [5,6], the authors discussed the soft handoff mechanism by the application of limited-availability group models with and without reservation. In [4], the Kaufman-Roberts recursion was applied to model an isolated cell with prioritized handoff traffic. The authors in [7] took into account the soft handoff mechanism by a simple increase in traffic offered to the system.

The proposed analytical method is an extension of the methodology proposed by the authors in several earlier works [8,9]. In the presented analytical model we assume that the WCDMA radio interface can be modelled by the full-availability group servicing a mixture of multi-rate traffic streams. In the model we apply the handoff mechanism using the fixed-point methodology (FPM) [10]. This method allows us to take into consideration the dependence between service processes in cells participating in the handoff connection.

The article is divided into five sections. Section 2 discusses basic dependencies describing the radio interface load for the uplink and the downlink direction. Section 3 presents an analytical model employed in blocking probability calculations. The following section includes a comparison of the results obtained in the calculation with the simulation data for a system comprising seven cells. The final section sums up the discussion.

2 Radio Interface in the UMTS Network

The following section summarizes the fundamentals of the UMTS standard which have been adopted by the authors in the modelling of the WCDMA radio interface. The section briefly discusses the assumptions concerning the radio interface in the UMTS networks which were elaborated in earlier works of the present authors [8].

Accurate signal reception in the WCDMA radio interface is possible only when the relation of energy per bit E_b to noise spectral density N_0 is appropriate [2]. A too low value of E_b/N_0 will cause the receiver to be unable to decode the received signal, while a too high value of the energy per bit in relation to noise will be perceived as interference for other users of the same radio channel. The relation E_b/N_0 for a user of the class i service can be calculated as follows [2]:

$$\left(\frac{E_b}{N_0}\right)_i = \frac{W}{\nu_i R_i} \frac{P_i}{I_{total} - P_i}, \tag{1}$$

where: W – chip rate of spreading signal, v_i – activity factor of a user of the class i service, R_i – bit rate of a user of the class i service, I_{total} – total received wideband power, including thermal noise power and P_i – signal power received from a user of the class i connection. After simple transformations, we can express the P_i parameter depending on I_{total}:

$$P_i = L_i I_{total}, \tag{2}$$

where L_i is the load factor for a user of the class i connection:

$$L_i = \left(1 + \frac{W}{(\frac{E_b}{N_0})_i R_i \nu_i}\right)^{-1}. \tag{3}$$

Table 1 shows sample values of the load factor L_i for different traffic classes [9].

Table 1. Examples of E_b/N_0, ν_i and L_i for different service classes [9]

Class of service (i)	Speech	Video call	Data	Data
W [Mchipps]	3.84			
R_i [kbps]	12.2	64	144	384
ν_i	0.67	1	1	1
$(E_b/N_0)_i$ [dB]	4	2	1.5	1
L_i	0.0053	0.0257	0.0503	0.1118

On the basis of the load factor obtained for particular users, it is possible to determine the total load for the uplink or the downlink connection [2]:

$$\eta = \begin{cases} (1+\overline{\delta}) \sum_{i=1}^{M} L_i n_i & \text{for uplink direction,} \\ (1-\xi_i+\overline{\delta}) \sum_{i=1}^{M} L_i n_i & \text{for downlink direction.} \end{cases} \tag{4}$$

In the Formula (4) the following notation is used, e.g. [8]:

- M is the number of services (traffic classes),
- n_i is the number of users of the class i service,
- $\bar{\delta}$ is the mean value of the other cell interference upon proper cell interference,
- ξ_i is the orthogonality factor,
- η is the assumed capacity of the radio interface in the uplink or downlink direction.

Therefore, it is assumed that the actual maximum use of the resources of a radio interface without lowering the level of the quality of service will amount to about 50 – 80% [11].

The determination of the capacity of the WCDMA interface depends then on the parameters related to the properties of the propagation channel and the nature and the degree of interference that are taken into account in the parameters $\bar{\delta}$ and ξ_i. Simulation studies make it possible to determine, with the application of advanced propagation models and digital maps of area, the values of the parameters $\bar{\delta}$ and ξ corresponding, with high accuracy, to the conditions of real cellular networks [3]. Therefore, in the considerations presented in the present paper, we have assumed, based also on the assumptions presented in [12, 13, 14, 15, 16], that the influence of interference on the flow capacity of the WCDMA radio interface can be determined by the pair of the parameters $\bar{\delta}$ and ξ.

3 Model of the System

Let us consider a group of a seven-cell system with omnidirectional antennas (Fig. 1). Each of the cells in the uplink and in the downlink directions can service a mixture of multi-rate traffic. In the description of the model let us designate the access cell z and let us assume that it is surrounded by six neighbouring cells. Additionally, in the model, we consider a soft handoff mechanism which occurs only in the uplink direction. Therefore, the scope of the analysis of the system is limited to the uplink direction.

Dimensioning of the capacity of the WCDMA radio interface is possible at the call level only and can be treated as dimensioning of the traffic capacity of the physical channel (expressed in Radio Access Bearer). At the call level, a packet stream generated by a user of a given service is treated as a connection. This means that each packet stream of a given traffic class corresponds to the matching service bit rate resulting from the applied standard, e.g. Relese99 [2,1] or determined on the basis of equivalent bandwidth, e.g. [17]. Such an approach is widely used in modelling the traffic capacity of the WCDMA radio interface, e.g. [18, 19, 20, 21, 7, 22]

3.1 Basic Assumptions

A model of the full-availability group servicing a mixture of different multi-rate traffic streams can be used to model the WCDMA radio interface. The

occupancy distribution in such a group can be described on the basis of the so-called Kaufman-Roberts distribution [23,24]:

$$nP(n) = \sum_{i=1}^{M} a_i t_i P(n - t_i), \tag{5}$$

where

- M is the number of classes of Poisson traffic streams,
- $P(n)$ jest the occupancy probability of n BBUs,
- a_i is the mean traffic offered to the system by the class i traffic stream: $a_i = \frac{\lambda_i}{\mu_i}$, where λ_i is the intensity of class i of Poisson traffic stream and μ_i is the parameter in the exponential distribution of the holding time for calls of class i,
- t_i is the number of BBUs required for servicing class i call.

Therefore, we can determine the value of the parameter t_i. In analytical models of the WCDMA interface [8], the value of L_{BBU} is defined as the greatest common divisor (GCD) of load factors for all traffic classes carried by the interface.

$$L_{BBU} = \mathrm{GCD}(L_1, ..., L_M), \tag{6}$$

where L_i is defined by the Eq. (3) (typical values of L_i for services offered in the UMTS network are presented in Tab.1). Knowing the value of L_{BBU}, we can calculate the number of BBUs required by the calls of each traffic class:

$$t_i = \left\lfloor \frac{L_i}{L_{BBU}} \right\rfloor. \tag{7}$$

The capacity of the WCDMA interface expressed in BBUs based on (4), can be determined by the following equation [8]:

$$\mathrm{V} = \begin{cases} \frac{\eta}{1+\overline{\delta}-\xi} & \text{for downlink direction,} \\ \frac{\eta}{1+\overline{\delta}} & \text{for uplink direction,} \end{cases} \tag{8}$$

where η is the physical capacity of the WCDMA interface in the downlink or in the uplink direction, respectively.

After determining the occupancy distribution $P(n)$ in the radio interface carrying a mixture of different traffic classes, we can define the blocking probability for class i:

$$B_i = \sum_{n=V-t_i+1}^{V} P(n). \tag{9}$$

3.2 Soft Handoff Traffic Model

It is assumed in Figure 1 that a new call of class i is offered to the radio interface of the cell "1". Additionally, Fig. 1 presents two soft handoff connections: between

cells '1' and '4', and between cells '1','5' and '6'. In the model, we also assume that the amount of required resources by the call being in soft handoff is lower than the initial value, i.e. the value required by a call which is not serviced in soft handoff connections. It is worth emphasizing that the greater number of simultaneous connections in the soft handoff decreases the value of required resources in each cell participating in this connection. According to Eqs. (6) and (7), a call requires t_i BBUs in the access cell and $t_i^{'}$ BBUs in the neighbouring cells for calls being in soft handoff connections, and $t_i^{'} < t_i$.

Figure 1 shows a traffic distribution scheme for the system under consideration. The following notation is used: V_x – the cell x capacity; a_i – the mean traffic offered to the system by users of class i ; $a^{SH}_{z,h,i}$ – the mean traffic of the users of class i (being in the soft handoff connection) to the system composed of 2 cells z and h; $a^{z}_{z,h,i}$ – the mean traffic of the users of class i (being in the soft handoff connection) to the system composed only of one cell z, and $a_{z,i}$– the mean traffic offered in cells z by a user of class i.

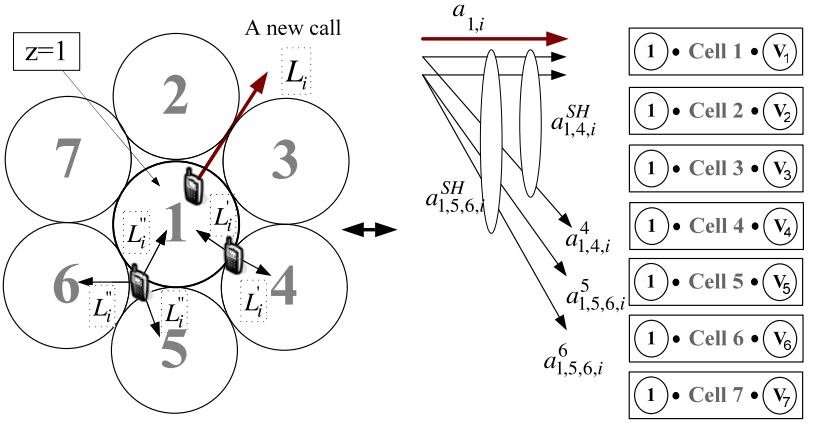


Fig. 1. A seven-cell cellular network with soft handoff connections

In the proposed model it is assumed that a new call in the soft handoff is rejected when the assumed increase in the load, in one of the cells participating in the connection, exceeds the allowed threshold.

Let us discuss the soft handoff connection of class i in which only cells z and h are involved. The blocking probability $B^{z}_{z,h,i}$ for class i calls in this handoff connection in the cell z and the blocking probability $B^{h}_{h,z,i}$ for class i calls in the cell h depends on the traffic streams offered to the cell z and to the neighbouring cell h. Thus, taking into consideration this assumption, we can express the above with the following functions:

$$B^{z}_{z,h,i} = f\left\{\begin{array}{l}(a_{z,1}, t_{z,1}), \ldots, (a_{z,M}, t_{z,M}),\\ (a^{z}_{z,h,1}, t^{'}_{z,h,1}), \ldots, (a^{z}_{z,h,M}, t^{'}_{z,h,M})\end{array}\right\}, \tag{10}$$

and

$$B^h_{h,z,i} = f\left\{\begin{array}{l}(a_{h,1}, t_{h,1}), \ldots, (a_{h,M}, t_{h,M}),\\ (a^h_{h,z,1}, t^{'}_{h,z,1}), \ldots, (a^h_{h,z,M}, t^{'}_{h,z,M})\end{array}\right\}. \tag{11}$$

The parameters $t^{'}_{z,h,1}, \ldots, t^{'}_{z,h,M}$ and $t^{'}_{h,z,1}, \ldots, t^{'}_{h,z,M}$ express the amount of resources required by the class i connection being in soft handoff, and can be determined in the following way:

$$t^{'}_{z,h,i} = \left\lfloor L^{'}_{z,h,i}/L_{z,BBU} \right\rfloor \quad \text{and} \quad t^{'}_{h,z,i} = \left\lfloor L^{'}_{h,z,i}/L_{h,BBU} \right\rfloor, \tag{12}$$

where $L_{z,BBU}$ for the cell z can be obtained on the basis of (6):

$$L_{z,BBU} = GCD(L_{z,1}, \ldots, L_{z,M}, L^{'}_{z,h,1}, \ldots, L^{'}_{z,h,M}), \tag{13}$$

and $L_{h,BBU}$ for the cell h:

$$L_{h,BBU} = GCD(L_{h,1}, \ldots, L_{h,M}, L^{'}_{h,z,1}, \ldots, L^{'}_{h,z,M}). \tag{14}$$

The functions $B^z_{z,h,i}$ and $B^h_{h,z,i}$ can be determined on the basis of FAG (Eqs. (5) and (9)).

A fixed-point methodology (FPM) was used to determine traffic $a^z_{z,h,i}$ offered to the cell z by a class i call being in soft handoff connection with the cell h. In keeping with this method, only such traffic which is not blocked in the neighbouring cell can be offered to a given cell. This phenomenon leads to a decrease in the traffic offered to a given cell and is called the thinning effect [10]. The class i traffic stream, which is offered to cell z by a call occurring in the access cell z, decreased by the thinning effect, is called effective traffic. This traffic - in accordance with FPM - can be determined on the basis of the following dependence:

$$a^z_{z,h,i} = a^{SH}_{z,h,i}\left(1 - B^h_{h,z,i}\right), \tag{15}$$

and the class i traffic stream, which is offered to cell h by a call being in soft handoff with cell z, can be calculated as follows:

$$a^h_{h,z,i} = a^{SH}_{z,h,i}\left(1 - B^z_{z,h,i}\right), \tag{16}$$

where $a^{SH}_{z,h,i}$ is class i traffic offered to the system by users being in soft handoff with cells z and h in the uplink direction.

It should be noted that to determine the effective class i traffic $a^z_{z,h,i}$, the information on the blocking probability $B^h_{h,z,i}$ of the traffic of this class in the neighbouring cells is indispensable. Therefore, to determine the value $a^z_{z,h,i}$ the iterative method is used.

If we know the blocking probability $B^h_{h,z,i}$ of class i call in the cell h offered simultaneously in the soft handoff connection in the cell z, we can determine the blocking probability $B^{SH}_{z,h,i}$ of class i calls in the soft handoff connection:

$$B^{SH}_{z,h,i} = 1 - (1 - B^z_{z,h,i})(1 - B^h_{h,z,i}). \tag{18}$$

Algorithm 1. Iterative algorithm for blocking probabilities calculation in the uplink direction for calls in the soft handoff connections with the cell z and h.

1. Determination of the values of $t'_{z,h,i}$ and $t'_{h,z,i}$ on the basis of Eqs. (12), (13) and (14).
2. Setting the iteration number $l = 0$.
3. Determination of initial values of: $B^{SH(0)}_{z,h,i} = 0$.
4. Determination of the values of the effective traffic $a^{z(l)}_{z,h,i}$, $a^{h(l)}_{h,z,i}$ on the basis of Eq. (16) and Eq. (15).
5. Increase in the iteration number: $l = l + 1$.
6. Determination of the values of blocking probabilities $B^{z(l+1)}_{z,h,i}$ and $B^{h(l+1)}_{h,z,i}$ on the basis of Eqs.(10) and (11).
7. Determination of the values of blocking probabilities $B^{SH(l+1)}_{z,h,i}$ on the basis of Eq. (18).
8. Repetition of Steps No. 4–7 until the assumed accuracy of the iterative process is obtained:
$$\forall_{1\le i\le M}\left(\left|\frac{B^{SH(l)}_{z,h,i} - B^{SH(l+1)}_{z,h,i}}{B^{SH(l+1)}_{z,h,i}}\right| \le \xi\right), \tag{17}$$
where $B^{SH(l)}_{z,h,i}$ and $B^{SH(l+1)}_{z,h,i}$ are the appropriate values of blocking probabilities, obtained in iteration l and $l + 1$, respectively.

To sum up, the iterative algorithm for a determination of blocking probability $B^{SH}_{z,h,i}$ can be written in the form of Algorithm 1.

Figure 1 also shows an example in which there are three cells involved in the soft handoff connection. Following the presented algorithm, we can easily determine the blocking probability in that case as well.

4 Numerical Examples

The proposed analytical model of the WCDMA interface carrying soft handoff traffic is an approximate one. Thus, the results of the analytical calculations of the WCDMA interface were compared with the results of the simulation experiments. The study was carried out for users serviced by 7-cell UMTS system (Fig. 1) which demanded a set of services (Tab. 1) in the uplink direction and it was assumed that:

- a call of the particular services demanded $t_1 = 53$, $t_2 = 257$, $t_3 = 503$ and $t_4 = 1118$ BBUs,
- the traffic classes were offered in the following proportions: $a_1\ t_1 : a_2\ t_2 : a_3\ t_3 : a_4\ t_4 = 15 : 5 : 40 : 40$,
- the L_{BBU} was equal to 0.0001,
- the accuracy of the iterative process was equal to 0.0001 (ξ in Algorithm 1),
- traffic of each class was divided into soft handoff and non-soft handoff traffic, and this division depended on the traffic classes as follows:

  - voice: 40% for soft handoff traffic (2 cells - 30%, 3 cells - 10%) and 60% for non-soft handoff traffic,
  - otherwise: 20% for soft handoff traffic (2 cells - 15%, 3 cells - 5%) and 80% for non-soft handoff traffic.
- the amount of required resources by calls of class i (t_i) carried the soft handoff connection also depended on the number of cells participating in this connection:
  - $t_i^{'}$ was equal to 80% of t_i for the soft handoff connection with 2 cells,
  - $t_i^{''}$ was equal to 60% of t_i for the soft handoff connection with 3 cells.
- the maximum uplink direction load of one cell were set to 80% of the theoretical capacity (Eq. (8)):

$$V_{UL} = \left\lfloor \frac{V}{L_{BBU}} \right\rfloor = \left\lfloor \frac{80\%}{0.0001} \right\rfloor = 8000 BBUs.$$

Table 2 shows the influence of the accuracy, expressed as the relative error (17), on the number of iterations for three exemplary values of traffic offered per BBU. It can be observed that the higher accuracy and the higher traffic offered per BBU, the higher the number of iterations.

