On Delays in Management Frameworks: Metrics, Models and Analysis*

Abdelkader Lahmadi, Laurent Andrey, and Olivier Festor

LORIA - INRIA Lorraine - Université de Nancy 2
615 rue du Jardin Botanique
F-54602 Villers-lès-Nancy, France
{Abdelkader.Lahmadi, Laurent.Andrey, Olivier.Festor}@loria.fr

Abstract. Management performance evaluation means assessment of scalability, complexity, accuracy, throughput, delays and resources consumptions. In this paper, we focus on the evaluation of management frameworks delays through a set of specific metrics. We investigate the statistical properties of these metrics when the number of management nodes increases. We show that management delays measured at the application level are statistically modeled by distributions with heavy tails, especially the Weibull distribution. Given that delays can substantially degrade the capacity of management algorithms to react and resolve problems it is useful to get a finer model to describe them. We suggest the Weibull distribution as a model of delays for the analysis and simulations of such algorithms.

Keywords: Management delays analysis, Management delays metrics, Management delays modelling.

1 Introduction

The objectives of a management framework is to maintain the service level objectives of managed systems by detecting, tracking and resolving problems that might occur on those systems. A key performance metric in management frameworks to meet these objectives is the feeding delay of management tasks and its respective quality as perceived by the management algorithms. These algorithms are susceptible to large number of managed systems with complex managed services.

Many approaches have been proposed for management systems, that varies from data-oriented approaches (SNMPv2, RMON), object-oriented approaches (Corba, JMX), mobile agents approaches [1], algorithmic approaches [2,3] and more recently autonomic management and control theory approaches [4]. Those approaches aim at minimizing the management overhead and costs but still accomplishing their management tasks with their desired goals. However, to consider a management system as efficient does not only mean reducing its cost and overhead, but also implies that the achievement of a management task has a certain level of quality, more specifically timeliness and temporal accuracy. The timeliness and the temporal accuracy aspects require that management

* This paper was supported in part by the IST-EMANICS Network of Excellence project.

R. State et al. (Eds.): DSOM 2006, LNCS 4269, pp. 13–24, 2006.

tasks and the retrieved or altered management data must be taken and fitted within a reasonable time window [1]. Thus, one fundamental performance metric of a management algorithm is the end-to-end delay required to retrieve/alter a management attribute within a group of management nodes involved in a management task. End-to-end delays can be expressed in terms of fixed and variable components [5]. In a management system, variable delays occur due to processing time at the management nodes (managers, mid-level-managers and agents), transmission time on the network, and queueing time while the network is busy. In this study we investigate the random nature of these delays and its effect on the performance of a monitoring algorithm on the manager side, more specifically the monitoring error. Previous studies of management frameworks delays led to the establishment of simple analytical descriptions [6] or measurement based analysis where management delays are described by their two first moments (mean and standard deviation). However, we could not find a real investigation on the statistical properties of these delays.

We have developed a benchmarking platform for JMX based management applications [7] to collect management performance metrics at the application-level. In this paper, we present our results towards defining adequate models to describe management frameworks delays. Our analysis is done through a single manager/many agents configuration using a polling scheme with a fixed monitoring interval. Thereby, the analysis of the polling is interesting since it remains the main way in which GetAttribute calls occur in a managed network. We analyse the effect of the number of management agents, presented as a scalability factor, on the properties of monitoring delays. We demonstrate that there is a statistical analysis methodology and modelling for describing management frameworks performance metrics. This statistical analysis resembles in many ways to the engineering reliability modelling using the Weibull model [8] and the IP protocol performance models described in [9].

The remainder of the paper is structured as follows: Section 2 reviews related works. Section 3 describes our metrics for end-to-end management delays, and the factors that affect it. Section 4 describes our measurement methodology and data sets. In section 5, we analyse statistical properties of the delays on a polling-based monitoring algorithm. Section 6 contains a summary of this contribution as well as an outlook for future work.

