Resource Management in Computing Systems

Payam Amani

LUND UNIVERSITY

Department of Electrical and Information Technology

Licentiate Thesis
Series of licentiate and doctoral theses No. 101
ISBN 978-91-7753-248-4 (print)
ISBN 978-91-7753-249-1 (web)
ISSN 1654-790X

Department of Electrical and Information Technology
Lund University
Box 118
SE-221 00 LUND
Sweden

To my wife Nassim, for all her love and support

Abstract

Resource management is an essential building block of any modern computer and communication network. In this thesis, the results of our research in the following two tracks are summarized in four papers.

The first track includes three papers and covers modeling, prediction and control for multi-tier computing systems. In the first paper, a NARX-based multi-step-ahead response time predictor for single server queuing systems is presented which can be applied to CPU-constrained computing systems. The second paper introduces a NARX-based multi-step-ahead query response time predictor for database servers. Both mentioned predictors can predict the dynamics of response times in the whole operation range particularly in high load scenarios without changes having to be applied to the current protocols and operating systems. In the third paper, queuing theory is used to model the dynamics of a database server. Several heuristics are presented to tune the parameters of the proposed model to the measured data from the database. Furthermore, an admission controller is presented, and its parameters are tuned to control the response time of queries which are sent to the database to stay below a pre-defined reference value.

The second track includes one paper, covering a problem formulation and optimal solution for a content replication problem in Telecom operator's content delivery networks (Telco-CDNs). The problem is formulated in the form of an integer programming problem trying to minimize the communication delay and cost according to several constraints such as limited content replication budget, limited storage size and limited downlink bandwidth of each regional content server. The solution of this problem is a performance bound for any distributed content replication algorithm which addresses the same problem.

Acknowledgments

First of all, I would like to thank my wife Nassim, my parents Habibollah and Najmehalsabah and my sister Pegah who have supported me through the years, and their unconditional love and support helped me pass the hardships, allowing me to get where I am now.

Furthermore, I would like to thank my supervisor Anders Robertsson who has not only been my mentor during my studies towards the PhD but also a very good friend. Your deep knowledge, great teaching skills, amazing personality and positive energy has inspired me during this time. I also would like to extend my gratitude to my supervisor Christian Nyberg for his valuable guidance and comments regarding the thesis.

Many thanks to all the co-authors of the research papers we have written over the years, especially my colleague Saeed Bastani with whom I spent many hours discussing topics regarding content replication. Moreover, I would like to thank the colleagues in the department of electrical and information technology (EIT), the department of automatic control, the Lund Center for Control of Complex Engineering Systems (LCCC) and the mobile and pervasive computing institute (MAPCI) whom I enjoyed working and collaborating with.

My many regards goes to Leif Andersson for kindly sharing his extensive knowledge and experience of LaTeX with me, which made the thesis look much nicer.

Financial support: Payam Amani was a member of LCCC, a Linnaeus Center at Lund University, funded by the Swedish Research Council.

Payam Amani
Lund, April 2017

List of Acronyms

Acronym	Description
Telco-CDN	Telecom operator's content delivery network
NARX	Nonlinear auto-regressive neural network with exogenous inputs
MLP	Multi-layer perceptron
CCNN	Cascade correlation neural network
PRNN	Pattern recognition neural network
KCCA	Kernel canonical correlation analysis
LDS	Load dependent server
MSS	Mobile service support system
HLR	Home location register
NE	Network element
TDNN	Time delay neural network
ARX	Auto-regressive with exogenous inputs
MAS	Management server
CDN	Content delivery network
MCS	Main content servers
KPI	Key performance indicator
RCS	Regional content server
LRU	Least recently used
LFU	Least frequently used
DTN	Delay tolerant network

Acronym	Description
AP	Access point
SP	Service provider
ISP	Internet service provider
CPU	Central processing unit
MLE	Maximum likelihood estimator
MSE	Mean squared error
MAE	Mean absolute error
I/O	Input/Output
JDBC	Java database connectivity
LAC	Load-adaptive controller

Contents

1

Modeling, prediction and control for multi-tier computing systems

Resource management of server systems is of great interest in the research community as poorly managed systems contribute to severe performance degradation in computing systems. Experience indicates that enterprise servers are usually the bottleneck in computing systems while the backbone is underutilized. Thus, performance models of server systems, particularly in the high traffic region, are important building blocks in the design of optimal resource management techniques in modern computing systems. In the next section, we will present the most widely used architecture in web based computing systems, i.e., the multi-tier architecture [Barry, 2003].

1.1 Multi-tier Computing Systems

Many modern computing systems use web technologies to provide their customers with a vast variety of services. Due to requirements imposed by the flexibility and re-usability of software, most web applications are designed in a multi-tier architecture. Each tier here is assigned a specific functionality. The most widely used example of this type of architecture is the 3-tier architecture which partitions the system into three sections, namely presentation, application and data tiers.

Web-servers such as Apache HTTP Server serve as the building blocks of the presentation tier. The second tier holds the business logic, consisting of application servers such as Apache tomcat and Glassfish server. The last tier, called the data tier, consists of database servers such as MySQL server and is responsible for data storage and accessing. This is illustrated in Figure 1.1.

Dynamical modeling of each of these tiers in their operation range is a required basis for the design of automatic resource management entities in modern

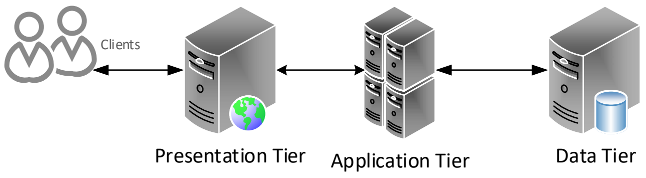


Figure 1.1 Multi-tier computing system.

computing systems, in particular overload protection and admission control entities.

1.2 Dynamical modeling of web servers

Previously in [Cao et al., 2003], it has been shown that CPU constrained computing systems such as web servers can be modeled as a single server queuing system. Also, [Kihl et al., 2003] states that a non-linear model is more capable of representing the dynamics of a single server queuing system compared to a linear model. Figure 1.2 illustrates a single server queuing system in which the distribution of the inter-arrival times and service times are general. The mean arrival rate and the mean service rates of the queuing system are denoted by λ and μ respectively.

The literature offers many attempts to develop analytical estimators or predictors for parameters of single server queuing systems. Clark in his pioneering work has presented a maximum likelihood estimator of arrival and service rates [Clarke et al., 1957]. Using the waiting time data, Basawa *et al.* in [Basawa et al., 1996] have presented a maximum likelihood estimator for single server queues. Some popular performance measures of queuing systems such as mean waiting time in the queue, mean waiting time in the system, mean number of customers in the queue and mean number of customers in the system were studied by Zheng and Seila in [Zheng and Seila, 2000] . They showed that estimators generated by replacing parameter estimators with the unknown parameters in the formula for the above mentioned performance measures have the undesirable characteristic that

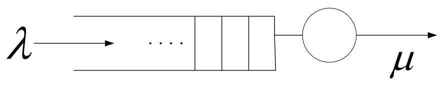


Figure 1.2 A single server queue with mean job arrival rate λ and mean service rate μ.

the expected value of the estimator is non-existing and the mean squared error of the estimator is infinite. In addition, they proposed a setup to fix these undesirable characteristics. For the first time, the concept of Bayesian statistical inference was applied by McGrath *et al.* in [McGrath et al., 1987] and [McGrath and Singpurwalla, 1987] to an M/M/1 queuing system in which the arrivals form a Poisson process and the service times of the single server are exponentially distributed. Their work has been considerably extended in [Armero, 1994] and [Choudhury and Borthakur, 2008].

The analytical approaches mentioned above to estimate parameters of single server queuing systems present some unfavourable characteristics when used in overload protection and admission control schemes. All these methods can only be applied to steady state and stationary scenarios where the queuing system's mean arrival and service rates are constant and time invariant. It should also be noted that none of these methods support multi-step-ahead prediction.

The requirement for a nonlinear multi-step-ahead response time predictor that can work well under stationary and time varying scenarios led us to a black box approach to identify the dynamics of a single server queuing system. In [Amani et al., 2011a], we have presented a multi-step-ahead response time predictor that has all the required characteristics which the analytical models lack and was designed based on a nonlinear auto-regressive neural network with exogenous inputs (NARX). This neural network and the structure of the predictor will be introduced further in section 1.5. The proposed response time predictor works very well in terms of both mean absolute and mean squared prediction errors. Tran *et al.* in [Tran et al., 2013] have compared the respective performance of five neural network based forecast models for web server workload including NARX, Multi-layer Perceptron (MLP), Elman, Cascade Correlation Neural Network (CCNN) and Pattern Recognition Neural Network (PRNN). They have established that the best prediction accuracy was provided by NARX. This is completely in line with our research in [Amani et al., 2011a].

1.3 Dynamical modeling of database servers in data tier

Database servers form the building blocks of the third tier in the 3-tier web technologies, i.e., the data tier. These servers require secure, reliable and real-time activation, modification and deactivation of both new and current customers and services.

There is a resource access conflict among queries related to the management of the system and queries related to the services for current users. As all these tasks should be performed fast and in an automated manner, control systems designed to avoid the resource access conflict and protect the database system from being overloaded are indispensable. As these control systems have to predict a resource access conflict well ahead in time so as to take proper action, they include a feed-forward controller. This will require a multi-step-ahead state predictor which can represent the dynamics of the data tier with fair precision in its operation range and in the areas close to the overload region with high precision. The query response time is considered to be the main state for this purpose. The main methods used to develop response time estimators or predictors for database queries can be divided into two main categories, namely analytical and data-driven methods.

Analytical models are only valid for certain types of database queries and assume a number of simplifying conditions [Zhang et al., 2007; Tomov et al., 2004; Watson et al., 2010]. Therefore, these models are not able to capture the complex dynamics of the data tier. Another of their shortcomings is that they are only valid under static and stationary scenarios and cannot represent the data tier under time-variant and dynamical scenarios.

On the other hand, several instances of data-driven methods for modeling the dynamics of the data tier are presented in the literature. Ganapathi et. al have utilized Kernel Canonical Correlation Analysis (KCCA) to predict several metrics for database queries including the response time [Ganapathi et al., 2009]. This work has been extended by the authors in [Ganapathi et al., 2010] to cover workload modeling for the cloud. In order to throttle long running queries, Tozer et. al in [Tozer et al., 2010] used a linear regression model for the query response time.

A Bayesian approach for online performance modeling of database appliances using gaussian models was proposed by Sheikh et. al in [Sheikh et al., 2011]. This model offered the possibility of adapting to changes in the workload and configuration. The requirement for a nonlinear multi-step-ahead query response time predictor that can work under both steady state and stationary scenarios as well as under time varying and non-stationary scenarios led us to a gray box approach to dynamical identification of database servers. In [Amani et al., 2011b], by means of the same type of NARX neural network as in [Amani et al., 2011a], we have designed a query response time predictor that has all the aforementioned required characteristics and can represent the dynamics of query response times of the database servers under various load and query mix conditions with a high precision represented by very small mean absolute, mean squared and sum of squared prediction errors.

Queuing theory can be utilized for the performance modeling of database servers. The concept of load dependent server (LDS) models in which the response time of the jobs in the system is a function of the service time of the jobs and the current number of jobs waiting to be served in the system to the best of our knowledge was first introduced in [Perros et al., 1992]. Rak et. al [Rak and Sgueglia, 2010], Curiel et. al [Curiel and Puigjaner, 2001] and Perros *et al.* [Perros et al., 1992] used standard benchmarks for database workload generation as well as regression models to capture the system dynamics. A multi-step model parameter calibration strategy was used to fine-tune the model's parameters. The resulting models belong to the data-driven model class. In [Mathur and Apte, 2004], a queuing network representing the load-dependent behavior of the LDS was presented and validated only by simulations. Two queuing systems, i.e., D/G/1 and M/G/1 with load dependency assumptions, were theoretically analyzed in [Leung, 2002] by Leung. The D/G/1 is a single server queuing system with fixed regular inter-arrival times and a general service time distribution. The M/G/1 is a single server queuing system with exponentially distributed inter-arrival times and a general service time distribution. These models were developed to be used in congestion control schemes in broadband networks.

In [Kihl et al., 2012], we added the concept of load dependency to an M/M/m queuing system, which is a queuing system with exponentially distributed inter-arrival times, exponentially distributed service times, *m* servers and an infinite queue, and also to an M/M/m/n queuing system, which is the same as M/M/m except that it employs n queuing positions instead of an infinite queue.

The properties of the load-dependent M/M/m model (M/M/m-LDS) are set by exponentially distributed service times with mean service time equal to base processing time $x_{base} = 1/\mu$ and a dependency factor f. When a request enters the system, it will be assigned on average the base processing time x_{base}. A single request in the system will on average have a processing time of x_{base}. Each additional request inside the system increases the residual work for all requests inside the system (including itself) by a percentage equal to the dependency factor f of the base processing time x_{base}. When a request leaves the system, all other requests have their residual work decreased by f percent of the base processing time x_{base}. This means that if n concurrent requests enter the system at the same point, they will all have a processing time of $x_s(n) = x_{base} \cdot (1+f)^{n-1}$. A special case is when $f = 0$. It means that there is no load dependency, and all requests will have a processing time x_{base}. The system can process a maximum of m concurrent requests at each time instance. Any additional request will have to wait in the queue. New requests arrive according to a Poisson process with the average rate λ. Therefore, the system can be modeled as a Markov chain as illustrated below in Figure 1.3.

The steady state probabilities, average number of jobs in the system and average response times were calculated by means of queuing theory for both queuing systems. In order to tune the model's parameters to represent the

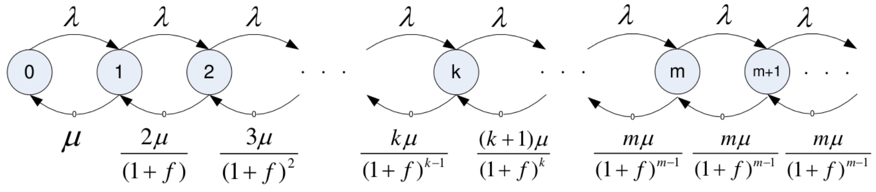

λ
0
1
2
k
m
m+1
μ
$\frac{2\mu}{(1+f)}$
$\frac{3\mu}{(1+f)^2}$
$\frac{k\mu}{(1+f)^{k-1}}$
$\frac{(k+1)\mu}{(1+f)^k}$
$\frac{m\mu}{(1+f)^{m-1}}$

Figure 1.3 Load dependent M/M/m queuing system as a Markov chain

dynamics of the current database, query combination and load set-up, the effects of variations of each parameter on the mean response time of queries sent to the database as a function of the mean effective arrival rate were studied.

Furthermore, some heuristics for fine-tuning the model parameters were introduced. Finally, via some experimental results, we have shown that the M/M/m/n-LDS model is able to represent the response time of the queries sent to the database as a function of the mean arrival rate of the queries. This result was further presented and used in [Amani et al., 2012] in an admission control scheme for Ericsson's mobile service support system.

1.4 Mobile Service Support system and admission control design

Application tier is the middle layer between presentation and data tiers and holds the business logic. Application servers are the building blocks of this tier. In this section we will introduce an entity in the mobile network which is developed by Ericsson AB based on the application tier architecture.

The Mobile Service Support system (MSS), which Ericsson AB develops, handles the set-up of new subscribers and services into a mobile network. To the operator and its business support systems, it offers a unified middle-ware where complex functions, such as setting up a new subscriber or modifying services for an existing subscriber, can be easily invoked. The software architecture is complex, with several layers and distributed infrastructures, which means that specific parts of the system will not have complete knowledge of the interactions among other parts of the system. The system architecture is illustrated in Figure 1.4.

A request to the MSS from an upstream system such as customer administration system normally results in a number of requests send out on the mobile network to several different network elements (NEs). A network element is usually a database storing subscriber and service data, such as for example the Home Location Register (HLR).

A user ID, which needs to be fetched from one database, has to be supplied in a query to another database to ensure the system's consistency.

In parallel to the changes and set-ups performed by the MSS, the network is also employed by the end users, i.e., mobile subscribers. Services set up by the MSS are queried by base stations and other systems requiring that information. With respect to the MSS, this traffic can be considered as unknown background traffic, in contrast to the known traffic flowing through the MSS.

The experience from deployed Ericsson systems shows the possibility of overload situations in the NEs. The measurable (known) load coming from the MSS and the not directly measurable (unknown) load coming from the the mobile users may compete in a race for resources in an NE that may lead to overload in that NE. If such an overload happens and the NE becomes unresponsive, all

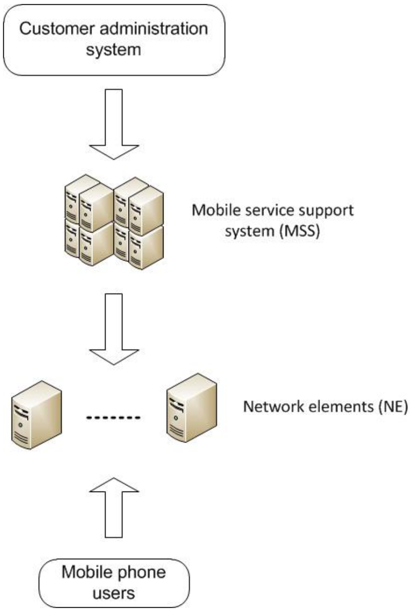

Customer administration system
Mobile service support system (MSS)
Network elements (NE)
Mobile phone users

Figure 1.4 Mobile service support system (MSS).

transactions sent to that NE need to be rolled back manually to prevent inconsistency in the databases. This roll-back process requires manual work which is costly for the operators in terms of time and expenditure.

In order to avoid such overload situations, traffic monitoring and admission control are vital. In cooperation with Ericsson AB, we have identified and addressed several control challenges for the MSS system. Performance models and control designs are based on the response time of queries sent to the NEs as this metric is easily measurable without requiring a change of protocols and systems and also because it well represents the dynamics of the load of the NEs, i.e., a highly loaded NE will have a long response time and a lightly loaded NE will have a short response time accordingly. The presented performance models in [Kihl et al., 2012] and [Amani et al., 2012], namely the M/M/m/n-LDS model is shown to be a suitable candidate to represent the dynamics of the response times of queries sent to a NE. In Figure 1.5, the known and unknown load sources to a NE with their respective mean arrival rates of queries λ and λ_u are illustrated.

By monitoring the response time of the requests sent from the MSS to the NE, we are able to identify the overload situation. When the average response time of the requests sent to a NE reaches a threshold, the MSS can classify the NE as being overloaded and thus take action to reduce the mean arrival rate of the requests sent to that particular NE. This reduced arrival rate is denoted by λ_c.

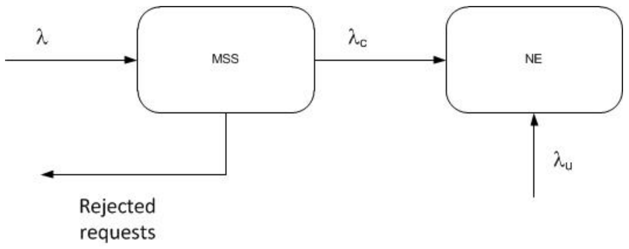


Figure 1.5 Load at the NEs.

The MSS includes a control system which ensures that the mean response time of the queries sent to a NE is kept below an acceptable level, thus also keeping the load on the NE equally acceptable. The control system includes a controller and a gate. The control system is depicted in Figure 1.6.

A response time reference value, T_{ref}, and response time measurements are used by the controller to determine an acceptable workload offered to the database server. The admitted workload is defined by the normalized mean admittance ratio of the requests , λ_A which is defined as the mean arrival rate of the admitted requests divided by the maximum mean arrival rate of the requests. Robust performance of the controller in the presence of fluctuations in the average arrival rate of the queries sent to the database is desired. The gate ensures that ratio λ_A of all arriving queries is admitted to the database. In the MSS, this is handled by delaying the transmission of the requests to the NEs. Since the communication with the customer administration system is synchronous, adding delays to the responses will lower the arrival rate of requests. Details of the controller design are presented in [Amani et al., 2012].

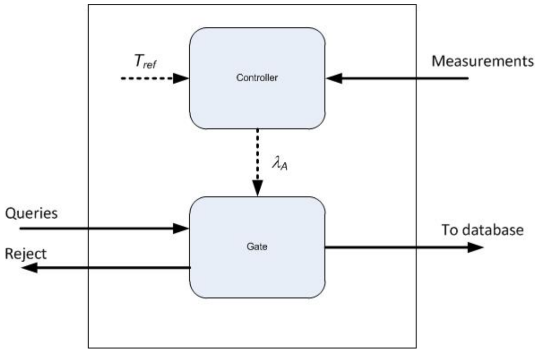


Figure 1.6 Control system.

1.5 NARX-based multi-step-ahead predictor

NARX Neural Network

Recurrent neural networks have been widely used for modeling of nonlinear dynamical systems [Haykin, 1998; Ljung, 1999]. Among various types of the recurrent neural networks such as distributed time delay neural networks (TDNN) [Haykin, 1998], layer recurrent networks [Haykin, 1998] and NARX [Haykin, 1998], the latest is of great interest in input-output modeling of nonlinear dynamical systems and time series prediction [Siegelmann et al., 1997; Lin et al., 1996; Xie et al., 2009; Menezes and Barreto, 2006; Parlos et al., 2000].

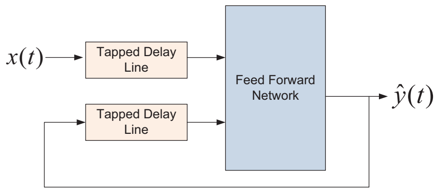
NARX is a dynamical recurrent neural network based on the linear ARX model. The next value of the dependent output signal $y(t)$ is regressed over the latest n_x values of the independent input signal and n_y values of the dependent output signal. n_x and n_y respectively represent the dynamical order of the inputs and outputs of the NARX. A mathematical description of the NARX model is summarized in (1.1) in which f is a non-linear function.

$$y(t) = f(y(t-1), y(t-2), \dots, y(t-n_y), x(t-1), x(t-2), \dots, x(t-n_x)) \tag{1.1}$$

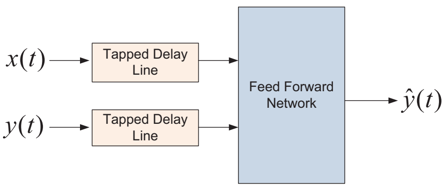
A NARX neural network can be implemented in two set-ups, namely parallel and series-parallel architectures. These are depicted in Figure 1.7.

This network consists of three main layers, i.e., the input layer, hidden layer and output layer. The input layer consists of the current and previous inputs as well as previous outputs. These are fed into the hidden layer. This layer consists of one or several neurons resulting in a nonlinear mapping of an affine weighted combination of the values from the input layer. The output layer consists of an affine combination of the values from the hidden layer. In this network, the dynamical order of inputs and outputs and the number of neurons in each layer are pre-determined. Several methods for determination of these values are presented in [Haykin, 1998]. A suitable training algorithm and performance measure should also be chosen. Finally, the type of non-linear map needs to be defined.