Table 2. Number of iterations in the functions of the assumed accuracy

offered traffic per BBU	$\xi = 10^{-3}$	$\xi = 10^{-4}$	$\xi = 10^{-5}$	$\xi = 10^{-6}$	$\xi = 10^{-7}$
0.3	2	3	3	3	4
0.7	4	5	5	6	7
1.1	5	6	7	8	9

Table 3. Exemplary values of blocking probability obtained for particular traffic classes in relation to the successive iterations of the calculation algorithm ($a = 0.7$, $\xi = 10^{-5}$)

Step	Speech	Accuracy	Video	Accuracy	Data 144	Accuracy	Data 384	Accuracy
1	0.005897	0.0058976	0.027853	0.0278536	0.058272	0.0582726	0.151271	0.1512713
2	0.005555	0.0003422	0.026291	0.0015622	0.055146	0.0031266	0.144140	0.0071303
3	0.005572	0.0000172	0.026370	0.0000790	0.055304	0.0001585	0.144504	0.0003633
4	0.005571	**0.0000008**	0.026366	**0.0000040**	0.055296	**0.0000080**	0.144485	0.0000184
5	0.005571	0.0000000	0.026366	0.0000002	0.055296	0.0000004	0.144486	**0.0000009**

Tables 3 and 4 present a relation between the obtained values of the blocking probability for particular traffic classes shown in Tab. 1 to the number of iterations of the calculation Algorithm 1 for two exemplary values of traffic offered to the system per BBU and for the assumed accuracy (expressed as the absolute error). The number of iterations of the calculation process shown in Tab. 3 is lower in comparison with the corresponding values in Tab. 4. Thus,

Table 4. Exemplary values of blocking probability obtained for particular traffic classes in relation to the successive iterations of the calculation algorithm ($a = 1.1$, $\xi = 10^{-5}$)

Step	Speech	Accuracy	Video	Accuracy	Data 144	Accuracy	Data 384	Accuracy
1	0.026769	0.0267698	0.117664	0.1176649	0.2252515	0.2252515	0.466586	0.4665867
2	0.023523	0.0032465	0.104281	0.0133830	0.201721	0.0235303	0.428465	0.0381214
3	0.023836	0.0003136	0.105583	0.0013018	0.204029	0.0023084	0.432288	0.0038233
4	0.023805	0.0000309	0.105455	0.0001285	0.203801	0.0002277	0.431912	0.0003764
5	0.023809	**0.0000030**	0.105467	0.0000126	0.203824	0.0000224	0.431949	0.0000371
6	0.023808	0.0000003	0.105466	**0.0000012**	0.203822	**0.0000022**	0.431945	0.0000036
7	0.023808	0.0000000	0.105466	0.0000001	0.203822	0.0000002	0.431946	**0.0000003**

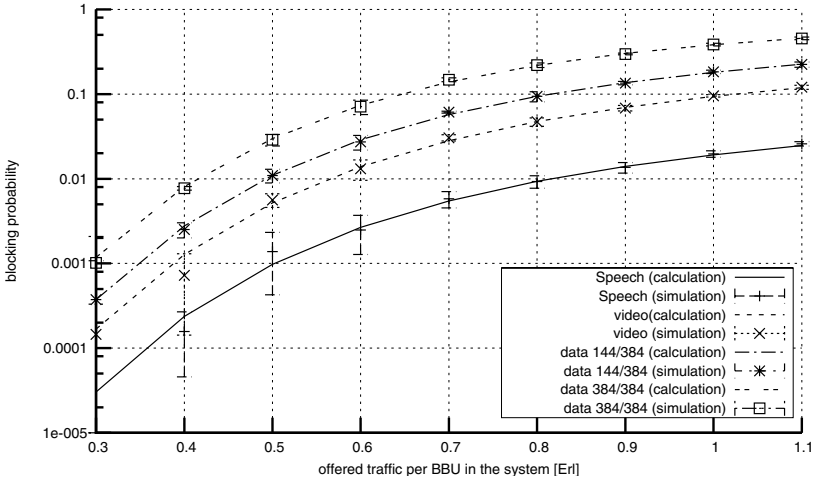


Fig. 2. Blocking probabilities for the part of calls of all traffic classes serviced in the system by the soft handoff mechanism (connection with 2 cells)

the number of iterations increases when offered traffic increases. Notice that the required number of iterations also depends on the traffic class - the higher number of BBUs demanded by calls of a given class the higher number of iteration steps (required to obtain the assumed accuracy). We can conclude that the total number of iterations for each value of traffic offered per BBU, with the assumed accuracy, depends on the number of BBUs require by this traffic class which demanded the highest number of BBUs for set up connection.

Figures 2 and 3 present the values of the blocking probability obtained for all the traffic classes in relation to traffic offered per BBU in the group of cells presented in Fig. 1, in the UMTS network, in which soft handoff mechanism was applied.

The results presented in Figure 2 were obtained for this part of all traffic classes which was serviced by the soft handoff mechanism in which simultaneous

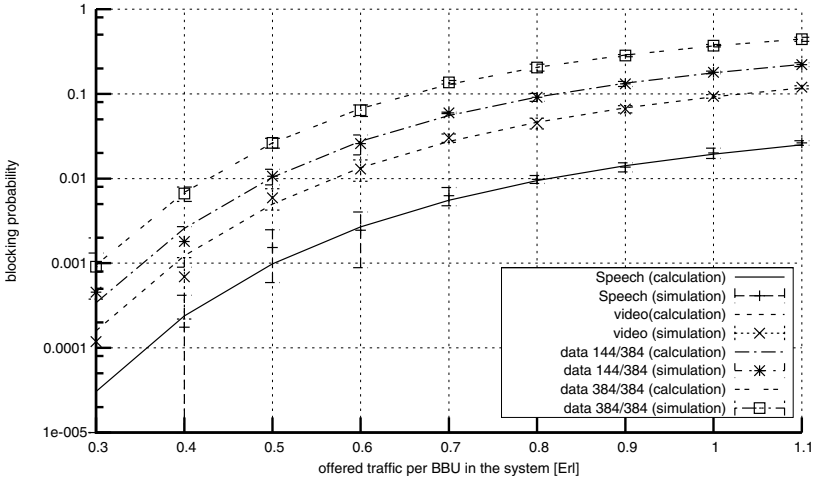


Fig. 3. Blocking probabilities for the part of calls of all traffic classes serviced in the system by the soft handoff mechanism (connection with 3 cells)

connections with two cells were established. Figure 3 shows the results obtained for the art of all traffic classes which was serviced by the soft handoff mechanism in which parallel connections with three cells were established.

Comparing the results presented in Figs. 2 and 3 it is noticeable that the increase in the number of cells participating in the soft handoff connection leads to the increase in blocking in these parts of traffic streams which are involved in handoff connections. This phenomenon results from the increase of resources required to set up a handoff connection in the group of cells (e.g. $3t_i^{''} > 2t_i^{'}$).

The analytical results were validated by simulation experiments which were carried out on the basis of our own simulation program. The simulator was implemented in the Python language and it used the event scheduling simulation method [25]. The simulation model, however, does not take into consideration many technological properties of the UMTS system such as the propagation model of the radio channel or the mobility of users. The devised simulator is intended for modelling the traffic capacity of the system, thus for modelling of a system at the so-called call level [26], and at this level, technological parameters have no significant influence upon the mapping accuracy of a modelled system [27]. In the conducted simulation research shown in Figs. 2 and 3, each point of the plot is the average value of the blocking probabilities obtained in 5 series. It was assumed that in particular simulation series 10^7 of the incoming calls of the "oldest"[1] class were offered. The results of the simulations are shown in the charts in the form of marks with 95% confidence intervals calculated after the t-Student distribution. 95% confidence intervals of the simulation are almost included within the marks plotted in the figures.

[1] The class which demands the highest number of BBUs.

5 Conclusions

The paper proposes a new simple calculation method for blocking probability determination in the WCDMA radio interface carrying a mixture of multi-rate traffic in the network, in which soft handoff mechanism is applied.

In the presented model, the influence of soft handoff traffic on the service processes for calls of all traffic classes in each cell is taken into consideration by the application of the fixed-point methodology. The obtained results show that taking into account three cells instead of two cells in the soft handoff connections leads to the increase in the value of blocking probability for each traffic classes.

The results of the analytical calculations were compared with the results of the simulation experiments, which confirmed the good accuracy of the proposed method. The proposed scheme can be applicable for a cost-effective radio resource management in 3G mobile networks and can be easily applied to network capacity calculations.

References

1. Nawrocki, M., Aghvami, H., Dohler, M.: Understanding UMTS Radio Network Modelling, Planning and Automated Optimisation: Theory and Practice. John Wiley and Sons, Ltd., Chichester (2006)
2. Holma, H., Toskala, A.: WCDMA for UMTS. Radio Access For Third Generation Mobile Communications. John Wiley and Sons, Ltd., Chichester (2000)
3. Laiho, J., Wacker, A., Novosad, T.: Radio Network Planning and Optimization for UMTS. John Wiley and Sons, Ltd., Chichester (2006)
4. Subramaniam, K., Nilsson, A.A.: Tier-based analytical model for adaptive call admission control scheme in a UMTS-WCDMA system. In: Proceedings of Vehicular Technology Conference, vol. 4, pp. 2181–2185 (2005)
5. Głąbowski, M., Sobieraj, M., Stasiak, M.: Evaluation of traffic characteristics of UMTS with bandwidth reservation and handoff mechanism. In: Proceedings of The 14th IEEE International Conference On Telecommunications, Penang, Malaysia (2007)
6. Głąbowski, M., Sobieraj, M., Stasiak, M.: Blocking probability calculation in UMTS networks with bandwidth reservation, handoff mechanism and finite source population. In: Proceedings of the 7th International Symposium on Communications and Information Technologies, Sydney, Australia, pp. 433–438 (2007)
7. Vassilakis, V.G., Logothetis, M.D.: The wireless Engset multi-rate loss model for the handoff traffic analysis in W-CDMA networks. In: Proccedings of 19th International Symposium on Personal, Indoor and Mobile Radio Communications, Cannes, France, pp. 1–6 (2008)
8. Stasiak, M., Zwierzykowski, P., Wiewióra, J., Parniewicz, D.: An Approximate Model of the WCDMA Interface Servicing a Mixture of Multi-rate Traffic Streams with Priorities. In: Thomas, N., Juiz, C. (eds.) EPEW 2008. LNCS, vol. 5261, pp. 168–180. Springer, Heidelberg (2008)
9. Stasiak, M., Wiśniewski, A., Zwierzykowski, P., Głąbowski, M.: Blocking probability calculation for cellular systems with WCDMA radio interface servicing PCT1 and PCT2 multirate traffic. IEICE Transactions on Communications E92-B(4) (2009)

10. Kelly, F.: Loss networks. The Annals of Applied Probability 1(3), 319–378 (1991)
11. Laiho, J., Wacker, A., Novosad, T.: Radio Network Planning and Optimization for UMTS, 2nd edn. John Wiley and Sons, Ltd., Chichester (2006)
12. Hiltunen, K., De Bernardi, R.: WCDMA downlink capacity estimation. In: The IEEE Semiannual Vehicular Technology Conference. IEEE, Los Alamitos (2000)
13. Pedersen, K.I., Mogensen, P.E.: The downlink code orthogonality factors influence on WCDMA system performance. In: Proceedings of the 56th IEEE Vehicular Technology Conference, Vancouver, Canada, pp. 2061–2065 (2002)
14. Mehta, N., Willis, L.G.T., Kostic, Z.: Analysis and results for the orthogonality factor in WCDMA downlinks. IEEE Transactions on Wireless Communications 2(6), 1138–1149 (2003)
15. El-Sallabi, H.M., Salo, J., Vainikainen, P.: Investigations on impact of bandwidth on orthogonality factor for WCDMA systems. In: Proceedings of the IEEE Wireless Communications and Networking Conference, Atlanta, pp. 1138–1142 (2004)
16. Kennerstedt, F.: Estimation of non-orthogonality and other-to-own cell interference in a WCDMA radio network. Internal Technical Report EAB/PD-06:0082 Uen, Ericsson (2006)
17. Kelly, F.P.: Notes on effective bandwidths. In: Stochastic Networks: Theory and Applications, pp. 141–168. Oxford University Press, Oxford (1996)
18. Iversen, V., Epifania, E.: Teletraffic engineering of multi-band W-CDMA systems. In: Network control and engineering for QoS, security and mobility II, pp. 90–103. Kluwer Academic Publishers, Dordrecht (2003)
19. Staehle, D., Mäder, A.: An analytic approximation of the uplink capacity in a UMTS network with heterogeneous traffic. In: 18th International Teletraffic Congress (ITC18), Berlin, pp. 81–91 (2003)
20. Iversen, V., Benetis, V., Trung, H.: Evaluation of multi-service CDMA networks with soft blocking. In: ITC 16th Specialist Seminar on Performance Evaluation of Mobile and Wireless Systems, Antwerp, Belgium, pp. 212–216 (2004)
21. Staehle, D.: Analytic Methods for UMTS Radio Network Planning. PhD thesis, Bayerische Julius-Maximilians-Universität Würzburg (2004)
22. Kallos, G.A., Vassilakis, V.G., Logothetis, M.D.: Call blocking probabilities in a W-CDMA cell with fixed number of channels and finite number of traffic sources. In: Proceedings of 6th International Conference on Communication Systems, Networks and Digital Signal Processing, Graz, Austria, pp. 200–203 (2008)
23. Kaufman, J.: Blocking in a shared resource environment. IEEE Transactions on Communications 29(10), 1474–1481 (1981)
24. Roberts, J.: A service system with heterogeneous user requirements – application to multi-service telecommunications systems. In: Pujolle, G. (ed.) Proceedings of Performance of Data Communications Systems and their Applications, pp. 423–431. North Holland, Amsterdam (1981)
25. Tyszer, J.: Object-Oriented Computer Simulation Of Discrete-Event Systems. Kluwer Academic Publishers, Dordrecht (1999)
26. Roberts, J. (ed.): Performance Evaluation and Design of Multiservice Networks, Final Report COST 224. Commission of the European Communities, Brussels (1992)
27. Wiśniewski, A.: Modelling of the mobile systems with WCDMA radio interface. PhD thesis, Poznan University of Technology, Faculty of Electronics and Telecommunications, Poznan, Poland (2009)

Analytical Model of TCP NewReno through a CTMC

Nimbe L. Ewald and Andrew H. Kemp

School of Electronic and Electrical Engineering
University of Leeds, UK

Abstract. An analytical model of the Transmission Control Protocol (TCP) New Reno [7] performance through a Continuous-Time Markov Chain (CTMC) is presented and its theoretical predictions are corroborated by the well known network simulator ns-2 [15]. An existing TCP Reno model [6] is modified in order to characterise the NewReno algorithm. The NewReno version is modelled given its proven better performance over channels presenting high loss rates and its extensive deployment in current web servers [14].

Keywords: Continuous-time Markov Chains, TCP NewReno, TCP Reno, ns-2.

1 Introduction

TCP is the transport protocol most widely used over the Internet therefore an analytical model which accurately characterises its performance will result in a more efficient use of network resources which will provide a better quality of service for the user and higher revenue to service providers. Furthermore, a better understanding of TCP dynamics will be gained, which will lead to the development of TCP optimisations, extensions and new TCP-friendly protocols. For instance, the Reno model of [16] results in the well known PFTK00 throughput equation which has been widely used as a reference to predict the performance of other characterisations and modifications of TCP. This is the approach taken by [3] & [19] where TCP performance over a wireless link implementing a hybrid Automatic Repeat reQuest (ARQ) scheme in the link-layer (LL) is measured by the PFTK00 equation. It is also used in the design of the TCP-Friendly Rate Control protocol [9] for time-sensitive applications where its sending rate is calculated by it.

Multiple mathematical characterisations of TCP have been developed in order to predict, evaluate and optimise TCP performance. The most relevant analytical models are shown in table 1. As it can be seen, model features differ among them: all of the models represent the TCP Reno version but only [11,6,22,18] characterise all its phases[1]: Slow Start (SS), Congestion Avoidance (CA), Fast Recovery (FR) and Timeout (TO). However, only [22] specifies the type of simulator used to corroborate its theoretical predictions which makes easier the result

[1] It is assumed that the reader is familiar with TCP congestion control algorithms.

J.T. Bradley (Ed.): EPEW 2009, LNCS 5652, pp. 183–196, 2009.

Table 1. TCP analytical models

Model	Version(s)	Phases Modelled	Validation Method(s)
LM97 [12]	Reno	SS, CA & TO	Custom-made simulator
AK98 [11]	OldTahoe, Tahoe, Reno & NewReno	SS, CA, FR & TO	Custom-made simulator
CLM99 [5]	Tahoe & Reno	CA, FR & TO	Custom-made simulator
PFTK00 [16]	Reno	CA, FR & TO	Internet traces
CM02 [6]	Reno	SS, CA, FR & TO	Not specified
ASM03 [1]	Reno	CA, FR & TO	ns-2
WOO03 [22]	Reno, SACK & Vegas	SS, CA FR & TO	ns-2 &
SKV03 [20]	Tahoe, Reno & SACK	CA, FR & TO	ns-2 & Internet traces
RVZ04 [18]	Tahoe, Reno & NewReno	SS, CA, FR & TO	Not specified
VVB06 [21]	Reno	CA, FR & TO	ns-2

comparisons and the consequent selection of an analytical characterisation. Papers [11,18] additionally model the NewReno version and all its algorithms. The model presented here differs from those of [11,18] in the analytical approach: these papers model TCP algorithms as discrete-time processes whilst our characterisation consider them as continuous-time. In addition, as a known network simulator such as ns-2 is used here, it is easier to reproduce the reported results.