2 Related Works

Many investigations have studied partially management delays as part of their proposed management frameworks, but to our knowledge none of them did propose a delay models allowing demonstrating the timeliness of management operations. Such models need the matching of delays with well known underlying statistical distributions. In literature, the main used statistics to summarize management delays was the mean and the standard deviation [10,11]. In addition, many works assume that management delays and specifically monitoring delays are uniform [12]. Such assumption is heavy and becomes invalid when the number of managed resources and management agents increase [6]. In such cases, monitoring algorithms do not scale well from a delay perspective where algorithms lost the desired quality (timeliness and temporal accuracy) when achieving tasks. This aspect of delays scaling or timeliness of management tasks has been take up by Liotta et al [1] for mobile agents based management applications as an important

quality parameter for critical management tasks. Chen et al [6] assume in their work of management approaches evaluation that management delays have a non uniform random variable component that fluctuates according to the size of the managed network, but no additional study has been done to describe this non uniformness.

3 Management Delays Metrics

To define management delay metrics, we start from the IPPM framework [9] for IP packets delays measurements. We identify a monitoring delay metric to be relative to a monitoring attribute defined by its name, its delay metric type, the management operation type used to achieve the management task, the agents identifier and the number of management objects carried by the operation. For example, the delay to retrieve the *FreeMemory* attribute from a single agent to a manager is represented as following: *FreeMemory*-AttributeDelay-*GetAttribute*-Manager-Agent-1.

We identify two types of delay metrics: the attribute delay and the group delay. The attribute delay is relative to a single monitoring attribute retrieved from a single agent. This metric is for interest to capture the delay to move an attribute value between two management nodes (for example from an agent to a manager). The group delay captures the delay to move an attribute value from a group of agents with a fixed size. This metric is sensitive to perturbation that might arise on retrieving one attribute or computing an aggregated attribute from a group of agents.

The attribute delay is the total delay that an attribute experiences to retrieve/alter its values with a specific interaction mode (polling or notification), using a specific management operation. The attribute delay is measured in a time unit (seconds or milliseconds) per target attribute. If the interaction mode is notification, then it is a one way delay experienced from a sender to a receiver. If the interaction mode is polling, it is the round trip time between two or more management nodes.

The group delay is the amount of time required for a management algorithm to retrieve/alter the value of an attribute from a group of agents. The group delay is indeed the longest delay served, during a polling round, from any agent of the group. The unit of measure of the group delay is time units per agents group size. Many management algorithms are sensitive to this amount of time since it impacts the algorithm reaction time. During a management task period and according to its scheduling operations approach (sequential/serial or concurrent/parallel) [13], the group delay is defined by the maximum of attribute delays from the agent's group in a parallel management operations scheduling approach, or the sum of the attribute delays in a serial approach.

4 Measurement Methodology

4.1 Application-Level Versus Packet-Level Measurement

There are three approaches to delay measurement [14]. One is to collect packet-level information somewhere in a network path between the manager and agents. Another is

to collect kernel-level information on the manager side. The third is to collect information at the application level on a manager or a mid-level managers side. The packet and the kernel level approaches provide the most detailed information and the most accurate timings. Application-level measurements are easy to implement, and the benchmarking tools are portable. We use application-level measurement to develop and evaluate management delays. The benefit of application-level measurement, in contrast to other levels measurement, is that it reflects delays as perceived by a decision point implemented on a manager or a mid-level manager sides. Thus, the measured delays will include network delays, requests and responses processing delays on each management node, matching and searching attributes values delays on the agent, and security and privacy processing. Figure 1 depicts a time series of measured delays on the manager side at the

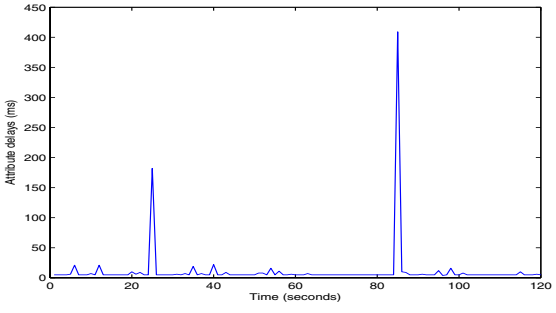


Fig. 1. A time series of measured delays at the application-level on a manager side while sending 1 request per second to a single agent over JMX framework

application-level in a Java-based management framework. We observe some periodic spikes where delay increases due to garbage collector activities each 60 seconds. Therefore, only an application-level measurement technique captures easily these spikes that might affect the performance of the management algorithm. The disadvantages of the application-level measurement are the overhead that introduces when logging measurements datasets, and the loss of information about each individual component delays among agent delays, manager delays and messages delays.