Some pre- and post-processing on the input and target values should be performed in order to ensure a valid training [Haykin, 1998]. These processes



(a) Parallel



(b) Series-Parallel

Figure 1.7 NARX (a): parallel and (b): series-parallel architectures.

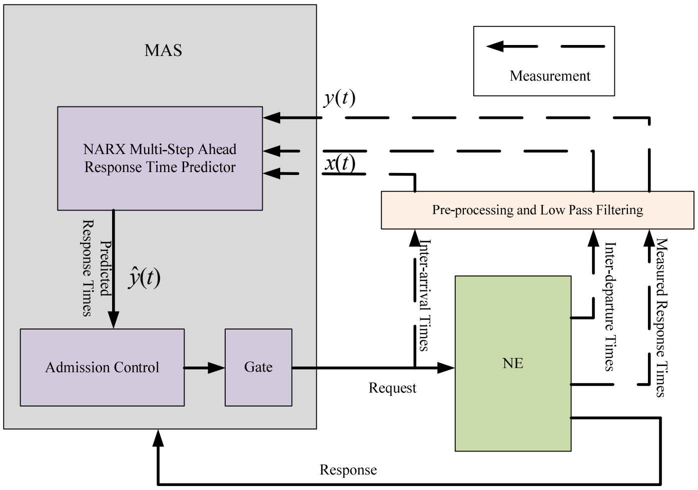


Figure 1.8 Multi-step-ahead response time predictor set-up.

include mapping of the input and target data to values in the range of $[-1,1]$, normalization of the inputs and targets to have zero mean and unity variance and removal of constant inputs and outputs and processing of unknown inputs.

NARX-based multi-step-ahead predictor set-up

Figure 1.8 depicts the structure of the multi-step-ahead response time predictor which was applied in [Amani et al., 2011a] to a single server system and in [Amani et al., 2011b] to a database server as the NE. Hereby we predict the response time of a request sent to a NE from the management server (MAS) by means of three measured time values, specifically the inter-arrival, inter-departure and response times of the requests sent to NE from the MAS.

In this setup, the input vector of the NARX predictor includes current inter-arrival and inter-departure times of the requests sent to the NE by the MAS. The output of the NARX predictor is the predicted response time in some steps ahead. The measured response times are required for training and evaluation of the NARX predictor and thus are fed back to the input layer of the proposed predictor. The measured data is divided into training, evaluation and test data sets. The prediction horizon m is defined as the time shift between corresponding inputs and output values so that the current input is used for prediction of the output in m time steps into the future. Details of the NARX predictor setup and the test beds used for its performance evaluation are presented in [Amani et al., 2011a] and [Amani et al., 2011b].

References

Amani, P., B. Aspernäs, K. J. Åström, M. Dellkrantz, M. Kihl, G. Radu, A. Robertsson, and A. Torstensson (2012). "Application of control theory to a commercial mobile service support system". *International Journal on Advances in Telecommunications Volume 5, Number 3 & 4, 2012*.

Amani, P., M. Kihl, and A. Robertsson (2011a). "Multi-step ahead response time prediction for single server queuing systems". In: *Proceedings of the 16th IEEE Symposium on Computers and Communications (ISCC2011)*. IEEE, pp. 950–955.

Amani, P., M. Kihl, and A. Robertsson (2011b). "NARX-based multi-step ahead response time prediction for database servers". In: *Proceedings of the 2011 11th International Conference on Intelligent Systems Design and Applications*. IEEE, pp. 813–818.

Armero, C. (1994). "Bayesian inference in Markovian queues". *Queueing Systems* **15**:1, pp. 419–426.

Barry, D. K. (2003). *Web services, service-oriented architectures, and cloud computing*. Morgan Kaufmann.

Basawa, I. V., U. N. Bhat, and R. Lund (1996). "Maximum likelihood estimation for single server queues from waiting time data". *Queueing systems* **24**:1-4, pp. 155–167.

Cao, J., M. Andersson, C. Nyberg, and M. Kihl (2003). "Web server performance modeling using an M/G/1/K*PS queue". In: *Proceedings of the 10th International Conference on Telecommunications (ICT 2003)*. Vol. 2. IEEE, pp. 1501–1506.

Choudhury, A. and A. C. Borthakur (2008). "Bayesian inference and prediction in the single server Markovian queue". *Metrika* **67**:3, pp. 371–383.

Clarke, A. B. et al. (1957). "Maximum likelihood estimates in a simple queue". *The Annals of Mathematical Statistics* **28**:4, pp. 1036–1040.

Curiel, M. and R. Puigjaner (2001). "Using load dependent servers to reduce the complexity of large client-server simulation models". In: *Performance Engineering*. Springer, pp. 131–147.

Ganapathi, A., Y. Chen, A. Fox, R. Katz, and D. Patterson (2010). "Statistics-driven workload modeling for the cloud". In: *Proceedings of the 2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010)*. IEEE, pp. 87–92.

Ganapathi, A., H. Kuno, U. Dayal, J. L. Wiener, A. Fox, M. Jordan, and D. Patterson (2009). "Predicting multiple metrics for queries: better decisions enabled by machine learning". In: *Proceedings of the IEEE 25th International Conference on Data Engineering (ICDE 2009)*. IEEE, pp. 592–603.

Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*. 2nd. Prentice Hall PTR.

Kihl, M., P. Amani, A. Robertsson, G. Radu, M. Dellkrantz, and B. Aspernäs (2012). "Performance modeling of database servers in a telecommunication service management system". In: *IARIA 7th International Conference on Digital Telecommunications (ICDT)*.

Kihl, M., A. Robertsson, and B. Wittenmark (2003). "Analysis of admission control mechanisms using non-linear control theory". In: *Proceedings of the 8th IEEE Symposium on Computers and Communications (ISCC 2003)*. IEEE, pp. 1306–1311.

Leung, K. K. (2002). "Load-dependent service queues with application to congestion control in broadband networks". *Performance Evaluation* **50**:1, pp. 27–40.

Lin, T., B. G. Horne, P. Tiño, and C. L. Giles (1996). "Learning long-term dependencies is not as difficult with NARX networks." *Advances in Neural Information Processing Systems*, pp. 577–583.

Ljung, L. (1999). *System Identification: Theory for the User*. Prentice Hall Information and System Sciences Series. Prentice Hall PTR.

Mathur, V. and V. Apte (2004). "A computational complexity-aware model for performance analysis of software servers". In: *Proceedings of the 12th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS 2004*. IEEE, pp. 537–544.

McGrath, M. F., D. Gross, and N. D. Singpurwalla (1987). "A subjective Bayesian approach to the theory of queues I-modeling". *Queueing Systems* **1**:4, pp. 317–333.

McGrath, M. F. and N. D. Singpurwalla (1987). "A subjective Bayesian approach to the theory of queues II—inference and information in M/M/1 queues". *Queueing Systems* **1**:4, pp. 335–353.

Menezes, J. M. P. and G. A. Barreto (2006). "A new look at nonlinear time series prediction with NARX recurrent neural network". In: *2006 9th Brazilian Symposium on Neural Networks (SBRN'06)*, pp. 160–165.

Parlos, A. G., O. T. Rais, and A. F. Atiya (2000). "Multi-step-ahead prediction using dynamic recurrent neural networks". *Neural networks* **13**:7, pp. 765–786.

Perros, H. G., Y. Dallery, and G. Pujolle (1992). "Analysis of a queueing network model with class dependent window flow control". In: *Proceedings of the 11th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 92*. IEEE, pp. 968–977.

Rak, M. and A. Sgueglia (2010). "Instantaneous load dependent servers (iLDS) model for web services". In: *Proceedings of the 4th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS-2010)*. IEEE, pp. 1075–1080.

Sheikh, M. B., U. F. Minhas, O. Z. Khan, A. Aboulnaga, P. Poupart, and D. J. Taylor (2011). “A Bayesian approach to online performance modeling for database appliances using Gaussian models”. In: *Proceedings of the 8th ACM International Conference on Autonomic Computing*. ACM, pp. 121–130.

Siegelmann, H. T., B. G. Horne, and C. L. Giles (1997). “Computational capabilities of recurrent NARX neural networks”. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **27**:2, pp. 208–215.

Tomov, N., E. Dempster, M. H. Williams, A. Burger, H. Taylor, P. J. King, and P. Broughton (2004). “Analytical response time estimation in parallel relational database systems”. *Parallel Computing* **30**:2, pp. 249–283.

Tozer, S., T. Brecht, and A. Aboulnaga (2010). “Q-cop: avoiding bad query mixes to minimize client timeouts under heavy loads”. In: *Proceedings of the IEEE 26th International Conference on Data Engineering (ICDE 2010)*. IEEE, pp. 397–408.

Tran, V. G., V. Debusschere, and S. Bacha (2013). “Neural networks for web server workload forecasting”. In: *2013 IEEE International Conference on Industrial Technology (ICIT)*, pp. 1152–1156.

Watson, B. J., M. Marwah, D. Gmach, Y. Chen, M. Arlitt, and Z. Wang (2010). “Probabilistic performance modeling of virtualized resource allocation”. In: *Proceedings of the 7th International Conference on Autonomic Computing*. ACM, pp. 99–108.

Xie, H., H. Tang, and Y.-H. Liao (2009). “Time series prediction based on NARX neural networks: an advanced approach”. In: *Proceedings of the 2009 International Conference on Machine Learning and Cybernetics*. Vol. 3. IEEE, pp. 1275–1279.

Zhang, Q., L. Cherkasova, and E. Smirni (2007). “A regression-based analytic model for dynamic resource provisioning of multi-tier applications”. In: *Proceedings of the 4th International Conference on Autonomic Computing (ICAC’07)*. IEEE, pp. 27–27.

Zheng, S. and A. F. Seila (2000). “Some well-behaved estimators for the M/M/1 queue”. *Operations Research Letters* **26**:5, pp. 231–235.

2

On centralized and decentralized content replication algorithms in content delivery networks

2.1 Introduction

The main trend in Internet usage covers content generation, distribution and sharing. Among the vast variety of content types being generated, video is by far in the lead, expected to correspond to 79% of the whole Internet traffic in 2016 [Wang et al., 2015]. In [Cisco, 2016] it is projected by Cisco that globally, IP video traffic will be 82 percent of all consumer Internet traffic by 2020.

Content delivery networks(CDNs) are the main medium used for efficient and reliable distribution of contents to the end users on the Internet. A CDN consists of a distributed system of servers which are geographically distributed around the globe.

CDN providers not only control the placement of content in the distributed system of servers located in different geographical locations but also decide on which server should serve a client's request. These two functions of CDN providers are called content replication and request routing in the literature.

Content replication and request routing algorithms can be implemented in a centralized approach or a distributed approach. Centralized approaches which are mainly implemented in commercial CDNs have the requirement of a global knowledge of the network architecture and related parameters. On the other hand, distributed approaches are presented in research CDNs and require only local knowledge of network architecture and related parameters.

CDNs are developed in-house by internet giants such as Google and Microsoft. Some other companies such as Akamai and Limelight have developed designated

CDN solutions and serve the content providers in the Internet. Some CDN providers have used cloud based technologies to provide a vast variety of content generators with cheap and pay as you go solutions which are called cloud CDNs in the literature. Finally some Telecom operators have deployed their own CDNs to have more control on delivery of content to their customers. These solutions are called Telecom operator's content delivery networks (Telco-CDNs) in the literature [Anjum et al., 2017].

2.2 Content replication in CDNs

In order to study the content replication problem in a CDN we have considered two layers of content servers. The content servers in the commercial CDNs are herein called main content servers (MCSs). It is assumed that a unique copy of each piece of content is located in the MCSs, thus MCSs cover the space of whole available contents for the content consumers.

The second layer of content servers are considered to be located in Internet service providers (ISPs). ISPs often own several regional content servers (RCSs) as part of a solution called Telco-CDNs [Li and Simon, 2013].

One of the main key performance indicators (KPIs) that can quantify quality of experience (QoE) of users of video content, is content retrieval latency. In order to minimize the content retrieval latency, contents should be disseminated towards consumers by replicating them in RCSs based on their popularity in the domain of each RCS.

Content replication in the RCSs can be performed in several ways. In a traditional paradigm, each RCS autonomously and independently decides which content to replicate and which content to discard when a new content is requested by end users while there is not sufficient storage available in the RCS for replicating it. Present web-caching architectures such as Least Recently Used (LRU) and Least Frequently Used (LFU) are built based on this paradigm [Androutsellis-Theotokis and Spinellis, 2004]. Despite its architectural simplicity, this paradigm has some drawbacks in terms of non-optimal storage usage due to the existence of redundant contents in RCSs. Even RCSs in the domain of one service provider are oblivious to the replicated contents of each other.

Distributed content replication is an alternative to the independent replication mechanism described above in which RCSs participate in a distributed content replication process. Upon arrival of a request for a content which is not already locally replicated, an RCS should decide to fetch the content from a neighboring RCS which has already replicated this content or from the MCS and whether to replicate it locally or not. The algorithms for distributed replication of contents can be divided into two categories, namely selfish and cooperative replication. In a selfish replication system, each RCS seeks replication strategies that maximize its own pay-off. On the other hand, in a cooperative replication, RCSs seek replication strategies that maximize the social pay-off [Borst et al., 2010]. The choice of selfish or cooperative strategies depends mainly on the business relations among the involved parties.

Content valuation is a vital part of any selfish or cooperative content replication algorithm through which decisions for local replication of a content is made in RCSs. The major notion of content valuation is content popularity which is used by almost all state of the art content replication algorithms. The popularity of a content has been addressed through various definitions such as the recency of usage of content in LRU and the frequency of usage in LFU [Mohan, 2001]. The demand rate of a content is another notion of content popularity which is used in most of the recent distributed content replication strategies.

An RCS can decide either to replicate a content completely or partially based on the replication strategy. Many content replication strategies implement the former option [Laoutaris et al., 2005; Borst et al., 2010]. For partial replication of contents, the division of contents into chunks has been studied in [Bo et al., 2013]. A chunk is considered as a fixed-size piece of content where the chunk size and hence the number of chunks in a content is determined with respect to that fixed-size-based chunk. The introduction of chunks into content replication has various advantages. Chunk-based replication enables RCSs to locally replicate initial chunks of the most popular contents with higher probability and providing the users lower access delay for these chunks. While these chunks are consumed (in the case of videos being played), the rest of the chunks could be fetched and consumed afterwards. Hereby, the QoE for the end user is improved by lowering the content retrieval latency. Another advantage is that by improving the granularity of the storage space, we can replicate the important chunks (initial chunks) of the most popular contents more efficiently compared to replication of the whole content in terms of storage utilization. In Delay Tolerant Networks (DTNs), chunk-based replication can ease opportunistic content retrieval. If a user did not fetch all chunks of a content in a meeting incidence with the current Access Point (AP), he will still have the chance to get the rest of the chunks from the next AP as he moves around the network.

In [Amani et al., 2013], we addressed the problem of optimal content replication in RCSs in the form of a minimization integer programming problem. The cost function for this problem is the accumulated weighted content retrieval latency among RCSs and also between the RCSs and the MCS. Various practical constraints such as a limited total content replication budget in each SP, limited storage size and downlink bandwidth of each RCS have been considered in the minimization problem to model the optimal content replication problem more realistically. A solution to this centralized minimization problem will provide the performance bound for any decentralized content replication with similar formulation. The topology of this scenario is depicted in Fig 2.1.

As this integer programming problem is NP-hard [Li and Simon, 2013], light weight centralized and distributed algorithms to approximate the solution of this problem are of practical interest.

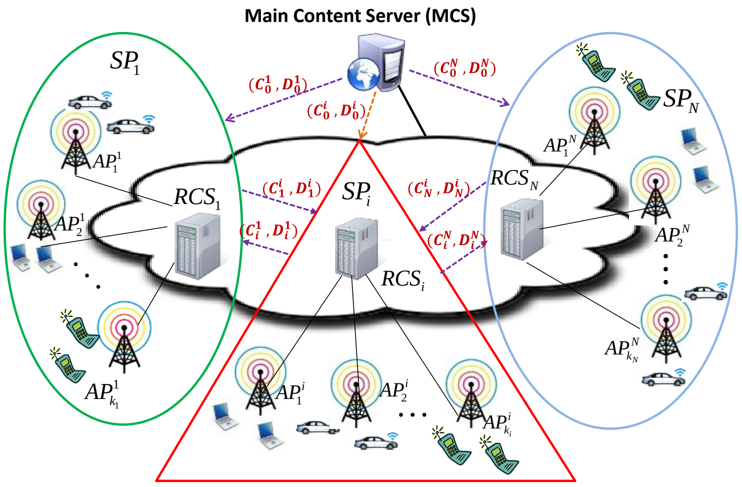


Figure 2.1 Multiple service providers, optimization for waited delay and bounded cost.

In [Liu et al., 2016], the authors have extended the presented problem formulation by assuming that the APs are equipped with a type of storage called storage helpers. Afterwards, they presented an integer programming formulation for the content placement problem. Furthermore, they have proposed some heuristics to estimate the optimal solution of the content placement problem.

We have extended the research results presented in [Amani et al., 2013] further in [Amani et al., 2015] where two popularity-based cooperative content replication and request routing algorithms have been proposed which minimize the content access delay in a general CDN topology. The proposed algorithms are examined under broad ranges of cache sizes and content popularity parameters via simulation. The results show that the proposed methods outperform similar cooperative recency-based methods and demonstrate close to optimal performance in representative scenarios of real world situations.

2.3 Future works

Two possible extensions of the research presented in [Amani et al., 2015] are considering the effect of server loads in content access delay as well as studying the power consumption of the proposed algorithms and developing joint power and content access delay optimization algorithms for content replication and request routing in a general CDN topology.

References

Amani, P., S. Bastani, and B. Landfeldt (2013). “Optimal content retrieval latency for chunk based cooperative content replication in delay tolerant networks”. In: *Proceedings of the 9th Swedish National Computer Networking Workshop*. SNCNW.

Amani, P., S. Bastani, and B. Landfeldt (2015). “Towards optimal content replication and request routing in content delivery networks”. In: *Proceedings of the 2015 IEEE International Conference on Communications (ICC 2015)*. IEEE, pp. 5733–5739.

Androutsellis-Theotokis, S. and D. Spinellis (2004). “A survey of peer-to-peer content distribution technologies”. *ACM Computing Surveys (CSUR)* **36**:4, pp. 335–371.

Anjum, N., D. Karamshuk, M. Shikh-Bahaei, and N. Sastry (2017). “Survey on peer-assisted content delivery networks”. *Computer Networks* **116**, pp. 79–95.

Bo, C., Z. F. Li, and W. Can (2013). “Research on chunking algorithms of data de-duplication”. In: *Proceedings of the 2012 International Conference on Communication, Electronics and Automation Engineering*. Springer, pp. 1019–1025.

Borst, S., V. Gupta, and A. Walid (2010). “Distributed caching algorithms for content distribution networks”. In: *Proceedings of the 2010 IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, pp. 1–9.

Cisco (2016). “Cisco VNI forecast and methodology, 2015-2020”. In: *Cisco Visual Networking Index (Cisco VNI)*. Cisco.

Laoutaris, N., V. Zissimopoulos, and I. Stavrakakis (2005). “On the optimization of storage capacity allocation for content distribution”. *Computer Networks* **47**:3, pp. 409–428.

Li, Z. and G. Simon (2013). “In a Telco-CDN, pushing content makes sense”. *IEEE Transactions on Network and Service Management* **10**:3, pp. 300–311.

Liu, J., Q. Yang, and G. Simon (2016). “Optimal and practical algorithms for implementing wireless CDN based on base stations”. In: *Proceedings of the 2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*. IEEE, pp. 1–5.

Mohan, C. (2001). “Caching technologies for web applications”. In: *Proceedings of the 27th International Conference on Very Large Data Bases*. VLDB ’01. Morgan Kaufmann Publishers Inc., pp. 726–726.

Wang, M., P. P. Jayaraman, R. Ranjan, K. Mitra, M. Zhang, E. Li, S. Khan, M. Pathan, and D. Georgeakopoulos (2015). "An overview of cloud based content delivery networks: research dimensions and state of the art". In: *Transactions on Large-Scale Data-and Knowledge-Centered Systems XX*. Springer, pp. 131–158.

3

Publications and Contributions

This thesis is based on four papers which summarizes the result of our research in two tracks . The contents of the two research tracks and contributions of each paper are described as follows.

Track 1—Modelling, prediction and control for multi-tier computing systems

Paper I

Amani, P., M. Kihl, and A. Robertsson (2011). "Multi-step ahead response time prediction for single server queuing systems". In: *Proceedings of the 16th IEEE Symposium on Computers and Communications (ISCC2011)*. IEEE, pp. 950–955.

Multi-step ahead response time prediction of CPU constrained computing systems is vital for admission control, overload protection and optimization of resource allocation in these systems.

CPU-constrained computing systems such as web servers can be modeled as single server queuing systems. These systems are stochastic and non-linear. Thus, a well-designed non-linear prediction scheme would be able to represent the dynamics of such a system much better than a linear scheme. A NARX-based multi-step-ahead response time predictor has been developed. The proposed estimator offers many promising characteristics making it a viable candidate for being implemented in admission control products for CPU constrained computing systems. It has a simple structure, is non-linear, supports multi-step-ahead prediction, and works very well under time variant and non-stationary scenarios, such as single server queuing systems under a time varying mean arrival rate. The performance of the proposed predictor is evaluated through simulation.

Simulations show that the proposed predictor is able to predict the response times of single server queuing systems in multi-step-ahead with very good precision represented by very small mean absolute and mean squared prediction errors.

I am the main contributor to this paper, and I was involved in all parts of the scientific work and writing of the paper.

Paper II

Amani, P., M. Kihl, and A. Robertsson (2011). “NARX-based multi-step ahead response time prediction for database servers”. In: *Proceedings of the 2011 11th International Conference on Intelligent Systems Design and Applications*. IEEE, pp. 813–818.

Advanced telecommunication applications are often based on a *multi-tier architecture*, with application servers and database servers. With a rapidly increasing development of cloud computing and data centers, characterizations of the dynamics for database servers during changing workloads will be a key factor for analysis and performance improvements in these applications. We propose a multi-step-ahead response time predictor for database queries based on a NARX neural network. The estimator shows many promising characteristics which make it a viable candidate for being implemented in admission control products for database servers. Performance of the proposed predictor is evaluated through experiments on a lab set-up with a MySQL server.

I am the main contributor to this paper, and I was involved in all parts of the scientific work and writing of the paper.

Paper III

Amani, P., B. Aspernäs, K. J. Åström, M. Dellkrantz, M. Kihl, G. Radu, A. Robertsson, and A. Torstensson (2012). “Application of control theory to a commercial mobile service support system”. *International Journal on Advances in Telecommunications Volume 5, Number 3 & 4, 2012*.

The Mobile Service Support system (MSS), which Ericsson AB is developing, handles the set-up of new subscribers and services into a mobile network. Experience from deployed systems shows that traffic monitoring and control of the system will be crucial for handling overload situations that may occur during sudden traffic surges. In this paper, we identify and explore some important control challenges for this type of system. Furthermore, we present analysis and experiments showing some advantages of the proposed solutions. First, we develop a load-dependent server model for the Database system, which is validated in test-bed experiments. Subsequently, we propose a control design based on the model and a method for estimating response times and arrival rates of the queries sent to the Database. The main contribution of this paper is that we show how control theory methods and analysis can be used for commercial Telecom systems.