Some of these TCP characterisations such as [12,16,20,22] are designed specifically for wired topologies whilst others such as [1,6,5,11,18,21] represent TCP performance over wired-wireless networks. Performance is mainly modelled by two approaches: i) [1,11,18] analyse TCP efficiency directly over the wireless link. ii) [5,6,21] study TCP interactions with LL mechanisms. The main common characteristic of the models in table 1 is the analytical tool used to represent either the radio link, the LL mechanisms or the TCP algorithms: a Markovian analysis. In addition, the loss distribution assumptions are usually the same. Independent losses are considered to take place in wireless channels undergoing fast-fading or using error control techniques in the LL such as ARQ mechanisms and Forward Error Correction (FEC) codes or in wired links with router buffers implementing Active Queue Management schemes such as Random Early Detection. On

the other hand, correlated losses are assumed to occur in slow-fading wireless channels or wired links with DropTail router buffers. The model presented here considers independent losses therefore it can be used to represent TCP NewReno performance in the mentioned network scenarios.

This work is divided as follows: a brief description of the TCP versions which have been implemented in web servers is presented in section 2. TCP NewReno analytical model is developed in section 3. Assessment of the accuracy of the theoretical model is reported in section 4. Finally, section 5 concludes the work.

2 TCP Versions

TCP has evolved, in the form of different versions, to present a better response to lost segments. In order to understand the differences among the various TCP versions, a brief description of its loss indication signals is needed. TCP has two mechanisms to detect segment loss: triple-duplicate acknowledgements (TD ACKs) and TOs. The receiver acknowledges a segment reception by indicating the next expected byte in the TCP header Acknowledgement field. If segments other than those expected arrive, the receiver keeps generating ACKs with the same expected byte which are duplicate ACKs. The reception of TD ACKs triggers TCP congestion control algorithms. On the other hand, the TCP sender maintains a retransmission timer, the duration of which indicates the expected arrival of ACKs. If the expected ACKs are not received during this period, the timer expires causing TCP to enter the SS phase.

The outdated OldTahoe version [17] only recovers a lost segment by a TO. Meanwhile, the Tahoe version [10] implements the Fast Retransmit algorithm but it does not have the FR[2] algorithm. Therefore, if a lost segment was detected by TD ACKs, the TCP sender would re-initiate in SS after its retransmission. TCP Reno [2] introduces FR. In the case of loss detection by TD ACKs, the Fast Retransmit algorithm is invoked: the lost segment is retransmitted as soon as they are received. If the lost packet is acknowledged, the FR algorithm halves the *cwnd* in which the loss occurred and TCP enters the CA phase. Reno keeps halving its *cwnd* each time a segment belonging to the same *cwnd* is detected, by TD ACKs, and only one segment is retransmitted each time these additional Fast Retransmissions are triggered. This leads to small *cwnds* causing fewer segments to be sent during FR and eventually may cause unnecessary TOs. It also causes CA to initiate with very small *cwnds*. In order to overcome this problem, TCP NewReno was proposed. NewReno also implements Fast Retransmit when TD ACKs arrive and immediately enters FR where the arrival of the first partial ACK[3] triggers the retransmission of the next expected segment (instead of waiting for other set of TD ACKs as in the Reno case) and the reset of the retransmission timer. Furthermore, the *cwnd* is halved just once whilst an extra segment is retransmitted each time a new partial ACK is received. Therefore, the

[2] The abbreviation FR refers only to the Fast Recovery algorithm.

[3] An ACK elicited from a segment retransmission which does not acknowledge all data in flight.

recovery of multiple lost packets is faster than with Reno and bandwidth is better used. As a result of this modification, the NewReno Fast Recovery algorithm becomes more aggressive than that of Reno. Finally, TCP Selective Acknowledgement (SACK) Options [13] implements Reno algorithms with the added feature of reporting in the TCP header Options field the segments which have been received thus the sender only retransmits those which were not advertised.

TCP deployment has been dominated by the NewReno and SACK implementations in recent years [14]. However, the Reno version still exists in numerous web servers and it has been the most represented version in analytical models as shown in table 1. As NewReno performs better than the extensively modelled Reno version, the aim of this paper is to present a TCP NewReno analytical model which represents its main characteristics: the retransmission of segments when partial ACKs are received and the single halving of its *cwnd* during FR mode. In addition, its theoretical predictions are validated through ns-2 simulations.

3 TCP NewReno Analytical Model

The mathematical procedure of [6] is extended in order to analytically represent the NewReno Fast Recovery algorithm. TCP segment generation rate λ and its subsequent TCP throughput are the performance targets to be obtained through this analysis where the following assumptions are made:

1. The higher-layer application passes data to the transport layer such that TCP is the protocol which carries out the data transfer.
2. The higher-layer application source presents a bursty nature. That is, it sends and stops sending data to the TCP transmitter intermittently. The time durations spent in each of these *ON* and *OFF* states are random variables, α and β, with negative exponential distribution.
3. TCP *cwnd*, is measured in Maximum Segment Sizes (MSS) instead of bytes. The *cwnd* values are integers and particularly powers of 2 during the SS phase.
4. The same average rtt is considered for the whole length of the transfer.
5. TCP connection establishment is not included in the characterisation.
6. Mainly TCP transmitter operation is modelled given that it is assumed that TCP ACKs are never lost.

3.1 TCP NewReno CTMC

The CTMC representing the TCP NewReno transmitter dynamics conveying data from a bursty application is shown in figure 1 where a maximum *cwnd* equal to 10 MSS is considered. It can be observed that all TCP algorithms such as SS, CA, FR and TO are represented. It is important to note that in order to clarify the figure, transitions from all states to OFF are not totally represented as well as some transitions from CA to FR. The chain state space is given by: $S_{OFF} \bigcup S_{SS} \bigcup S_{CA} \bigcup S_{FR} \bigcup S_{TO}$.

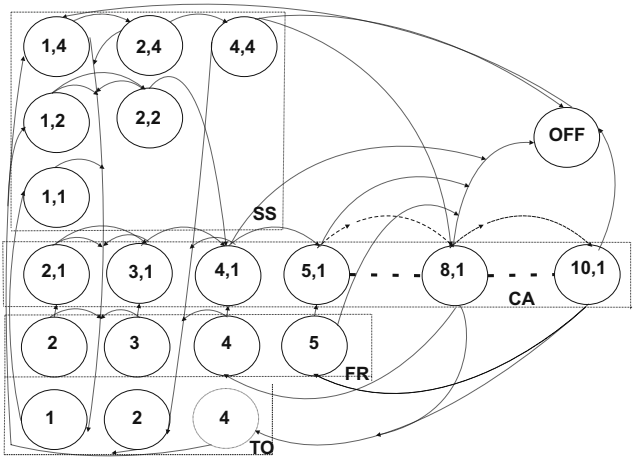


Fig. 1. CTMC representing TCP algorithms

OFF. The state *(OFF)* is the only element of set S_{OFF}.
SS. The set of the SS mechanism, S_{SS}, collects the states corresponding to this exponential growth phase behaviour:

$$S_{SS} = \{s = (cwnd, Th)\}$$

where $cwnd \in (1, \lfloor \frac{cwnd_{max}}{2} \rfloor)$. $cwnd_{max}$ corresponds to the maximum allowed receiver window size. If $\lfloor \frac{cwnd_{max}}{2} \rfloor$ value is not equal to a power of 2, the lower nearest power of 2 is chosen. Th is the threshold value and is initially set to $\lfloor \frac{cwnd_{max}}{2} \rfloor$ or to the lower nearest power of 2. The initial settings of TCP in SS mode are $(1, \lfloor \frac{cwnd_{max}}{2} \rfloor)$. As the threshold is halved every time a segment loss occurs, $Th \in (1, \lfloor \frac{cwnd_{max}}{2} \rfloor)$. If $cwnd$=Th and if no segment losses are detected, the system enters the CA phase.
CA. The states representing the CA algorithm belong to the set S_{CA}:

$$S_{CA} = \{s = (cwnd, 1)\}$$

In this case, $cwnd \in (2, cwnd_{max})$, whilst the second term, *1*, symbolises the dynamics of the congestion window growth in this phase: an extra segment per rtt if no losses are detected. Therefore, $cwnd$ sizes do not necessarily correspond to powers of 2 as in S_{SS}. Once the system reaches state $s_{(cwnd_{max},1)}$, it remains there unless a segment loss occurs and the FR mechanisms take place.
FR. The set S_{FR} comprises the states representing the $cwnd$ of the NewReno Fast Recovery algorithm once Fast Retransmit has been implemented:

$$S_{FR} = \{s = (cwnd)\}$$

where $cwnd \in (2, \lfloor \frac{cwnd_{max}}{2} \rfloor)$. If a loss event occurs and if the $cwnd$ is equal to or higher than 4, states from both S_{SS} and S_{CA} go to a state in S_{FR} which is

equal to the nearest lower integer of half their current *cwnds*. When loss(es) are not recovered in this phase, the system enters the TO mode.

TO. Finally, the states of the TO mechanism are collected in the set S_{TO}:

$$S_{TO} = \{s = (Th)\}$$

where $Th \in (1, \lfloor \frac{cwnd_{max}}{2} \rfloor)$. The elements of this set are all powers of 2. If the states in $S_{SS} \bigcup S_{CA} \bigcup S_{FR}$ cannot recover from lost segments, they will move to a corresponding state in S_{TO}. These states represent the threshold to be set in the next SS phase after the timeout expiration period.

All states, but those belonging to S_{TO}, have transitions to s_{OFF} given that the application source can stop sending data at any time.

TCP NewReno transition rate matrix. In order to clearly present the construction of TCP transition rate matrix, Q_{TCP}, the order of its states is displayed in equation 1 whilst its transition rates are shown in table 2 for a generic *cwnd* value. ν is defined as $\lfloor log_2(cwnd_{max}) \rfloor$ and is assumed equal to 3 in the construction of Q_{TCP} of equation 1.

$$Q_{TCP} = \begin{pmatrix} (1, \lfloor \frac{cwnd_{max}}{2} \rfloor) \in S_{SS} \\ \vdots \\ (\lfloor \frac{cwnd_{max}}{2} \rfloor, \lfloor \frac{cwnd_{max}}{2} \rfloor) \in S_{SS} \\ (1, \lfloor \frac{cwnd_{max}}{2^{\nu-1}} \rfloor) \in S_{SS} \\ \vdots \\ (\lfloor \frac{cwnd_{max}}{2^{\nu-1}} \rfloor, \lfloor \frac{cwnd_{max}}{2^{\nu-1}} \rfloor) \in S_{SS} \\ (1, \lfloor \frac{cwnd_{max}}{2^{\nu}} \rfloor) \in S_{SS} \\ (2, 1) \in S_{CA} \\ \vdots \\ (cwnd_{max}, 1) \in S_{CA} \\ (2) \in S_{FR} \\ \vdots \\ (\lfloor \frac{cwnd_{max}}{2} \rfloor) \in S_{FR} \\ (1) \in S_{TO} \\ \vdots \\ (2^{\nu-1}) \in S_{TO} \\ (OFF) \in S_{OFF} \end{pmatrix} \quad (1)$$

The number of elements in S_{SS}, S_{CA}, S_{FR} and S_{TO} is $\frac{\nu(\nu+1)}{2}$, $cwnd_{max} - 1$, $\lfloor \frac{cwnd_{max}}{2} \rfloor$ and ν, respectively. Therefore, the dimension of Q_{TCP} is given by: $q = \frac{\nu(\nu+1)}{2} + \lfloor \frac{3cwnd_{max}}{2} \rfloor + \nu - 1$.

The relations between Q_{TCP} transition rates and the actual TCP algorithms are explained as follows. As soon as the higher-layer application starts passing data to the transport layer, TCP divides the data in MSS and begins the connection by setting its threshold to the nearest lower power of 2 of $\lfloor \frac{cwnd_{max}}{2} \rfloor$.

Table 2. Q_{TCP} transition rates

From state	To state	Transition rate	Conditions
$(OFF) \in S_{OFF}$	$(1, Th') \in S_{SS}$	α	$Th' = \frac{cwnd_{max}}{2}$
$(cwnd, Th) \in S_{SS}$	$(cwnd', Th) \in S_{SS}$	$\delta p_{NL}(cwnd)$	$cwnd' = 2cwnd$ $cwnd' \leq \lfloor \frac{cwnd_{max}}{2} \rfloor$
$(cwnd, Th) \in S_{SS}$	$(cwnd', 1) \in S_{CA}$	$\delta p_{NL}(cwnd)$	$cwnd' = 2cwnd$ $cwnd = Th$
$s \in S_{SS} \bigcup S_{CA}$	$(Th') \in S_{TO}$	$\delta[1 - p_{NL}(cwnd)]$	$Th' = max(1, \lfloor \frac{cwnd}{2} \rfloor)$ if $cwnd \leq 4$
$s \in S_{SS} \bigcup S_{CA}$	$(cwnd') \in S_{FR}$	$\delta p_L(cwnd)$	$cwnd' = \lfloor \frac{cwnd}{2} \rfloor$ if $cwnd \geq 4$ and if $losses \leq \frac{TO}{rtt} + 1$
$(cwnd, 1) \in S_{CA}$	$(cwnd', 1) \in S_{CA}$	$\delta p_{NL}(cwnd)$	$cwnd' = (cwnd + 1)$ if $cwnd < cwnd_{max}$
$s \in S_{SS} \bigcup S_{CA}$	$(Th') \in S_{TO}$	$\delta[1 - p_{NL}(cwnd) - p_L(cwnd)]$	$Th' = \lfloor \frac{cwnd}{2} \rfloor$ if $cwnd \geq 4$ and if $losses \leq \frac{TO}{rtt} + 1$
$(cwnd) \in S_{FR}$	$(cwnd', 1) \in S_{CA}$	$\delta(p_{NL}(cwnd) + p_L(cwnd)$	$cwnd' = cwnd$ if $losses \leq \frac{TO}{rtt}$
$(cwnd) \in S_{FR}$	$(Th') \in S_{TO}$	$\frac{1}{(\frac{TO}{rtt}+1)rtt}[1 - p_{NL}(cwnd) - p_L(cwnd)]$	$Th' = \lfloor \frac{cwnd}{2} \rfloor$ if $losses > \frac{TO}{rtt}$
$(Th) \in S_{TO}$	$(1, Th') \in S_{SS}$	ζ	$Th' = Th$
$s \in S_{SS} \bigcup S_{CA} \bigcup S_{FR}$	$(OFF) \in S_{OFF}$	β	–

The chain transition from s_{OFF} to $s_{(1,Th)} \in S_{SS}$ represents the beginning of the TCP transfer. The transition rate α is the result of multiplying the occupancy time in s_{OFF} by the probability that the source passes data to TCP, which is 1.

Once the system is in Slow Start mode, if there is not a segment loss, state $(cwnd, Th)$ goes to $(cwnd', Th)$ with rate $\delta p_{NL}(cwnd)$. $cwnd'$ is equal to the double of the previous $cwnd$ value whilst δ is the parameter of the exponential distribution governing the occupancy time of the states and is equal to $\frac{1}{rtt}$. $p_{NL}(cwnd)$ is the embedded Markov chain probability of not having losses during the transmission of a certain $cwnd$ and is calculated as:

$$p_{NL}(cwnd) = (1 - p_{total})^{cwnd} \tag{2}$$

where p_{total} is the TCP segment loss probability.

If a $cwnd$ of a state in S_{SS} reaches the threshold value, it will enter $(cwnd', 1) \in S_{CA}$ with the transition rate of $\delta p_{NL}(cwnd)$. $cwnd'$ is twice its previous value in SS.

As previously explained, TCP detects losses through the reception of TD ACKs. Therefore, $cwnd \geq 4$ is the condition to recover losses by the Fast Retransmit and FR algorithms. If at least one loss occurs and if $cwnd < 4$, states in $S_{SS} \bigcup S_{CA}$ will inevitably enter a corresponding state in S_{TO}. In the TO phase,

states will take a value of (Th') where *Th'* represents the nearest lower power of 2 of half the *cwnd*. The transition rate to TO states is equal to $\delta[1-p_{NL}(cwnd)]$.

In contrast, as soon as TCP detects a lost segment through TD ACKs, it implements Fast Retransmit and enters FR, i.e. FR algorithm can only take place if $cwnd \geq 4$. Any state of the set $S_{SS} \bigcup S_{CA}$ experiencing 1 to $(\frac{TO}{rtt}+1)$ lost segments, will enter the FR algorithm. The state suffering from losses will move to $(cwnd') \in S_{FR}$ with $cwnd' = \lfloor \frac{cwnd}{2} \rfloor$ and transition rate δp_L.

The maximum number of allowed lost segments per *cwnd* is conditional to its size. That is, at least 3 segments have to be successfully received in order to trigger the TD ACKs indication, i.e. if *cwnd*=6, the maximum number of lost segments is 3 to enter S_{FR}, otherwise it will move to S_{TO}.