4.2 Data Collection

The data we present and analyse in this paper was collected from our JMX benchmarking platform [15] running on a cluster of 100PC (I-Cluster2) [1] where nodes are connected via a gigabyte Ethernet. We built a synthetic benchmarking application based on some widely used JMX implementations (SUN Reference implementation 1.2 and MX4J 2.0). The synthetic application is used because it provides a flexibility in experimentation with various levels of workload: number of requests per second, type of requests, type of MBeans and the number of attributes on each registered MBean within the MBean server. We used a BEA WebLogic JRockit JVM from a JDK 1.4.2_04 running on an Itanium 2 with 2x900MHZ CPU and 3GB memory. We kept the default

[1] http://i-cluster2.imag.fr

options values of all running JVMs on nodes. For all experiments we fixed the monitoring rate in terms of number of requests per second and we varied the number of agents. On the manager side, we used a concurrent monitoring operations scheduling strategy. The manager creates for each agent a pool of threads that generates a fixed number of requests per second. We use the JMX *getAttribute* operation that carries out a single attribute value from a single MBean. Thus, the size of requests is the same for all experiments. For each agent we stored its requests delays in a separate file, during a measurement of 20 minutes after a warm-up period of 1 minute as recommended in [16]. Within the measurement period, we record the timestamps before and after calling the *getAttribute* operation on the manager side. The difference between the two timestamps represents the measured attribute delay. We use the Java instruction *currentTimemillis* to get the current time in milliseconds with a resolution of 1 ms on Linux operating system[2]. All measurement data are collected and stored on a separate node. Confronted with 20GB of collected data to analyse, it is clear that we cannot hope to individually analyse each trace. We must indeed turn to *automated analysis*. That is, we use developed Perl scripts to analyse our data and generate the corresponding statistical properties. For the statistical analysis of delays, we are refereed to the same methodology proposed by Paxon in his PhD thesis [17] on which the IPPM framework lies [9].

4.3 Statistical Analysis Techniques

The first analysis technique we applied is summarizing a data set. If we wish to summarize the attribute delays we might at first think to express them in terms of their sample mean and variance (or standard deviation). However, in practice we find that often the collect of an attribute experiences one or more delays that are much higher than the reminder. These extreme delays greatly skew the sample mean and the variance, so that the resulting summaries do not accurately describe a typical behavior. A typical management delays description is reflected by the median and the IQR (Inter-Quantile-Range) that remain resilient in the presence of extremes or outliers. This *typical description* is suitable for management algorithms that require unbounded delays to retrieve management data. However, an *extreme description* is reflected by the mean and the standard deviation since they are less robust to outliers. Hence, the choice of statistical estimators for summarizing delays datasets depends on the evaluated management algorithm quality requirements. For a real time monitoring algorithm, the mean and the standard deviation are more adequate to summarize monitoring delays since they reflect any perturbations that might occur on the monitoring system. One other technique we apply, is describing delays using statistical distributions of collected data. The *empirical distribution function* (EDF) [9] of a set of scalar measurements is a function $F(x)$ which for any x gives the fractional proportion of the total measurements that were less equal than x. If x is less than the minimum value observed, then $F(x)$ is 0. If it is greater or equal to the maximum value observed, the $F(x)$ is 1. Moreover, in our analysis we match the group and attribute delays data sets against the popular Weibull distribution

[2] The resolution or granularity of a clock is the smallest time difference that can be measured between two consecutive calls.

model [8] to approximate the underlying statistics. When using this model to approximate the empirical data, we always apply *Maximum Likelihood Estimator* (MLE) [8] for parameters estimation.

5 Statistical Properties of Monitoring Delays

5.1 Attribute Delays Analysis

Firstly, we analyse the attribute delay when transferring the values of the same attribute from a single agent within a variable group size of agents to a manager under a fixed monitoring rate of 1 request per second. We know that the performance bottleneck is located at the manager side since the management approach is centralized [6]. Table 1 shows the attribute delay statistics measured on the manager side when retrieving values of an attribute from an arbitrary agent within a group. We observe from table 1 that

Table 1. Summary of attribute delays statistics measured at the manager side of a single agent within a varied group size