Parts of our results have been implemented in commercial products, validating the strength of our work. This paper is a summary of a long cooperation between researchers in the department of electrical and information technology (EIT) in Lund university and Ericsson in Karskrona.

I have contributed to the following parts of the scientific work and written the paper: calculations for the mean response time and mean number of customers for the M/M/m/n-LDS; parameter tuning for LDS models, setting up the experiments (There has been two parallel test beds one implemented by me and one by Manfred Dellkrantz) and fitting the model to the experimental data. I also contributed to the design of the linear LAC and tuned the controller for a LDS model fitted to the data from the database server in Simulink and provided the parameters for testing in the lab set-up. In a nutshell, I have directly contributed to all parts of this scientific work except for the section on monitoring and estimation.

Track 2—On centralized and decentralized content replication in content delivery networks

Paper IV

Amani, P., S. Bastani, and B. Landfeldt (2013). “Optimal content retrieval latency for chunk based cooperative content replication in delay tolerant networks”. In: *Proceedings of the 9th Swedish National Computer Networking Workshop*. SNCNW.

Modern content distribution networks face an increasing multitude of content generators. In order to reach the minimal content retrieval latency in content distribution networks, content shall be disseminated towards consumers based on its popularity taken from the content distribution networks. This, combined with dividing media into chunks (heterogeneous valuation of information) and contact duration of the consumers with the access points in delay tolerant networks, led us to a novel system for content management in large scale distributed systems. In order to determine where to replicate content, we formulated the problem as an integer programming problem. The cost function of this minimization problem is the accumulated weighted communication delay among the content replication servers and also the main content server. Various practical constraints such as a limited total budget for content replication in each service provider, limited storage size and downlink bandwidth of the content replication servers are considered. A centralized solution to the problem is derived which yields the performance bound for any decentralized content replication strategy for the presented scenarios.

I am the main contributor to this paper and I was involved in all parts of the scientific work and writing of the paper.

Other publications not included in the thesis

Paper V

Kihl, M., P. Amani, A. Robertsson, G. Radu, M. Dellkrantz, and B. Aspernäs (2012). “Performance modeling of database servers in a telecommunication service management system”. In: *IARIA 7th International Conference on Digital Telecommunications (ICDT)*.

Paper VI

Amani, P., S. Bastani, and B. Landfeldt (2015). “Towards optimal content replication and request routing in content delivery networks”. In: *Proceedings of the 2015 IEEE International Conference on Communications (ICC 2015)*. IEEE, pp. 5733–5739.

Paper I

Paper I

Multi-step ahead response time prediction for single server queuing systems

Payam Amani   Maria Kihl   Anders Robertsson

Abstract

Multi-step ahead response time prediction of CPU constrained computing systems is vital for admission control, overload protection and optimization of resource allocation in these systems. CPU constrained computing systems such as web servers can be modeled as single server queuing systems. These systems are stochastic and non-linear. Thus, a well-designed non-linear prediction scheme would be able to represent the dynamics of such a system much better than a linear scheme.

A non-linear auto-regressive neural network with exogenous inputs based multi-step ahead response time predictor has been developed. The proposed estimator has many promising characteristics that make it a viable candidate for being implemented in admission control products for computing systems. It has a simple structure, is non-linear, supports multi-step ahead prediction, and works very well under time variant and non-stationary scenarios such as single server queuing systems under time varying mean arrival rate. Performance of the proposed predictor is evaluated through simulation. Simulations show that the proposed predictor is able to predict the response times of single server queuing systems in multi-step ahead with very good precision represented by very small mean absolute and mean squared prediction errors.

Originally published in the Proceedings of the 16th IEEE Symposium on Computers and Communications (ISCC)", Kerkyra (Corfu), Greece, pp.950-955, Jun. 2011. Reprinted with permission.

1. Introduction

Computing systems enable the Telecom operators to provide their customers with a vast variety of services which are aimed to meet their demands and desires. An operator usually uses a network of several such computing systems to facilitate providing the end users with an ever growing variety of services. Optimization of resource allocation in computing systems has attracted much interest in recent years as it directly relates to the performance of these systems.

Node elements (NEs) as building blocks of this network of computing systems, have a requirement for secure, reliable and real-time activation, modification and deactivation of both new and current customers or services. These tasks should be performed fast and in an automated manner. A resource access conflict exists among performing those tasks and providing current customers with their requested services in the network. This fact raises the necessity for a new enterprise provisioning system in the network which is hereby named as management system (MAS). The MAS is equipped with an admission control mechanism which enables it to avoid the resource access conflict by delaying sending of requests to a highly loaded NE and protecting it from becoming overloaded [Kihl et al., 2008; Chen et al., 2003; Liu et al., 2006]. This control mechanism usually includes a feed forward controller as it should predict the resource access conflict well before it happens and take action to avoid it. Therefore there is a requirement for a multi-step ahead state predictor for the NEs which precisely represent the dynamics of the NE in its whole operation range. NEs are desired to be loaded as much as possible close to their capacity meanwhile protected from becoming overloaded. One of the main performance measures of computing systems is the response time of the requests sent to them. Businesses and their customers like to minimize the system's response times while maximizing system utilization. Doing this, users will have a positive experience during delivery of services which would lead to increased customer retention and revenue.

It has been shown in [Cao et al., 2003] that CPU constrained computing systems such as web servers with dynamic content can be modeled as single server queuing systems. These are nonlinear stochastic systems. A nonlinear model is much more capable of representing the dynamics of a single server queuing system compared to a linear model [Kihl et al., 2003].

Many attempts to develop analytical estimators or predictors for single server queuing systems have been presented in the literature. Clarke, in his pioneer work published in 1957, presented maximum likelihood estimator (MLE)'s of arrival and service rates [Clarke et al., 1957]. Basawa *et. al* . in [Basawa et al., 1996] have presented a maximum likelihood estimator for single server queues from waiting time data. In [Zheng and Seila, 2000], Zheng and Seila have investigated some popular performance measures like waiting time and queue length under frequentist setup and showed their undesirable characteristics like nonexistence of expected value of the estimator and infinite mean-squared error of the estimator. Further, they proposed a set-up to fix that property. For the first time, McGrath *et. al* . in [McGrath et al., 1987; McGrath et al., 1987] have applied the concept of Bayesian statistical inference to the $M/M/1$ queuing system. Their work has been considerably extended in [Armero, 1994; Choudhury and Borthakur, 2008].

The above mentioned analytical approaches to the estimation of single server queuing systems have some unfavorable characteristics for overload protection admission control schemes. Firstly, all of the above mentioned estimation methods can only be applied to steady state and stationary scenarios. Secondly, mean service and arrival rates are assumed to be constant and time invariant. However, in the real world, there are many cases where we are interested in a state estimator that can be applied to a CPU constrained computing system with at least one time varying parameter. Time varying mean arrival rate can be a good example of these parameters. Finally it should be noted that none of the above mentioned methods support multi-step-ahead prediction.

The requirement for a non-linear multi-step ahead response time predictor that can work under stationary and steady state scenarios as well as time varying and non-stationary scenarios led us to a grey box approach to identification of single server queuing systems. By means of a non-linear auto-regressive network with exogenous inputs (NARX) neural network we have designed a predictor that covers all the above mentioned characteristics that the other methods lack and also is able to predict the response times of single server systems with very good precision represented by very small mean absolute and mean squared prediction errors.

This paper is structured as follows. System configuration containing the use case scenario, the NARX neural network and the predictor is investigated in section 2. Section 3 is dedicated to specifications of simulation environment and scenarios. Simulation results are summarized in section 4. Finally section 5 concludes the paper.

2. System Configuration

This section covers three sub-sections. In subsection 2.1 the pilot system for which a non-linear multi-step-ahead predictor is developed is introduced. Sub-section 2.2 is dedicated to the introduction of NARX recurrent neural networks. The proposed NARX multi-step ahead response time predictor is presented in sub-section 2.3.

2.1 Management System and Node Elements

Communication and computer networks are the media used by Telecom operators to inspire their subscribers with an ever growing variety of services. These networks are usually inter-connected and operators provide services to their customers via several of them. The MAS is responsible for real-time, secure and reliable activation, modification and deactivation of subscribers and services in an

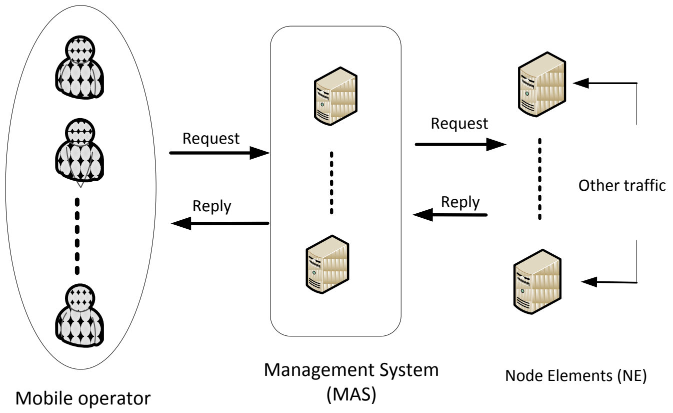


Figure 1. A generic distributed service management system.

automated manner. Such management system interacts with many NEs in the network and is usually implemented in distributed server clusters. Figure 1 depicts a generic distributed service management system.

The interactions between the MAS and the NEs should not lead the high loaded NEs to become overloaded. This brings the necessity of an admission control mechanism for overload protection of the NEs into picture. As a real network usually consists of various parts which are provided by several vendors with their own protocol implementations, the admission control mechanism should be implemented in the MAS. Also, it should be based on measurements which can be provided without a need for changing the current protocols and operating systems. A close review of the described set-up led us to figure out three measurements that have the above mentioned characteristics. These include inter-arrival times, inter-departure times and response times of the services sent to the node elements from the service management system. These measurements can be retrieved from the time tagged logs of requests traveling in the network. The response time of a service is defined as the duration of time that it spends from the moment that it leaves the MAS for the NE to the time that it leaves the NE. A high response time corresponds to a highly loaded NE and a low response time to a lightly loaded one. Thus response time can be used as an indicator for the NEs' internal state.

The control mechanism for overload protection of the NEs is in the form of admission control so that NEs' traffic from the subscribers is given higher priority compared to the traffic sent from the MAS to the NEs. In case that the NE is heavily loaded and tending to become overloaded, the MAS will back off sending more requests to the NE allowing it to process some of them and reduce its load. As the control action should take place well before an overload occurs in the network, the admission control scheme will consist of not only a feedback loop but also a feed

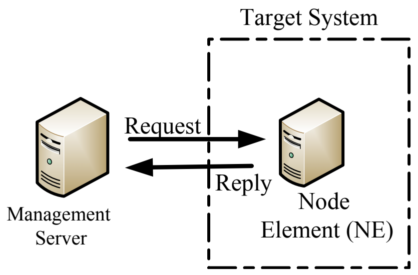


Figure 2. The pilot computing system configuration.

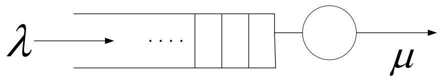


Figure 3. A single server queue with mean arrival rate λ and mean service rate μ.

forward part. The requirement for a feed forward controller raises the need for a multi-step ahead predictor for the NEs. In this paper we focus on interaction of one Management Server with one NE. Configuration of the computing system which is to be investigated in this paper is illustrated in Figure 2.

NEs can be modeled as single server queuing systems. In this paper the problem of non-linear multi-step ahead prediction of response times of single server queuing systems is investigated. Figure 3 illustrates a single server queuing system in which the distribution of the inter-arrival times and service times are general. The mean arrival rate and the mean service rates of the queuing system are denoted by λ and μ respectively.

2.2 NARX Neural Network

Recurrent neural networks have been widely used for modeling of nonlinear dynamical systems [Haykin, 1998; Ljung, 1999]. Among various types of the recurrent neural networks such as distributed time delay neural networks (TDNN) [Haykin, 1998], layer recurrent networks [Haykin, 1998] and NARX [Haykin, 1998], the latest is of great interest in input output modeling of nonlinear dynamical systems and time series prediction [Siegelmann et al., 1997; Lin et al., 1996; Xie et al., 2009; Menezes and Barreto, 2006; Parlos et al., 2000].

NARX is a dynamical recurrent neural network based on the linear ARX model. The next value of the dependent output signal $y(t)$ is regressed over the latest n_x values of the independent input signal and n_y values of the dependent output signal. n_x and n_y respectively represent the dynamical order of the inputs and outputs of the NARX. A mathematical description of the NARX model is summarized in (1)

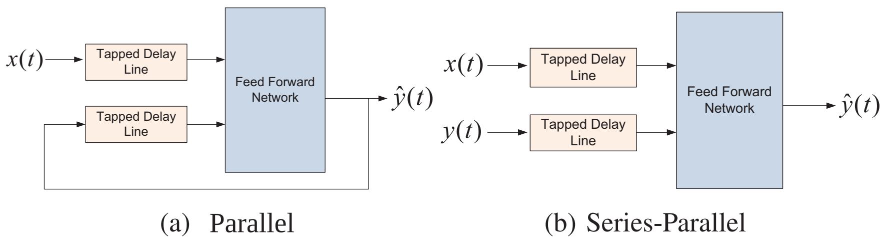


Figure 4. NARX (a): Parallel and (b): Series-Parallel Architectures.

in which f is a nonlinear function.

$$y(t) = f(y(t-1), y(t-2), \ldots, y(t-n_y), x(t-1), x(t-2), \ldots, x(t-n_x)) \quad (1)$$

A NARX neural network can be implemented in two set-ups namely parallel and series-parallel architectures. These are depicted in Figure 4. In this paper we have used the series-parallel architecture in which during the training period the actual values of the output are fed back to the neural network. This will improve the training precision.

This network consists of three main layers namely input layer, hidden layer and output layer. The input layer consists of the current and previous inputs and previous outputs. These are fed into the hidden layer. The hidden layer consists of one or several neurons resulting in a nonlinear mapping of affine weighted combination of the values from input layer. The output layer consists of an affine combination of the values from hidden layer. In this network the dynamical order of inputs and outputs and number of neurons in each layer are pre-determined. Several methods for determination of these values are presented in [Haykin, 1998]. A suitable training algorithm and performance measure also should be chosen. Finally, the type of the non-linear map needs to be defined.

Some pre and post processing on the input and target values should be performed in order to have a valid training [Haykin, 1998]. These processes include mapping of the input and target data to values in range of $[-1, 1]$, normalization of the inputs and targets to have zero mean and unity variance and removal of constant inputs and outputs and processing of unknown inputs. As these measurements are very noisy, after normalization we filter both input and target values with a designed Butterworth low pass filter. The bandwidth of the filter is chosen so it suppresses noise as much as possible while not affecting the characteristics of in band part of input and output data sets.

2.3 NARX Multi-Step Ahead Response Time Predictor Set-up

Our application requires the prediction of response times of the single server queuing system in some time steps into the future before the actual output measurements become available.

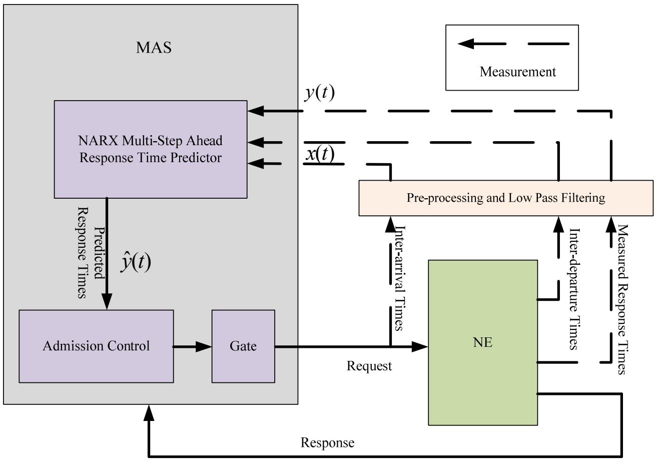


Figure 5. Multi-Step ahead Response Time Predictor Set-up.

A grey box identification approach was chosen to predict the response time of a service sent to a NE by the Management Server from three measured time values namely inter-arrival, inter-departure and response times of the services. The input vector consists of current inter-arrival times and inter-departure times as two inputs. Output of the neural predictor is the predicted response time. Measured response times are required for training and evaluation of the NARX multi-step ahead response time predictor and are fed back to the input layer of the proposed predictor.

Measured data is divided into training, evaluation and test data sets. Prediction horizon m is defined as the shift between corresponding inputs and output values so that current input is used for prediction of output in m time steps in the future. The proposed multi-step ahead response time predictor set-up is illustrated in Figure 5.

The overload protection admission controller uses a gate for controlling the flow of requests to the NE. The flow of requests from the MAS to the NE cannot get negative values as negative requests do not exist. Also the gate cannot send more requests than the available requests in the MAS. This imposes an input nonlinearity to the NE. We already know that NEs are nonlinear and stochastic computing systems. Thus, a NARX based predictor as a nonlinear predictor has a much better opportunity to grasp the dynamics of the response times of the NE compared to linear predictors [Kihl et al., 2003].

The off-line training process is described as follows. The NE is simulated under high load conditions. The acquired data is then divided to training, validation and test data sets. The NARX multi-step ahead response time predictor is trained using the train data and training is validated and its performance is tested using validation and test data sets. This trained predictor is used for all static and dynamic load conditions. Performance of the predictor is investigated in the following section.

3. Simulation environment and Scenarios

The simulation environment and the simulation scenarios are defined in the subsections 3.1 and 3.2 respectively.

3.1 Simulation environment

The NE and the Management Server are implemented in the MATLAB Simulink tool for event based simulations called SimEvents. Considering the pilot system configuration, the MAS is simulated by a request generator and the NE is simulated by a single server queuing system. The NARX multi-step ahead response time predictor is designed using MATLAB Neural Network Toolbox. The low pass filter is designed using the Filter Design Toolbox of MATLAB. All the mentioned tools are included in MATLAB R2010b.

3.2 Simulation Scenarios

In order to test the performance of the multi-step ahead NARX response time predictor we apply it to the $M/M/1$ queuing system [Kleinrock, 1975]. Also we consider four main test scenarios plus an extra scenario which deals with smoothly changing mean arrival rate.

Response Time Predictor's Parameters The NARX multi-step ahead response time predictor is configured as follows. The two dimensional input vector $x(t)$ consists of inter-arrival and inter departure times. The one dimensional output $y(t)$ represents the predicted response times.

The measured response times are fed back to the input layer for training. The tapped delay line in the hidden layer consists of two delays. Three neurons are considered in the hidden layer. The hidden layer neuron's activation function is considered as tangential sigmoid function $tansig$. The output layer consists of one neuron with activation function chosen as linear function $purelin$. This is summarized in Figure 6.

Several criteria such as sampling time, control structure and constraints affect the choice of the prediction horizon m. As in this paper we only focus on response time prediction of a single server system regardless of the control structure and other criteria that affect the choice of the prediction horizon, a suitable value for m cannot be decided here. However as we know that a multi-step ahead prediction is required we chose $m = 4$ to show that the NARX response time predictor is able to predict the response times of the single server queuing systems in several time steps into the future.

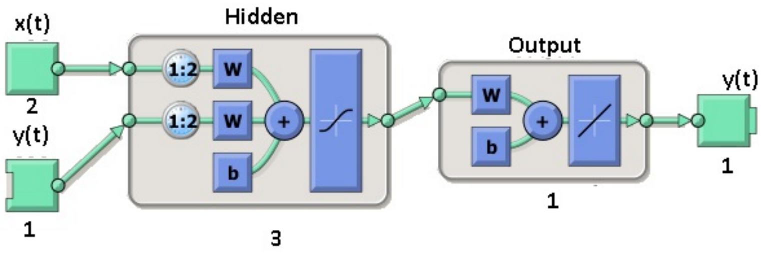


Figure 6. Neural predictor configuration and parameters.

The Levenberg-Marquardt algorithm [Haykin, 1998] is chosen as the training algorithm and the performance metric is set to mean squared error (MSE). The predictor is trained with the data from the high load scenario then tested over high load, low load and two varying load conditions.

Single Server Queuing System Parameters Our test set includes a $M/M/1$ queuing system with the following parameters. Mean service rate is set to 1 for all the scenarios. Simulation time is set to 20000. All time values are in simulated seconds. The static scenarios are designed for evaluation of performance of the NARX response time predictor in steady state under low $(\rho = 0.30)$ and high $(\rho = 0.95)$ load conditions. The dynamical scenarios are meant to evaluate performance of the NARX response time predictor with arrival rate changing from high to low load or vice versa with a step function at time 5000. The extra simulation scenario covers time variant mean arrival rate as a saturated ramp function starting from low$(\rho = 0.30)$ load at start time and saturating at high $(\rho = 0.95)$ load. Static Scenarios:

- S1: Mean arrival rate is set to 0.95 to simulate a high $(\rho = 0.95)$ load scenario. The NARX multi-step ahead response time predictor is then trained, validated and tested using train, validation and test data sets.
- S2: Mean arrival rate is set to 0.3 to simulate a low $(\rho = 0.3)$ load condition. Performance of the NARX multi-step ahead response time predictor is tested using the acquired data.

Dynamic Scenarios:

- S3: Mean arrival rate is a step, denoted by $step1$, from 0.3 to 0.95 with the step time set to 5000. Performance of the NARX multi-step ahead response time predictor is tested using the acquired data.
- S4: Mean arrival rate is a step from 0.95 to 0.3 with the same step time as before shown as $step2$. Performance of the NARX multi-step ahead response time predictor is tested using the acquired data.

4. Simulation Results

Performance of the proposed predictor applied to the defined M/M/1 queuing system is summarized in Table 1. In this section MAE stands for mean absolute error and MSE stands for mean squared error. It should be considered that the data used in these simulations is normalized to its maximum value. This is the reason why the maximum value of the response times is equal to one.

As it can be seen from the results in TABLE 1, the multi-step ahead NARX response time predictor is well trained and shows a promising performance under S1 and S2 considering both MSE and MAE. This shows that the proposed response time predictor is able to accurately predict the response time of the described $M/M/1$ system in 4 steps ahead under static and steady state load conditions.

Performance of the response time predictor under dynamic load conditions especially its performance in transient load conditions is of interest in the following tests. Under S3 both MSE and MAE indicate very good performance of the predictor. Figure 7 depicts measured response time vs. estimated response time

Table 1. Performance (MAE and MSE) of NARX m step ahead response time predictor for $M/M/1$ queuing system in S1 to S4 scenarios with prediction horizon m set to 4.