Since every segment sent in the *cwnd* represents an independent event and can be equally affected by the segment loss probability p_{total}, it is assumed that the segment loss pattern follows a binomial distribution. Hence, p_L is defined as:

$$p_L(cwnd) = \sum_{losses=1}^{\frac{TO}{rtt}+1} C_{losses}^{cwnd}\, p_{total}^{losses}\, (1-p_{total})^{(cwnd-losses)} \tag{3}$$

The number of segments, $(\frac{TO}{rtt}+1)$, which can be lost in CA and recovered in FR depends on the NewReno algorithm timer which is explained below. This assumption of making the number of losses proportional to the retransmission timer to model NewReno Fast Recovery algorithm is one of the main differences from the analytical model of [6].

If no losses take place in the Congestion Avoidance mode, any state $(cwnd, 1)$, will move to $(cwnd+1, 1)$ with a rate of $\delta p_{NL}(cwnd)$.

In the FR phase it is only possible to recover $(\frac{TO}{rtt}+1)$ losses occurred in CA. If more losses occur, a state from $s \in S_{SS} \bigcup S_{CA}$ will move to S_{TO} with transition rate $\delta(1-p_{NL}-p_L)$ as long as $cwnd \geq 4$.

Unlike Reno, NewReno version avoids multiple reductions of *cwnd* every time a loss is detected. The reception of TD ACKs caused by the first loss occurring in SS or CA, triggers NewReno FR algorithm and its inherent rate halving. In contrast, the reception of partial ACKs caused by the subsequent $\frac{TO}{rtt}$ losses do not trigger any further sending rate reduction. The assumption of allowing up to $(\frac{TO}{rtt}+1)$ losses in CA and $(\frac{TO}{rtt})$ losses in FR is based on the following explanation. During the NewReno FR mode, the retransmission timer is reset after the first partial ACK is received, as stated in RFC 3782 [7]. Therefore, it will ultimately timeout after $\frac{TO}{rtt}$ rtts in which $\frac{TO}{rtt}$ segments are retransmitted given that it is only possible one retransmission per rtt.

TCP retransmits a segment as soon as it receives the TD ACKs indication and it has to wait one rtt once in FR, in order to detect a subsequent loss. If a cumulative ACK of the data in flight is received after the first rtt, no further losses occurred in CA and the TCP system can move back to CA. In other words, no losses were detected in the FR mode. Otherwise, if a partial ACK is received, more than 1 loss has occurred and the lost segment is retransmitted immediately. As, the timer is reset in reception of the first partial ACK, up to $\frac{TO}{rtt}$ segments can be retransmitted before the timer expires, i.e. the system remains in FR

mode ($\frac{TO}{rtt}+1$) rtts before it timeouts. Thus, the system will move from $s \in S_{FR}$ back to a corresponding $s \in S_{CA}$ with a rate of $\delta[p_{NL}(cwnd)+p_L(cwnd)]$, where the second term $p_L(cwnd)$ can be calculated through equation 3 considering the maximum number of losses equal to $\frac{TO}{rtt}$.

As it has been explained, it is considered that losses occurred during the CA mode are recovered during the FR phase. However, the loss probabilities of the embedded DTMC are calculated considering the *cwnd* in FR, i.e. it is considered that losses occur during the FR mode. This consideration does not significantly affect the NewReno model performance as shown in its validation in the next section.

If the number of losses during the FR is higher than $\frac{rtt}{TO}$, $s \in S_{FR}$ will enter a corresponding state $(Th) \in S_{TO}$ with a transition rate $\left(\frac{1}{(\frac{TO}{rtt}+1)rtt}\right)(1-p_{NL}-p_L)$. $\frac{1}{(\frac{TO}{rtt}+1)rtt}$ represents the state occupancy in FR during the retransmission of lost segments.

Any state $s \in S_{SS} \bigcup S_{CA} \bigcup S_{FR}$ going to the TO phase will halve its *cwnd*. This value will be the threshold of $s \in S_{TO}$. It will also be the threshold value in the SS phase when the chain moves back to $(cwnd, Th) \in S_{SS}$ in order to continue the data transfer. The transition rate from the TO states to the SS mode is, as expected, $\zeta = \frac{1}{TO}$, the inverse of the timeout duration.

If the bursty application stops sending data to TCP, the chain will enter s_{OFF} with a rate β regardless of its current state $s \in S_{SS} \bigcup S_{CA} \bigcup S_{FR}$. However, there are no transitions among $s \in S_{TO}$ and s_{OFF} because once the retransmission timer timeouts, it still needs to retransmit the whole *cwnd* which contains the lost segment(s).

3.2 TCP Performance Parameters

In order to calculate the segment generation rate of the TCP model, it is necessary to obtain the steady-state probabilities of the CTMC, $\pi_{TCP}(s)$. It can be observed in figure 1 that all states from the different sets eventually communicate hence the embedded DTMC is an irreducible chain and consequently so is the TCP CTMC. The TCP chain is also positive recurrent given that the probability of going back to any state exists. A system under these conditions will reach equilibrium therefore it will be possible to calculate its steady-state probabilities. In this work, these are calculated by the iterative algorithm developed in [8].

The average MSS generation rate per second, whilst the higher-layer application is active, is calculated as follows:

$$\lambda = \frac{1}{\frac{t_{ON}}{t_{ON}+t_{OFF}}} \Big[\sum_{s \notin S_{TO}} \delta \cdot cwnd \cdot \pi_{TCP}(s) + \sum_{s \in S_{TO}} \zeta \cdot \pi_{TCP}(s) \Big] \tag{4}$$

where $\pi_{TCP}(s)$ are the steady-state probabilities of the TCP chain and t_{ON} and t_{OFF} are the average time durations in states s_{ON}[4] and s_{OFF}, respectively,

[4] $s_{ON} = S_{SS} \bigcup S_{CA} \bigcup S_{FR} \bigcup S_{TO}$

given by equation 5.

$$t_{ON} = \frac{1}{\alpha} \qquad t_{OFF} = \frac{1}{\beta} \tag{5}$$

The receiving TCP throughput in bps is given by:

$$Throughput_{TCP} = \lambda(1 - p_{total})MSS \tag{6}$$

4 Simulations

The main aim of this section is to validate the correctness and accuracy of the TCP NewReno analytical model through ns-2 simulations.

4.1 Reference Scenario

The network topology set up is showed in figure 2. The following methodology was used:

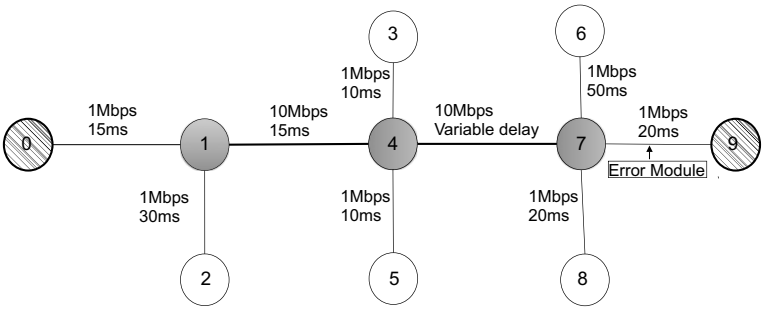


Fig. 2. Network topology set up in ns-2 validation

1. The TCP transfer under analysis is that from TCP server in host 0 to TCP host in host 9.
2. File sizes of 5, 10 and 20MB were transferred by the FTP/TCP flow under consideration in separate simulations with different traffic conditions.
3. The 1Mbps last-hop link, from node 7 to 9, is used solely by the TCP flow under analysis.
4. Traffic conditions include no-traffic and a mixture of Reno and NewReno as well as UDP flows which were started by a random generator in all hosts. Up to 20 flows, including the monitored TCP flow, were at once interacting in the network. From these, 15 were TCP and 5 were UDP flows in order to represent the greater percentage of TCP traffic in the Internet. All but the TCP receiver in host 8 were sending and receiving different flows. The file sizes being sent vary from 0.5 to 20 MB.
5. Attention was given to obtain an average rtt equal to 200ms for the TCP flow under consideration. This was achieved by varying the delay of the link

between nodes 4 and 7 to obtain the required rtt. This was corroborated by tracing every single rtt during the connection and calculating the transfer average rtt.

6. Segment losses are caused by an error module with a uniform distributed random variable which is inserted in the last-hop link. This error module simulates the MSS loss probability detected at the transport layer.
7. TCP segments of 1500B (including IP/TCP headers) are considered.
8. A *cwnd* of 42 segments is used in most of the simulations to approximately represent a *cwnd* of 64KB.
9. Every TCP transfer, with a given *cwnd*, rtt, file size and loss probability, was simulated 20 times under different traffic conditions and starting times. The average throughput was then obtained.
10. The average state durations, t_{ON} and t_{OFF}, are set equal to 1 in the analytical model.
11. Timeout duration is set to $\frac{\delta}{5}$, five times the average rtt, as generally suggested in the literature [21,4].

4.2 Simulation Results

TCP NewReno performance and its improvement over Reno is shown in figure 3 where analytical and ns-2 simulation throughputs are depicted for different segment loss probabilities p_{total}. Reno analytical characterisation is that proposed in [6]. As mentioned before, the main difference with this model is the assumption of considering the number of losses proportional to the TCP retransmission timer. The agreement between the analytical model and the simulations is more than satisfactory.

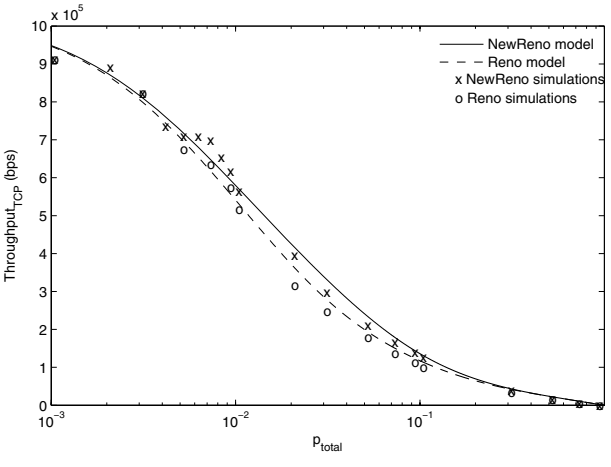


Fig. 3. TCP NewReno and Reno performance with MSS=1500B, cwnd=42MSS and rtt=200ms

In figure 3 it can be observed that both versions behave similarly for high loss probabilities, particularly when $p_{total} \geq 0.2$. Since there is a high number of lost segments per *cwnd*, neither NewReno nor Reno can recover them during FR. Both have to retransmit the whole *cwnds* through TOs which causes them to present the same performance. In contrast, for probabilities in the range of 0.002 to 0.2 the impact of the NewReno Fast Recovery algorithm can be noticed. NewReno lost segments are mostly recovered during the FR mode unlike Reno which presents a higher number of retransmissions through TOs. Thus, NewReno keeps transmitting new data during FR, causing a higher throughput. Any TCP throughput improvement impacts positively on the Internet performance given the high number of TCP transfers taking place on it.

For low segment loss probabilities, $p_{total} < 0.002$, NewReno throughput improvement is negligible given that the low number of lost segments makes it behave like a traditional Reno, i.e. NewReno and Reno FR's phases present approximately equal durations.

Figure 4 shows the throughput for the same flow with different segment sizes in order to emphasise its importance on TCP performance. It is important to remember that the *cwnd* size is being given in segments. In such a manner that segment sizes of 1000 and 576B would correspond to maximum *cwnds* of approximately 42KB and 24KB, respectively. Thus, the impact of the maximum *cwnd* on the throughput is actually shown: smaller *cwnds* lead to lower receiving bit rates which vary according to the maximum *cwnd* size.

To corroborate the accuracy of the model with different parameters, the cwnd sizes were varied in other set of simulations. As expected, the TCP throughput decreases according to the cwnd size, i.e. the lower the cwnd size, the lower the throughput, and the analytical results show good agreement with those of the ns-2 simulations. This last set of simulations is shown in figure 5.

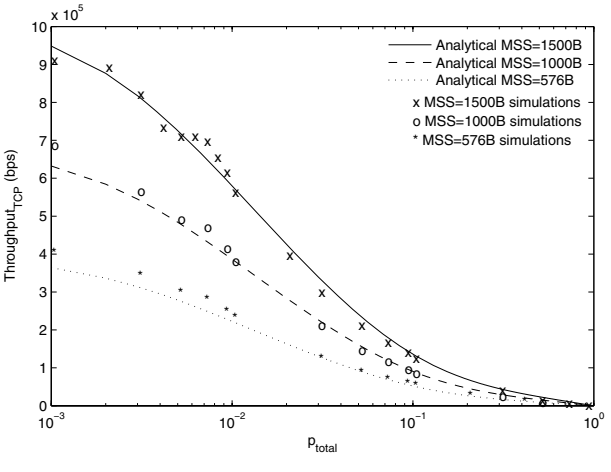


Fig. 4. TCP NewReno with different MSS with cwnd=42MSS and rtt=200ms

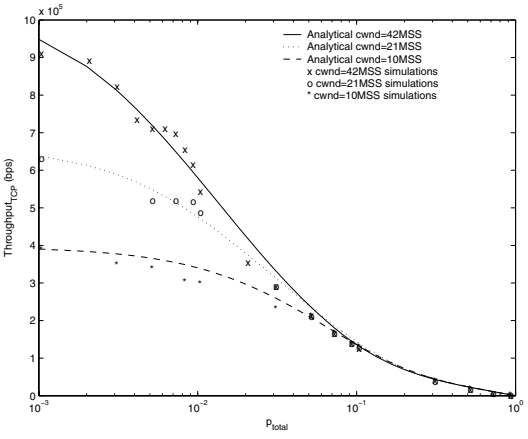


Fig. 5. TCP NewReno with different congestion window sizes, MSS=1500B and rtt=200ms

As observed in table 1, just two more analytical models characterise NewReno, [11] and [18]. The results produced by the analytical model presented in this work show close agreement with those of [11]. However, it is not possible to compare them with those of [18] given that their mathematical characterisation was developed specifically for wireless channels and uses parameters which are not possible to map to those used here.

5 Conclusions

In this paper, a CTMC was used to develop the NewReno Fast Recovery algorithm over an existing Reno analytical model. The main assumption to characterise it was to consider the number of losses proportional to its retransmission timer. As shown, the agreement between NewReno analytical predictions and the simulations is more than satisfactory and its throughput exceeds that of Reno under certain range of high loss rates. Therefore, this model can be safely used to represent TCP NewReno performance over different network scenarios where losses at the transport layer appear as independent losses such as wireless channels undergoing fast-fading or using error control techniques in the LL (e.g. ARQ mechanisms, FEC codes) or in wired links with router buffers implementing Active Queue Management schemes such as Random Early Detection.

References

1. Abouzeid, A., Sumit, R., Murat, A.: Comprehensive Performance Analysis of a TCP Session over a Wireless Fading Link with Queuing. IEEE Trans. on Wireless Communications 2, 344–356 (2003)
2. Allman, M., Paxson, V., Stevens, W.: TCP Congestion Control. RFC 2581 (1999)

3. Barakat, C., Al Fawal, A.: Analysis of link-level hybrid FEC/ARQ-SR for wireless links and long-lived TCP traffic. Performance Evaluation 57, 453–476 (2004)
4. Casetti, C., Meo, M.: Modeling the Stationary Behavior of TCP Reno Connections. In: Ajmone Marsan, M., Bianco, A. (eds.) QoS-IP 2001. LNCS, vol. 1989, pp. 141–156. Springer, Heidelberg (2001)
5. Chaskar, H., Lakshman, T.V., Madhow, U.: TCP Over Wireless with Link Level Error Control: Analysis and Design Methodology. ACM/IEEE Trans. on Networking 7, 605–615 (1999)
6. Chiasserini, C., Meo, M.: A Reconfigurable Protocol Setting to Improve TCP over Wireless. IEEE Trans. on Vehicular Technology 51, 1608–1620 (2002)
7. Floyd, S., Henderson, T., Gurtov, A.: The NewReno Modification to TCP's Fast Recovery Algorithm. RFC 3782 (2004)
8. Grassmann, W.K., Taksar, M.I., Heyman, D.P.: Regenerative Analysis and Steady State Distributions for Markov Chains. Operations Research 33, 1107–1116 (1985)
9. Handley, M., Floyd, S., Padhye, J., Widmer, J.: TCP Friendly Rate Control TFRC: Protocol Specification. RFC 5348 (2008)
10. Jacobson, V.: Congestion Avoidance and Control. In: Proc. ACM SIGCOMM Communications Architectures and Protocols Symposium, pp. 314–329 (1988)
11. Kumar, A.: Comparative Performance Analysis of versions of TCP in a Local Network with a Lossy Link. ACM/IEEE Trans. on Networking 6, 485–498 (1998)
12. Lakshman, T.V., Madhow, U.: The performance of TCP/IP for networks with high bandwidth-delay products and random loss. ACM/IEEE Trans. on Networking 5, 336–350 (1997)
13. Mathis, M., Mahdavi, J., Floyd, S., Romanow, A.: TCP Selective Acknowledgement Options. RFC 2018 (1996)
14. Medina, A., Allman, M., Floyd, S.: Measuring the Evolution of Transport Protocols in the Internet. ACM SIGCOMM Computer Communications Review 35, 37–51 (2005)
15. The Network Simulator, ns-2, http://www.isi.edu/nsnam/ns/
16. Padhye, J., Firoiu, V., Towsley, D.F., Kurose, J.F.: Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation. ACM/IEEE Trans. on Networking 8, 133–145 (2000)
17. Postel, J.: The Transmission Control Protocol. RFC 793 (1981)
18. Rossi, M., Vicenzi, R., Zorzi, M.: Accurate Analysis of TCP on Channels with Memory and Finite Round-Trip Delay. IEEE Trans. on Wireless Communications 3, 627–640 (2004)
19. Sarkar, J., Sengupta, S., Chatteerjee, M., Ganguly, S.: Differential FEC and ARQ for Radio Link Protocols. IEEE Trans. on Wireless Communications 55, 1458–1472 (2006)
20. Sikdar, B., Kalyanaramanand, S., Vastola, K.: Analytical Models for the Latency and Steady-State Throughput of TCP Tahoe, Reno and SACK. ACM/IEEE Trans. on Networking 11, 959–971 (2003)
21. Vacirca, F., De Vendictis, A., Baiocchi, A.: Optimal Design of Hybrid FEC/ARQ schemes for TCP over Wireless Links with Rayleigh Fading. IEEE Trans. on Mob. Computing 5, 289–302 (2006)
22. Wierman, A., Osogami, T., Olsen, J.: A Unified Framework for Modeling TCP-Vegas, TCP-SACK, and TCP-Reno. In: Proc. 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunication Systems MASCOTS 2003, pp. 269–278 (2003)

Packet Loss Analysis of Load-Balancing Switch with ON/OFF Input Processes

Yury Audzevich[1], Levente Bodrog[2], Yoram Ofek[1], and Miklós Telek[2]

[1] Department of Information Engineering and Computer Science,
University of Trento, Italy
{audzevi,ofek}@disi.unitn.it
[2] Department of Telecommunications,
Technical University of Budapest, Hungary
{bodrog,telek}@hit.bme.hu

Abstract. Lately, the number of Internet users and, correspondingly, the amount of traversing traffic is growing extremely fast. In spite of the fact that transmission links – mostly optical fibres – have high capacity, the internet routers still remain a point of traffic bottleneck. The construction of highly scalable switches for high-speed transmission still remains a real challenge for designers. In this paper we focus our efforts on the analysis of Load-Balancing Birkhof-von Neumann switch which is lately considered to be a highly efficient distributed switch with simple control and high scalability. Due to the fact that Internet traffic represents an asynchronous traffic which supports a variety of applications, we have introduced the analysis of possible loss inside the load-balanced switch under consideration of *variable size packets* and *finite central stage buffers* previously in [1]. Although the analysis has showed some interesting features of the switch, it has exponential complexity of $O\left(N^N\right)$ which makes that model inapplicable for the switches with large number of ports, N. The main goal of this paper is to approximate the switch analysis with lower complexity, i.e., $O\left(2^N\right)$ which can be useful for evaluation of packet loss in the larger load-balanced switches.