Group size	Attribute Delay (ms) statistics					
	Mean	Std	Median	IQR	Min	Max
70	1.02	0.20	1	0	1	4
140	4.45	46.34	1	0	1	973
210	68.47	432.37	1	0	1	3614
280	105.08	983.41	1	0	1	9960
350	392.73	2250.64	1	0	1	20269
420	675.24	3689.65	1	0	1	33365
490	561.55	4254.14	1	0	1	39816
560	1787.82	8179.39	1	0	1	50395
630	1916.86	8764.2	1	0	1	60646
700	2394.55	12073.7	1	0	1	78994

the attribute delay description varies according to the used statistics estimators as mentioned in the section 4.3. When increasing the size of the group of agents, we observe first that the minimum delay is constant, the mean and the standard deviation increases. Indeed, the median and the IQR [3] remain invariant and constant. Figure 2 shows the time series of attribute delays of different agents from groups of size 70 and 700. We observe that the attribute delays become more randomness while the group of agents size increases. We also observe that the number of delays greater than the monitoring interval of 1 second with a group of 700 agents increases. This is due to queueing delays at the manager side that experiences monitoring calls when the number of agents increases and the manager saturates. Figure 3 shows the empirical distribution of an attribute delay from an agent within a group size of 700. We did not find a suitable statistical distribution fitting these delays. Indeed, we observe that the distribution of attribute delays from an agent is multi-modal as depicted in the figure 3. The percentile

[3] The Inter Quartile Range is the $0.75 - quantile$ minus the $0.25 - quantile$.

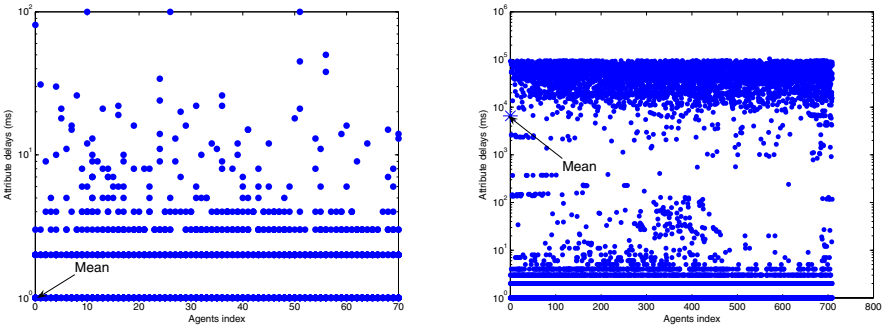


Fig. 2. Empirical time series of attribute delays per agent from two groups of agents of different sizes (y-axis is in log scale)

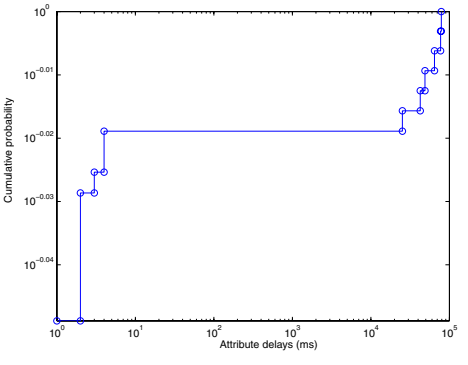


Fig. 3. Log-Log plot of the Empirical Distribution of the attribute delays of an agent within a group of 700 agents

α of attribute delays less than the monitoring interval of 1 second follow closely a normal distribution $N(1.10, 0.44)$. These delays represent monitoring calls that experience less queueing delays on the manager side. However, the $1-\alpha$ of delays greater than the monitoring interval follow closely a Weibull distribution. Despite the multi-modal behavior of the attribute delay, we claim that the attribute delays follow a weibull distribution since the normal distribution is a particular case of a weibull one. [4]

5.2 Group Delay Analysis

Secondly, we analyse the group delay of an attribute from a varied size group during a management period (a polling round for example) with a given management task (monitoring or configuration) and using a given management operation. Within a management period, the group delay is defined as a random process $X = \{X_i\}_{i=1,2,..m}$ of a group of

[4] If the shape parameter is close to 3.602 the weibull distribution is close to a normal.

agents with size m. In this section, we are interested in the behavior of X. We plot the EDF function as depicted in figure 4 of the mean attribute delays from each agent with different group sizes varying from 70 to 700. The group mean attribute delays shift right when the group size increases that means the monitoring messages are more burstiness and the queueing delays increase on the manager side. We use the maximum likelihood