Performance of the Proposed Predictor in Training Phase			
Scenario	Server load ρ	Performance Measure	Value
S1	$\rho = 0.95$	MSE	$5.1915e-10$
		MAE	0.00174
Performance of the Proposed Predictor in Testing Phase			
Scenario	Server load ρ	Performance Measure	Value
S1	$\rho = 0.95$	MSE	$1.3728e-9$
		MAE	0.002
S2	$\rho = 0.30$	MSE	$4.1312e-9$
		MAE	0.0163
S3	$\rho = step1$	MSE	$2.585e-9$
		MAE	0.002
S4	$\rho = step2$	MSE	$1.8481e-7$
		MAE	0.0037

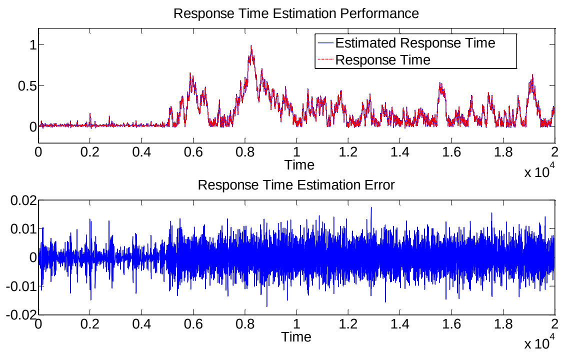


Figure 7. NARX *m* step-ahead response time prediction of the M/M/1 queuing system with mean arrival rate changing with a step from 0.3 to 0.95 at time 5000. The prediction horizon *m* is set to 4. (upper) Measured response times vs. estimated response times. (lower) Difference between measured and estimated response times.

under S3.

As it can be seen in the upper side of Figures 7, 8 and 9, the measured and predicted response time values are so close that it is really hard to distinguish between them. Thus an additional figure depicting the difference between the measured and predicted response time values or simply prediction error has been added to the lower part of Figures 7, 8 and 9.

Under S4 we observed that the prediction performance degrades compared to S3 but still holds a very small value for MSE and MAE. This performance degradation is most likely caused by the non-linearity of this queuing system. Figure 8 illustrates the transient behavior of the proposed NARX multi-step ahead response time predictor under S4.

The two last simulations showed that the proposed response time predictor is able to handle transient regimes very well. In the last two simulation scenarios, the time variation in the mean arrival rate was specified as step functions representing sudden change of levels. In the extra simulation scenario performance (MAE and MSE) of the NARX m step-ahead response time predictor applied to a $M/M/1$ queuing system under a smooth slowly varying mean arrival rate is studied. The prediction horizon m is set to 4. The mean arrival rate is changed with a saturated ramp which is depicted in Figure 9. The minimum and maximum values of the mean arrival rate are respectively 0.3 and 0.95. By studying Figure 9 closely we can conclude that the proposed predictor can handle smooth time variant mean arrival rates.

Performance of the proposed predictor under some more single server queuing system configurations such as $M/D/1$ and $D/M/1$ for the same sets of scenarios S1-S4 has been investigated via simulations and very small MAE and MSE for the prediction error has been confirmed. Due to the lack of space, we skipped presenting

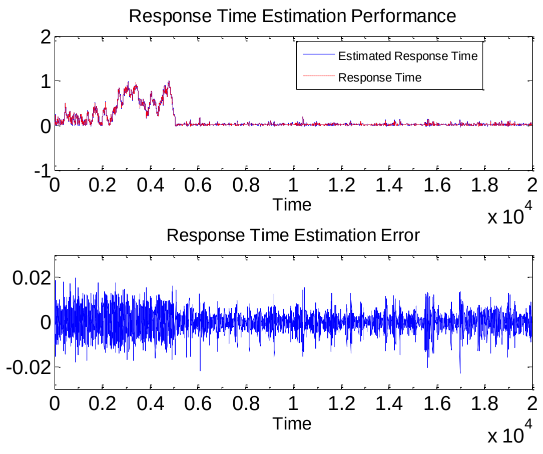


Figure 8. NARX *m* step-ahead response time prediction of the M/M/1 queuing system with mean arrival rate changing with a step from 0.95 to 0.3 at time 5000. The prediction horizon *m* is set to 4. (upper) Measured response times vs. estimated response times. (lower) Difference between measured and estimated response times.

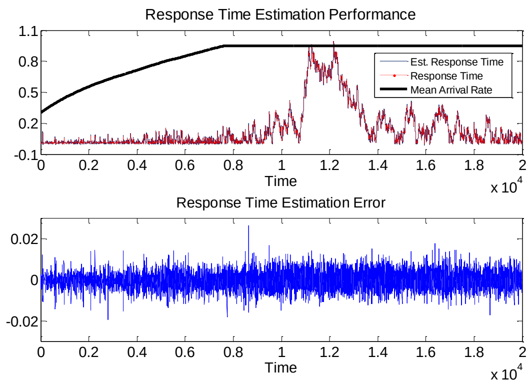


Figure 9. NARX *m* step-ahead response time prediction of the M/M/1 queuing system with mean arrival rate as a saturated ramp function.(upper) Measured response times vs. estimated response times and the saturated ramp. (lower) Difference between measured and estimated response times.

those results.

5. Conclusion

A multi-step ahead NARX response time predictor for single server queuing systems, which represents a CPU constrained computing system, has been proposed and its performance under several test scenarios has been studied. The proposed predictor benefits from several promising characteristics which turns it into a viable candidate for being implemented in admission control products for computing systems. It is non-linear, it supports multi-step ahead prediction, its structure is simple and its required measurements can be obtained without any requirement on changing communication protocols or operating systems. It has been shown that with being trained in only one high load scenario, it can predict the response times of a single server queuing system in multiple step ahead under high and low load steady state scenarios with a high accuracy. Very good performance of the proposed predictor under time variant and non-stationary scenarios has been confirmed by very small MAE and MSE of the response time prediction. It has also been shown that the proposed predictor is capable of accurate m step ahead response time prediction under time varying mean arrival rate scenarios. For the future work, the proposed predictor will be implemented in our web server lab and its capability of predicting the response times of the node elements will be studied.

Acknowledgment

Payam Amani, Maria Kihl and Anders Robertsson are members of Lund Center for Control of Complex Engineering Systems (LCCC), a Linnaeus Center at Lund University, funded by the Swedish Research Council. Maria Kihl is partly funded by the VINNMER program at the Swedish Governmental Agency for Innovation Systems (VINNOVA). Hereby, the first author appreciates the valuable comments of Dr. Kaan Bur and Mr. Jens Andersson from Department of Electrical and Information Technology of Lund University regarding this paper.

References

Armero, C. (1994). “Bayesian inference in Markovian queues”. *Queueing Systems* **15**:1, pp. 419–426.

Basawa, I. V., U. N. Bhat, and R. Lund (1996). “Maximum likelihood estimation for single server queues from waiting time data”. *Queueing systems* **24**:1-4, pp. 155–167.

Cao, J., M. Andersson, C. Nyberg, and M. Kihl (2003). “Web server performance modeling using an M/G/1/K*PS queue”. In: *Proceedings of the 10th International Conference on Telecommunications (ICT 2003)*. Vol. 2. IEEE, pp. 1501–1506.

Chen, X., H. Chen, and P. Mohapatra (2003). “Aces: an efficient admission control scheme for QoS-aware web servers”. *Computer Communications* **26**:14, pp. 1581–1593.

Choudhury, A. and A. C. Borthakur (2008). “Bayesian inference and prediction in the single server Markovian queue”. *Metrika* **67**:3, pp. 371–383.

Clarke, A. B. et al. (1957). “Maximum likelihood estimates in a simple queue”. *The Annals of Mathematical Statistics* **28**:4, pp. 1036–1040.

Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*. 2nd. Prentice Hall PTR.

Kihl, M., A. Robertsson, M. Andersson, and B. Wittenmark (2008). “Control-theoretic analysis of admission control mechanisms for web server systems”. *World Wide Web* **11**:1, pp. 93–116.

Kihl, M., A. Robertsson, and B. Wittenmark (2003). “Analysis of admission control mechanisms using non-linear control theory”. In: *Proceedings of the 8th IEEE Symposium on Computers and Communications (ISCC 2003)*. IEEE, pp. 1306–1311.

Kleinrock, L. (1975). *Queueing Systems Theory, Volume 1*. Wiley-Interscience.

Lin, T., B. G. Horne, P. Tiño, and C. L. Giles (1996). “Learning long-term dependencies is not as difficult with NARX networks.” *Advances in Neural Information Processing Systems*, pp. 577–583.

Liu, X., J. Heo, L. Sha, and X. Zhu (2006). “Adaptive control of multi-tiered web applications using queueing predictor”. In: *Proccedings of the 10th IEEE/IFIP Network Operations and Management Symposium. IEEE/IFIP NOMS 2006*. IEEE, pp. 106–114.

Ljung, L. (1999). *System Identification: Theory for the User*. Prentice Hall Information and System Sciences Series. Prentice Hall PTR.

McGrath, M. F., D. Gross, and N. D. Singpurwalla (1987). “A subjective Bayesian approach to the theory of queues I-modeling”. *Queueing Systems* **1**:4, pp. 317–333.

Menezes, J. M. P. and G. A. Barreto (2006). “A new look at nonlinear time series prediction with NARX recurrent neural network”. In: *2006 9th Brazilian Symposium on Neural Networks (SBRN’06)*, pp. 160–165.

Parlos, A. G., O. T. Rais, and A. F. Atiya (2000). “Multi-step-ahead prediction using dynamic recurrent neural networks”. *Neural networks* **13**:7, pp. 765–786.

Siegelmann, H. T., B. G. Horne, and C. L. Giles (1997). “Computational capabilities of recurrent NARX neural networks”. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **27**:2, pp. 208–215.

Xie, H., H. Tang, and Y.-H. Liao (2009). “Time series prediction based on NARX neural networks: an advanced approach”. In: *Proceedings of the 2009 International Conference on Machine Learning and Cybernetics*. Vol. 3. IEEE, pp. 1275–1279.

Zheng, S. and A. F. Seila (2000). “Some well-behaved estimators for the M/M/1 queue”. *Operations Research Letters* **26**:5, pp. 231–235.

Paper II

Paper II

NARX-based multi-step ahead response time prediction for database servers

Payam Amani   Maria Kihl   Anders Robertsson

Abstract

Advanced telecommunication applications are often based on a *multi-tier architecture*, with application servers and database servers. With a rapidly increasing development of cloud computing and data centers, characterizations of the dynamics for database servers during changing workloads will be a key factor for analysis and performance improvements in these applications. We propose a multi-step ahead response time predictor for database queries based on a non-linear auto-regressive neural network model with exogenous inputs. The estimator shows many promising characteristics which make it a viable candidate for being implemented in admission control products for database servers. Performance of the proposed predictor is evaluated through experiments on a lab setup with a MySQL-server.

Originally published in Proceedings of the 11th International Conference on Intelligent Systems Design and Applications, Cordoba, Spain, pp.813-818, Nov. 2011. Reprinted with permission.

1. Introduction

Telecom and Internet operators need to provide their customers with a vast variety of services which are aimed at meeting their demands and desires.

Multi-tier server clusters are used to host the service logic and user data. The optimization of resource allocation in server cluster systems has attracted much interest in recent years as it directly relates to the performance of these systems.

Database servers, as important entities of these server clusters require secure, reliable and real-time activation, modification and deactivation of both new and current customers or services. These tasks should be performed fast and in an automated manner. Therefore, control mechanisms can be introduced, which enable the system to avoid the resource access conflict and protect it from becoming overloaded [Kihl et al., 2008; Chen et al., 2003; Liu et al., 2006]. This control mechanism usually includes a feed-forward controller as it should predict the resource access conflict well before it happens and take action to avoid it. Therefore, there is a need for a multi-step-ahead state predictor, which fairly represents the dynamics of the database in its whole operation range and also provides high precision state representation of the system near the overload region.

Many attempts to develop response time estimators or predictors for database queries have been presented in the literature. They can be divided into two categories namely analytical and experiment-driven methods. Analytical models [Liu et al., 2006; Tomov et al., 2004; Watson et al., 2010], designed by experts, usually cover specific types of queries and database servers and assume some simplifying conditions. Thus they are not able to capture the complex dynamics of the database server. These models only support static cases and cannot be used in dynamic scenarios.

Several instances of experiment-driven methods have been recently presented in the literature. Ganapathi *et. al* in [Ganapathi et al., 2009] predict several metrics for database queries including the response time by means of Kernel Canonical Correlation Analysis (KCCA). Tozer in [Tozer et al., 2010] used a linear regression model for the response time in order to throttle long running queries. Sheikh *et. al* in [Sheikh et al., 2011] have presented a Bayesian approach for on-line performance modeling of database appliances using Gaussian models. Their proposed model has the possibility of adaptation to changes in workload and configuration. The smallest prediction error of their method is 14%.

In [Amani et al., 2011], we have presented a (non-linear auto-regressive neural network with exogenous inputs) NARX-based multi-step-ahead response time predictor for single server queuing systems. We have shown via simulations that the suggested response time predictor is capable of predicting the response time of the single server queuing systems in multiple steps ahead with very small mean squared errors and mean absolute prediction errors respectively under both static and dynamic workload scenarios without adapting the model parameters to the changes in the workload.

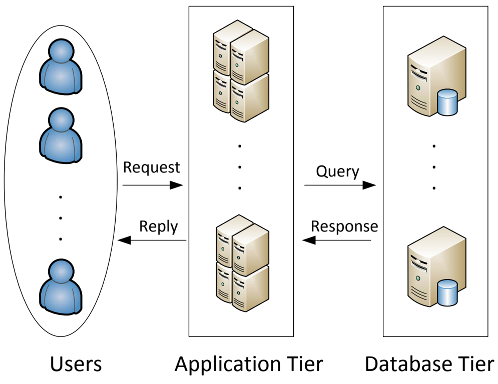


Figure 1. A generic multi-tier server cluster.

The requirement for a non-linear multi-step-ahead query response time predictor that can work under stationary and steady state scenarios, as well as under time varying and non-stationary scenarios led us to a gray box approach to identification of database servers. Thus we have used the same type of response time predictor for database servers. By means of a NARX neural network, we have designed a predictor that covers all the aforementioned characteristics and is also able to very well predict the response times of queries of database servers with very good precision represented by very small mean absolute, mean squared and sum of squared prediction errors.

This paper is structured as follows: system description, the NARX neural network and the predictor are investigated in section 2. Section 3 is dedicated to specifications of the experiment set-up and scenarios. Experimental results are summarized in section 4 and finally, section 5 concludes the paper.

2. System Configuration

This section covers three sub-sections. In subsection 2.1 the pilot system for which a non-linear multi-step-ahead predictor is developed is introduced. Sub-section 2.2 is dedicated to the introduction of NARX recurrent neural networks. The proposed NARX multi-step ahead response time predictor is presented in sub-section 2.3.

2.1 System description

Figure 1 depicts a generic multi-tier server cluster. The system can correspond to a broad range of Telecom and Internet applications, as data centers, cloud networking systems, web shops, enterprise systems or service management systems.

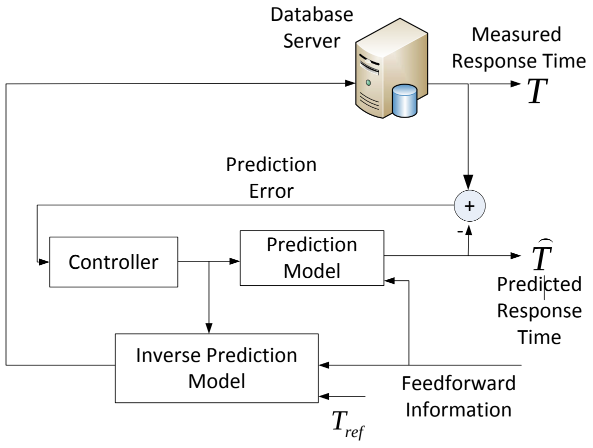


Figure 2. A sample scheme with combined feed-forward and feedback for control of database servers using response time prediction.

Here, we focus on the database tier. The interactions between the application tier and the database tier should not lead to the database servers to become overloaded. Therefore, control mechanisms should be implemented in the application servers that limit the traffic to the databases. The control system should be based on measurements which are available and which can be provided without a need for changing the current protocols and operating systems.

In this paper, we use inter-arrival, inter-departure and response times of the queries sent to the database servers from the application servers. These measurements can easily be retrieved from the time-tagged logs of the queries traveling in the system. A high response time (compared to the reference response time) corresponds to a highly loaded database and a low response time to a lightly loaded one. Thus, response time can be used as an indicator of the databases' internal state. In this paper, we focus on the interaction of one application server with one database server.

As the control action should take place well before an overload occurs in the system, the control scheme will consist of not only a feedback loop but also a feed-forward part. The requirement for a feed-forward controller raises the need for a multi-step-ahead query response time predictor for the databases. Figure 2 shows a controller scheme combining feedback and feed-forward, which requires response time prediction, presented by Kjær *et. al* in [Kjær et al., 2007].

Two main MySQL query types, Select and Update, are investigated in this paper. These requests have very different contributions to the response times of the queries sent to the database. Select queries are based on read actions while Update queries are based on write actions. Select queries are CPU restricted actions while

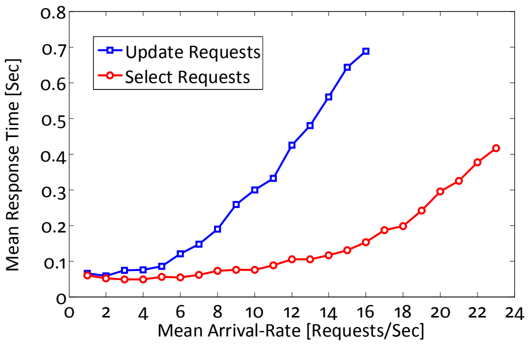


Figure 3. Mean response times of Select and Update queries sent to a 1E7 tuples relation in a scalable Wisconsin benchmark table in a MySQL database server vs. mean arrival rates of the queries.

Update queries are I/O restricted actions. As it can be seen in Figure 3, the non-linear behavior of these two types of queries are very different. Processing of an Update query is much more time consuming compared to a Select query.

2.2 NARX Neural Network

Recurrent neural networks have been widely used for modeling of nonlinear dynamical systems [Haykin, 1998; Ljung, 1999]. Among various types of the recurrent neural networks such as distributed time delay neural networks (TDNN) [Haykin, 1998], layer recurrent networks [Haykin, 1998] and NARX [Haykin, 1998], the latest is of great interest in input output modeling of nonlinear dynamical systems and time series prediction [Siegelmann et al., 1997; Lin et al., 1996; Xie et al., 2009; Menezes and Barreto, 2006; Parlos et al., 2000].

NARX is a dynamical recurrent neural network based on the linear ARX model. The next value of the dependent output signal $y(t)$ is regressed over the latest n_x values of the independent input signal and n_y values of the dependent output signal. n_x and n_y respectively represent the dynamical order of the inputs and outputs of the NARX. A mathematical description of the NARX model is summarized in (1) in which f is a non-linear function.

$$y(t) = f(y(t-1), y(t-2), \ldots, y(t-n_y), x(t-1), x(t-2), \ldots, x(t-n_x)) \quad (1)$$

This network consists of three main layers namely input layer, hidden layer, and output layer. The input layer consists of the current and previous inputs and outputs. These are fed into the hidden layer. The hidden layer consists of one or several neurons resulting in a nonlinear mapping of affine weighted combination of the values from the input layer. The output layer consists of an affine combination of the values from the hidden layer. In this network, the dynamical order of inputs and outputs and number of neurons in each layer are pre-determined. Several methods for determination of these values are presented in [Haykin, 1998]. A suitable training algorithm and performance measure should also be chosen. Finally, the type of the non-linear map needs to be defined.

Some pre- and post-processing on the input and target values should be performed in order to have a valid training set [Haykin, 1998]. These processes include mapping of the input and target data to values in the range of $[-1,1]$, normalization of the inputs and targets to have zero mean and unity variance and removal of constant inputs and outputs and processing of unknown inputs. As the measurements are very noisy, after normalization we filter both input and target values with a designed Butterworth low pass filter. The bandwidth of the filter is chosen so it suppresses noise as much as possible while not affecting the characteristics of in band part of input and output data sets.

2.3 NARX Multi-Step Ahead Response Time Predictor Set-up

Our application requires the prediction of response times of the queries sent to the database server in some time steps into the future, before they are processed in the database server. A gray box identification approach was chosen to predict the response times of such queries from three measured time values, namely inter-arrival, inter-departure, and response times of the queries. The predictor is designed by means of the Neural Networks Toolbox of MATLAB R2010b. The input vector consists of current inter-arrival times and inter-departure times as two inputs. Output of the neural predictor is the predicted response time. Measured response times are required for training and evaluation of the NARX multi-step ahead response time predictor and are fed back to the input layer of the proposed predictor. Measured data is divided into training, evaluation and test data sets. Prediction horizon m is defined as the shift between corresponding inputs and output values so that current input is used for prediction of output in m time steps in the future. The proposed multi-step ahead response time predictor is illustrated in Figure 4. The overload protection admission controller uses a gate for controlling the flow of queries to the database server. The flow of queries from the application server to the database server cannot get negative values as negative requests do not exist. Also, the gate cannot send more requests than the available requests in the application server. This imposes an input non-linearity to the database server. We already know that database servers are nonlinear and stochastic computing systems under high load conditions. Thus, a NARX based predictor as a nonlinear predictor has a much better opportunity to capture the dynamics of the response time of the database server compared to linear predictors [Kihl et al., 2003].

The off-line training process is described as follows: The database server is stressed under high load conditions. The acquired data is then divided to training, validation and test data sets. The NARX multi-step ahead response time predictor

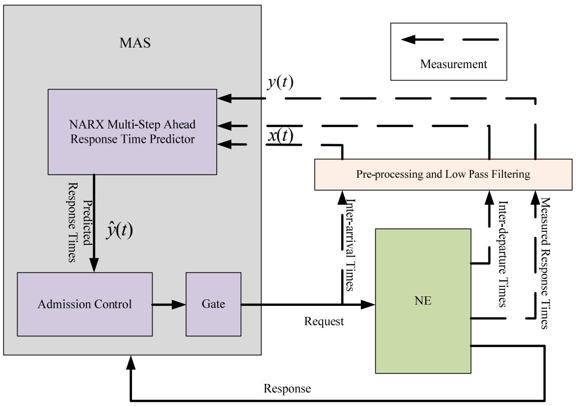


Figure 4. Multi-Step ahead Response Time Predictor in Admission Control Set-up.

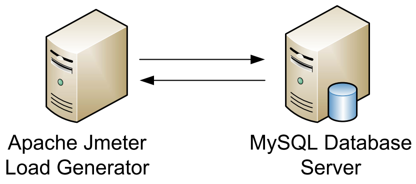


Figure 5. Database lab set-up.

is trained using the train data and training is validated and its performance is tested using validation and test data sets. This trained predictor is used for all static and dynamic load conditions containing various combinations of database queries. The performance of the predictor is investigated in the following sections.

3. Database server lab set-up

The database server lab consists of two main computers: one hosting the database server and one hosting the traffic generator which represents the application tier. One objective of this lab is to test the performance of the proposed response time predictor for I/O constrained systems such as database servers. The mentioned computers are connected via an Ethernet switch. This is depicted in Figure 5.

