Analytical and Numerical Means to Model Transient States in Computer Networks

Tadeusz Czachórski[1], Monika Nycz[2], Tomasz Nycz[2], and Ferhan Pekergin[3]

[1] Institute of Theoretical and Applied Informatics, Polish Academy of Sciences
Bałtycka 5, 44-100 Gliwice, Poland
`tadek@iitis.gliwice.pl`

[2] Institute of Informatics, Silesian University of Technology
Akademicka 16, 44-100 Gliwice, Poland
`{monika,tomasz.nycz}@polsl.pl`

[3] LIPN, Université Paris-Nord, 93 430 Villetaneuse, France
`pekergin@lipn.univ-paris13.fr`

Abstract. Transient queue analysis is needed to model the influence of time-dependent flows on congestion in computer networks. It may be applied to the networks performance evaluation and the analysis of the transmissions quality of service. However, the exact queuing theory gives us only few practically useful results, concerning mainly M/M/1 and M/M/1/N queues. The article presents potentials of three approaches: Markovian queues solved numerically, the diffusion approximation, and fluid-flow approximation. We mention briefly a software we implemented to use these methods and summarise our experience with it.

Keywords: diffusion approximation, fluid flow approximation, Markov chains, transient states.

1 Introduction

Transient queue analysis is needed to model time-dependent flows and the dynamics of changes of router queues in computer networks. It is needed in stability analysis of Internet connections, It helps to predict packet loss probability and queueing delays which are the major factors of the quality of service. Furthermore, efficient modelling tools are indispensable for coping with large network topologies typical for modern Internet.

The use of analytical models known in queueing theory is limited to M/M/1 and M/M/1/N single queues and even there the solution is complex. Transient states of these models were investigated more than half a century ago. Chapman-Kolmogorov equations (first-order linear differential equations) define state probabilities $p(n, t; n_0)$ of n customers present in the system at time t if $n = n_0$ at time $t = 0$. If we apply the Laplace transform to make these equations algebraic ones, solve them and then find the original functions of the solutions in time domain, we obtain [1]:

$$p(n,t;n_0=i)=e^{(\lambda+\mu)t}\Big[\varrho^{\frac{n-i}{2}}I_{n-n_0}(at)+ \\ +\varrho^{\frac{n-n_0-1}{2}}I_{n+n_0+1}(at)+(1-\varrho)\varrho^{n}\sum_{j=n+n_0+2}^{\infty}\varrho^{\frac{-j}{2}}I_j(at)\Big] \quad (1)$$

where λ is input flow intensity, μ is service intensity (i.e. $1/\mu$ is mean service time), $\varrho=\lambda/\mu$ is server utilisation factor, $a=2\mu\sqrt{\varrho}$ and $I_k(x)$ is the modified Bessel function of the first type and order k. Similarily, transient distributions for the limited queue M/M/1/N were derived [2, 3]. Some simplifications of the solution (1) were proposed, e.g. the generating function of the distribution $\bar{p}(n,s;n_0)$ may be replaced by expressions having simpler originals in time domain [4] or Bessel funcions may be replaced by easier to compute functions, [5].

These results do not fit well to the problem of modelling IP routers, where the incoming streams are not Poisson and the size of packets is not exponentially distributed. Note that the solution (1) refers to transient states but it is assumed that the model parameters, λ in particular, are constant. Hence, in case of time dependent flows we should make them piecewise constant. We need models treating constantly changing non-Poisson flows and assuming general distributions of service times. We need also the possibility to include in these models the description of self similarity of flows. The models should also be scalable to meet very large topologies characteristic to the Internet.

In the sections below we describe our experience with three approaches: Markovian queues solved numerically, diffusion approximation, and fluid-flow approximation.

2 Markov Models Solved Numerically

Markov models are essential for the evaluation of the performance of computer networks. The models support the design of new communication protocols, mechanisms for regulating the intensity of Internet transmissions and mechanisms to ensure the quality of transmission services. However, they are not scalable: the number of states is growing very rapidly with the complexity of a modelled object: each state of the Markov chain corresponds to one state of the system. It is necessary to construct and solve the system of equations defining the probability of states – the number of equations equals the number of states. The existing solvers as e.g. Markovian solver in QNAP, XMARCA, PEPS, PEPSY, PRISM consider only steady state Markov chains and solve algebraic systems of equations.