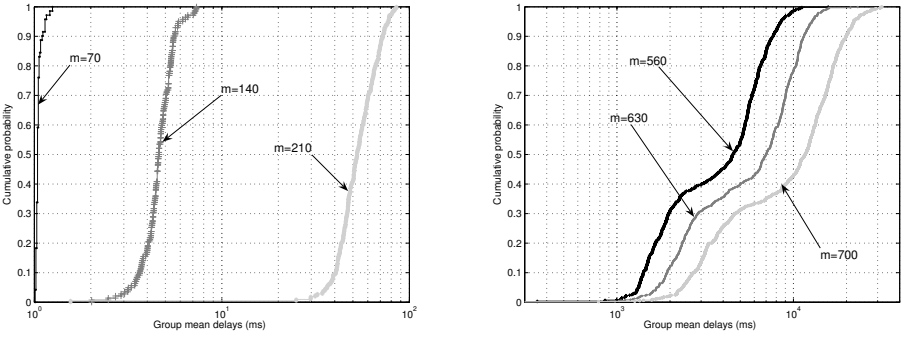
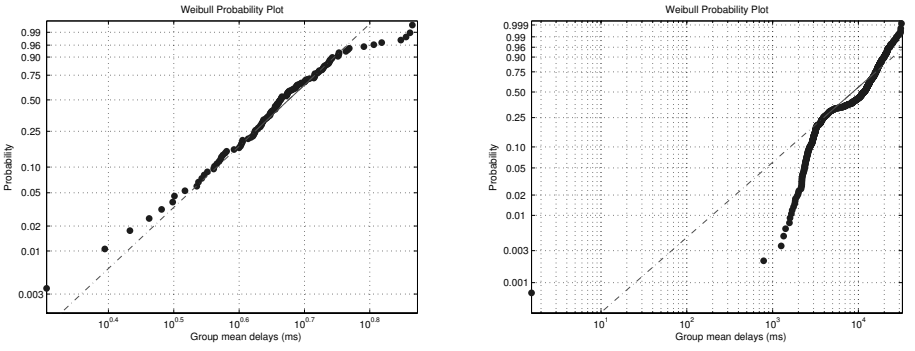


Fig. 4. Empirical Distribution of the group mean delays of a monitoring task with a monitoring period of 1 second, using a polling interaction between one manager and a group of agents of size m, and using the JMX GetAttribute operation (x-axis is in log scale)

estimators [8] to find an adequate classical statistical underlying distribution of group delays with a fixed size. Our finding is that the Weibull distribution best fits the group delays data set. This distribution has been recognized as a good model for TCP inter-arrival times [18] and for many fields in engineering reliability components lifetimes [8]. Figure 5 reports the Weibull probability plots [5] of the group attribute mean delays distribution of respectively 140 and 700 agents. We find that for a size of 140 agents the group mean delays best fit the weibull distribution than a larger group of 700 agents. We find that as the group size becomes larger the group mean delays best fit a normal distribution. It is known that the normal distribution well fits sample means of a large number of independent observations from a given distribution [19][page 494]. Consequently, samples mean delays of a group of agents well fit a normal distribution as the size of the group becomes larger. The parameters a, b of the Weibull distribution represent respectively the so called *scale* and *shape* parameters which determine its structure and statistics. Varying the values of these two parameters highly impacts the appearance of the Weibull distribution. The parameter a is closely related to the distribution peak, and the parameter b is concerned with the tail behaviour.

When the shape parameter $b \gg 1$, the mode, the median and the mean are close to the value of the parameter a. Thus, parameter a represents the mean delays values where data are mainly located (distribution peak). We observe from table 2 that with a small group of 70 agents, we have a shape parameter of 17.11 large greater than 1. Therefore,

[5] The purpose of a Weibull probability plot is to graphically assess whether the data in the random variable X could come from a Weibull distribution. If the data are Weibull the plot will be linear.



140 agents **700 agents**

Fig. 5. The Weibull probability plot of group attribute mean delays of group sizes of 140 and 700 agents. The fitted Weibull distributions have respectively a scale parameters $a(140) = 5.02$, $a(700) = 12349$ and a shape parameters $b(140) = 5.52$,$b(700) = 2$.

for small groups the group mean delays well fit a weibull distribution and the first and second-order statistics are close to the scale parameter. We also observe that the shape parameter decreases, as the size of the group of agents increases. When the group of agents becomes larger enough, the shape parameter is close to 2 and the Weibull distribution approximates a normal distribution. It is obviously that the normal distribution is used whenever the randomness is caused by several independent sources acting additively; for example sample means of a large number of independent observations from a given distribution.