1 Introduction

The traditional ways of packet switching are designed to connect multiple area networks (LANs, WANs, etc.) and forward asynchronous traffic between the communication links. Usually packet switches are implementing centralized control, in order to find the best possible link to forward data traffic from the source to the specific destination. Although in most of the cases these architectures are capable to provide high throughput, they have poor scalability for switches of large size. In this context, the switches with distributed control are more attractive with the advantage of their scalability due to the fact that each stage is making its own calculations for packet forwarding.

In this paper we examine the Load-Balanced switch (LB switch) [2,3], which is considered to be a particular case of two-stage switch. The first stage of the

J.T. Bradley (Ed.): EPEW 2009, LNCS 5652, pp. 197–211, 2009.

switch is balancing the arriving traffic to the intermediate inputs of the second switch, which is in fact an input buffer switch with deterministic control (see Figure 1). Since all the interconnections inside are deterministic and periodic, the switch has a simple distributed control and can be highly scalable. Among the first significant results shown in [2] and [3] was the fact that under certain assumptions the switch can achieve high throughput (up to 100%) and low packet traversing delay. However these results were obtained under consideration that all the packets have equal length, traffic is admissible and central stage buffers are infinite. Even under these strong assumptions some important issues of packets mis-sequencing were investigated in detail in [4,5,6,7,8]. It is important to mention that some of the architectures to resolve packets mis-sequencing require extra control, introducing different overheads (communication and computational), that basically increases the control complexity of the LB switch. However, keeping correct sequence of packets through the system avoids unnecessary retransmissions of packets in the network protocol layer.

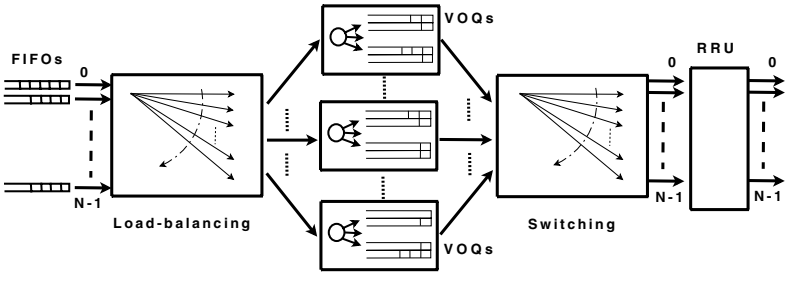


Fig. 1. The load-balanced switch considered for the analysis

Taking into account the fact that some of the assumptions mentioned in [2,3,8] are not practical, in [9] and [1] we examined the behaviour of the LB switch with finite size central stage buffers. Under these considerations, the LB switch can experience a packet loss due to congestion. The first simulation results on this issue were presented in [10] and detailed mathematical analysis in [9]. However, the analysis in [9] was done only for fixed size packets (cells), and there were not taken into account variable size packets (multiple number of cells going to the same destination). It is considered that most of the internet switches are operating on the cell-based level (to increase buffer utilization), that means that arriving variable size packets are segmented at the inputs and reassembled at the outputs. The issue of possible cell and correspondingly a packet loss inside the switch, can introduce some significant posterior problems to the LB switch reassembly part [11]. That is why in [1] we presented the analysis of a packet loss experienced by the switch operating with variable size packets and finite central stage buffers.

We assumed Markovian behaviour to be able to use numerically efficient algorithms to solve the model. This means geometrically distributed packet lengths

and interarrival times, which allows us to capture the mean of these distributions. Real internet traffic shows different packet size distributions [12] and one can fit more parameters using other, more complex Markovian structures like discrete Phase Type (DPH) distributions or discrete Markovian arrival processes (DMAPs). The number of fitted parameters can be increased at an arbitrary level, but it would greatly increase the complexity of the model as well and that would also hide the main contribution of our approach.

In spite of the same assumption in [1] the complexity of that model resulted unresolvable high Markov chains even in case of very small switches ($N \geq 4$). The main goal of this paper is to introduce the approximate model of the initial analysis – with complexity $O\left(2^N\right)$ – in order to make the evaluation of packet loss probabilities feasible for larger number of ports – at least in terms of the exact analysis provided in [1].

As the present model is still exponentially complex with regard to the number of ports – $O\left(2^N\right)$ – we have kept on with the research and introduced the model of complexity $O\left(N\right)$ in [13]. However, in the least complex model [13], we assumed stochastically identical input processes. As a consequence the reader should take into mind that the present model is less complex than that of [1] and more complex than that of [13], but it takes into consideration inhomogeneous input processes. These features of the three models are summarized in Table 1.

Table 1. The authors' recent work on the LB switch topic

citation	[1]	this paper	[13]
submission order	1st	2nd	3rd
complexity	$O\left(N^N\right)$	$O\left(2^N\right)$	$O\left(N\right)$
homogeneous inputs	✗	✗	✓

The rest of the paper is organized as follows. We summarize the LB switch's operation principles and main assumptions in Section 2. Next, in Section 3 we introduce the "ON/OFF" model of the system. In Section 4 we verify the result by comparing it with initial analytical model as well as with simulation results. Finally, Section 5 concludes the paper.

2 The Main Assumptions and Operating Principles

Let denote $N \times N$ the LB switch with both N input and output ports. The single-stage buffering LB switch is equipped with First-In-First-Out (FIFO) buffers in the inputs, N sets of N Virtual Output Queues (VOQs) in the central stage and re-sequencing and reassembly units (RRU) in the output (see the illustration in Figure 1). In the kth set of VOQs there is one VOQ (VOQ_{kj}) dedicated to store cells directed to output j. Hereinafter the term VOQ_k with a single index denotes the kth set of VOQs and the term VOQ_{kj} with the pair of indices denotes the specific VOQ stores cells directed to output $j \quad j, k \in [0, N-1]$. As

it is out of the scope of this paper and it does not affect the modelled parts of the switch, the implementation of the RRU is not discussed in this paper, but it can be taken from the ones proposed in research, e.g., in [11]. In this analysis there is no feedback link between the switch stages and each stage is operating independently. After segmentation of an incoming packet, the cells are load-balanced between the central stage VOQs according to the final destination [2]. The interconnections between stages are made by means of crossbar switches without buffers inside (contrary to [14]). The crossbar switches are indicated as "Load-balancing" and "Switching" in Figure 1. In the tth time slot – the transmission time of one cell – the interconnection pattern is the periodic round-robin sequence according to the rules

$$\begin{aligned} k &= (i+t) \mod N \\ j &= (k+t) \mod N, \end{aligned} \tag{1}$$

where i denotes the ordinal number of the input port, j the output port and k the set of VOQs, $i, j, k \in [0, N-1]$ which implies the periodic behaviour of the system. This N cell transmission time long period – hereinafter referred to as time period – will be the time unit of the discrete time Markov chain (DTMC) modelling the VOQ. As all the stages are synchronized, the transmission of cells is possible from all inputs simultaneously during a time slot [8].

If a single cell is lost in the central stage, there is no possibility to drop all the remaining cells of this "broken" packet from VOQs without sophisticated centralized controller (which is not the case in this paper). Such packets will waste the capacity of the central stage buffers, will increase the possibility of further packet loss and definitely will make impossible packets reassembling operation [11].

In a time slot, first, the VOQs are connected to the outputs and then the inputs to the VOQs. This order of interconnections inhibits a cell from traverse the switch in a single time slot. The transmission rate inside the switch is fixed and it is the service time of a cell. The mean service rate of the switch assumed to be greater than the mean arrival rate of the variable size packets – the switch is not overloaded.

The arrival pattern consists of packets with random distributed number of cells idle periods in between in time slots. The details of these distributions are

packet length geometric distributed with probability mass function (PMF) $\Pr(X = i) = p(1-p)^{i-1} \quad \forall i = 1, 2, \ldots$ and

idle period length geometric distributed with PMF $\Pr(Y = i) = q(1-q)^{i} \quad \forall i = 0, 1, \ldots$

The geometric distribution of the packets arrive from input i to output j have the parameter p_{ij} and the idle periods between packet arrivals at input i have the parameter q_i.

The destinations of the packets can be set via matrix $\mathbf{T}$ whose ijth element (t_{ij}) gives the probability that if a packet arrives to input i is directed to output j. The row-sum of $\mathbf{T}$ thus equals to $\mathbf{h}$ an appropriate size column vector of ones.

Moreover, as shown in our analytical results, the packet loss probability of the specific VOQ strongly depends on the specific traversing path of the traffic inside the switch (i.e., input, VOQ and output), which is an interesting phenomenon described in Section 2.1 for the interconnection pattern applied.

2.1 Properties of the Different Paths

An important finding of our analysis of the LB switch is that there are differences between the loss probabilities of paths traversing the switch. Here path means the triple, $\{i, j, k\}$, containing the ordinal number of the input, the output and the VOQ respectively.

Using the interconnection pattern policy given in (1) the time difference between the service of the VOQ and the arrival to it can be expressed as

$$d = (2k - i - j) \mod N. \tag{2}$$

d also expresses the number of inputs that have the right to send a packet to VOQ_{kj} before input i in the same time period.

A particular VOQ is served once a time period. It is also true that in a time period all the inputs have the right – in a particular order determined by (2) – to send cells to the VOQ. In case of "almost full" buffer the higher the d value is the higher the probability that there are enough inputs that can fill up the buffer, i.e., make the cell of the observed input to be lost. According to this observation we introduce the notation type-d for paths with value d.

For example, using the above introduced notation, we can say that the type-0 paths cannot have cell loss. Its short explanation is that even if the buffer is full the cell in the head of the queue is served and thus there is always a free position in the tail accordingly which is used by the type-0 path to push its cell into it. This makes impossible the cell loss at a type-0 path.

3 The ON/OFF Model of the 3×3 Switch

In this section we give the approximate model of a VOQ of the 3×3 LB switch. Compared to the exact analysis in [1] the approximation is that we model the input process, i.e., the arrival process of the VOQ, with a two state – ON/OFF – model. By this the state space of the model of the same VOQ can be reduced compared to the exact model of [1] where a size (N) dependent full characterization of the input process is given. Once we have the model of an input the complete model of the chosen VOQ is given in the same way as in case of the full characterization in [1]. Indeed the ON/OFF based model of the LB switch differs from the complete characterization in the DTMCs describing the input processes.

As we described in Section 2.1 it is relevant which type of path is considered. Here we describe a type-2 path lead through the 3×3 switch – as also done in [1] in case of the full characterization. For example it is path $\{1, 0, 0\}$ but we will also investigate all types of path later in Section 4.

3.1 Model of an Input

In this section we will introduce the approximate – ON/OFF – input model of path $\{1, 0, 0\}$ of the 3×3 switch.

The ON/OFF model of the first input is derived from its complete characterization depicted in Figure 2 using the notations introduced for the input processes in Section 2. According to the geometric assumptions for the packet length and idle period length this is a DTMC having four states, 1 *id* corresponds to the idle period, and the other three states corresponds to packet arrival from input 1 to either output 0 (state 10) or output 1 (state 11) or output 2 (state 12). The exact state transition probability matrix describing the behaviour of input 1 is

$$\mathbf{P}_1^{\mathcal{C}} = \begin{pmatrix} (1-p_{10}) + p_{10}q_1t_{10} & p_{10}q_1t_{11} & p_{10}q_1t_{12} & p_{10}\,(1-q_1) \\ p_{11}q_1t_{10} & (1-p_{11}) + p_{11}q_1t_{11} & p_{11}q_1t_{12} & p_{11}\,(1-q_1) \\ p_{12}q_1t_{10} & p_{12}q_1t_{11} & (1-p_{12}) + p_{12}q_1t_{12} & p_{12}\,(1-q_1) \\ q_1t_{10} & q_1t_{11} & q_1t_{12} & 1-q_1 \end{pmatrix}. \tag{3}$$

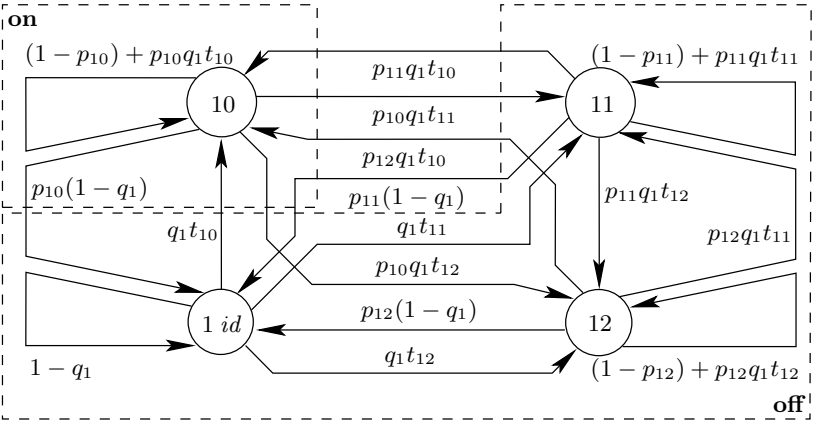


Fig. 2. The graph of the DTMC fully characterizing the first input of the 3×3 switch

In terms of path $\{1, 0, 0\}$ the states of the DTMC modelling input 1 can be divided into two subsets

on this is a one-element subset containing state 10 in which there are cell arrivals from input 1 to output 0 and

off the other states in which there is no arrival from input 1 to output 0

which is also indicated in Figure 2. Using this division we create the two state ON/OFF model of the input processes. Hereinafter lowercase bold **on** and **off** denotes these two subsets and uppercase ON and OFF the two states of the newly derived DTMC model of the inputs.

In the following sections the detailed description of the ON and OFF states are given based on the aforementioned division.

OFF properties. The OFF state is used to approximate the set of **off** states. Its properties are determined based on the absorbing time of a discrete phase type (DPH) distribution given in Figure 3 with transient states identical to the **off** states and absorbing state given as the **on** state. Its initial distribution then given as the renormalization of the zeroth row of $\mathbf{P}_1^{\mathcal{C}}$ in (3) without its zeroth element

$$\boldsymbol{\beta}_1 = \left(\frac{q_1 t_{11}}{q_1 t_{11} + q_1 t_{12} + (1-q_1)} \; \frac{q_1 t_{12}}{q_1 t_{11} + q_1 t_{12} + (1-q_1)} \; \frac{1-q_1}{q_1 t_{11} + q_1 t_{12} + (1-q_1)} \right). \tag{4}$$

$\mathbf{B}_1$, the transition probability matrix of the transient states, is the $N \times N$ matrix given as $\mathbf{P}_1^{\mathcal{C}}$ without its zeroth row and zeroth column

$$\mathbf{B}_1 = \begin{pmatrix} (1-p_{11}) + p_{11} q_1 t_{11} & p_{11} q_1 t_{12} & p_{11}\,(1-q_1) \\ p_{12} q_1 t_{11} & (1-p_{12}) + p_{12} q_1 t_{12} & p_{12}\,(1-q_1) \\ q_1 t_{11} & q_1 t_{12} & 1-q_1 \end{pmatrix}. \tag{5}$$

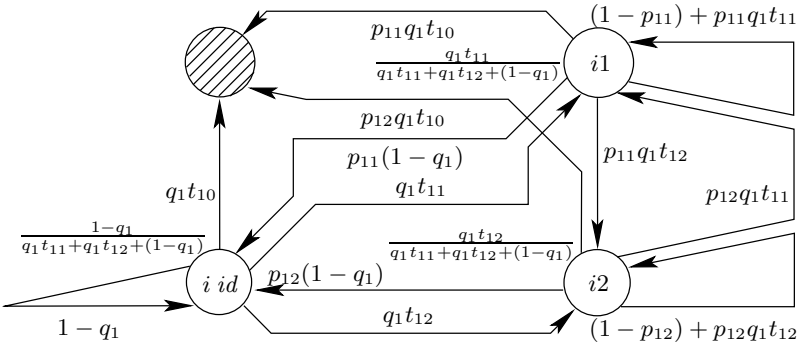


Fig. 3. The graph of the DPH substitution of the **off** states in terms of the pair input 1 - output 0

The mean absorbing time of this DPH is

$$\mu_1 = \boldsymbol{\beta}_1 \left(\mathbf{I} - \mathbf{B}_1 \right)^{-1} \mathbf{h}, \tag{6}$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{h}$ is the column vector of ones of appropriate size.