Theoretically, for any continuous time Markov chain the Chapman-Kolmogorov equations with transition matrix Q

$$\frac{d\pi(t)}{dt}=\pi(t)\mathrm{Q}\ , \quad (2)$$

have the analytical transient solution:

$$\pi(t) = \pi(0)e^{\mathrm{Q}t} \ , \tag{3}$$

where $\pi(t)$ is the probability vector and $\pi(0)$ is the initial condition. However, it is not easy to compute the expression $e^{\mathrm{Q}t}$ where Q is a large matrix, see e.g. [6]. It may be done by its expansion to Taylor's series

$$e^{\mathrm{Q}t} = \sum_{k=0}^{\infty} \frac{(\mathrm{Q}t)^k}{k!} \ , \tag{4}$$

but the task is numerically unstable, especially for large Q. Additionally, to consider $\lambda(t)$, we should make the parameters of the model piecewise constant in small intervals and apply the solution (3) at each of these intervals.

We are developping our own package Olymp. It is a library generating transition matrices of continuous time Markov chains (CTMC), solving them. Olymp uses Java language to define network nodes and the interactions between them. Due to the potentially very large sizes of the models' transition matrices, their generation is parallelized, and they can be compressed on-the-fly using a dedicated compression based on finite-state automata. Olymp has a quite different approach to represent CTMC in the comparison to typical model checkers. A move to another state involves a transfer of a token. A node that sends the token initiates the move asynchronously, in moments of time that adhere to an exponential distribution. A node that receives the token can accept it validating the move. The negotiation can be thought of as for example an agreed transfer of a packet between these two nodes or as a synchronisation on a signal, distributed to the network by a clock node. At the moment we are able to generate and solve Markov chains of the 150 million of states. The method of solution used is one of projection methods based on Krylov subspace with Arnoldi process to project the exponential of a large matrix approximately onto a small Krylov subspace, see [7]; the transition matrix is then small and the computation of the expression (3) with the use of uniformization method and Padé approximations is much easier, [8–11]. This approach is supplemented by direct numerical solution of large systems of ordinary differential equations (ODE) using uniformization, i.e. discretization of the CTMC, that is replacing the CTMC by a DTMC (a discrete-time Markov chain) and a Poisson process.

We are increasing the size of tractable Markov chains by several orders through the use of a GPU-CPU (graphical processing unit) and a better design of computational algorithms for parallel computing and optimization of memory usage, [12]. GPU capabilities go far beyond the computer graphics. It is well known that a potential computational power of GPUs is much greater than that of contemporary CPUs (in a sense of the performance measured by number of floating point operations per second). Thus, it is possible to shorten the time of computations. Due to the enormous amount of the data to be processed, methods must be developed to store vectors and matrices with intelligent management of memory.

3 Diffusion Approximation

This approach is merging states of the considered queueing system and thus needs much less computations than the Markov models. We present here the principles of the method following [13] where steady-state solution of a single G/G/1/N model was given and then extended to the network of queues in [14]. We supplemented these results with semi-analytical, semi-numerical transient state solution [15] given for constant model parameters but it could be applied also in case of time-dependent parameters if we only make them constant within small intervals, as demonstrate numerical results below.

Let $A(x)$, $B(x)$ denote the interarrival and service time distributions at a service station and $a(x)$ and $b(x)$ be their density functions. The distributions are general but not specified, the method requires only the knowledge of their two first moments. The means are denoted as $E[A] = 1/\lambda$, $E[B] = 1/\mu$ and variances are $\mathrm{Var}[A] = \sigma_A^2$, $\mathrm{Var}[B] = \sigma_B^2$. Denote also squared coefficients of variation $C_A^2 = \sigma_A^2\lambda^2$, $C_B^2 = \sigma_B^2\mu^2$. $N(t)$ represents the number of customers present in the system at time t.