5.3 Management Delays Quality

To gauge how well management delays scale when the size of a group of agents increases, we are interested to identify the proportion of attribute delays that are less or equal to a *maximum tolerable delay* that we consider, in this work, as the monitoring interval Δ (1 second in our case). Let θ be this proportion of attribute delays, statistically we obtain $\theta = P[X \leq \Delta] = EDF(\Delta)$. The quantity $1 - \theta$ represents the proportion of attribute delays that are late after the monitoring interval and θ is a quality parameter of a monitoring framework delay. For the group delay metric, the parameter θ becomes more interesting, since it captures the proportion of agents that respond within a delay less or equal to the monitoring interval. This quantity represents the monitoring error of the management system and captures the quality of the monitoring algorithm temporal accuracy. Table 2 shows the measured and predicted values from the fitted distribution of the $1 - \theta$ parameter.

5.4 Cross-Validation

In order, to give a first insight on the validity of our model, we use the group mean attribute delays dataset from the study of Pras et al [20] to partially verify that group delays approximate a Weibull distribution. The validation is partial because their datasets

Table 2. The measured and the predicted monitoring error $1-\theta$ from group mean delays of different sizes group of agents

Group size (m)	Measured monitoring error $1-\theta$	Predicted monitoring error $1-\theta$	Weibull parameters	
			Shape (b)	Scale (a)
70	0	0	17.11	1.07
140	0	0	5.52	5.02
210	0	0	4.54	59.16
280	0	0	1.81	77.54
350	0.05	0.04	2.28	610.36
420	0.51	0.62	1.9	1468.88
490	0.91	0.86	2.1	2519.79
560	0.99	0.95	1.8	4902.03
630	0.99	0.98	1.8	7437.27
700	0.99	0.98	2	12349

are measured for SNMP based monitoring system and are packet-level measurements. We focus on the dataset delays of retrieving a single object from the group of agents. Our finding that their group mean attribute delays of their data set with a group size of 23 agents fits closely a Weibull distribution as depicted in the figure 6. In their paper, the authors record SNMP agents Round-Trip delays at the UDP-level, indeed our dataset delay are recorded on the application-level with RMI as underlying protocol that lies on the TCP protocol. As noted in [18], wired TCP flow arrivals are well modeled by a 2-parameters Weibull distribution and claimed that it gives a better fit than other distribution models (lognormal, pareto and exponential). Hence, according to these works based on packet-level measurement and our work based on application-level measurement, we claim that the Weibull model is a good candidate to model delays in management frameworks.

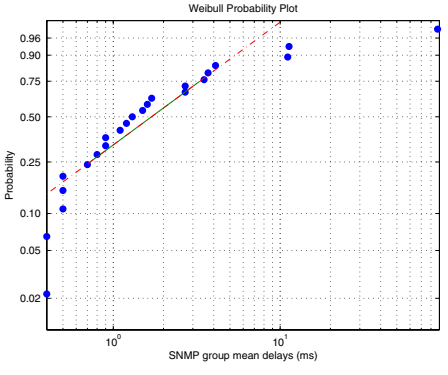


Fig. 6. The Weibull Probability plot of group attribute mean delays of SNMP agent's group size of 23. The fitted Weibull distribution has a scale parameter $a = 3.46$ and a shape parameter $b = 0.62$.

6 Conclusion and Future Works

In this paper, we have analysed management delays within a simple centralized monitoring algorithm. Our primary objective is to identify a set of metrics that characterize delays. We based our study on standardized work from other fields, specially the IPPM framework and Vern Paxon work [17]. We identified how to use the statistical estimators (mean, standard deviation or median and IQR) to characterize management delays based on the quality requirements (delays dependant or independent) of the evaluated algorithm. Our analysis of a synthetic JMX based management benchmark datasets, shows that group delays have a statistical underlying distribution, identified as the Weibull model. Understanding the statistical modelling of management delays is important for the simulation and analytical performance evaluation studies of management frameworks. It is also interesting for the proper designing of management applications (buffer sizing and underlying transport protocol). In this work we initiate an empirically derived analytical models of management frameworks, that could be exploited in simulation environments or performance prediction. As a further interesting work, we will investigate the coupling of packet-level and application-level delays measurement datasets to better understand management frameworks delays and develop more accurate models. Although the polling model is probably the main way in which Get calls occur in a managed network, it will be interesting to study how the network would behave with notifications, particularly when notifications signal errors, and this is the time that the network is most critical.