3.1 Hardware and Software

The database server is a Dell precision workstation 340 computer with an Intel Pentium 4 CPU running at 1.7 GHz, 768 MB RAM and a 72GB Hard Disk hosting MySQL Server 5.1.4.1. It has Solaris 11 Express as its operating system.

The application server, in this case, is represented by a Dell precision workstation 340 computer with an Intel Pentium 4 CPU running at 1.7 GHz, 512 MB RAM and a 36 GB Hard Disk hosting Apache Jmeter 2.4 as load generator sending queries to the database server. It has UBUNTU 10.04 LTS as its operating system.

3.2 Apache Jmeter

Apache Jmeter [Halili, 2008] is a Java-based load generator with support for plugins that can be used to stress test various types of servers such as web servers, mail servers and database servers. Support for database queries is provided via Java database connectivity, JDBC. Various load distributions can be generated by means of timer plugins. A timer plugin for generation of Poisson distributed database queries via JDBC has been used [Kihl et al., 2011]. Apache Jmeter generates the load to the supported servers by means of blocking I/O, and a fixed number of threads. This imposes an upper limit for the maximum number of concurrent requests which is equal to the number of Jmeter's threads. During the time intervals that all the threads are busy, no new queries can be sent out before the processing of an old query is finished. This will change the distribution of the load to the database server. Thus, all experiments that use all of the threads at the same time shall be invalidated.

3.3 Tracing and D-Trace

Tracing in software engineering terminology is a specialized use of logging for recording information about execution of an application. Dynamic Tracing, D-Trace [Gregg and Mauro, 2011], is a detailed dynamic tracing tool introduced by Oracle for Unix-like operating systems. D-Trace has the option to provide not only information regarding the whole application like CPU and memory demand, but also information regarding each function in the application. D-trace scripts are written in a C based programming language which is equipped with variables and functions required for tracing, called D. D programs include a set of one or more probes and each probe is associated with an action. When the condition of the probe is satisfied, the associated action is executed. We have used these probes to get exact time stamps of arrival of a Select or Update query to the MySQL database server and the time that the database server is done with processing of the mentioned queries. From these time stamps, we can calculate the inter-arrival and inter-departure times of the queries.

3.4 Structure of the Database

The database server has several relations all with the same structure from the Scalable Wisconsin Benchmark [DeWitt, 1993] with different number of tuples. Two types of MySQL queries which are most frequently used in the database servers, namely Select and Update, are taken into consideration in this paper. The structure of the queries are as follows:

Select queries:

```
SELECT unique2 from tenmil where unique1 equals ?;
```

Update queries:

```
UPDATE tenmil SET unique3=? where unique1=?;
```

In the above queries, *tenmil* is a ten million tuple relation from the Scalable Wisconsin Benchmark and ? represents a uniformly distributed random number between 0 and 1E7.

In order to test the performance of the multi-step ahead NARX response time predictor, we apply it to the described MySQL database server. Also, we consider 4 main test scenarios, consisting of two static and two dynamic scenarios.

3.5 Response Time Predictor's Parameters

The NARX multi-step ahead response time predictor is configured as follows: the two dimensional input vector $x(t)$ consists of inter-arrival and inter departure times. The one dimensional output $y(t)$ represents the predicted response times.

The measured response times are fed back to the input layer for training. The tapped delay line in the hidden layer consists of three delays. Three neurons are considered in the hidden layer. The hidden layer neuron's activation function is considered to be tangential sigmoid function $tansig$. The output layer consists of one neuron with activation function chosen as linear function $purelin$. Several criteria such as sampling time, control structure and constraints affect the choice of the prediction horizon m. Since support for multi-step ahead prediction is required, we chose $m = 4$ to show that the NARX response time predictor is able to predict the response times of the queries sent to the MySQL database server in several time steps into the future.

The Bayesian Regularization algorithm [Foresee and Hagan, 1997] is chosen as the training algorithm and the performance metric is set to the sum of squared errors (SSE). The predictor is first trained with the data from the high load scenario then tested over high load, low load and two varying load scenarios.

3.6 Database load and queries specifications

Our test set includes an Apache Jmeter load generator with 30 concurrent threads. Duration of each experiment is set to 600 seconds. The effective mean arrival rate for which all the threads are busy in case of Update requests corresponds to 16 requests per second and for Select queries corresponds to 23 requests per second. This defines the maximum effective mean arrival rates in case of each type of the

Table 1. Experiment Scenarios

	Scenario	Mean Arrival Rate [Req/Sec]			Predictor State
		Update	Select	Mixed	
Static	*S1*	15	22	15	Train,Test
	S2	5	7	5	Test
Dynamic	*S3*	*Step1*	*Step1*	*Step1*	Test
	S4	*Step2*	*Step2*	*Step2*	Test

queries. The types of queries in real world applications are usually mixed of both the mentioned query types. In order to represent a more realistic case we have also considered a mix of 75% Select and 25% Update queries. The maximum allowed mean arrival rate in order to keep the Poisson distribution of the arrivals in this case is equal to 16 requests per second.

The static scenarios are designed for evaluation of performance of the NARX response time predictor in steady state under low ($\rho \approx 0.30$) and high ($\rho \approx 0.95$) load conditions. By load, here we mean the ratio between the mean arrival rate and the maximum effective mean arrival rate. The dynamical scenarios are meant to evaluate performance of the NARX response time predictor with arrival rates changing with a step function at time 200 seconds from low to high load (*Step1*) or vice versa (*Step2*). These are summarized in Table 1.

4. Experimental Results

Performance of the proposed predictor applied to the MySQL database server is summarized in Table 2. In this section, MAE stands for mean absolute error, MSE for mean squared error and SSE stands for the sum of squared errors. It should be noted that the data used in these tests is normalized to its maximum value. This is the reason why the maximum value of the response times is equal to one.

As it can be seen from the results in Table 2, the multi-step ahead NARX response time predictor is well trained and shows a promising performance under *S1* and *S2* considering MSE, MAE and also SSE. This shows that the proposed response time predictor is able to accurately predict the response times of the queries sent to the described MySQL server in 4 steps ahead under static and steady state load conditions.

Looking at the presented experimental results in Table 2, one can observe a large difference between the performance of the proposed response time predictor

Table 2. Performance (MAE, MSE and SSE) of NARX *m* step ahead response time predictor for MySQL database server in scenarios *S1*-*S4* with prediction horizon *m* set to 4.

Predictor's Performance Select Queries			
Scenario	Server load ρ	Measure	Value
S1	$\rho = 0.956$	MSE	$2.8716e-8$
		MAE	0.0061
		SSE	0.7964
S2	$\rho = 0.318$	MSE	$2.8582e-7$
		MAE	0.0055
		SSE	0.2056
S3	$\rho = Step1$	MSE	$1.9117e-6$
		MAE	0.0094
		SSE	2.4692
S4	$\rho = Step2$	MSE	$6.2844e-7$
		MAE	0.0071
		SSE	0.7352

Predictor's Performance Update Queries			
Scenario	Server load ρ	Measure	Value
S1	$\rho = 0.935$	MSE	$1.2331e-6$
		MAE	0.0073
		SSE	0.8717
S2	$\rho = 0.333$	MSE	$4.2261e-8$
		MAE	0.0065
		SSE	0.4208
S3	$\rho = Step1$	MSE	$9.3608e-8$
		MAE	0.0055
		SSE	0.9220
S4	$\rho = Step2$	MSE	$1.4715e-8$
		MAE	0.0055
		SSE	0.4

Table 2. (*Continued*)

Predictor's Performance Mixed Queries			
Scenario	Server load ρ	Measure	Value
S1	$\rho = 0.935$	MSE	$8.1621e-9$
		MAE	0.0065
		SSE	0.7968
S2	$\rho = 0.333$	MSE	$4.3381e-7$
		MAE	0.0.0049
		SSE	0.1487
S3	$\rho = Step1$	MSE	$9.3608e-8$
		MAE	0.0053
		SSE	0.4254
S4	$\rho = Step2$	MSE	$2.2948e-7$
		MAE	0.0055
		SSE	0.3344

under *S1* for update queries compared to the select and mixed queries. This can be related to the different nature of Select and Update queries. Select queries are CPU constrained while the Update queries are I/O constrained. This leads to a very different non-linear behavior of the MySQL database server depending on the query types. As the response time predictor has been trained using the mixed queries, we can expect that the scenario *S1* for Update queries should have the worst prediction performance as it is the extreme case which has the longest distance from the mixed queries.

Performance of the response time predictor under dynamic load conditions especially its performance in transient load conditions is of interest in the following tests. Under both scenarios *S3* and *S4*, all the performance measures namely MSE, MAE and SSE indicate very good performance of the predictor. Figure 6 depicts measured response times vs. estimated response times under *S3* for the mixed queries. As it can be seen in the upper diagram of Figure 6, the measured and predicted response time values are so close that it is really hard to distinguish between them. Thus an additional figure depicting the difference between the measured and predicted response time values or simply prediction error has been added to the lower part of Figure 6. As it can be seen in this Figure, the maximum prediction error for each mixed query is less than 5% which is a very promising performance under dynamic mean arrival rates.

Performance of the proposed predictor under some more query mixes such as (50% Select, 50% Update queries) and (25% Select and 75% Update queries) for

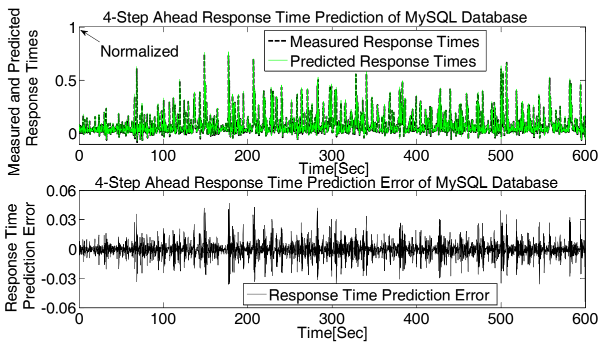


Figure 6. NARX *m* step-ahead response time prediction of the MySQL Server Database with mean arrival rate changing with a step from 5 to 15 requests per second at time 200. The prediction horizon *m* is set to 4. (upper) Measured response times vs. estimated response times. (lower) Difference between measured and estimated response times.

the same sets of scenarios *S1*-*S4* has been investigated via experiments and very small MAE, MSE, and SSE for the prediction error has been confirmed. Due to the lack of space, we skipped presenting those results.

5. Conclusion

A multi-step ahead NARX response time predictor for MySQL database server, has been proposed and its performance under several test scenarios has been studied. The proposed predictor benefits from several promising characteristics which turns it into a viable candidate for being implemented in admission control products for computing systems. It is non-linear, it supports multi-step ahead prediction, its structure is simple and its required measurements can be obtained without any requirement on changing communication protocols or operating systems.

It has been shown that with being trained in only one high load scenario, it still can predict the response times of queries in MySQL database server under both high and low load steady state scenarios with a high accuracy. Very good performance of the proposed predictor under time varying and non-stationary scenarios has been confirmed by very small MAE, MSE and SSE of the response time prediction.

Acknowledgment

Payam Amani, Maria Kihl and Anders Robertsson are members of Lund Center for Control of Complex Engineering Systems (LCCC), a Linnaeus Center at Lund

University, funded by the Swedish Research Council. Maria Kihl is partly funded by the VINNMER program at the Swedish Governmental Agency for Innovation Systems (VINNOVA). Hereby, Anders Robertsson acknowledges the support by the Swedish Research Council (grant VR 2010-5864). The first author acknowledges valuable guidance from Gustav Cedersjö in setting up the lab.

References

Amani, P., M. Kihl, and A. Robertsson (2011). “Multi-step ahead response time prediction for single server queuing systems”. In: *Proceedings of the 16th IEEE Symposium on Computers and Communications (ISCC2011)*. IEEE, pp. 950–955.

Chen, X., H. Chen, and P. Mohapatra (2003). “Aces: an efficient admission control scheme for QoS-aware web servers”. *Computer Communications* **26**:14, pp. 1581–1593.

DeWitt, D. J. (1993). *The Wisconsin Benchmark: Past, Present, and Future.* Ed. by J. Gray. Morgan Kaufmann.

Foresee, F. D. and M. T. Hagan (1997). “Gauss-Newton approximation to Bayesian learning”. In: *Proceedings of the 1997 IEEE International Conference on Neural Networks*. Vol. 3. IEEE, pp. 1930–1935.

Ganapathi, A., H. Kuno, U. Dayal, J. L. Wiener, A. Fox, M. Jordan, and D. Patterson (2009). “Predicting multiple metrics for queries: better decisions enabled by machine learning”. In: *Proceedings of the IEEE 25th International Conference on Data Engineering (ICDE 2009)*. IEEE, pp. 592–603.

Gregg, B. and J. Mauro (2011). *DTrace: Dynamic Tracing in Oracle Solaris, Mac OS X and FreeBSD*. 1st. Prentice Hall Press.

Halili, E. (2008). *Apache JMeter*. Packt Publishing.

Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*. 2nd. Prentice Hall PTR.

Kihl, M., G. Cedersjö, A. Robertsson, and B. Aspernäs (2011). “Performance measurements and modeling of database servers”. In: *Proceedings of the 6th International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks*.

Kihl, M., A. Robertsson, M. Andersson, and B. Wittenmark (2008). “Control-theoretic analysis of admission control mechanisms for web server systems”. *World Wide Web* **11**:1, pp. 93–116.

Kihl, M., A. Robertsson, and B. Wittenmark (2003). “Analysis of admission control mechanisms using non-linear control theory”. In: *Proceedings of the 8th IEEE Symposium on Computers and Communications (ISCC 2003)*. IEEE, pp. 1306–1311.

Kjær, M. A., M. Kihl, and A. Robertsson (2007). “Response-time control of a single server queue”. In: *Proceedings of the 46th IEEE Conference on Decision and Control, (CDC 2007)*. IEEE, pp. 3812–3817.

Lin, T., B. G. Horne, P. Tiño, and C. L. Giles (1996). “Learning long-term dependencies is not as difficult with NARX networks.” *Advances in Neural Information Processing Systems*, pp. 577–583.

Liu, X., J. Heo, L. Sha, and X. Zhu (2006). “Adaptive control of multi-tiered web applications using queueing predictor”. In: *Proccedings of the 10th IEEE/IFIP Network Operations and Management Symposium. IEEE/IFIP NOMS 2006*. IEEE, pp. 106–114.

Ljung, L. (1999). *System Identification: Theory for the User*. Prentice Hall Information and System Sciences Series. Prentice Hall PTR.

Menezes, J. M. P. and G. A. Barreto (2006). “A new look at nonlinear time series prediction with NARX recurrent neural network”. In: *2006 9th Brazilian Symposium on Neural Networks (SBRN’06)*, pp. 160–165.

Parlos, A. G., O. T. Rais, and A. F. Atiya (2000). “Multi-step-ahead prediction using dynamic recurrent neural networks”. *Neural networks* **13**:7, pp. 765–786.

Sheikh, M. B., U. F. Minhas, O. Z. Khan, A. Aboulnaga, P. Poupart, and D. J. Taylor (2011). “A Bayesian approach to online performance modeling for database appliances using Gaussian models”. In: *Proceedings of the 8th ACM International Conference on Autonomic Computing*. ACM, pp. 121–130.

Siegelmann, H. T., B. G. Horne, and C. L. Giles (1997). “Computational capabilities of recurrent NARX neural networks”. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **27**:2, pp. 208–215.

Tomov, N., E. Dempster, M. H. Williams, A. Burger, H. Taylor, P. J. King, and P. Broughton (2004). “Analytical response time estimation in parallel relational database systems”. *Parallel Computing* **30**:2, pp. 249–283.

Tozer, S., T. Brecht, and A. Aboulnaga (2010). “Q-cop: avoiding bad query mixes to minimize client timeouts under heavy loads”. In: *Proceedings of the IEEE 26th International Conference on Data Engineering (ICDE 2010)*. IEEE, pp. 397–408.

Watson, B. J., M. Marwah, D. Gmach, Y. Chen, M. Arlitt, and Z. Wang (2010). “Probabilistic performance modeling of virtualized resource allocation”. In: *Proceedings of the 7th International Conference on Autonomic Computing*. ACM, pp. 99–108.

Xie, H., H. Tang, and Y.-H. Liao (2009). “Time series prediction based on NARX neural networks: an advanced approach”. In: *Proceedings of the 2009 International Conference on Machine Learning and Cybernetics*. Vol. 3. IEEE, pp. 1275–1279.

Paper III

Paper III

Application of Control Theory to a Commercial Mobile Service Support System

Payam Amani Bertil Aspernäs Karl Johan Åström
Manfred Dellkrantz Maria Kihl Gabriela Radu
Anders Robertsson Andreas Torstensson

Abstract

The Mobile Service Support system (MSS), which Ericsson AB develops, handles the setup of new subscribers and services into a mobile network. Experience from deployed systems show that traffic monitoring and control of the system will be crucial for handling overload situations that may occur at sudden traffic surges. In this paper we identify and explore some important control challenges for this type of systems. Further, we present analysis and experiments showing some advantages of proposed solutions. First, we develop a load-dependent server model for the system, which is validated in test-bed experiments. Further, we propose a control design based on the model, and a method for estimation of response times and arrival rates. The main contribution of this paper is that we show how control theory methods and analysis can be used for commercial Telecom systems. Parts of our results have been implemented in commercial products, validating the strength of our work.

Originally published in International Journal On Advances in Telecommunications, vol.5, no. 3 & 4, pp. 204-215, Dec. 2012. Reprinted with permission.

1. Introduction

Resource management of computer systems, which has gained increased attention during recent years, was explored already in the late 60's [Brawn and Gustavson, 1968; Crocus, 1975]. It is an essential mechanism to handle load disturbances such as traffic surges and changes in user behavior. Poorly managed resources can severely degrade the performance of a system with potentially large financial consequences.

The work presented in this paper is motivated by a commercial Mobile Service Support System (MSS), developed and produced by Ericsson AB. Mobile Service Support Systems are used by the network operators for all processing regarding new subscribers and services in the network. Each new subscriber or service requires processing and data storage in several network nodes. The systems are in general multi-tier systems, implemented as distributed server clusters, where web and application servers process the incoming requests and database servers are used for data storage. The resource management of these systems, based on measurements of the system states such as actual utilization and response times, is crucial for the optimization of operation cost and the guarantee of service level agreements during load surges, for example during marketing campaigns or various events.

Therefore, the challenge is how to control system performance while providing guarantees on convergence and disturbance rejection. The solution is based on *dynamic control schemes*, which monitors the systems and provides actions when needed. Several types of resource–management mechanisms have been proposed and evaluated in the literature. In larger computer systems, *load balancing* is performed in order to distribute the demand for resources uniformly over a number of resource units (computers, CPUs, memory, etc.), thus avoiding the case that among the nodes with similar functionalities some are under-utilized while others are overloaded [Diao et al., 2005; Fu et al., 2006]. During overload periods, when more resources are requested than are available, *admission control* mechanisms reduce the load to the system by blocking or delaying some of the requests [Kihl et al., 2008; Chen et al., 2003; Liu et al., 2006; Voigt and Gunningberg, 2002]. For Internet applications, virtualized server systems can be used to divide physical resources into a number of separated platforms where different web applications are allowed to operate without affecting one another. *Dynamic resource allocation* between the virtualized platforms serves as a new and easy way to perform resource optimization on web server systems [Kjær et al., 2009; Xu et al., 2006; Wang et al., 2007]. In the last years, the field of *power and energy management* has become important. Large software systems have high energy consumption, which means that dynamic resource optimization of these systems may considerably lower the operating costs for the network operators [Bianchini and Rajamony, 2004; Claussen et al., 2009; Horvath et al., 2007; Elnozahy et al., 2002].

However, all optimization techniques require accurate performance models of the involved computing systems. The operation region is mainly high traffic load scenarios, which means that the computing systems show non-linear dynamics that needs to be characterized accurately [Kihl et al., 2003]. A software system is basically a network of queues, as examples, the CPU ready queue, semaphore queues, socket queues, and I/O device queues, which store requests in waiting of service in the processors. Therefore, queuing models can be used when describing the dynamic behavior of server systems [Brawn and Gustavson, 1968; Dilley et al., 1998; Menasce and Almeida, 2002; Mei et al., 2001].

The concept of Load-Dependent Server (LDS) models, in which the response time of the jobs in the system is a function of the service time of the jobs and current number of jobs waiting to be served has, to the best of our knowledge, firstly been introduced in [Perros et al., 1992]. In [Perros et al., 1992; Rak and Sgueglia, 2010; Curiel and Puigjaner, 2001], standard benchmarks were used for workload generation and also regression models to capture the system dynamics. In [Mathur and Apte, 2004], a queuing network model which represents the load dependent behavior of the LDS was presented and validated with simulations. In [Leung, 2002], a theoretical analysis of the D/G/1 and M/G/1 models with load dependency assumptions was presented.

In this paper, we investigate solutions to some important control challenges identified for the commercial MSS developed by Ericsson AB. We present a load-dependent server model, which is validated in experiments. The model has been previously published in [Kihl et al., 2012]. Further, we extend [Kihl et al., 2012] by proposing and validating an admission control mechanism based on a load-adaptive controller. A modified version of the controller has been implemented in the Ericsson product. Finally, we show how extended Kalman filters can be used for estimating the response times and arrival rates in the system.

The paper is organized as follows. In Section 2, the Ericsson product is described and the control challenges identified for the system are presented. In Section 3, the test-bed used for some of the experiments is described. In Section 4, the load-dependent server model is presented and validated. In Section 5, the load-adaptive controller is presented and experiments validating its performance are described. In Section 6, our work on response time estimation based on extended Kalman filters is presented. Finally, in Section 7, some conclusions are presented.

2. System and Problem Description

The Mobile Service Support system (MSS), which Ericsson AB develops, handles the set-up of new subscribers and services into a mobile network. It presents to the operator and its business support systems a unified middle-ware where complex functions, such as setting up a new subscriber or modifying services for an existing subscriber, can be easily invoked. The software architecture is complex with several layers and distributed infrastructures, which means that specific parts of the

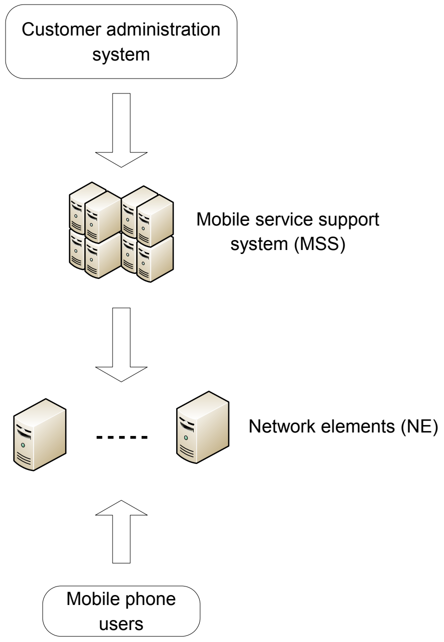

Customer administration system
Mobile service support system (MSS)
Network elements (NE)
Mobile phone users

Figure 1. Mobile service support system (MSS).

system will not have complete knowledge of the interactions among other parts of the system.