Diffusion approximation replaces the process $N(t)$ by a continuous diffusion process $X(t)$, the incremental changes $dX(t) = X(t+dt) - X(t)$ of which are normally distributed with the mean βdt and variance αdt, where β, α are coefficients of the diffusion equation

$$\frac{\partial f(x,t;x_0)}{\partial t} = \frac{\alpha}{2}\frac{\partial^2 f(x,t;x_0)}{\partial x^2} - \beta\frac{\partial f(x,t;x_0)}{\partial x} \ . \tag{5}$$

This equation defines the conditional pdf of $X(t)$:

$$f(x,t;x_0)dx = P\left[x \le X(t) < x + dx \mid X(0) = x_0\right] \ .$$

The both processes $X(t)$ and $N(t)$ have normally distributed changes; the choice $\beta = \lambda - \mu$, $\alpha = \sigma_A^2\lambda^3 + \sigma_B^2\mu^3 = C_A^2\lambda + C_B^2\mu$ ensures that the parameters of these distributions grow at the same rate with the length of the observation period. In the case of G/G/1/N station, the process evolves between barriers placed at $x = 0$ and $x = N$. When it comes to $x = 0$, it remains there for a time exponentially distributed with the parameter λ and then it returns to $x = 1$; when it comes to $x = N$, it remains there for a time which is exponentially distributed with the parameter μ and then to $x = N - 1$. These are not typical bordary conditions for differential equations, therefor we developed a special method to solve the Equation (5): the function $f(x,t;x_0)$ is expressed by a superposition of density functions of the diffusion process with the absorbing barriers $x = 0$ and $x = N$, [15]. The solution is obtained in terms of the Laplace transform of $f(x,t;x_0)$ and then inverted numerically.

We studied the errors of this heuristic approach with the use of a wide range numerical examples. An example is given below. Consider a G/G/1/30 queue (in fact, it is M/M/1/30 queue, as we assume $C_A^2 = C_B^2 = 1$) with the input rate $\lambda(t)$ varying in time as presented in Fig. 1. It represents a typical TCP flow with additive increases and multiplicative decreases in case of packet losses, the range of time is $[0, 100]$ time units. In computations, the values of diffusion parameters are changed each 0.5 time unit. Figs. 2–3 present results of the diffusion model compared with the simulation results (in the latter case it is the average of 500 thousands independent runs). The mean queue is presented both in linear and logarithmic scales, to see better the differences for very small values.

However, when the traffic is different from Poisson and the service times have nonexponential distributions, the errors of the approximation are growing: Fig. 4 presents some exemplary errors for G/G/1/100 station ($\varrho = 0.75$) as a function of C_A^2, C_B^2 (transient state, $t = 100$, at the beginning the queue was empty). The errors of the approximation increase with C_A^2, C_B^2.

In our method the density function of the diffusion process is obtained in form of its Laplace transform which is then inverted numerically with the use of Stehfest algorithm [16]. In this algorithm a function $f(t)$ is obtained from its transform $\bar{f}(s)$ for any fixed argument t as

$$f(t) = \frac{\ln 2}{2} \sum_{i=1}^{N} V_i \, \bar{f}\left(\frac{\ln 2}{t} i\right) \quad , \tag{6}$$

where V_i are known constants and N is an even integer and depends on a computer precision; we used $N = 20$.

The algorithm brings some numerical difficulties, the inversion requires computation of hiberbolic sinus function with arguments that may exceed double-precision floating-point number exponent size which for 64-bit double permits only maximum positive value of 308. That often happens when maximum queue's buffer N is big (like 100). Possible solution would be to use 80-bit extended precision double which permits maximal positive exponent of 4932.

The method may be applied to a network of queueus of any topology [14] although in case of transient states the computations of the time-dependent flows evry small time-interval is of course time consuming. We tested it on an examplary network of 1000 stations.

4 Fluid-Flow Approximation

This approximation method assumes a much simpler flow model determining the mean values of traffic intensity and service times of network stations. In contrary to diffusion approximation, it is based only on first-order ordinary linear differential equations, so that the values are obtained in a much shorter calculation time. Therefore it is suited for modeling transient states of large TCP/IP networks, in particular the Internet. However it is quite difficult to analyze Internet-scale topology manually, thus we prepared a tool that is easily adaptable for analysis of different cases.