We set the sojourn probability of the state OFF to $1 - \frac{1}{\mu_1}$ which sets the mean sojourn time to μ_1. Then the state transition probability from OFF to ON is $\frac{1}{\mu_1}$.

ON properties In case of ON the sojourn probability remain the same as in the complete characterization, i.e. in case of output 0 the upper left element

of $\mathbf{P}_1^{\mathcal{C}}$ in (3). The state transition probability from ON to OFF is the summation of the remaining elements of the zeroth row of $\mathbf{P}_1^{\mathcal{C}}$ which is 1 minus the sojourn probability.

Summation of the ON/OFF DTMC. Here we summarize all the properties of the ON/OFF DTMC by giving its graph for the general path $\{i, j, k\}$ in Figure 4 together with its state transition probability matrix

$$\mathbf{P}_i = \begin{pmatrix} \left(\mathbf{P}_i^{\mathcal{C}}\right)_{jj} & 1 - \left(\mathbf{P}_i^{\mathcal{C}}\right)_{jj} \\ \frac{1}{\mu_i} & 1 - \frac{1}{\mu_i} \end{pmatrix} = \begin{pmatrix} (1 - p_{ij}) + p_{ij}q_it_{ij} & p_{ij} - p_{ij}q_it_{ij} \\ \frac{1}{\mu_i} & 1 - \frac{1}{\mu_i} \end{pmatrix}, \tag{7}$$

where $(*)_{ij}$ denotes the ijth element of a matrix.

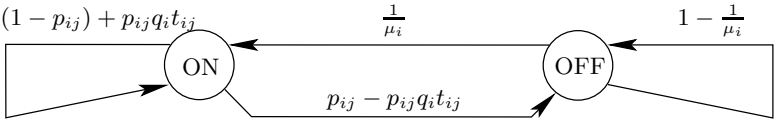


Fig. 4. The graph of the ON/OFF DTMC describing the pair input i - output j

3.2 The Cell Level Model

Up to now we have introduced the differences between the full model of [1] and the ON/OFF model of the input processes. From now on we recall the remaining part of building the model of the VOQ using the ON/OFF model of each input. Here we keep on with building the model of the VOQ of path $\{1, 0, 0\}$.

First of all we give the cell level model of VOQ_{00} which is a quasi birth-deathlike (QBD-like) DTMC where the level represents the queue length and the phase is the combined state $\left(0, 1, \ldots, 2^N - 1\right)$ of the inputs.

According to the periodic operation of the switch mentioned in Section 2 the time unit of the QBD-like model is N time slots – the time period of the operation of the switch.

Since the DTMC given in Figure 4 and in (7) gives the behaviour of the input process in a single time slot we raise all of them to the Nth $= 3$rd power to have the model of the input processes in a time period.

Then the joint behaviour of the input processes – for all inputs $(i = 0, 1, 2)$ – gives the phase process of the QBD-like model which is the Kronecker product of their 3rd power as

$$\boldsymbol{\mathcal{P}} = \mathbf{P}_0^3 \otimes \mathbf{P}_1^3 \otimes \mathbf{P}_2^3. \tag{8}$$

The number of arrivals to the observed VOQ is determined as the sum of the arrivals from each input, but we cannot forget that each input can transmit a cell into the VOQ in its dedicated time slot. This is determined by the interconnection pattern given in (1), i.e. input 0 sends cell to VOQ_{00} in the 1st time slot of a time period, input 1 sends in the 3rd time slot of a period and input 2 sends in the 2nd time slot of a time period. Here we note that the ordinal number of

the dedicated time slot equals to $d+1$ for each input i in any path. According to this we replace the 1st, the 3rd and the 2nd factor of the powers of $\mathbf{P}_0^3$, $\mathbf{P}_1^3$ and $\mathbf{P}_2^3$ respectively in (8) to

$$\mathbf{P}_i = \mathbf{A}_i + \mathbf{K}_i \quad \forall i \in [0, N-1]\,, \tag{9}$$

in which the first term corresponds to arrival from input i and the second term corresponds to the case when there is no arrival from input i. The substitution is then

$$\boldsymbol{\mathcal{P}} = \mathbf{P}_0^3 \otimes \mathbf{P}_1^3 \otimes \mathbf{P}_2^3 = (\mathbf{A}_0 + \mathbf{K}_0)\,\mathbf{P}_0^2 \otimes \mathbf{P}_1^2\,(\mathbf{A}_1 + \mathbf{K}_1) \otimes \mathbf{P}_2\,(\mathbf{A}_2 + \mathbf{K}_2)\,\mathbf{P}_2 \tag{10}$$

based on the d values of the inputs calculated as given in (2). Expanding this expression and collecting the terms according to $0, 1, 2$ and 3 arrivals we get

$$\begin{aligned}
\boldsymbol{\mathcal{P}} = {} & \underbrace{\mathbf{K}_0\mathbf{P}_0^2 \otimes \mathbf{P}_1^2\mathbf{K}_1 \otimes \mathbf{P}_2\mathbf{K}_2\mathbf{P}_2}_{\text{no arrivals} - \mathbf{B}} + \underbrace{\mathbf{A}_0\mathbf{P}_0^2 \otimes \mathbf{P}_1^2\mathbf{K}_1 \otimes \mathbf{P}_2\mathbf{K}_2\mathbf{P}_2}_{\text{1 arrival} - \mathbf{L}} + \\
& + \underbrace{\mathbf{K}_0\mathbf{P}_0^2 \otimes \mathbf{P}_1^2\mathbf{A}_1 \otimes \mathbf{P}_2\mathbf{K}_2\mathbf{P}_2 + \mathbf{K}_0\mathbf{P}_0^2 \otimes \mathbf{P}_1^2\mathbf{K}_1 \otimes \mathbf{P}_2\mathbf{A}_2\mathbf{P}_2}_{\text{1 arrival} - \mathbf{L}} + \\
& + \underbrace{\mathbf{K}_0\mathbf{P}_0^2 \otimes \mathbf{P}_1^2\mathbf{A}_1 \otimes \mathbf{P}_2\mathbf{A}_2\mathbf{P}_2 + \mathbf{A}_0\mathbf{P}_0^2 \otimes \mathbf{P}_1^2\mathbf{K}_1 \otimes \mathbf{P}_2\mathbf{A}_2\mathbf{P}_2}_{\text{2 arrivals} - \mathbf{F}_1} + \\
& + \underbrace{\mathbf{A}_0\mathbf{P}_0^2 \otimes \mathbf{P}_1^2\mathbf{A}_1 \otimes \mathbf{P}_2\mathbf{K}_2\mathbf{P}_2}_{\text{2 arrivals} - \mathbf{F}_1} + \underbrace{\mathbf{A}_0\mathbf{P}_0^2 \otimes \mathbf{P}_1^2\mathbf{A}_1 \otimes \mathbf{P}_2\mathbf{A}_2\mathbf{P}_2}_{\text{3 arrivals} - \mathbf{F}_2} = \\
& = \mathbf{B} + \mathbf{L} + \mathbf{F}_1 + \mathbf{F}_2,
\end{aligned} \tag{11}$$

where we have also indicated the level transition based decomposition, $\boldsymbol{\mathcal{P}} = \mathbf{B} + \mathbf{L} + \mathbf{F}_1 + \mathbf{F}_2$, of such a QBD-like model.

Using these level transition matrices the state transition probability matrix has the QBD-like structure

$$\mathbf{P} = \begin{pmatrix}
\mathbf{B} & \mathbf{L} & \mathbf{F}_1 & \mathbf{F}_2 & 0 & \dots \\
\mathbf{B} & \mathbf{L} & \mathbf{F}_1 & \mathbf{F}_2 & 0 & \dots \\
\dots & \dots & \dots & \dots & \dots & \dots \\
\dots & 0 & \mathbf{B} & \mathbf{L} & \mathbf{F}_1 & \mathbf{F}_2 \\
\dots & 0 & 0 & \mathbf{B} & \mathbf{L} & \mathbf{F}_1' \\
\dots & 0 & 0 & 0 & \mathbf{B} & \mathbf{L}'
\end{pmatrix}, \tag{12}$$

where $\mathbf{F}_1' = \mathbf{F}_1 + \mathbf{F}_2$ and $\mathbf{L}' = \mathbf{L} + \mathbf{F}_1 + \mathbf{F}_2$.

The building of this kind of QBD-like DTMC for $N = 3$ is given in Algorithm 1.

The steady state solution of this QBD-like model is the solution of the linear equation system

$$\boldsymbol{\pi}\mathbf{P} = \boldsymbol{\pi}, \qquad \boldsymbol{\pi}\mathbf{h} = 1. \tag{13}$$

Algorithm 1. Building the QBD-like model of a VOQ

INPUT: $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2$ from (7)
OUTPUT: $\mathbf{P}$ the QBD-like model similar to (12)
1: **for** $i = 0$ to 2 **do**
2: compute $\mathbf{A}_i, \mathbf{K}_i$ as given in (9)
3: calculate d for the ith input as given in (2)
4: replace the $(d+1)$st factor of $\mathbf{P}_i^3$ in (8) with $\mathbf{A}_i + \mathbf{K}_i$ as given in (10)
5: **end for**
6: expand the resulting expression for $\boldsymbol{\mathcal{P}}$ and
7: identify the level transition matrices $\mathbf{B}, \mathbf{L}, \mathbf{F}_1, \mathbf{F}_2$ as given in (11)
8: build $\mathbf{P}$ as in (12)
9: **return** $\mathbf{P}$

3.3 The Packet Level Model

With the geometric assumption for the packet length, given in Section 2, the life cycle of a packet in the observed path can be modelled by a transient DTMC in which the two absorbing states corresponds to the two possible ending of a packet transmission – the successful transmission (ST) of the packet or its lost (PL), as given in Figure 5. In this section we present this transient DTMC with its state transition probability matrix and initial distribution.

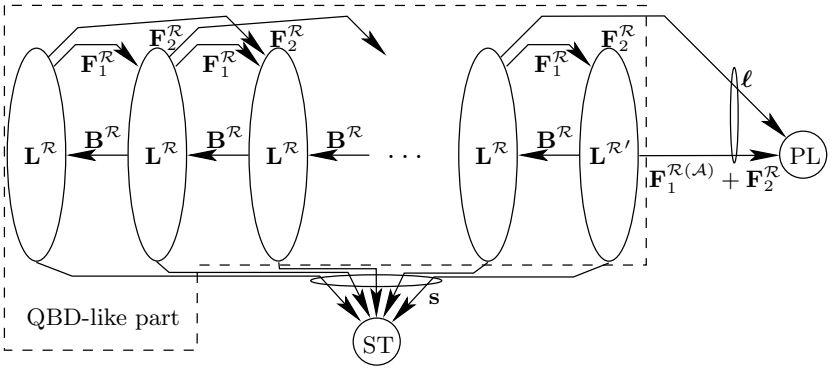


Fig. 5. The transient DTMC modelling the VOQ during the life cycle of a packet

The state transition probability matrix of the transient part. The transient DTMC is mainly built in the same way as the QBD-like model of the VOQ on the cell level in Section 3.2. The exceptions are

- the state transitions responsible for packet completion in the observed path are removed (its DTMC is given in Figure 6(a)) and
- the cell losses in case of "nearly" full buffer are considered.

The removal of the state transitions is explained by the introduction of absorbing state ST. Indeed this transient DTMC move to state ST when the transmission of a packet is completed. Then according to these modifications the state transition probability matrix of the modified DTMC of input 1, with such state transitions removed (see Figure 6(a)), is

$$\mathbf{P}_1^{\mathcal{R}} = \begin{pmatrix} 1-p_{10} & 0 \\ 1-\frac{1}{\mu_i} & \frac{1}{\mu_i} \end{pmatrix}, \tag{14}$$

where superscript $\mathcal{R}$ refers to the DTMC with absorbing states PL and ST, in Figure 5. The DTMC of the other two inputs remain as in (7).

The state transition probability matrix of the QBD-like part of the DTMC in Figure 5 is $\mathbf{P}^{\mathcal{R}}$. It is determined by Algorithm 1 with input parameters $\mathbf{P}_0, \mathbf{P}_1^{\mathcal{R}}, \mathbf{P}_2$ with one exception in line 8.

Having the level transition matrices $\left(\mathbf{B}^{\mathcal{R}}, \mathbf{L}^{\mathcal{R}}, \mathbf{F}_1^{\mathcal{R}}, \mathbf{F}_2^{\mathcal{R}}\right)$ the construction of the QBD-like structure and the state transition vector to state PL are

$$\mathbf{P}^{\mathcal{R}} = \begin{pmatrix} \mathbf{B}^{\mathcal{R}} & \mathbf{L}^{\mathcal{R}} & \mathbf{F}_1^{\mathcal{R}} & \mathbf{F}_2^{\mathcal{R}} & 0 & \dots \\ \mathbf{B}^{\mathcal{R}} & \mathbf{L}^{\mathcal{R}} & \mathbf{F}_1^{\mathcal{R}} & \mathbf{F}_2^{\mathcal{R}} & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & 0 & \mathbf{B}^{\mathcal{R}} & \mathbf{L}^{\mathcal{R}} & \mathbf{F}_1^{\mathcal{R}} & \mathbf{F}_2^{\mathcal{R}} \\ \dots & 0 & 0 & \mathbf{B}^{\mathcal{R}} & \mathbf{L}^{\mathcal{R}} & \mathbf{F}_1^{\mathcal{R}} \\ \dots & 0 & 0 & 0 & \mathbf{B}^{\mathcal{R}} & \mathbf{L}^{\mathcal{R}\prime} \end{pmatrix}, \qquad \boldsymbol{\ell} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \mathbf{F}_2^{\mathcal{R}}\mathbf{h} \\ \left(\mathbf{F}_1^{\mathcal{R}(\mathcal{A})} + \mathbf{F}_2^{\mathcal{R}}\right)\mathbf{h} \end{pmatrix}, \tag{15}$$

where $\mathbf{L}^{\mathcal{R}\prime} = \mathbf{L}^{\mathcal{R}} + \mathbf{F}_1^{\mathcal{R}(\mathcal{K})}$. Here the forward level transition matrix $\left(\mathbf{F}_1^{\mathcal{R}}\right)$ is decomposed into two parts both of them corresponds to two cell arrivals. In the first case one of the cells arrives from input 1

$$\mathbf{F}_1^{\mathcal{R}(\mathcal{A})} = \mathbf{K}_0\mathbf{P}_0^2 \otimes \mathbf{P}_1^{\mathcal{R}\,2}\mathbf{A}_1^{\mathcal{R}} \otimes \mathbf{P}_2\mathbf{A}_2\mathbf{P}_2 + \mathbf{A}_0\mathbf{P}_0^2 \otimes \mathbf{P}_1^{\mathcal{R}\,2}\mathbf{A}_1^{\mathcal{R}} \otimes \mathbf{P}_2\mathbf{K}_2\mathbf{P}_2$$

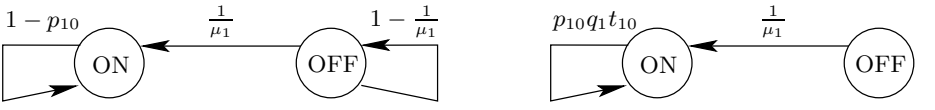
and in the second case none of them arrive from input 1

$$\mathbf{F}_1^{\mathcal{R}(\mathcal{K})} = \mathbf{A}_0\mathbf{P}_0^2 \otimes \mathbf{P}_1^{\mathcal{R}\,2}\mathbf{K}_1^{\mathcal{R}} \otimes \mathbf{P}_2\mathbf{A}_2\mathbf{P}_2.$$

Due to this there is cell loss and accordingly packet loss in the observed path $\{1, 0, 0\}$ if at the beginning of the time period either

- there is one free position in VOQ_{00} and there are cell arrivals from all three inputs $\left(\mathbf{F}_2^{\mathcal{R}}\right)$ or
- the buffer is full and there are cell arrivals either
  - from all the three inputs $\left(\mathbf{F}_2^{\mathcal{R}}\right)$ or
  - there is two arrivals from which one arrives from input 1 $\left(\mathbf{F}_1^{\mathcal{R}(\mathcal{A})}\right)$.

Accordingly, if the buffer is full at the beginning of the time period and there is two arrival, but none of them from input 1 the DTMC stays in the last level $\left(\mathbf{F}_1^{\mathcal{R}(\mathcal{K})}\right)$.