2.1 System architecture

The system architecture is illustrated in Figure 1. One request to the MSS from an upstream system normally results in a number of requests downstream out on the mobile network to several different network elements (NEs). A network element is usually a database storing subscriber and service data, for example, the Home Location Register (HLR). A user id, which needs to be fetched from one database, needs to be supplied in a query to another database to get the system consistent.

In parallel to the changes and set-ups that the MSS performs, the network is also used by the end users. Services being set up by the MSS are queried by base stations and other systems requiring that information. In respect to the MSS, this traffic can be considered as unknown background traffic, in contrast to the known traffic flowing through the MSS.

2.2 Control challenges

The experience from deployed Ericsson systems shows that there can be problems with overload in the NEs. The measurable load arriving from the MSS and the

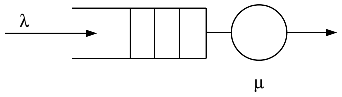


Figure 2. M/M/1 model.

unknown (not directly measurable) load arriving from mobile users may interfere with each other, creating a race for resources that may lead to overload in a NE. When one NE becomes overloaded and unresponsive, this may result in the entire transaction requiring rollback to avoid in-consistencies in the network. Such a rollback may require manual work which is of course costly for the operator.

To protect against such situations, traffic monitoring and control are crucial. In cooperation with Ericsson AB, some important control challenges have been identified for this type of system. These challenges are described below. In the following sections our collaborative work on these challenges will be presented. The models and control designs are based on response times, as this metric is rather easily measurable in the real system and because the response times can be mapped to the load status of the controlled system using the proposed model.

Performance models The first challenge is to design a performance model for the NEs, since good control designs are based on sufficiently accurate system models. The model should capture the dominant load dynamics of the NEs. Most service performance metrics such as response times and service rates depend on queue state dynamics, which means that queue models are suitable for these systems.

For the objective of performance control, simple models, such as single server queues, are often preferred. The model should only capture the dominating load dynamics of the system, since a well-designed control system can handle many model uncertainties [Åström and Wittenmark, 1997].

The classical M/M/1 model, where a single-server queue processes requests that arrive according to a Poisson process with exponential distributed service times, see Figure 2, has been shown to accurately capture the response time dynamics of a web server system [Cao et al., 2003]. However, experience from deployed systems and lab measurements have shown that databases may not have M/M/1 dynamics [Kihl et al., 2011]. Therefore, other models are required that more accurately captures the dynamics of database servers.

Admission control in MSS The NEs are loaded by two traffic sources, the measurable traffic coming to the MSS and the unknown (unmeasurable) traffic coming from the mobile users, as illustrated in Figure 3. The average arrival rates can be denoted as λ for the measurable traffic and λ_u for the unknown traffic. Overload in the NEs can be detected by monitoring the response time of requests sent to each node. When the average requests' response times exceed some threshold, the MSS can classify the involved NE as overloaded and thereby start actions to lower the arrival rate to that particular NE, in order to achieve an

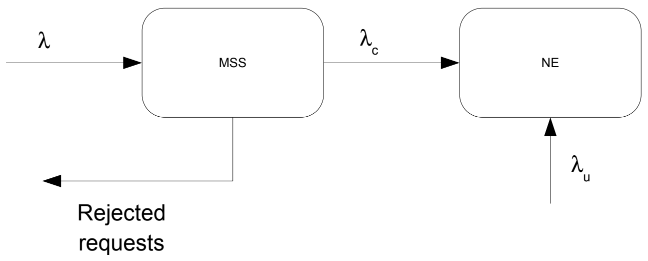


Figure 3. Load at the NEs.

acceptable arrival rate, denoted as λ_c. Therefore, the second control challenge is to design an admission control scheme that can handle the unknown traffic at the NEs and further can handle the time varying mean measured traffic rates experienced in the systems.

Monitoring and estimation One of the problems when designing control mechanisms in these types of systems is the lack of performance information. The designed protocols basically provide no means of control communication between the MSS and the NEs that can be used by a control system. Therefore, the third control challenge that has been identified is the design of monitoring and estimation mechanisms that could help in the design of, for example, an admission control scheme. The estimation scheme can be used as feed-forward control in the control system, and thereby improving the performance of the control system compared to when only using feedback control. In collaboration with Ericsson AB, some preliminary work on the application of extended Kalman filters for load estimation have been started for systems as in Figure 3.

3. Testbed

To validate some of the proposed solutions, we have performed a series of experiments in our server lab. We developed a MSS test-bed with two traffic generators, one for the measurable traffic and one for the unknown traffic, and a MySQL 5.1.41 database server as depicted in Figure 4. The computers were connected to a local 100 Mbit/s Ethernet network.

The traffic generators were implemented in Java, using the JDBC MySQL connector, and they were executed on computers with an AMD Phenom II X6 1055T Processor at 2.8 GHz and 4 GB main memory. The operating system was Ubuntu 10.04.2 LTS. The traffic generators use 200 working threads and generate MySQL queries according to a Poisson process with average rate λ and λ_u queries per second. Both traffic generators were validated in order to guarantee that they were not a bottleneck in the experiments.

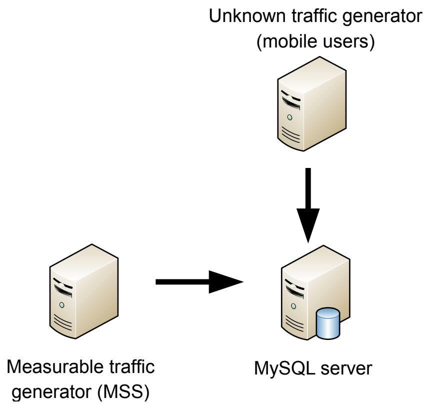

Unknown traffic generator (mobile users)
Measurable traffic generator (MSS)
MySQL server

Figure 4. Test-bed for the experiment.

The database server has several relations with the same structure but with different number of tuples. The maximum number of allowed concurrent connections is set to 100. The structure of the relations comes from the Scalable Wisconsin Benchmark [DeWitt, 1993] with 10 million tuples. Two basic types of queries are used, `SELECT` (read) and `UPDATE` (write).

The queries look like this:

```
SELECT * FROM <relation> WHERE unique1=?;
UPDATE <relation> SET unique2=? WHERE unique1=?;
```

The question marks are replaced with uniformly distributed random numbers from zero to ten million.

4. Performance Models

In this section, we focus on the modeling aspects of database servers. The objective is to develop a performance model for the database server that captures the dynamics during high loads. The performance model can be used in resource optimization schemes, as admission control systems, in order to maximize the throughput of the database server, while keeping some latency constraints. One of the challenges for these database servers is that they have a write-heavy workload, which means that the CPU is not the bottleneck during high loads. This means that previous work on performance modeling of server systems may not be applicable since they assume CPU-intensive workload.

4.1 M/M/m model with load dependency (M/M/m-LDS)

We propose to add load-dependency to an M/M/m system. In all load-dependent server models, the service time for a request will be dependent on the number of

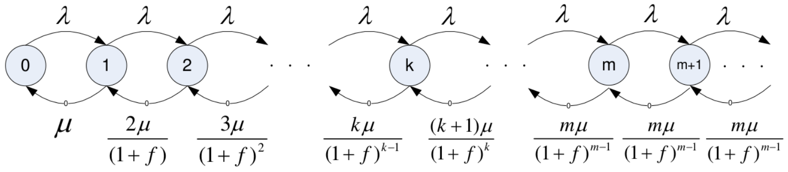


Figure 5. Illustration of M/M/m-LDS model as a Markov chain.

concurrent requests in the system. This load-dependency will model effects of the operating system, memory use, etc., which may cause service degradation when there are many concurrent jobs in a computing system [Curiel and Puigjaner, 2001]. In the experiment section, we will show that the M/M/m-LDS model accurately captures the behavior of various database workload.

The properties of the load dependent M/M/m model (*M/M/m-LDS*) are set by an exponential distributed base processing time, $x_{base} = 1/\mu$ and a dependency factor, f. When a request enters the system, it gets the base processing time x_{base} assigned to it. A single request in the system will always have a processing time of x_{base}. Each additional request inside the system increases the residual work for all requests inside the system (including itself) by a percentage equal to the dependency factor f. When a request leaves the system all other requests have their residual work decreased by f percent again. This means that if n concurrent requests enter the system at the same point, they will all have a processing time of

$$x_s(n) = x_{base} \cdot (1+f)^{n-1} \tag{1}$$

A special case is when $f = 0$. It means that there is no load dependency, and all requests will have processing time x_{base}.

The system can process a maximum of m concurrent requests at each time instance. Any additional request will have to wait in the queue. New requests arrive according to a Poisson process with average rate λ.

Therefore, the system can be modeled as a Markov chain as illustrated in Figure 5.

The average service rate of the system depends on the number of concurrent requests in the system, k, derived as follows:

$$\mu_k = \begin{cases} \dfrac{k\mu}{(1+f)^{k-1}} & \text{if } 0 < k < m \\ \dfrac{m\mu}{(1+f)^{m-1}} & \text{if } k \geq m \end{cases} \tag{2}$$

By solving the balance equations, stationary probability distribution of existence of k concurrent requests in the system is calculated as below:

$$\pi_k = \begin{cases} \dfrac{\left(\frac{\lambda}{\mu}\right)^k}{k!}(1+f)^{\frac{k(k-1)}{2}}\pi_0 & \text{if } 0 < k < m \\ \dfrac{\left(\frac{\lambda}{\mu}\right)^k}{m^{k-m}\cdot m!}(1+f)^{(m-1)(k-\frac{m}{2})}\pi_0 & \text{if } k \geq m \end{cases} \tag{3}$$

As the sum of the probabilities of all possible states equals to one, π_0 can be derived as follows:

$$\textstyle\sum_{k=0}^{\infty}\pi_k = 1 \rightarrow$$

$$\pi_0 = \frac{1}{1+\sum\limits_{k=1}^{m-1}\dfrac{\left(\frac{\lambda}{\mu}\right)^k}{k!}(1+f)^{\frac{k(k-1)}{2}} + \dfrac{\mu\left(\frac{\lambda}{\mu}\right)^m(1+f)^{\frac{m(m-1)}{2}}}{(m-1)!(\mu m-\lambda(1+f)^{m-1})}} \tag{4}$$

The stability condition in this case is:

$$\frac{\lambda}{\mu m}(1+f)^{m-1} < 1 \tag{5}$$

The average number of requests in the system, N, can be calculated as below:

$$\begin{aligned} N &= \sum_{k=1}^{\infty} k\cdot\pi_k = N_1 + N_2 \\ N_1 &= \sum_{k=0}^{m-1}\frac{\left(\frac{\lambda}{\mu}\right)^k(1+f)^{\frac{k(k-1)}{2}}}{(k-1)!}\pi_0 \\ N_2 &= \frac{\left(\frac{\lambda}{\mu}\right)^m(1+f)^{\frac{m(m-1)}{2}}(\mu m^2-\lambda(m-1)(1+f)^{m-1})\mu}{(m-1)!(m\mu-\lambda(1+f)^{m-1})^2}\pi_0 \end{aligned} \tag{6}$$

Finally by means of Little's theorem [Kleinrock, 1975], the average time each request spends in the system, T, can be derived as follows.

$$T = \frac{N}{\lambda} \tag{7}$$

4.2 M/M/m/n model with load dependency (M/M/m/n-LDS)

In case that the queue is limited to n positions, the probability for an empty system, π_0, can be determined as follows. This queuing system is named as *M/M/m/n-LDS*.

$$
\begin{aligned}
\pi_0 &= \frac{1}{I+II+III} \\
I &= 1 + \sum_{k=1}^{m-1} \frac{\left(\frac{\lambda}{\mu}\right)^k (1+f)^{\frac{1}{2}k(k-1)}}{k!} \\
II &= \frac{(1+f)^{\frac{1}{2}m^2+\frac{1}{2}m+mn-n-1}\lambda^{n+m+1}}{m^n\mu^{n+m}m!(\lambda(1+f)^{m-1}-\mu m)} \\
III &= -\frac{(1+f)^{\frac{1}{2}m(m-1)}\lambda^m}{\mu^{m-1}(m-1)!(\lambda(1+f)^{m-1}-\mu m)}
\end{aligned}
\tag{8}
$$

Further, the average number of requests in the system is as follows:

$$
\begin{aligned}
N &= N_1 - N_2 \\
N_1 &= \sum_{k=0}^{m-1} \frac{k\left(\frac{\lambda}{\mu}\right)^k (1+f)^{\frac{1}{2}k(k-1)} \cdot \pi_0}{k!} \\
N_2 &= \frac{\mu(1+f)^{\frac{1}{2}m^2-\frac{1}{2}m-1}}{m^{m-1}\left(-\lambda(1+f)^{m-1}+\mu m\right)} \cdot \frac{N_{2_{n1}}+N_{2_{n2}}-N_{2_{n3}}}{N_{2_{D1}}+N_{2_{D2}}+N_{2_{D3}}} \\
N_{2_{n1}} &= -\lambda\,(n+m)\,(1+f)^{\left(\frac{1}{2}`m^2+\frac{3}{2}m+mn-n-1\right)}\left(\frac{\lambda}{\mu}\right)^{n+m+1}\left(\frac{1}{m}\right)^{n+1} \\
N_{2_{n2}} &= \left(m(1+f)^m\mu\,(n+m+1)\,(1+f)^{\left(\frac{1}{2}`m^2+\frac{1}{2}m+mn-n\right)}\right)\left(\frac{\lambda}{\mu}\right)^{n+m+1}\left(\frac{1}{m}\right)^{n+1} \\
N_{2_{n3}} &= \left(-\lambda(1+f)^m\mu\,(m-1)+(1+f)\mu m^2\right)\left(\frac{\lambda}{\mu}\right)^m(1+f)^{\frac{1}{2}m(m-1)} \\
N_{2_{D1}} &= \left(\frac{1}{m}\right)^m(1+f)^{\frac{1}{2}m(m-1)}m!\left(-\lambda(1+f)^{m-1}+\mu m\right)\left(\sum_{k=1}^{m-1}\frac{\left(\frac{\lambda}{\mu}\right)^k(1+f)^{\frac{1}{2}k(k-1)}}{k!}\right) \\
N_{2_{D2}} &= \left(\frac{-\lambda(1+f)^{m^2+mn-n-1}}{(\mu m)^{n+m}}\right)+\left(\frac{1}{m}\right)^m(1+f)^{\frac{1}{2}m(m-1)}m!\left(-\lambda(1+f)^{m-1}+\mu m\right) \\
N_{2_{D3}} &= \mu m\left(\frac{\lambda(1+f)^{m-1}}{\mu m}\right)^m
\end{aligned}
\tag{9}
$$

Finally, the average response time for a request can be derived using Little's theorem.

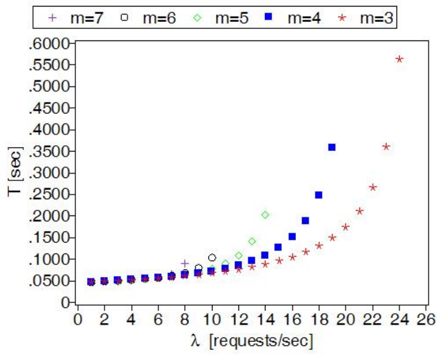


Figure 6. Variations of the λ/T graph for a special scenario with m as variable when $(f,\mu) = (0.7,22)$.

4.3 Parameter Tuning

In a Telecom system with latency constraints, the dominant dynamic of the system is often characterized by the average response time, T, when varying the average arrival rate, λ. Tuning of the parameters of the LDS model in a way that it fits the measured data from the actual server system is a necessary step in modeling of such systems. Assuming that λ and T are measurable, there are three main parameters for the M/M/m-LDS model, m, f and μ to tune in order to fit the model on the measured data. Further, for the M/M/m/n-LDS there is an extra parameter, n, to tune.

Therefore, in Figures 6-10, the effects of changing model parameters on dynamics of average response time versus mean arrival rate of queries are illustrated. In the rest of the paper, this graph will be called the λ/T graph. In each figure, it is assumed that two (three) of the parameters are fixed and the one that is mentioned is the variable. As the equations for calculating the mean response times are rather complex and the parameters are interdependent, more than one set of parameters can be fit on the measured data. Thus using these figures, a heuristic rule for tuning the parameters of the LDS model can be achieved.

In the cases where the M/M/m-LDS model is used, the first parameter to be tuned is the number of servers, m. As it can be seen in Figure 6, by increasing the maximum number of concurrent requests that can be processed in the system, the linear part of the λ/T graph will be shorter and the exponential rising rate of the graph is increased. In this case it is assumed that $(f,\mu) = (0.6,22)$.

The second parameter to be tuned is the dependency factor, f. As shown in Figure 7, by decreasing the dependency factor, the linear part of the λ/T graph is increased, however, the change is slower than in the case where m is decreased. On the other hand the exponential rising rate of the graph is increased in comparison with the case where m is decreased. Here, it is assumed that $(m,\mu) = (3,22)$.

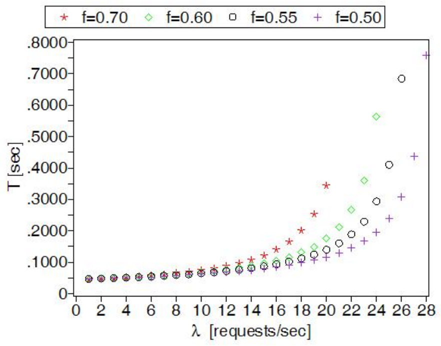


Figure 7. Variations of λ/T graph for a special scenario with f as variable when $(m,\mu) = (3,22)$.

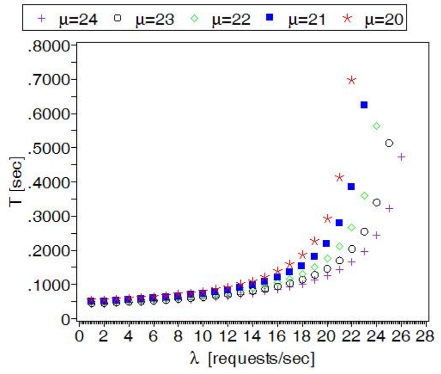


Figure 8. Variations of λ/T graph for a special scenario with μ as variable when $(m,f) = (3,0.6)$.

The effects of changing μ on the λ/T graph while fixing the two other parameters is illustrated in Figure 8. As shown in the figure, by increasing μ in equal steps, the λ/T graph will be shifted to the right in equal steps. In this case, where $(m,f) = (3,0.6)$, the rate of rising of the graph is decreased.

In cases where the M/M/m/n-LDS model is used, there will be a saturation of the response times when the load is high enough to overload the queue. Here, it is assumed that the default values are $(m,n,f,\mu) = (4,15,0.6,22)$. Figure 9 and Figure 10 show the effects when varying m and f respectively. In each case, the values of the other three parameters are constant. The general effect of changing the parameters is similar as for the case with the infinite queue, with the difference that the response times saturate when the load is high.

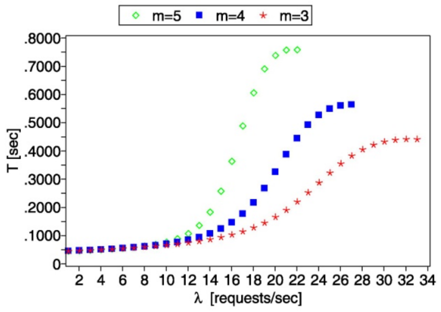


Figure 9. Variations of λ/T graph for a special scenario with m as variable when $(n, f, \mu) = (15, 0.6, 22)$.

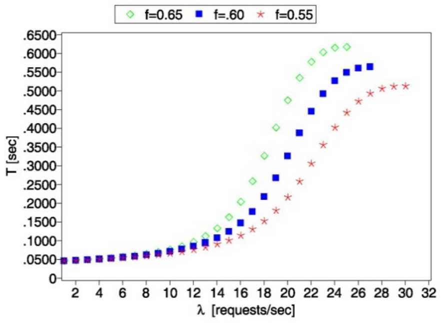


Figure 10. Variations of λ/T graph for a special scenario with f as variable when $(m, n, \mu) = (4, 15, 22)$.

4.4 Experiments

In order to validate the model, we have performed a series of experiments in our test-bed, as described in Section 3. In this case, the arrival rate of the unknown traffic was set to zero. The dynamics of the database server highly depends on the mix of requests, since SELECT and UPDATE queries require different amount of server capacity. Therefore, experiments with varying workload mix have been performed.

Figure 11, Figure 12, and Figure 13 show the results from experiments where the arrival rate is varied from low load to high load. The graphs show the average response times of queries as a function of the arrival rate. We have fitted M/M/m/n-LDS models for the data using the tuning steps described in the previous section. In both scenarios, the CPU utilization was very low, also for high loads. The maximum CPU load was about 5%.

In order to model the network delays, we have added a bias of 0.023 seconds in the average response times of the proposed models.

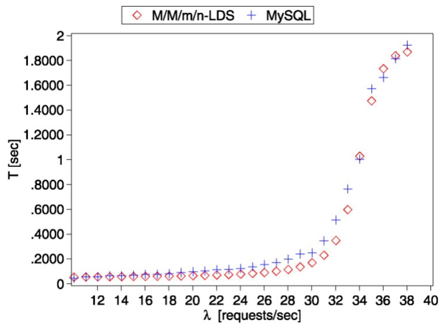


Figure 11. Performance of the M/M/m/n-LDS queuing model in modeling steady state dynamics of a MySQL database server using UPDATE queries.

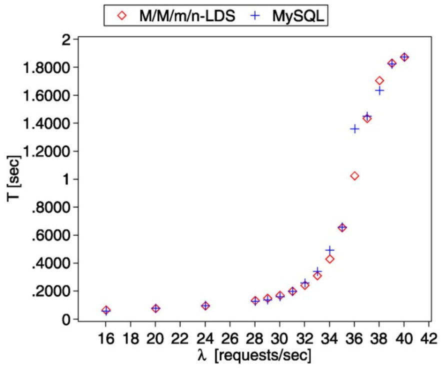


Figure 12. Performance of the M/M/m/n-LDS queuing model in modeling steady state dynamics of a MySQL database server using mixed queries.

In Figure 11, the workload is based on 100% UPDATE queries. The fitted model in this case has the following parameters $(m,n,f,\mu) = (3,81,0.75,37.1)$. Figure 12 depicts the same experiment set-up when using a mix of 25% SELECT queries and 75% UPDATE queries. The fitted M/M/m/n-LDS model in this case has the following parameters $(m,n,f,\mu) = (6,73,0.44,35.2)$. In Figure 13 only SELECT queries are used. In this case the model parameters are $(m,n,f,\mu) = (6,240,1.39,38)$.

The results verify that the proposed model can represent the average dynamics of a database server with various workloads very well.

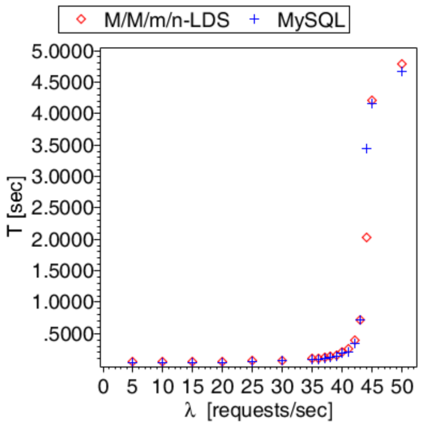


Figure 13. Performance of the M/M/m/n-LDS queuing model in modeling steady state dynamics of a MySQL database server using SELECT queries.