Mathematical model implemented in our software, that is already adapted to TCP congestion window mechanism and RED algorithm in routers, was presented in [17, 18]. A modelled network V is described by characteristic values of routers (instantaneous and average queue length and discard probability per

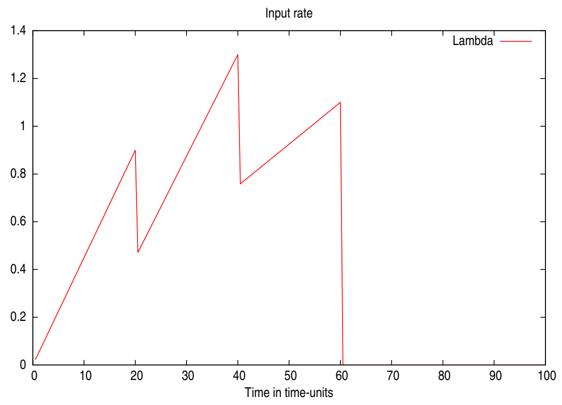

Fig. 1. Input traffic intensity $\lambda(t)$ in the studied example

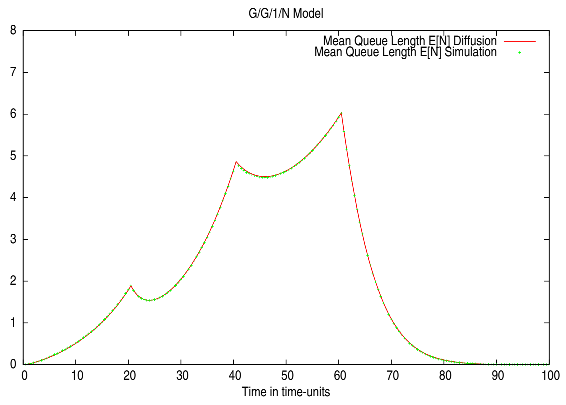

Fig. 2. Numerical example: mean number of customers (linear scale) as a function of time; diffusion approximation and simulation results

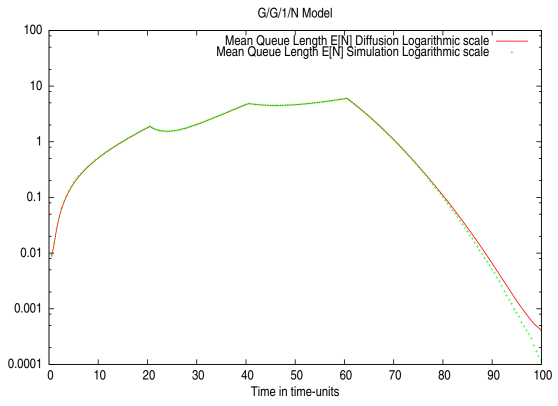

Fig. 3. Numerical example: mean number of customers (logarithmic scale) as a function of time; diffusion approximation and simulation results

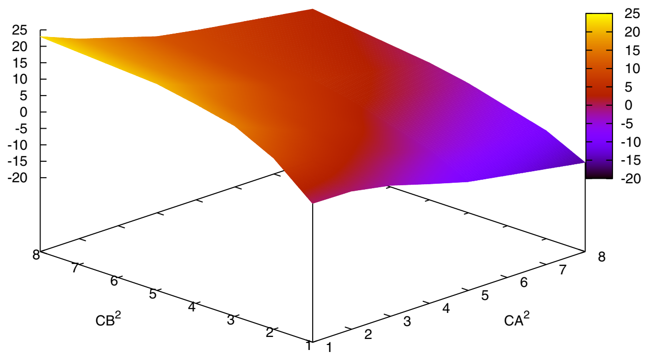

Fig. 4. G/G/1/100 station with $\varrho = 0.75$, relative error of the mean queue as a function of C_A^2, C_B^2; at $t = 100$, at the beginning the queue was empty

router) and flows (congestion window size and round trip time on the whole path of an individual flow) traversing network nodes.

The main parameter of a network node v is its mean queue length q_v; with the transmission capacity C_v, the time change of q_v is defined as

$$\frac{dq_v(t)}{dt} = \sum_{i=1}^{N} \frac{W_i(t)}{R_i(q_v(t))} - \mathbf{1}(q_v(t) > 0) \cdot C_v \tag{7}$$

where W_i is the congestion window of i-th TCP connection (one of N) crossing this node, R_i is the round trip time (RTT) of this connection. Following the TCP Reno policy, for each flow the size W of its congestion window (8) is increasing in average by 1 every RTT (the arrival of a new acknowledgement) or is divided by 2 with the intensity of losses. The loss factor is calculated on the basis of matrix $\mathbf{B}$ which stores the drop probabilities of particular routers for each TCP flows, [19].