(a) Packet completion (b) New packet arrival

Fig. 6. The modified graphs of the ON/OFF DTMC describing input 1

Finally according to Figure 6(a) and (14) and using the notations of Figure 5 and (15) the state transition probability vector to the absorbing state ST is

$$\mathbf{s} = \mathbf{h} - \left(\mathbf{P}^{\mathcal{R}}\mathbf{h} - \boldsymbol{\ell}\right). \tag{16}$$

The initial distribution of the transient DTMC. The initial distribution of $\mathbf{P}^{\mathcal{R}}$ in (15) is determined as the state of the system right after the arrival of an incoming customer. In this section we determine the probability distribution of the system at this time instance, right after a new packet arrival.

A new packet arrives at input 1 according to the state transitions depicted in Figure 6(b). Its state transition probability matrix is

$$\mathbf{P}_1^{\mathcal{N}} = \begin{pmatrix} p_{10}q_1t_{10} & 0 \\ \frac{1}{\mu_1} & 0 \end{pmatrix}, \tag{17}$$

where superscript $\mathcal{N}$ refers to the DTMC according to new packet arrival.

Here we build a QBD-like model also using Algorithm 1 with input parameters $\mathbf{P}_0, \mathbf{P}_1^{\mathcal{N}}, \mathbf{P}_2$ with an exception in line 4 which also affects lines 6 and 7.

Instead of replacing the third factor of ${\mathbf{P}_1^{\mathcal{N}}}^3$ (remind that $d = 2$ for $i = 1$) we give the state transition probability matrix of input 1 in a time period as

$$\mathbf{P}_1^3 - \left(\mathbf{P}_1 - \mathbf{P}_1^{\mathcal{N}}\right)^3. \tag{18}$$

It expresses the behaviour of input 1 at new packet arrivals in a three time slots long time period. According to Algorithm 1 we expand (18), replace the third factors of its terms and simplify it we get

$$\mathbf{P}_1^3 - \left(\mathbf{P}_1 - \mathbf{P}_1^{\mathcal{N}}\right)^3 = \underbrace{\mathbf{P}_1^2\mathbf{A}_1^{\mathcal{N}} + \left(\mathbf{P}_1\mathbf{P}_1^{\mathcal{N}} + \mathbf{P}_1^{\mathcal{N}}\left(\mathbf{P}_1 - \mathbf{P}_1^{\mathcal{N}}\right)\right)\left(\mathbf{A}_1 - \mathbf{A}_1^{\mathcal{N}}\right)}_{\boldsymbol{\mathcal{A}}_1^{\mathcal{N}}} +$$
$$+ \underbrace{\mathbf{P}_1^2\mathbf{K}_1^{\mathcal{N}} + \left(\mathbf{P}_1\mathbf{P}_1^{\mathcal{N}} + \mathbf{P}_1^{\mathcal{N}}\left(\mathbf{P}_1 - \mathbf{P}_1^{\mathcal{N}}\right)\right)\left(\mathbf{K}_1 - \mathbf{K}_1^{\mathcal{N}}\right)}_{\boldsymbol{\mathcal{K}}_1^{\mathcal{N}}} = \boldsymbol{\mathcal{A}}_1^{\mathcal{N}} + \boldsymbol{\mathcal{K}}_1^{\mathcal{N}}, \tag{19}$$

where we have also indicated the two terms according to cell arrival $\left(\boldsymbol{\mathcal{A}}_1^{\mathcal{N}}\right)$ into VOQ_{00} and no cell arrival $\left(\boldsymbol{\mathcal{K}}_1^{\mathcal{N}}\right)$ in the time period. These two matrices are

used to replace the whole middle operand of (8) in line 6 of Algorithm 1 as

$$\begin{aligned}\boldsymbol{\mathcal{P}}^{\mathcal{N}} &= (\mathbf{A}_0+\mathbf{K}_0)\,\mathbf{P}_0^2 \otimes \left(\boldsymbol{\mathcal{A}}_1^{\mathcal{N}}+\boldsymbol{\mathcal{K}}_1^{\mathcal{N}}\right) \otimes \mathbf{P}_2\,(\mathbf{A}_2+\mathbf{K}_2)\,\mathbf{P}_2 = \\ &= \underbrace{\mathbf{K}_0\mathbf{P}_0^2 \otimes \boldsymbol{\mathcal{K}}_1^{\mathcal{N}} \otimes \mathbf{P}_2\mathbf{K}_2\mathbf{P}_2}_{\text{no arrivals}\,-\,\mathbf{B}^{\mathcal{N}}} + \underbrace{\mathbf{A}_0\mathbf{P}_0^2 \otimes \boldsymbol{\mathcal{K}}_1^{\mathcal{N}} \otimes \mathbf{P}_2\mathbf{K}_2\mathbf{P}_2}_{1\text{ arrival}\,-\,\mathbf{L}^{\mathcal{N}}} + \\ &+ \underbrace{\mathbf{K}_0\mathbf{P}_0^2 \otimes \boldsymbol{\mathcal{A}}_1^{\mathcal{N}} \otimes \mathbf{P}_2\mathbf{K}_2\mathbf{P}_2 + \mathbf{K}_0\mathbf{P}_0^2 \otimes \boldsymbol{\mathcal{K}}_1^{\mathcal{N}} \otimes \mathbf{P}_2\mathbf{A}_2\mathbf{P}_2}_{1\text{ arrival}\,-\,\mathbf{L}^{\mathcal{N}}} + \\ &+ \underbrace{\mathbf{K}_0\mathbf{P}_0^2 \otimes \boldsymbol{\mathcal{A}}_1^{\mathcal{N}} \otimes \mathbf{P}_2\mathbf{A}_2\mathbf{P}_2 + \mathbf{A}_0\mathbf{P}_0^2 \otimes \boldsymbol{\mathcal{K}}_1^{\mathcal{N}} \otimes \mathbf{P}_2\mathbf{A}_2\mathbf{P}_2}_{2\text{ arrivals}\,-\,\mathbf{F}_1^{\mathcal{N}}} + \\ &+ \underbrace{\mathbf{A}_0\mathbf{P}_0^2 \otimes \boldsymbol{\mathcal{A}}_1^{\mathcal{N}} \otimes \mathbf{P}_2\mathbf{K}_2\mathbf{P}_2}_{2\text{ arrivals}\,-\,\mathbf{F}_1^{\mathcal{N}}} + \underbrace{\mathbf{A}_0\mathbf{P}_0^2 \otimes \boldsymbol{\mathcal{A}}_1^{\mathcal{N}} \otimes \mathbf{P}_2\mathbf{A}_2\mathbf{P}_2}_{3\text{ arrivals}\,-\,\mathbf{F}_2^{\mathcal{N}}} = \\ &= \mathbf{B}^{\mathcal{N}} + \mathbf{L}^{\mathcal{N}} + \mathbf{F}_1^{\mathcal{N}} + \mathbf{F}_2^{\mathcal{N}}.\end{aligned} \tag{20}$$

Here we also indicated the level transition matrices used in line 8 of Algorithm 1 to build the state transition probability matrix $\left(\mathbf{P}^{\mathcal{N}}\right)$ in the same way as in (12).

Using (13) and $\mathbf{P}^{\mathcal{N}}$ the initial distribution of the DTMC in Figure 5 is

$$\boldsymbol{\pi}^{\mathcal{N}} = \frac{\boldsymbol{\pi}\mathbf{P}^{\mathcal{N}}}{\boldsymbol{\pi}\mathbf{P}^{\mathcal{N}}\mathbf{h}}. \tag{21}$$

The packet loss of the system. Using (15), (16) and (21) the packet loss probability (p_ℓ) is given as the probability of absorbing in state PL and the probability of successful packet transmission (p_s) as absorbing in state ST

$$p_\ell = \boldsymbol{\pi}^{\mathcal{N}}\left(\mathbf{I}-\mathbf{P}^{\mathcal{R}}\right)^{-1}\boldsymbol{\ell} \qquad p_s = \boldsymbol{\pi}^{\mathcal{N}}\left(\mathbf{I}-\mathbf{P}^{\mathcal{R}}\right)^{-1}\mathbf{s} = 1-p_\ell. \tag{22}$$

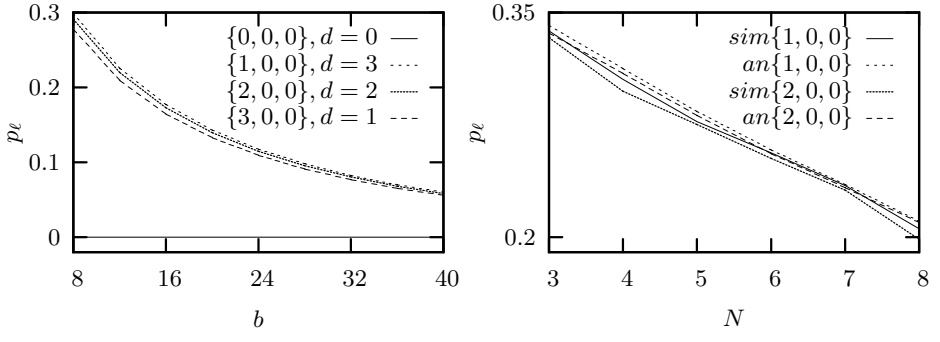
4 Computation Study

In this section we present the comparative study of the analysis with ON/OFF model and the simulation results using the memoryless (geometric) assumptions and the notations introduced in Section 2. We executed two studies with different sets of parameters given in Table 2 representing a set of considered parameters in detail, instead of just the ON and the OFF parameters (the model is derived from the detailed parameters). Although the independent variables are discrete we used continuous plots to improve visibility of Figure 7.

Study 1. Figure 7(a) plots the packet loss probability of different types of paths through VOQ_{00} versus the buffer size. The loss of a single queue is decreasing with increase of the buffer size, which is obvious with increase of system capacity. Here the dependence of packet loss on the chosen paths is also shown. The set of parameters of study 1 is given in the left hand side of Table 2. The experimental

Table 2. The main parameters of the computation

study 1		study 2	
variable	value	variable	value
N	4	N	3,...,8
p_{ij}	$\frac{1}{20}$ (av. 20 cells)	p_{ij}	$\frac{1}{50}$ (av. 50 cells)
q_i	$\frac{1}{3}$ (av. 2 cells)	q_i	$\frac{1}{6}$ (av. 5 cells)
t_{ij}	$\frac{1}{N}$	t_{ij}	$\frac{1}{N}$
b	$8, \ldots, 40$	b	20



(a) The packet loss probability versus the buffer size (study 1)

(b) The packet loss probability versus the switch size (study 2)

Fig. 7. Numerical results for the packet loss analysis of LB switches

results proof the validity of our assumptions. In particular, in Figure 7(a) we show, that the queue does not experience any loss for the type-0 path $\{0,0,0\}$, and, as expected, the higher the d value is the higher the loss probability of the path is. It is also shown in Figure 7(a) that the higher the buffer size (b) is the less the difference between the loss values for types.

Study 2. Due to lower analysis complexity in comparison with [1], the packet loss of a single queue can be evaluated for larger switches – than those ones in [1]. Figure 7(b) plots the packet loss of the queue if the switch size is increasing – up to the solvable highest size of this model. The detailed set of parameters used in Study 2 is shown in the right hand side of Table 2. We present packet loss only for those two traffic path ($\{1,0,0\}$ and $\{2,0,0\}$) which exist for all considered switch sizes. As it is shown on the plot, with the increase of the switch size, the packet loss decreases. As the average packet size and idle period size keeps to be the same, the increase in number of ports increases the number of queues at the central stage and consequently the buffering capacity for the same set of parameters. Correspondingly, the higher is the LB switch buffering capacity the lower packet loss is experienced.

5 Conclusions

In this paper we have presented an approximate analytical model for evaluation of loss probabilities inside the load-balanced switch with finite buffers and variable length packets. In comparison to the analysis presented in [1], we reduced the complexity of the model from $O\left(N^N\right)$ to $O\left(2^N\right)$. Although the complexity has remained exponential, the new approach has extended the range of packet/cell loss probability evaluation for switches with $N \geq 4$ and large VOQ sizes. Since the load-balanced switch is the architecture of choice when N is large, our next step is the presentation of approximated analysis with linear complexity in [13]. This will enable us to remove restrictions on the port/buffer size of the switch in order to calculate the systems important characteristics (like different kinds of loss, delays, average buffers occupancy).

References

1. Audzevich, Y., Bodrog, L., Telek, M., Ofek, Y.: Variable Size Packets Analysis in Load-balanced Switch with Finite Buffers. Technical report, Technical University of Budapest (2009)
2. Chang, C., Lee, D., Jou, Y.: Load-Balanced Birkhoff-von Neumann switches, Part I: One-Stage Buffering. Computer Communications 25, 611–622 (2002)
3. Chang, C., Lee, D., Lien, C.: Load-Balanced Birkhoff-von Neumann switches, Part II: Multi-Stage Buffering. Computer Communications 25, 623–634 (2002)
4. Yu, C., Chang, C., Lee, D.: CR Switch: A Load-Balanced Switch with Contention and Reservation. In: IEEE INFOCOM 2007, Anchorage, USA, May 2007, pp. 1361–1369 (2007)
5. Chang, C., Lee, D., Shih, Y.: Mailbox Switch: A Scalable Two-Stage Switch Architecture for Conflict Resolution of Ordered Packets. In: Proceedings of IEEE INFOCOM 2004, Hong Kong, March 2004, vol. 3, pp. 1995–2006 (2004)
6. Shen, Y., Jiang, S., Panwar, S., Chao, H.: Byte-Focal: A Practical Load-Balanced Switch. In: IEEE HPSR 2005, Hong Kong, May 2005, pp. 6–12 (2005)
7. Lin, B., Keslassy, I.: The Concurrent Matching Switch Architecture. In: IEEE INFOCOM 2006, Barcelona, Spain, April 2006, pp. 1–12 (2006)
8. Keslassy, I., Chuang, S., Yu, K., Miller, D., Horowitz, M., Solgaad, O., McKeown, N.: Scaling Internet Routers Using Optics. In: ACM SIGCOMM 2003, Karlsruhe, Germany (2003)
9. Audzevich, Y., Ofek, Y., Telek, M., Yener, B.: Analysis of load-balanced switch with finite buffers. In: IEEE Globecom 2008, New Orleans, LA, USA, pp. 1–6 (2008)
10. Tu, C., Chang, C., Lee, D., Chiu, C.: Design a Simple and High Performance Switch Using a Two-Stage Architecture. In: IEEE GLOBECOM 2005, St. Louis, MO, USA, November 2005, vol. 2, pp. 6–11 (2005)
11. Turner, J.: Resilient Cell Resequencing in Terabit Routers. Technical report, Washington University, Department of Computer Science (June 2003)
12. Thompson, K., Miller, G., Wilder, R.: Wide-area Internet traffic patterns and characteristics. IEEE Network 11, 10–23 (1997)
13. Audzevich, Y., Bodrog, L., Telek, M., Ofek, Y.: Scalable model for packet loss analysis of load-balancing switches with identical input processes. In: ASMTA 2009, Madrid, Spain. LNCS, vol. 5513, pp. 249–263. Springer, Heidelberg (2009) (accepted for publication)
14. Turner, J.: Strong Performance Guarantees for Asynchronous Crossbar Schedulers. In: IEEE INFOCOM 2006, Barcelona, Spain, April 2006, pp. 1–11 (2006)

Approximate Analysis of a Round Robin Scheduling Scheme for Network Coding*

Omer H. Abdelrahman and Erol Gelenbe

Department of Electrical and Electronic Engineering,
Imperial College London, UK
{oha06,e.gelenbe}@imperial.ac.uk

Abstract. Network coding (NC) has been proposed as a way to compress flows of packets by combining different packets, provided that there is sufficient redundancy through multiple transmission paths so that the receiving nodes can then decode the flows to reconstruct all of the individual packets. However NC does introduce additional computational overhead, and also creates different additional delays which are the subject of this paper. In particular, we evaluate the queueing and delay performance of NC at intermediate nodes of a store and forward packet network when the cross-encoding of packets is restricted within disjoint subsets of the traffic streams so that packets from different subsets cannot be encoded together. We propose a round robin scheduling scheme for serving these disjoint subsets, and present a queueing model for a single encoding node that captures the effect of NC. The model is analyzed approximately using a decoupling approach, and can be used to predict the additional delays incurred by packets in nodes that use NC. The accuracy of the analytical solution is validated via simulations.

Keywords: Network coding, performance evaluation, vacation models.

1 Introduction

Network coding (NC) [1] allows the cross encoding of packets at intermediate nodes before forwarding them towards their destination, and can offer greater communications efficiency and better usage of overall network bandwidth, at the cost of more processing and additional delays.

In [2], we analyzed the delay performance of NC at intermediate nodes of a store and forward network when the nodes encode together all traffic streams that pass through them. Our analysis showed that NC can improve overall performance significantly provided that it is used *opportunistically*. In this paper, we extend the previous analysis to a more general setting where encoding nodes perform NC within disjoint subsets of the packet streams so that packets from different subsets are not allowed to be mixed. This constraint arises when some source-destination pairs cannot use a sufficient number of redundant paths

* Research supported by the EU FP6 CASCADAS and FP7 DIESIS Projects of the EU Future and Emerging Technologies Program.

J.T. Bradley (Ed.): EPEW 2009, LNCS 5652, pp. 212–217, 2009.

for decoding purposes. To see this, consider the directed network depicted in Fig. 1(a), which has three independent unicast information flows a, b and c between the source destination pairs (s_i, t_i). Without NC, the three flows must share the link $u_1 \rightarrow u_2$ which becomes a bottleneck, while some of the links in the network will not be utilized. However, combining the three flows together at node u_1 will leave non of the receivers able to recover its desired information. We propose a scheduling scheme which can alleviate this problem. In particular, node u_1 can alternate between transmitting the coded flow $a \oplus b$ and the uncoded flow c along the bottleneck link to node u_2, which then forwards the coded flow to both t_a and t_b and the uncoded flow to t_c. Receivers t_a and t_b can then reconstruct the original flows from $\{b, a \oplus b\}$ and $\{a, a \oplus b\}$, respectively. Thus, combining NC and scheduling in this scenario can reduce traffic rate on the bottleneck link while distributing traffic on a larger number of paths. The equivalent queueing model representation of the encoding node u_1 with the proposed coding scheme is presented in Fig. 1(b).