5. Admission Control

As part of the collaboration with Ericsson AB, we have designed an admission control mechanism for the measurable traffic to the NEs, as illustrated previously in Figure 3. As a direct effect of this work, a modified version of the control mechanism has been implemented in the Ericsson product. In this section, the controller design and its validation are described.

5.1 Control structure

The MSS includes a control system, as illustrated in Figure 14, which should ensure that the load on a specific NE is kept at an acceptable level. The control objective is to keep the mean response times of the NE queries below a desired value while maximizing the throughput. The control actions must be based on a limited amount of control information, due to the standardized protocols and the layered software architecture. The control system includes a controller and a gate.

The controller uses a response time reference value, T_{ref}, and measurements to determine an acceptable workload to the database server. The acceptable workload is defined by the normalized rate of admitted queries, λ_A, which corresponds to the ratio of the average arrival rate of the admitted requests over the higher bound of the average arrival rate of the requests. It is desired that the control system performs robustly in presence of fluctuations in the average arrival rate of the queries sent to the database. Therefore, the controller design is crucial for guaranteeing the control objectives.

The gate ensures the ratio λ_A of arriving queries is admitted to the database. In the experiments, the gate rejects requests that cannot be admitted. However, in

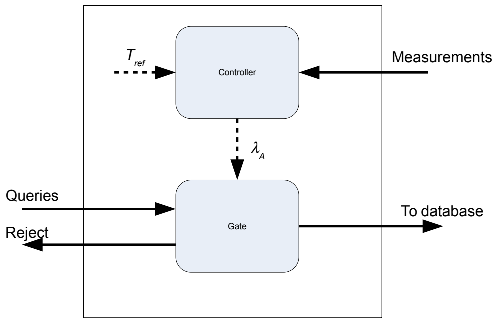


Figure 14. Control system.

the real product, this is not feasible. Instead, the real product has a traffic shaping mechanism that adds delays to the responses to the customer administration system. Since the communication with the customer administration system is synchronous, adding delays to the responses will lower the arrival rate of requests.

In this paper, we focus on the controller performance. Therefore, the implementation of the gate is not the main focus as long as it can be assumed that the gate actuates the control signal accurately.

5.2 Controller design

We have designed a controller that can guarantee the control objectives for the system. The controller, called the Load-Adaptive Controller (LAC), only uses measurements of the query response times. A classical PID controller [Åström and Wittenmark, 1997] includes one Proportional part (P), one Integral part (I), and one Derivative part (D) that determines the control signal based on the deviation of the input signal from the reference value. For stochastic systems, the derivative part will amplify the effect of high frequency noise in the response time error and thus deteriorate the overall performance of the system.

Therefore, the LAC is based on a modified PI controller with anti-windup. The LAC adapts its proportional gain with the variations in the mean arrival rates of queries sent to the database. The structure of the modified PI controller is illustrated in Figure 15.

The total load of the NE is determined by the aggregated arrival rates of the measurable and the unknown traffic streams. However, assuming that the unknown traffic is stationary during a limited time period and that the database server behaves as a conservative queuing system [Kleinrock, 1975], a specific admitted ratio of the traffic will correspond to a specific mean response time, as illustrated in Figure 16.

The controller continuously keeps track of two points in this graph, one low point, (λ_{low}, T_{low}), which is situated below the reference response time, T_{ref}, and

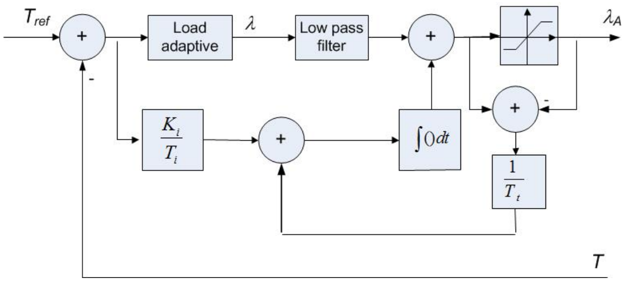


Figure 15. Load-adaptive controller (LAC).

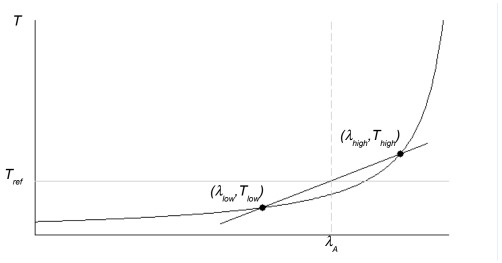


Figure 16. An illustration of the LAC calculations.

one high point, $(\lambda_{high}, T_{high})$, which is situated above T_{ref}. As the control system operates only based on measured response times of NE queries, λ_{low} guarantees that those measurements exist for all sampling intervals. The upper limit for mean arrival rates of the queries processed by the NE while not overloading the database is represented by λ_{high}. The starting values for λ_{low} and λ_{high} are set to 5% and 100% respectively.

The admittance rate of the incoming queries is iteratively updated so that its corresponding response time meets the desired value. Every sampling time, the controller calculates the average response time, T, over the last period. If the average response time during sampling period k, T_k, is too high, $(T_k > T_{ref})$, the high point is updated as $(\lambda_{high}, T_{high}) = (\lambda_k, T_k)$ where λ_k is the normalized admitted arrival rate during interval k. If the average response time during interval k is too low, $(T_k < T_{ref})$, the low point is updated as $(\lambda_{low}, T_{low}) = (\lambda_k, T_k)$. It is now assumed that the optimal normalized arrival rate, λ_o, which gives a response time of exactly T_{ref} is in the interval $[\lambda_{low}, \lambda_{high}]$. Therefore, the next normalized admitted arrival rate, λ_{k+1}, can be interpolated from these points using classic geometry:

$$\lambda_{k+1} = \lambda_k + \frac{\lambda_{high} - \lambda_{low}}{T_{high} - T_{low}}(T_{ref} - T_k) \tag{10}$$

Therefore, the quotient $(\lambda_{high} - \lambda_{low})/(T_{high} - T_{low})$ is used as proportional gain in the P-part of the controller. The algorithm will converge to the desired response time value assuming that the arrival process is stationary or slowly changing. It is obvious that the control gate cannot admit more queries than the incoming ones. This upper limit will be noted in the calculations and treated as a saturation limit of the control signal.

The integral I-part of the controller is used when the P-part is not enough for keeping the steady state error to zero. The integral part uses a controller parameter, K_i, which in conventional PI controllers are equal to the proportional gain. However, in this case, as the proportional gain changes drastically due to the load-adaptive algorithm, using the conventional PI structure will lead to a reduced phase margin which will drive the system to unstable region. Therefore, K_i is chosen as a static gain and its suitable value is determined in tuning phase of the controller.

Further, the parameter T_i is the integration time constant and T_t is the integrator's reset time constant in the anti-windup mechanism. Anti-windup is added to avoid building up of the integration part when the control gate is saturated or completely open. It is desired to choose small values for T_t so that the integrator resets quickly. Generally, T_t is chosen to be less than T_i.

A low pass filter is added after the proportional gain to smoothen the response time error signal as it is very noisy. The bandwidth of this filter should be suitably chosen so that its effect on the in-band characteristics of the response time errors is minor while attenuating high frequency components of that signal.

5.3 Experiments

To investigate the controller performance, a Java implementation of the controller was deployed as a web application to a Glassfish application server, placed on the server acting as traffic generator in Figure 4. The web application also included the traffic generator that generated requests for the web application. For each request, the admission control decides whether to allow the request to be sent to the database or rejected. The traffic generator for unknown traffic did not have an admission control, and was set to a specific average arrival rate that could be altered during run time. All requests sent to the database server were SELECT queries (according to the query structure described earlier). The λ/T graph for this particular scenario setting is shown in Figure 17. The saturation of the system is not shown in the graph for clarity reasons, since the operation region is around the "knee".

To test the performance of the controller, a scenario was chosen where the load changed from slight overload to high overload. The reference response time, T_{ref}, was set to 0.2 seconds. According to the λ/T graph in Figure 17., this corresponds to a total arrival rate of approximately 40 queries per second.

In this paper, two experiments are shown, one with a step in the unknown traffic and one with a step in the measurable traffic. The controller parameters were set to $T_i = 4$, $K_i = 0.5$, $T_r = 1$, and the sampling time $h = 0.5$ seconds. T_i was determined as

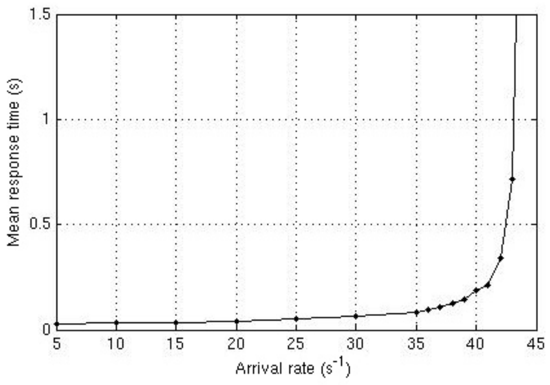


Figure 17. λ/T graph for the admission control experiments.

a multiple of the sampling time, chosen so that the controller was able to maximize the throughput while keeping the mean response times below T_{ref}. K_i was set equal to the sampling time. To give the controller time to settle this state was kept for 100 seconds after which a step in the traffic was performed. The resulting graphs are shown in Figure 18 and Figure 19. The graphs show the average dynamics from 100 runs.

In the first experiment, shown in Figure 18, the starting arrival rate was set to 23 requests per second for the measured traffic and 22 requests per second for the unknown traffic. The step increased the arrival rate of the unknown traffic by 10 (to

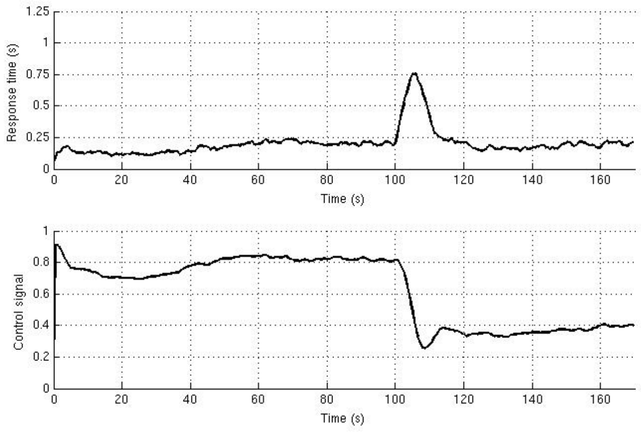


Figure 18. Performance of the LAC with step in unknown traffic.

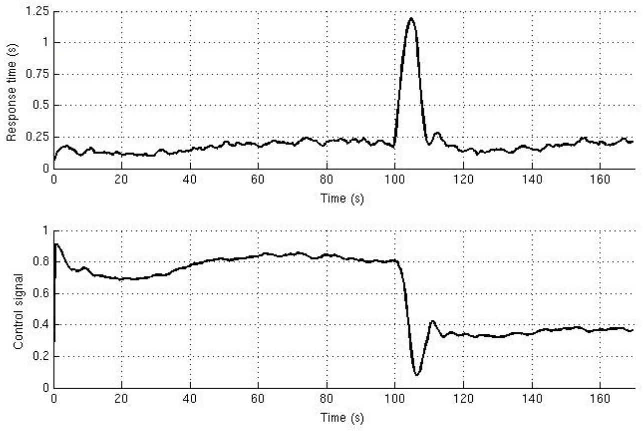


Figure 19. Performance of the LAC with step in observable traffic.

32) requests per second, resulting in a more severe overload situation.

The second experiment, shown in Figure 19, was similar to the first experiment. However, the arrival rate step was in the measurable traffic instead. To obtain a similar control signal response as in the first experiment, the step in the controllable traffic had to be larger. Therefore, the observable arrival rate was increased from 23 requests per second to 51 requests per second.

Both experiments show a well-behaved controller, with a reasonable settling time and smooth dynamics after the step.

6. Monitoring and Estimation

The system in Figure 1 is complicated with many different queues, caches and databases. Attempting to capture all details gives models that are too complex for on-line control. Extensive experience in the field of control has clearly demonstrated that simple models that capture essential behavior can be very beneficial [Åström and Murray, 2008]. One aspect of the collaboration with Ericsson has been to explore if benefits can also be obtained for monitoring and control of the MSS. A crucial issue is what complexity of the models is required for estimation and control of the MSS.

Response time and arrival rates are variables of prime concern. The variables have strong variations, which can be reduced by averaging. A more effective way is to construct estimators that exploit the dynamic behavior of the system. Exploration of such estimators has been one of the goals of the project.

A key feature of the system shown in Figure 1 is that there are two traffic

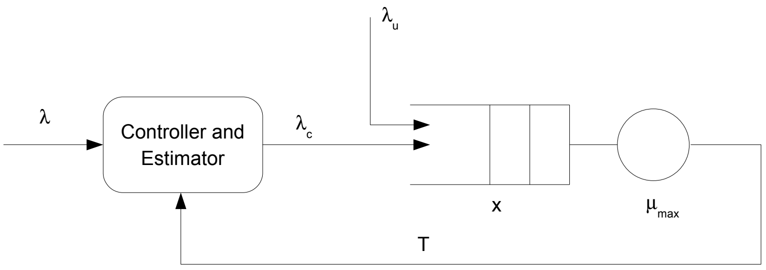


Figure 20. Schematic diagram of an abstraction of the MSS in Figure 1 with a controller and estimator.

streams. The measured traffic, generated by the customer administration system has a known arrival rate λ_c, can be controlled. The unknown stream, which is created by the mobile phone users, has an arrival rate λ_u that cannot be controlled. Monitoring and control of the system can be improved if good estimates of the average service time are available.

An abstraction of the system in Figure 1 is shown in Figure 20, where an estimator and the controller have been included. In this section, we will focus on the estimator, which only has access to measurements of the measured arrival stream λ and the response time T. All actions by the NEs and the MSS have been represented by one queue that represents the aggregated behaviors.

The queue length is represented by the variable x, which captures the aggregated behavior of many different queues in the real system. The variable x can be interpreted as a virtual queue length. The queue length cannot be measured. The actual response time T and the actual arrival times can, however, be measured. Variations in x reflect changes in the system‘s load.

6.1 Flow Model

To model the system, we will make an additional abstraction by assuming that the variables x and T are continuous and that they vary continuously in time. The behavior of the system can then be captured by the simple flow model:

$$\frac{dx}{dt} = \lambda - \mu_{\max} f(x) \tag{11}$$

where x is the virtual queue length, λ_c is the known arrival rate, λ_u is the unknown arrival rate, μ_{max} is the maximum service rate and f is a monotone function with the range [0, 1]. The response time is given by:

$$T = t_0(1+x) = t_0(1+f^{-}(\rho)) \tag{12}$$

where $t_0 = 1/\mu_{max}$ is the average time to serve one job when the queue is empty and ρ is the normalized service rate or the utility $\rho = \lambda/\mu_{max}$.

The response time goes to infinity as λ approaches μ_{max} if the range of the function f is [0, 1]. The function f gives significant freedom in adjusting the behavior to real queue behavior.

The model (11), (12) has been used extensively to model queuing systems [Agnew, 1976]. The simple M/M/1 queue can be represented by (12) with $f = x/(x+1)$ [Tipper and Sundareshan, 1990].

Even if the model (11), (12) is simple it captures some important features of real queuing systems, for example the fact that response time increases with queue length. The model also captures the behavior that the rate of change of the response time increases with increasing arrival rate. The behavior of the system can be shaped by the function f.

In the project, we have investigated simulated models with servers and we have demonstrated that it is possible to find functions f which matches the steady state behavior of simulated systems. An illustration is given in Figure 21.

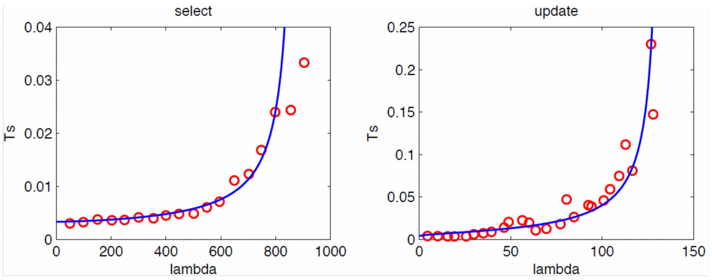


Figure 21. Service times for the operations SELECT (left) and UPDATE on an SQL server and predictions based on the model (12) with $f(x) = (1/(1+x))^n$, $n = 1.5$ and $\mu_{max} = 880$ for SELECT and $n = 0.15$ and $\mu_{max} = 132$ for UPDATE.

6.2 Estimation Algorithm

There are significant variations in the arrival and response times due to their discrete nature. To monitor and control the system it is necessary to smooth these variations. For example, the average arrival rate of the controlled stream can be estimated the simple exponential smoother

$$\begin{aligned} \hat{t}_i^+ &= \hat{t}_i + k_3(h_a - \hat{t}_i) \\ \hat{\lambda}_c^+ &= 1/\hat{t}_i^+ \end{aligned} \tag{13}$$

where t_i is the arrival time and h_a is the time since the last arrival update.

One advantage with the model (11), (12) is that it is possible to use Kalman filtering [Åström and Murray, 2008] to combine the model, which captures the gross behavior of the queuing system, with measured data.

If continuous data was available, an extended Kalman filter for the service time is given by:

$$\begin{aligned}\frac{d\hat{x}}{dt} &= \lambda_c + \lambda_u - \mu_{\max} f(\hat{x}) + k_1(T - t_0(1+\hat{x})) \\ \frac{d\lambda_u}{dt} &= k_2(T - t_0(1+\hat{x}))\end{aligned} \tag{14}$$

This filter will capture the behavior that response time increases with increasing queue length and arrival rate. The detailed behavior can be shaped by the function f.

It must be considered that the real measurements are events that represent arrival of a request or a completed response. To deal with this, we have developed an event-based Kalman filter. At arrivals, the queue length is updated according to the flow model:

$$\hat{x}^+ = \hat{x} + h_a(\hat{\lambda}_c + \hat{\lambda}_u - \mu_{\max} f(\hat{x})) \tag{15}$$

This difference equation is simply a forward Euler approximation of (11). Equation (15) is simply a prediction of x based on the model (11). Information about x is obtained when a service is completed. The queue length and the unknown arrival rate are then updated as:

$$\begin{aligned}\hat{x}^+ &= \hat{x} + h_d(\lambda_c + \lambda_u - \mu_{\max} f(\hat{x}) + k_1(T - \hat{T})) \\ \hat{\lambda}_u^+ &= \hat{\lambda}_u + h_d k_2(T - \hat{T})\end{aligned} \tag{16}$$

where h_d is the time since the last departure update. The arrival rate can be estimated because it results from the model (11) and (12) that the arrival rate is observable from a measurement of service time [Åström and Murray, 2008].

6.3 Experiment

The Kalman filter estimator was evaluated using a discrete-event simulation program written in Java, The program simulates a single server queue with exponentially distributed service times with mean $\mu_{max} = 100$ requests per second. The queue has two arrival processes, representing the measurable and unknown traffic. The Kalman filter has been evaluated for a number of scenarios validating its performance. However, in this paper we show the results of one specific scenario.

In this scenario, the unknown arrival process was a stationary Poisson process with mean 42.5 requests per second. The measurable arrival process was basically a Poisson process with changing average rate. The arrival rate, λ, was the sum of one constant part and one part represented by a sine function as given by:

$$\lambda(t) = C + a \cdot \sin(kt) \tag{17}$$

The parameters were chosen so that the system can handle the workload over long time but with periodic overloads, hence:

$$\mu_{\max} - a < C < \mu_{\max} \tag{18}$$

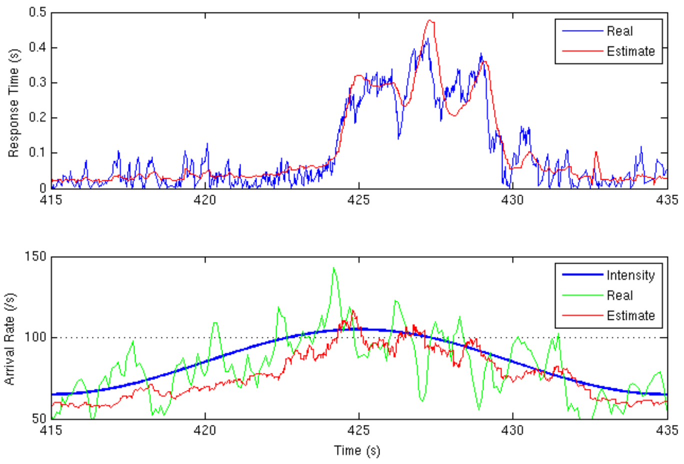


Figure 22. Kalman filter estimates of response times and estimation of arrival rate.

Therefore, the numerical values used in the simulations are $C = 42.5$ and $a = 20$ requests per second.

The differential equations describing the behavior of the estimates between events were approximated using first order forward Euler discretization.

Figure 22 shows the response times and the arrival rate, both real values and estimates for a time period of 20 seconds during the simulation. The estimate error is shown in Figure 23. It can be seen how the Kalman filter manages to follow the real system during the quick rises in response time around time 424 and 427. Here the mean square error is $\sigma = 7.4 \cdot 10^{-4}$ for the period $415 < t < 420$ and $\sigma = 1.1 \cdot 10^{-2}$ for the period $425 < t < 430$. The mean square error for the entire experiment is $\sigma = 1.9 \cdot 10^{-2}$.

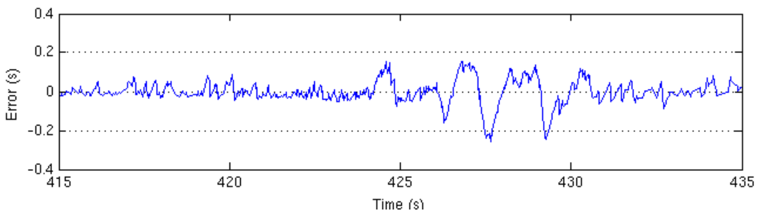


Figure 23. Proposed Kalman filter‘s response time prediction error.

7. Conclusion

Accurate control designs using control theory are essential for resource management in computer systems. In this paper we have presented work performed in collaboration with Ericsson AB, investigating how control theory can improve the performance of a commercial mobile service support system. Together with Ericsson AB, we have identified three major control challenges, and investigated solutions. The first challenge is to find accurate performance models for the system, with the objective to capture the system dynamics. The second challenge is to develop an admission control scheme that can handle unknown traffic and load surges. The final challenge is to develop estimation methods for accurate prediction of response times and arrival rates in systems with unknown traffic.

In this paper, the challenges have been treated rather independent of each other. However, the future goal is to be able to use all solutions together, in order to improve the system performance and speed up the development process. The performance model could be tuned using real data and then used for validating control designs, which is much easier than implementing the designs in testbeds or the real system. Also, in the future, the estimation algorithms should be incorporated in the control system, improving the control decisions.