$$\frac{dW_i(t)}{dt} = \frac{1}{R_i(\boldsymbol{q}(t))} - \frac{W_i(t)}{2} \cdot \frac{W_i(t-\tau)}{R_i(\boldsymbol{q}(t-\tau))} \cdot \\ \cdot \left(1 - \prod_{j \in V} (1 - B_{ij})\right) . \tag{8}$$

Round Trip Time (9) depends heavily on total queue delay and network topology, to be more precisely, on a total propagation delay (Tp_i) of the i flow route, which consists of K nodes. Total queue delay is computed as the sum of quotients of instantaneous queue length q and transmission capacity C of the routers on the route from source to destination. In our small model modification the total propagation delay is not a global flow parameter, but the sum of the links delays, where a single link is a connection between two network nodes that belongs to one flow path.

$$R_i(\mathbf{q}(t)) \;=\; \sum_{j=1}^{K} \frac{q_j(t)}{C_j} + \sum_{j=1}^{K-1} Tp_j \;\; . \tag{9}$$

Our software implementation allows to model large IP networks having thousands of nodes and thousands of flows, divided however into several categories, which consist of flows with identical route and starting parameters. In case of collection data of large topology we save the results in cumulative binary files, that are converted to plot files when needed. The timing results for such an exemplary topology was presented in [19].

The main drawback of our program was the necessity to manually defining the structure of nodes and flows topologies in text file. Thus, the more complex the network for analysis was, the more time was needed to create the configuration file with input parameters and flows routes. To eliminate the manual definition of flows paths, we recently implemented a converter that import the topology from network topology generator and produce a configuration file for our program. We decided to choose aSHIIP generator [20, 21] because of its ability to generate hierarchical network topologies, typical for the actual Internet networks.

Our converter parses the output file from the aSHIIP program to obtain the number of nodes, number of network layers and the directed graph that illustrates the network topology. The next step assumes to construct the flows routes based on selected options (categorization, partition method, total number of flows). Thus, we determine the border routers – the nodes that are entries points of the network – with the flag defining the node as source or destination. Then, we randomly pick out a pair of source and destination routers and perform the Dijkstra algorithm with the use of Fibonacci Heap (implementation is based on code [22] and [23]) on each source node. There are however some cases, where generated number of paths is less then required by the user. It is due to the fact that the generated hierarchical topology is a directed graph, hence it may happen that it is not possible to find a path – the existing links do not provide two-way connections between necessary nodes. The next and final step involves generation of the starting parameters values i.e. for a router: the maximum size of buffer, initial queue size, transmission capacity, weight parameter and probabilities for RED mechanism; and for flow: the initial window size, usually set to one. During the computations of the settings, the number of layers, clustering and node degree are taken into consideration. The phase ends with saving the settings to a configuration file, that is used as an input to our implementation of fluid flow approximation model.

In our next publication we will present the results of sample topology generated by aSHIIP and processed by our tool as described above.

5 Conclusions

Each of three approaches presented above has its highlights and drawbacks. Markovian models are flexible but have frequently enormous state space and therefore are time and space consuming. Fluid flow approximation is the simplest but it may be easily applied to very large configurations. Diffusion approximation is somewhere between. We believe that each of the methods has its place in the panoply of tools needed in performance evaluation and we develop appropriate software tools. We may also use simulation models. In this purpose we have developed an extension of OMNET++ (a popular simulation tool written in C++, [24]) allowing simulation of transient state models. In particular, random generators were modified to make possible the changes of their parameters as a function of time, a new software was added to collect the statistics of multiple runs and to aggregate them. We used this module to validate the diffusion approximation results. Basically, the simulation run in a transient state investigation should be repeated sufficient number of times (e.g. 500 thousands in our examples) and the results for a fixed time should be averaged. As the number of repetitions is high, the estimation of the errors is easy (confidence interval) on the basis of normal distribution. However, the number of repetitions is related to the value of the investigated probabilities and in case of rare events it should be high; this fact increases considerably the simulation time: typically in some of our examples, 5 min of computation on a standard PC station for a diffusion model is compared to 24 hours of simulations on the same machine.

Acknowledgments. This work was supported by Polish project NCN nr 4796/B/T02/2011/40 *Models for transmissions dynamics, congestion control and quality of service in Internet* and the European Union from the European Social Fund (grant agreement number: UDA-POKL.04.01.01-00-106/09) .