References

1. Liotta, A., Knight, G., Pavlou, G.: On the performance and scalability of decentralized monitoring using mobile agents. In: DSOM. (1999) 3–18
2. Chen, Y., Bindel, D., Song, H., Katz, R.H.: An algebraic approach to practical and scalable overlay network monitoring. In: Proceedings of the ACM SIGCOMM 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 30 - September 3, 2004, Portland, Oregon, USA, ACM (2004) 55–66
3. Chua, D., Kolaczyk, E.D., Crovella, M.: A statistical framework for efficient monitoring of end-to-end network properties. In: Proceedings of the International Conference on Measurements and Modeling of Computer Systems, SIGMETRICS 2005, June 6-10, 2005, Banff, Alberta, Canada, ACM (2005) 390–391
4. Hellerstein, J.L., Diao, Y., Parekh, S., Tilbuyr, D.M.: Feedback Control of Computing Systems. Wiley-Interscience (2004) ISBN: 0-471-26637-X.
5. Bolot, J.C.: End-to-end packet delay and loss behavior in the internet. In: SIGCOMM '93: Conference proceedings on Communications architectures, protocols and applications, New York, NY, USA, ACM Press (1993) 289–298
6. Chen, T.M., liu, S.S.: A model and evaluation of distributed network management approaches. IEEE journal on selected areas in communications **20** (2002)
7. Lahmadi, A., Andrey, L., Festor, O.: On the impact of management on the performance of a managed system: A jmx-based management case study. In: DSOM. Volume 3775 of Lecture Notes in Computer Science., Springer (2005) 24–35
8. Johnson, N.L., Kotz, S., Balakrishnan, N.: Continuous Univariate Distributions. 2 edn. Volume 1. John Wiley & Sons (1994) ISBN: 0-471-58495-9.

9. Paxson, V., Almes, G., Mahdavi, J., Mathis, M.: RFC 2330: Framework for IP performance metrics (1998) Status: INFORMATIONAL.
10. Lim, K.S., Stadler, R.: Weaver: Realizing a scalable management paradigm on commodity routers. In: Integrated Network Management. Volume 246 of IFIP Conference Proceedings., Kluwer (2003) 409–424
11. Gu, Q., Marshall, A.: Network management performance analysis and scalability tests: SNMP vs. CORBA. In: IEEE/IFIP Network Operations & Management Symposium, Seoul, Korea. (2004)
12. Moghe, P., Evengelista, M.: Rap-rate adaptive polling for network management applications. In: Network Operations and Management Symposium, 1998, NOMS 98, IEEE. Volume 3. (1998) 395–399 ISBN: 0-7803-4351-4.
13. Beverly, R.: RTG: A Scalable SNMP Statistics Architecture for Service Providers. In: Proceedings of the 6th Systems Administration Conference (LISA 2002). (2002) 167–174
14. Downey, A.B.: An empirical model of tcp performance. In: MASCOTS, IEEE Computer Society (2005) 45–54
15. Andrey, L., Lahmadi, A., Delove, J.: A jmx benchmark. Technical Report RR-5598, Loria-INRIA Lorraine (2005)
16. Demarey, C., Harbonnier, G., Rouvoy, R., Merle, P.: Benchmarking the round-trip latency of various java-based middleware platforms. Studia Informatica Universalis Regular Issue **4** (2005) 7–24 ISBN: 2-912590-31-0.
17. Paxson, V.E.: Measurements and analysis of end-to-end Internet dynamics. PhD thesis, Berkeley, CA, USA (1998)
18. A.Feldmann: Characteristics of TCP connections. In: Self-similar Network Traffic and Performance Evaluation. Jonhn Wiley and Sons (2000) 367–399
19. Jain, R.: The art of Computer Systems Performance Analysis. John Wiley & Sons, Inc (1991) ISBN : 0-471-50336-3.
20. Pras, A., Drevers, T., de Meent, R.V., Quartel, D.: Comparing the performance of SNMP and web services-based management. eTransactions on Network and Service Management(eTNSM) **1** (2004)