The rest of this paper is organized as follows. In section 2, the queueing model for a single encoding node is presented along with the approximation technique. In section 3, the accuracy of the proposed analysis is validated by comparison with simulation results, and the proposed scheme's performance is compared to the performance of a peer non-coding technique. Conclusions are summarized in section 4.

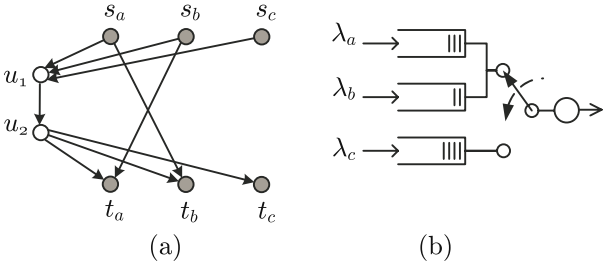


Fig. 1. (a) A directed network with three unicast sessions (s_i, t_i). (b) equivalent queueing model representation of the encoding node u_1 with the proposed coding scheme.

2 System Model

Consider a node receiving F distinct independent flows of packets, each assumed to be Poisson of arrival rate λ_i for the ith flow, which queue up in distinct buffers of infinite capacity. Assume that packet lengths in each stream are independent random variables, and that they are mutually independent between flows, with general distribution $S(x) = Prob[S \leq x]$, where S is the random variable representing packet length. Assume also that the node transmission time is directly proportional to packet length with a constant of proportionality of 1. When the server encodes n packets of different lengths, it packs the shorter packets with *zero*-bits to reach the length of the longest packet, and encodes the resulting

packet bit by bit so that its length will be equal to the largest of the n packet lengths. The transmission time of a packet is then assumed to be proportional to the length of the largest packet.

We divide the input flows $\{1, \ldots, F\}$ into J disjoint subsets which we denote as *coding classes*, and let C_j be the flows in class-j. We set the restriction that only those packets within the same class will be encoded together while packets of different classes will not be mixed. We assume that the server cycles the coding classes in a round robin manner and that it switches from one class to another instantaneously, i.e. we assume zero *switchover* time. When the server visits a coding class, it removes the head-of-the-line packets from non-empty queues in the class, forms a single coded packet and transmits the resulting packet on the output link. When the server arrives at a coding class and finds that all its queues are empty, it immediately switches to the next class.

We will analyze the performance of the system approximately by assuming that the complex model can be decomposed into F separate queues where each queue is treated as a *server of the walking type* [3,4]. In this single-server model, each time the queue is non-empty the server serves one customer (packet) for a service time S then becomes idle for a period T after which it examines the queue again. If it finds that the queue is empty then it takes off for a vacation time V after which it returns once again to examine the queue. Let U be the waiting time in a simple queueing system without vacations and with the same arrival process as the system considered, but with a modified service time $Y = S + T$. Then the following equality holds in distribution [3]:

$$W = U + \hat{V} \tag{1}$$

where $\hat{V}$ denotes the forward recurrence time of the vacation period V which has a distribution:

$$\hat{V}(x) = \frac{1}{E[V]} \int_0^x [1 - V(y)]\, dy \tag{2}$$

Thus the result summarized in (1) allows us to map all properties of interest of a queue with vacations in steady state to those of a system without vacations using the probability distribution of the vacation time V. Note also that (1) holds as long as the arrivals of customers to the queue constitute a renewal process, i.e. inter-arrival times are independent and identically distributed (iid) random variables [3].

We will study a queue, say the ith, in isolation from the others and consider that the queues interact with each other via the steady-state probabilities. Denote by C_{n_i} the coding class to which the ith queue belongs, and define $C_{n_i}(i) = \{x \in C_{n_i} | x \neq i\}$. If the ith queue is not empty when C_{n_i} is scheduled, then the subsequent service time S_i will be obtained from the maximum of the service times for the set of non-empty queues in the class including the ith queue. The server then moves to the next coding class and offers service in a similar manner. The vacation period T_i thus corresponds to the time interval from the server's departure from C_{n_i} until its next visit, which consists of the sum of service times at the other coding classes. Note that these service times can also

be of zero duration if all other coding classes are empty. If, however, the server finds queue i empty upon scheduling C_{n_i}, then a sequence of service times V_i involving the other queues in the same class $\hat{S}_i$ and the other coding classes T_i will take place, some of these services possibly being of zero duration if all the other queues are empty. We will assume that the vacation periods V_i and T_i are iid random variables. The decoupling approximation we propose is as follows. Let q_i be the probability that in steady state the ith server does not participate in encoding a packet when the corresponding coding class is scheduled for service. Let Z_j be the set of all subsets of C_j, including C_j and the empty set. We will assume that the steady-state probability that any subset $Z \in Z_j$ is busy when visited by the server is given by $\prod_{k\in Z}[1-q_k]\prod_{k\notin Z} q_k$. With these assumptions, the solution of the system can be summarized in the following steps:

Step 1. Assuming that the quantities q_i are known, find the distribution of the service and vacation times for each queue i as a function of q_k, $\forall k \neq i$:

$$Y_i = S_i + T_i \tag{3a}$$

$$S_i(x) = S(x) \sum_{Z\in Z_{n_i}(i)} S(x)^{|Z|} \prod_{k\in Z}[1-q_k] \prod_{k\notin Z} q_k \tag{3b}$$

$$T_i = \sum_{j=1, j\neq n_i}^{J} T_{C_j} \tag{3c}$$

$$T_{C_j}(x) = \sum_{Z\in Z_j} S(x)^{|Z|} \prod_{k\in Z}[1-q_k] \prod_{k\notin Z} q_k \tag{3d}$$

$$V_i = \hat{S}_i + T_i \tag{3e}$$

$$\hat{S}_i(x) = \sum_{Z\in Z_{n_i}(i)} S(x)^{|Z|} \prod_{k\in Z}[1-q_k] \prod_{k\notin Z} q_k \tag{3f}$$

where $|Z|$ denotes the number of elements in set Z.

Step 2. For each subsystem i, determine the steady state probability q_i approximately using one of the following expressions:

- The probability of the equivalent ith server being empty when it returns from a vacation period [4]:

$$\hat{q}_i = \frac{1-\lambda_i E[Y_i]}{1+\lambda_i(E[V_i]-E[Y_i])} \tag{4a}$$

- The probability of the ith queue being empty or that it is busy but the equivalent server is idle due to a vacation time:

$$\hat{q}_i = \max\{0, 1-\lambda_i E[Y_i]\} \tag{4b}$$

Step 3. Solve the system of non-linear equations $\hat{q}_i = q_i$, for $i = 1, \ldots, F$.

Now applying standard results for an $M/G/1$ queueing system and utilizing the decomposition property (1), we can write the mean waiting time in the ith queue as:

$$E[W_i] = \frac{\lambda_i E[Y_i^2]}{2(1-\lambda_i E[Y_i])} + E[\hat{V}_i] \tag{5}$$

The mean response time is then $E[R_i] = E[W_i] + E[S_i]$.

The output rate from the encoding node (throughput) can also be obtained approximately as:

$$\gamma = \sum_{j=1}^{J} \sum_{n=1}^{|C_j|} \sum_{\substack{Z \subseteq C_j \\ |Z|=n}} \prod_{k \in Z} [1-q_k] \prod_{k \notin Z} q_k \left(E[T_i]\mathbf{1}_{i \in C_j} + \int_0^\infty x \, nS(x)^{n-1} dS(x) \right)^{-1} \tag{6}$$

where $\mathbf{1}_x$ is the characteristic function which takes the value 1 if x is true and 0 otherwise.

3 Numerical Results

In this section we present numerical results which validate the accuracy of the mathematical model, and we compare the performance of the proposed scheme against a peer non-coding approach. The latter uses a technique known as store and forward in which packets received from different incoming links are stored in a single queue and served in a FIFO order. This can be modelled as the classical $M/G/1$ queueing system. For fair comparison of the two schemes, we provide the same total incoming traffic rate and packet length distribution.

In Fig. 2 we present results for the mean response time and bandwidth efficiency for the encoding node in Fig. 1(a). We assume that packet lengths are

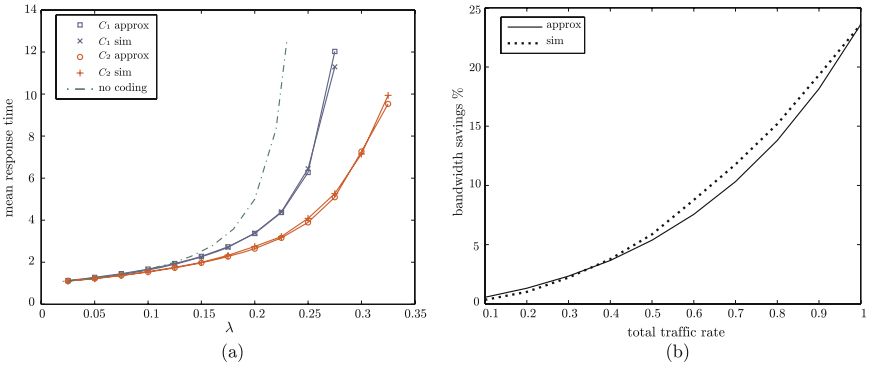


Fig. 2. (a) Mean response time and (b) bandwidth savings, for the encoding node u_1 in Fig. 1. The parameters are: $F = 3$, $J = 2$, $C_1 = \{a, b\}$, $C_2 = \{c\}$, arrival rates vector $\underline{\lambda} = \lambda[1.5 \quad 1.5 \quad 1]$ and exponential packet lengths with mean 1.

exponentially distributed and we vary the load on each queue from very light until a value that saturates the system. The figure indicates that the analytical results are in excellent agreement with those obtained from simulation. The model, however, tends to produce less accurate results when the queues are heavily loaded. This is because the queues are highly coupled in this instance, while the approximation is based on independence assumptions.

The mean response time results clearly demonstrate that the coding scheme outperforms the store and forward method, particularly when the node is heavily loaded. Moreover, since no packet loss is incurred, it follows that the scheme can deliver more packets per time unit, which translates to higher network throughput provided that destinations can reconstruct the original flows in a timely manner. Fig. 2(b) shows the percentage of saved bandwidth resulting from NC when the traffic rates do not saturate the store and forward node, i.e. $\sum_i \lambda_i E[S] < 1$. The figure indicates that coding can reduce bandwidth utilization by up to 25% when the node is heavily loaded. Note that bandwidth reductions of 37.5% could have been achieved if the flows were fully synchronized. In general, the bandwidth savings are more pronounced when the traffic rates within each coding class are balanced, since more coding opportunities arise in such instances.

4 Conclusions

In this paper, a round robin scheduling scheme based on NC has been proposed and analyzed approximately. The analysis was carried out using a decoupling approach, which was validated by simulation. The obtained results showed that the scheme outperforms the traditional store and forward technique, particularly when the system is heavily loaded. In future work, we will consider scheduling policies which aim at maximizing the throughput of the system, by employing priority levels, while maintaining acceptable QoS for the different flows. The analytical model, which seems to provide adequate delay approximation, will also be used to design efficient network algorithms such as routing and flow control under NC.

References

1. Ahlswede, R., Cai, N., Li, S.Y., Yeung, R.: Network information flow. IEEE Transactions on Information Theory 46(4), 1204–1216 (2000)
2. Abdelrahman, O.H., Gelenbe, E.: Queueing performance under network coding. In: IEEE Information Theory Workshop, 2009 (ITW 2009), Volos, Greece (2009)
3. Gelenbe, E., Iasnogorodski, R.: A queue with server of walking type (autonomous service). Annales de l'institut Henri Poincaré (B) Probabilités et Statistiques 16(1), 63–73 (1980)
4. Gelenbe, E., Mitrani, I.: Analysis and synthesis of computer systems. Academic Press, London (1980)

Analysis of Large Populations of Interacting Objects with Mean Field and Markovian Agents

Marco Gribaudo

Dip. di Informatica, Università di Torino
marcog@di.unito.it

Abstract. The necessity of studying sensor networks, rich internet applications, social networks and molecular biology have raised the need of being able to consider systems composed by very large population of similar objects. This lead to the development of new modelling paradigms, such as Fluid Process Algebra, Mean Field analysis and Markovian Agents. These methodologies produces exact results if the number of considered objects goes to the infinity, but provide reasonable approximations even for finite quantities. In this work Mean Field analysis and Markovian Agents models will be presented.

Keywords: Mean field, Fluid Models, Markovian Agents, Large systems.

1 Introduction

In recent years the study of systems composed by a large number of similar interacting objects has become an important issue in performance evaluation. The necessity of studying sensor networks, rich internet applications, social networks and molecular biology, just to name a few examples, have raised the need of being able to consider very large systems. However, the well known *state space explosion* problem has prevented most of the previously developed compositional techniques from being applied. For this reason new approximate analytical techniques have been introduced. For example *Fluid Process Algebra* [4] is an extension of the PEPA stochastic Process Algebra, that considers the number of components as continuous variables, and studies their evolution using o.d.e. *Mean Field Analysis* provides approximation of the counting process of objects described by both Discrete [2] and Continuous [1] Markov chains. *Markovian Agents* [3] instead considers a continuous population of entities (the agents) spread over a space, that communicates by sending and receiving messages.

All the aforementioned techniques are based more or less on the same assumptions: a continuous approximation of the counting process of the number objects in a given state. This allows to compute performance indices of both the complete system and of its component without having to consider the combinatorial state space. Although all these techniques are approximate, they provide very accurate solution as the population increases. In this work we present a short summary of the Mean Field Analysis and of the Markovian Agents models.

J.T. Bradley (Ed.): EPEW 2009, LNCS 5652, pp. 218–219, 2009.

1.1 Mean Field Analysis

Mean Field Analysis is a technique that approximates the counting process of set of partially dependent similar objects. All the entities have the same behaviour, which might depend either on their local state, and on the number of objects in each state. In particular, if there are N objects, each one described by m states, the evolution of the entire system can be approximated by $\boldsymbol{N}(\tau) = N_i(\tau)$. Here $N_i(\tau)$ counts the number of entities in state i, and is such that $\sum_{i=1}^{m} N_i(\tau) = N$ $\forall \tau \leq 0$. If we call $X(\tau)$ the state of a given (tagged) object and following [1] we define $K_{ij}(\boldsymbol{N}(\tau)) = \lim_{\Delta \to 0} \frac{Pr\{X(\tau+\Delta)=j | X(\tau)=i, N(\tau)\}}{\Delta}$ when $i \neq j$ and $N_i(\tau) > 0$, and we set $K_{ij}(\boldsymbol{N}(\tau)) = -\sum_{l \neq i} K_{il}(\boldsymbol{N}(\tau))$ for $i = j$, we can the compute the (approximate) transient evolution of the counting process by defining a matrix $\boldsymbol{K}(\boldsymbol{N}(\tau)) = |K_{ij}(\boldsymbol{N}(\tau))|$, and by solving the following o.d.e.:

$$\frac{d\boldsymbol{N}(\tau)}{d\tau} = \boldsymbol{N}(\tau)\boldsymbol{K}(\boldsymbol{N}(\tau)) \tag{1}$$

Note that $K_{ij}(\boldsymbol{N}(\tau))$ corresponds to the transition rate of from state i to state j of one object in insulation when there is at least one entity in state i.

1.2 Markovian Agents

Markovian Agents Models (MAMs) [3] represent systems as a collection of agents spread over a geographical space. Each agent is described by a finite state machine where two types of transitions can happen: *local transitions* and *induced transitions*. A local transition occurs whenever an agent changes its state due to information perceived on its environment. During a local transition, an agent can produce a message. Induced transitions occur when an agent receives a message. The Markovian Agents(MAs) are spread over a finite geographical area $\mathcal{V}$ that can be either continuous or discrete. We denote by $\rho(\boldsymbol{v}) : \mathcal{V} \to \mathbb{R}^+$ the spatial density function of the agents. For every volume A in $\mathcal{V}$ the number of agents in A is distributed according to a Poisson distribution with mean $\iint_A \rho(\boldsymbol{v}) d\boldsymbol{v}$. Message exchanging is governed by a function, called the *perception function*, that takes into account the agent distribution over the space, the message routing policy and the transmittance properties of the medium.

References

1. Bobbio, A., Gribaudo, M., Telek, M.: Analysis of large scale interacting systems by mean field method. In: 5th International Conference on Quantitative Evaluation of Systems, QEST 2008, St. Malo (2008)
2. Le Boudec, J.Y., McDonald, D., Mundinger, J.: A generic mean field convergence result for systems of interacting objects. In: 4th International Conference on Quantitative Evaluation of Systems, QEST 2007, Edinburgh (2007)
3. Gribaudo, M., Cerotti, D., Bobbio, A.: Analysis of on-off policies in sensor networks using interacting markovian agents. In: 4th International Workshop on Sensor Networks and Systems for Pervasive Computing, PerSens 2008, Hong Kong (2008)
4. Hillston, J.: Fluid flow approximation of PEPA models. In: 2nd International Conference on Quantitative Evaluation of Systems, QEST 2005, Turin (2005)

Author Index