References

1. Champernowne, D.C.: An elementary method of solution of the queueing problem with a single server and constant parameters. J. R. Statist. Soc. B 18, 125–128 (1956)
2. Takâcs, L.: Introduction to the Theory of Queues. Oxford University Press (1960)
3. Tarabia, A.M.K.: Transient Analysis of M/M/1/N Queue – An Alternative Approach. Tamkang Journal of Science and Engineering 3(4), 263–266 (2000)
4. Kotiah, T.C.T.: Approximate transient analysis of some queueing systems. Operations Research 26(2), 334–346 (1978)
5. Jones, S.K., Cavin, R.K., Johnston, D.A.: An Efficient Computational Procedure for the Evaluation of the $M/M/1$ Transient State Occupancy Probabilities. IEEE Trans. on Comm. COM-28(12), 2019–2020 (1980)
6. Moler, C., van Loan, C.: Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later! SIAM Review 45(1), 30–49
7. Stewart, W.: Introduction to the Numerical Solution of Markov Chains. Princeton University Press, Chichester (1994)

8. Scientifique, C., Philippe, B., Sidje, R.B.: Transient Solutions of Markov Processes by Krylov Subspaces. In: 2nd International Workshop on the Numerical Solution of Markov Chains (1989)
9. Sidje, R.B., Burrage, K., McNamara, S.: Inexact Uniformization method for computing transient distributions of Markov chains. SIAM J. Sci. Comput. 29(6), 2562–2580 (2007)
10. Sidje, R.B., Stewart, W.J.: A Numerical Study of Large Sparse Matrix Exponentials Arising in Markov Chains. Computational Statistics & Data Analysis 29, 345–368 (1999)
11. Sidje, R.B.: Expokit: A Software Package for Computing Matrix Exponentials. ACM, Transactions on Mathematical Software 24(1) (1998)
12. Numerical computation for Markov chains on GPU: building chains and bounds, algorithms and applications. Project POLONIUM 2012–2013, bilateral cooperation PRISM-Université de Versailles and IITiS PAN, Polish Academy of Sciences
13. Gelenbe, E.: On Approximate Computer Systems Models. J. ACM 22(2) (1975)
14. Gelenbe, E., Pujolle, G.: The Behaviour of a Single Queue in a GeneralQueueing Network. Acta Informatica 7(fasc. 2), 123–136 (1976)
15. Czachórski, T.: A method to solve diffusion equation with instantaneous return processes acting as boundary conditions. Bulletin of Polish Academy of Sciences, Technical Sciences 41(4) (1993)
16. Stehfest, H.: Algorithm 368: Numeric inversion of Laplace transform. Comm. of ACM 13(1), 47–49 (1970)
17. Misra, V., Gong, W.-B., Towsley, D.: Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED. In: ACM SIGCOMM (2000)
18. Liu, Y., Lo Presti, F., Misra, V., Gu, Y.: Fluid Models and Solutions for Large-Scale IP Networks. ACM/SigMetrics (2003)
19. Czachórski, T., Nycz, M., Nycz, T., Pekergin, F.: Transient states of flows and router queues – a discussion of modelling methods. In: Proc. of International Conference on Networking and Future Internet (ICNFI 2012), Istanbul (April 2012)
20. Weisser, M.-A., Tomasik, J.: Automatic Induction of Inter-Domain Hierarchy in Randomly Generated Network Topologies. In: 10th Communication and Networking Simulation Symposium CNS 2007 (2007)
21. Tomasik, J., Weisser, M.-A.: Internet topology on AS-level: model, generation methods and tool. In: 29th IEEE International Performance Computing and Communications Conference (IPCCC 2010) (2010)
22. Dijkstra's Algorithm for Network Optimization Using Fibonacci Heaps (September 2009),
`http://www.codeproject.com/Articles/42561/`
`Dijkstra-s-Algorithm-for-Network-Optimization-Usin`
23. Fibonacci heap implementation (March 2012),
`http://code.google.com/p/gerardus/source/browse/`
`trunk/matlab/ThirdPartyToolbox/dijkstra.cpp`
24. OMNET++ Community Site, `http://www.omnetpp.org`
25. Czachórski, T., Pekergin, F.: Diffusion Approximation as a Modelling Tool. In: Kouvatsos, D.D. (ed.) Next Generation Internet: Performance Evaluation and Applications. LNCS, vol. 5233, pp. 447–476. Springer, Heidelberg (2011)