Acknowledgment

The authors at Lund University are members of the Lund Center for Control of Complex Engineering Systems (LCCC). Maria Kihl and Anders Robertsson are members of the Excellence Center at Linköping-Lund in Information Technology (eLLIIT). The work is partly funded by the Swedish Research Council, grant VR 2010-5864.

References

Agnew, C. E. (1976). “Dynamic modeling and control of congestion-prone systems”. *Operations research* **24**:3, pp. 400–419.

Åström, K. J. and R. M. Murray (2008). *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press.

Åström, K. J. and B. Wittenmark (1997). *Computer-controlled systems: theory and design*. Prentice-Hall.

Bianchini, R. and R. Rajamony (2004). “Power and energy management for server systems”. *Computer* **37**:11, pp. 68–76.

Brawn, B. S. and F. G. Gustavson (1968). “Program behavior in a paging environment”. In: *Proceedings of the Fall Joint Computer Conference, part II*. ACM, pp. 1019–1032.

Cao, J., M. Andersson, C. Nyberg, and M. Kihl (2003). “Web server performance modeling using an M/G/1/K*PS queue”. In: *Proceedings of the 10th International Conference on Telecommunications (ICT 2003)*. Vol. 2. IEEE, pp. 1501–1506.

Chen, X., H. Chen, and P. Mohapatra (2003). “Aces: an efficient admission control scheme for QoS-aware web servers”. *Computer Communications* **26**:14, pp. 1581–1593.

Claussen, H., L. T. Ho, and F. Pivit (2009). “Leveraging advances in mobile broadband technology to improve environmental sustainability”. *Telecommunications Journal of Australia* **59**:1, pp. 4–1.

Crocus (1975). *Systemes d’exploitation des ordinateurs : principes de conception*. Collection Dunod universite. Dunod, Paris.

Curiel, M. and R. Puigjaner (2001). “Using load dependent servers to reduce the complexity of large client-server simulation models”. In: *Performance Engineering*. Springer, pp. 131–147.

DeWitt, D. J. (1993). *The Wisconsin Benchmark: Past, Present, and Future.* Ed. by J. Gray. Morgan Kaufmann.

Diao, Y., C. W. Wu, J. L. Hellerstein, A. J. Storm, M. Surenda, S. Lightstone, S. Parekh, C. Garcia-Arellano, M. Carroll, L. Chu, et al. (2005). “Comparative studies of load balancing with control and optimization techniques”. In: *Proceedings of the 2005 American Control Conference*. IEEE, pp. 1484–1490.

Dilley, J., R. Friedrich, T. Jin, and J. Rolia (1998). “Web server performance measurement and modeling techniques”. *Performance evaluation* **33**:1, pp. 5–26.

Elnozahy, E. M., M. Kistler, and R. Rajamony (2002). “Energy-efficient server clusters”. In: *International Workshop on Power-Aware Computer Systems*. Springer, pp. 179–197.

Fu, Y., H. Wang, C. Lu, and R. S. Chandra (2006). “Distributed utilization control for real-time clusters with load balancing”. In: *Proceedings of the 27th IEEE Real-Time Systems Symposium (RTSS 2006)*. IEEE, pp. 137–146.

Horvath, T., T. Abdelzaher, K. Skadron, and X. Liu (2007). “Dynamic voltage scaling in multitier web servers with end-to-end delay control”. *IEEE Transactions on Computers* **56**:4.

Kihl, M., P. Amani, A. Robertsson, G. Radu, M. Dellkrantz, and B. Aspernäs (2012). “Performance modeling of database servers in a telecommunication service management system”. In: *IARIA 7th International Conference on Digital Telecommunications (ICDT)*.

Kihl, M., G. Cedersjö, A. Robertsson, and B. Aspernäs (2011). "Performance measurements and modeling of database servers". In: *Proceedings of the 6th International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks*.

Kihl, M., A. Robertsson, M. Andersson, and B. Wittenmark (2008). "Control-theoretic analysis of admission control mechanisms for web server systems". *World Wide Web* **11**:1, pp. 93–116.

Kihl, M., A. Robertsson, and B. Wittenmark (2003). "Performance modelling and control of server systems using non-linear control theory". *Teletraffic Science and Engineering* **5**, pp. 1151–1160.

Kjær, M. A., M. Kihl, and A. Robertsson (2009). "Resource allocation and disturbance rejection in web servers using SLAs and virtualized servers". *IEEE Transactions on Network and Service Management* **6**:4, pp. 226–239.

Kleinrock, L. (1975). *Queueing Systems Theory, Volume 1*. Wiley-Interscience.

Leung, K. K. (2002). "Load-dependent service queues with application to congestion control in broadband networks". *Performance Evaluation* **50**:1, pp. 27–40.

Liu, X., J. Heo, L. Sha, and X. Zhu (2006). "Adaptive control of multi-tiered web applications using queueing predictor". In: *Proccedings of the 10th IEEE/IFIP Network Operations and Management Symposium. IEEE/IFIP NOMS 2006*. IEEE, pp. 106–114.

Mathur, V. and V. Apte (2004). "A computational complexity-aware model for performance analysis of software servers". In: *Proceedings of the 12th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS 2004*. IEEE, pp. 537–544.

Mei, R. D. van der, R. Hariharan, and P. Reeser (2001). "Web server performance modeling". *Telecommunication Systems* **16**:3, pp. 361–378.

Menasce, D. A. and V. A. Almeida (2002). *Capacity Planning for Web Services: metrics, models, and methods*. Prentice Hall PTR.

Perros, H. G., Y. Dallery, and G. Pujolle (1992). "Analysis of a queueing network model with class dependent window flow control". In: *Proceedings of the 11th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 92*. IEEE, pp. 968–977.

Rak, M. and A. Sgueglia (2010). "Instantaneous load dependent servers (iLDS) model for web services". In: *Proceedings of the 4th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS-2010)*. IEEE, pp. 1075–1080.

Tipper, D. and M. K. Sundareshan (1990). "Numerical methods for modeling computer networks under nonstationary conditions". *IEEE Journal on Selected Areas in Communications* **8**:9, pp. 1682–1695.

Voigt, T. and P. Gunningberg (2002). “Adaptive resource-based web server admission control”. In: *Proceedings of the 7th IEEE Symposium on Computers and Communications (ISCC 2002)*. IEEE, pp. 219–224.

Wang, Z., X. Liu, A. Zhang, C. Stewart, X. Zhu, and T. Kelly (2007). “Autoparam: automated control of application-level performance in virtualized server environments”. In: *Feedback Control Implementation and Design in Computing Systems and Networks*.

Xu, W., X. Zhu, S. Singhal, and Z. Wang (2006). “Predictive control for dynamic resource allocation in enterprise data centers”. In: *Proccedings of the 10th IEEE/IFIP Network Operations and Management Symposium, IEEE/IFIP NOMS 2006*. IEEE, pp. 115–126.

Paper IV

Paper IV

Optimal Content Retrieval Latency for Chunk Based Cooperative Content Replication in Delay Tolerant Networks

Payam Amani   Saeed Bastani   Björn Landfeldt

Abstract

Modern content distribution networks face an increasing multitude of content generators. In order to reach the minimal content retrieval latency in the content distribution networks, content shall be disseminated towards consumers based on its popularity taken from the content distribution networks. This, Combined with dividing media into chunks (heterogeneous valuation of information) and contact duration of the consumers with the access points in delay tolerant networks led us to a novel system for content management in large scale distributed systems.

In order to determine where to replicate content we formulated the problem as an integer programming problem. The cost function of this minimization problem is the accumulated weighted communication delay among the content replication servers and also the main content server. Various practical constraints such as limited total budget for content replication in each service provider, limited storage size and downlink bandwidth of the content replication servers are considered. A centralized solution to the problem is derived which gives the performance bound for any decentralized content replication strategy for the presented scenarios.

Originally published in Proceedings of The 9th Swedish National Computer Networking Workshop, SNCNW, Lund, Sweden, 2013. Reprinted with permission.

1. Introduction

Content replication has proved an efficient mechanism for offloading traffic from the content sources to the consumer edge. It has gained more significance in modern content distribution networks (CDNs) which face an increasing multitude of content generators, thus content, in delay tolerant networks (DTNs). An example of such content generators is the mobile users with smart-phones which upload videos to various possible video sharing websites.

The advantages of content replication in a Service Provider (SP) are twofold: first, the content access requests from the Main Content Servers (MCSs) -owning the content- are reduced by fulfilling some requests locally in the designated Replicated Content Servers (RCS) in the service provider's premises. This helps avoiding unnecessary extension of MCS which otherwise would incur costs on the Content Provider (CP). Second, by placing the content closer to the end consumers the delay-throughput performance is improved. Delay performance is a key factor influencing the user experience. A long content access delay results in reduced quality of experience among end users. Therefore, the limited contact duration of end consumers with the edge access devices in DTNs requires that the content is delivered with minimum delay.

Content replication can be realized in several ways. In a traditional paradigm, individual RCSs autonomously and independently decide what content to replicate and what content to discard when the available storage is full and new contents become available. Present web caching architectures are mainly built based on this paradigm where Least Recently Used (LRU) and Least Frequently Used (LFU) are the dominant content replacement policies used in web caching [Androutsellis-Theotokis and Spinellis, 2004]. While being simple from an architectural and implementation point of view, this paradigm comes with some drawbacks; the storage is not used efficiently due to potentially largely redundant contents in the individual RCSs. Even RCSs belonging to the same SP are oblivious to the replicated contents of one another. This implies that the same content may have as many copies as the number of RCSs which globally results in waste of storage. Distributed content replication is an alternative to the isolated replication mechanism as described above. With this paradigm, individual RCSs participate in a distributed content replication process. RCSs should decide whether to replicate content locally or fetch it from a (neighbour) RCSs currently having the content, or download the content on demand from the MCS. The neighbourhood of RCSs is defined with respect to some cost notion (e.g. delay, monetary, bandwidth, etc.). This mechanism while offloading the traffic and request load from the MCS, will potentially achieve a delay efficient content access to a level beyond the capacities of individual RCSs. The distributed replication can be realized in two ways. In a selfish replication system, each RCS seeks replication strategies which maximize its payoff. On the other hand, in a cooperative replication system RCSs seek strategies which maximize the social payoff (i.e., global payoff) [Borst et al., 2010]. The choice of selfish and cooperative replication would depend on the business relations. In its simplest form, the cooperative replication is the choice if the RCSs are maintained by the same SP and the objective will be maximizing the pay-off achievable by the whole system and not an individual RCS. However, in presence of multiple SPs, selfish replication is the natural strategy of choice. It should be noted that, business agreements between SPs may change these settings. For instance, SPs may decide to implement cooperative replications to enhance their individual pay-offs.

In practice, the realization of a replication system requires some form of content valuation, i.e., the importance of a typical content object from the standpoint of the consumers. Content valuation enables a SP (in a cooperative scenario) or individual RCSs (in traditional autonomous replication servers) to act selectively when they decide on content to replicate. Content popularity is one major metric used in almost all existing work to enable selective replication. In traditional web caching, LRU and LFU implicitly represent the popularity of web pages demanded so far [Mohan, 2001]. In the existing distributed replication proposals the notion of content popularity is expressed by the demand rate for a content. Furthermore they assume content popularity is fixed in the entire system. Popularity of content is considered to be different from region to region, where a region is the coverage area of an Access Point (AP) designated for consumers living or commuting within that area.

A service provider (or an individual RCS) may choose to replicate a typical content entirely or partially. In the previous work, the replication systems adopt the former option [Borst et al., 2010; Laoutaris et al., 2005] i.e., either replicate the entire content or nothing. In our proposed content replication strategy, the individual RCSs can decide on the fraction of the content to replicate and download on-demand the remaining content fraction from either the peer RCSs of from the MCS. We conjecture that partial replication is in stronger alignment with the limited contact duration or impatience of consumers; however, continuous fractions will introduce design challenges into the replication system. Thus, in this work we choose to discretize the content using the notion of chunks. Various chunking strategies have been studied in the literature [Bo et al., 2013]. We consider a chunk as a fixed size piece of a content where the chunk size and hence the number of chunks in a content is determined with respect to a fixed size base chunk. By introducing the notion of chunks into the content replication problem we aim for the following advantages:

- **Improved user experience:** The chunk based replication allows for replicating the initial chunks of contents locally with high probability thus allowing the user to access those chunks faster. While consuming those initial chunks, the remaining chunks are fetched and consumed gradually.

- **More efficient storage utilization:** RCSs are not required to replicate the entire content body. Instead, they choose a number of more important chunks from each content to replicate locally and access the remaining chunks either from peer RCSs or the MCS.

- **Avoiding bandwidth and storage usage for fake content:** With the enormous variety of contents, the emergence of fake content is inevitable, i.e., content masquerading authentic content. A natural mechanism of identifying fake content is to involve the direct consumers. If content is fake, then it is likely that the user abort demanding for the rest of the fake content by investigating the initial chunks.
- **Opportunistic content download made easier:** If a user did not fetch all chunks of a content in a meeting incidence with the current AP, they have the opportunity to download the remaining chunks form other APs as they move through the network.
- **Flexibility of cost optimization:** A RCS can partition the chunks of a content to replicate locally, fetch from neighbour RCS, or from the MCS in order to achieve a minimum cost of access to the content containing these chunks. This implies more flexibility in cost saving compared with 0/1 replication paradigm.
- **Low overall delay per content access:** It is conjectured that the chunk based replication will achieve a lower overall access delay per content.

In order to determine where to replicate content we formulated the problem as an integer programming problem with some practical constraints and solve it for the global optimal solution. This paper is structured as follows. After introduction, problem formulation is presented in section 2. System parameters are described in section 3. Results from the defined integer programming problem are summarized in section 4. Finally section 5, concludes the paper and gives some ideas for future work.

2. Problem Formulation

The topology of this scenario is depicted in Fig 1. In this scenario, we first assign weights to the chunks of contents. The weight of a chunk is a function of the popularity of the content containing the chunk, the contact duration of the AP where the content has non-zero popularity, and the rank of the chunk in the containing content. Define w^i_{cr} as the weight of a chunk with rank r in content c with non-zero popularity in RCS_i. We assign w^i_{cr} as:

$$w^i_{cr} = \sum_k \frac{M-r+1}{M} \left(\frac{\tau^i_{max} - \tau^i_k}{\tau^i_{max} - \tau^i_{min}} \right) I^i_c \tag{1}$$

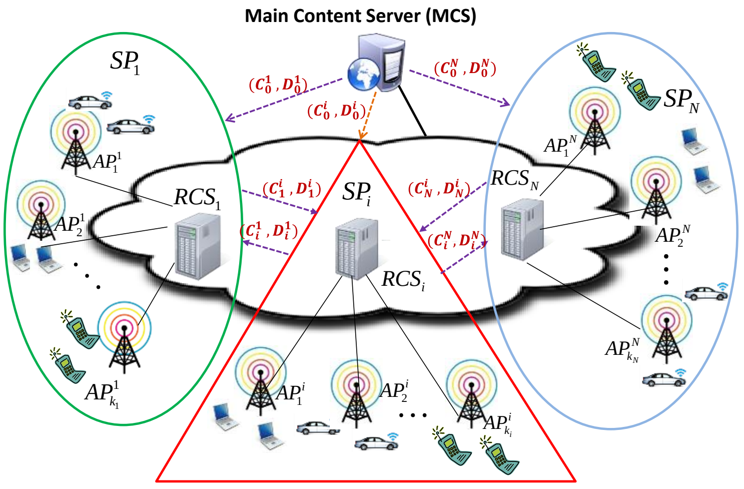


Figure 1. Multiple service providers, optimization for waited delay and bounded cost.

The weight assignment mechanism as described by (1) is expected to maintain these fundamental properties:

- Assign larger weights to chunks within more popular content.
- Assign larger weights to the initial chunks (high ranks) in a content as compared to the chunks positioned at the end of the content (with low ranks).
- Among two contents with equal popularity, assign weights to the content being demanded in an access point with shorter contact duration.
- Not be biased in favour of small contents (with small number of chunks).

2.1 System architecture

w_{cr}^i is interpreted as the valuation of an RCS_i of a chunk r within a content c and used as a metric for deciding which chunk is a good candidate for replication when there are a number of several available choices. In this scenario we address a replication system involving multiple service providers. For simplicity, we assume each service provider owns a single RCS (or hired it from a cloud). The business relations implies for some monetary cost calculated per chunk download. Denote C_j^i by the cost per chunk download in the RCS operated by service provider i from the RCS operated by service provider j , and C_0^i the cost per chunk download in the RCS operated by service provider i from the MCS. Assume a service provider i has a limited budget G_{max}^i . We define Q_j^i as follows:

$$Q_j^i = \sum_c \sum_r s_b d_c^i (C_j^i X_{crj}^i + C_0^i X_{cr0}^i) \tag{2}$$

Y_j^i is the total weighted delays of fetching chunks in RCS_i either from node RCS_j or from the MCS. Z_j^i is the total bandwidth needed for fetching chunks in RCS_i from RCS_j.

$$Y_j^i = \sum_c \sum_r w_{cr}^i (D_j^i X_{crj}^i + D_0^i X_{cr0}^i) \tag{3}$$

$$Z_j^i = \sum_c \sum_r s_b d_c^i X_{crj}^i \tag{4}$$

The optimization problem is defined as follows:

$$\min \sum_i \sum_j Y_j^i \tag{5}$$

Subject to:

$$X_{cr}^i - X_{crj}^i \geq 0 \;\; \forall i, j, c, r \tag{6}$$

$$X_{cr0}^i + \sum_j X_{crj}^i = 1 \;\; \forall i, c, r \tag{7}$$

$$\sum_c \sum_r X_{cr}^i \leq B_i \;\; \forall i \tag{8}$$

$$\sum_i Z_j^i \leq BW_j \;\; \forall j \tag{9}$$

$$\sum_j Q_j^i \leq G_{max}^i \;\; \forall i \tag{10}$$

$$X_{cr}^i, X_{crj}^i, X_{cr0}^i \in \{0, 1\} \;\; \forall i, j, c, r \tag{11}$$

Definition of parameters in the problem formulation are presented in Table 1. constraint (6) states that RCS_i cannot fetch a chunk r from RCS_j , unless chunk r is replicated in RCS_j. Constraint (7) states that a chunk r must be fetched in RCS_i either from the MCS or from one of RCSs (including itself). In other words, for each chunk it should be decided from where it is fetched. Constraint (8) shows that the total number of chunks replicated in RCS_i must not exceed its storage capacity. Constraint (9) states that the total amount of information downloaded from a RCS_j must not exceed its downlink bandwidth. Constraint (10) shows that the total cost of chunks downloaded in service provider i from the MCS and other service providers must not exceed the maximum budget of service provider i. Finally, (11) indicates the binary constraint for the decision variables.

Table 1. Notations

Notation	Description
MCS	Main Content Server
RCS_i	Replicated Content Server i
AP_k^i	Access Point k in domain of RCS_i
SP_i	Service Provider i
c	Content index
r	Chunk index
I_c^i	Popularity of content c from the standpoint of users of RCS_i,$I_c^i \in [0,1]$
τ_k^i	Average contact time in AP_k and in domain of RCS_i
$\tau_{min}^i, \tau_{max}^i$	Min and max contact duration of access points in domain of RCS_i
A_k^i	Arrival rate of users in AP_k in domain of RCS_i
d_c^i	Demand rate of content c in domain of RCS_i, $d_c^i = I_c^i \sum_k A_k^i$
B_i	Storage capacity of RCS_i (stated as the number of chunks)
BW_i	Downlink bandwidth of RCS_i
D_j^i	Chunk access delay in RCS_i from RCS_j
D_0^i	Chunk access delay in RCS_i from MCS
C_j^i	Generic chunk access cost in RCS_i from RCS_j
C_0^i	Generic chunk access cost in RCS_i from MCS
G_{max}^i	Maximum budget of service provider i
w_{cr}^i	Aggregate weight of chunk r of content c in domain RCS_i
X_{crj}^i	Chunk access binary decision variable: 1 if RCS_i accesses chunk r of content c from RCS_j, and 0 otherwise.
X_{cr0}^i	Chunk access binary decision variable: 1 if RCS_i accesses chunk r of content c from MCS, and 0 otherwise.
X_{cr}^i	Replication decision variable: 1 if RCS_i replicates chunk r of content c, and 0 otherwise.
s_b	The size of base chunk (in bits).
$M_c = \left\lceil \frac{s_c}{s_b} \right\rceil$	Number of chunks in content c. where s_c is the size of content c.

Table 2. Cost and Delay Matrix

		SP_1	SP_2	SP_3	MCS
SP_1	Delay	0	1	2	3
	Cost	0	2	1	0.5
SP_2	Delay	1	0	1	4
	Cost	2	0	3	0.5
SP_3	Delay	2	1	0	5
	Cost	1	3	0	0.5

3. System Parameters

We consider 400 files as our content pool with sizes uniformly ranging from 100KB to 5.18 MB. The size of base chunk is set to 100KB. This will translate to files of sizes ranging from 1 base chunk to 53 base chunks. Three service providers namely SP_1, SP_2, SP_3 are assumed. SP_1 consists of one RCS called RCS_1. SP_2, consists of two RCSs namely RCS_2, RCS_3. Finally, SP_3 covers three RCSs namely RCS_4, RCS_5 and RCS_6. We make a reasonable and realistic assumption that the cost of content access between two SPs is symmetric, i.e., $C^i_j = C^j_i$. If content is replicated inside a SP, no matter in which RCS, the cost for local access is considered as zero. This means that no cost is assumed for intra SP content delivery. We also assume that the cost of content access from the public main content server MCS is less than the cost of content access from a peer SP, i.e., $C^i_0 < C^i_j \quad \forall i, j = 1, 2, .., 6$. Likewise the symmetry of costs between two service providers, we assume that the communication delay between two RCSs belonging to two different SPs and also between two RCSs of the same SP are symmetric. Generally for two RCSs in the domain of two service providers namely SP_i and SP_j, $D^i_j = D^j_i$. The corresponding content access delay and cost are summarized in Table 2. Other parameters specific to the service providers are summarized in Table 3.

4. Summary of Results

The considered integer programming problem consists of 496656 binary variables. The problem was solved by means of Gurobi optimizer v5.1. Total number of chunks was 10347 from which 4472 chunks have been replicated. With this assignment the accumulated weighted chunk access delay is minimized while constraints on storage space, bandwidth and budgets for SPs are closely met. This gives us the optimal performance bound for any decentralized chunk based content

Table 3. Parameters Specific to the Service Providers

	SP_1	SP_2		SP_3		
	RCS_1	RCS_2	RCS_3	RCS_4	RCS_5	RCS_6
Percentage total storage	15	7.5	7.5	5	5	5
$t_{min}[sec]$	1	1	120	1	10	120
$t_{max}[sec]$	5	10	3600	10	120	3600
Zipfian index α	0.1	0.1	0.3	0.1	0.3	0.5
Bandwidth $Mbps$	1	2	2	1	2	3
Budget $units$	130	260		3600		

replication strategies for the same setup. We also have established that for the given system, chunk-based content replication led to a lower minimum accumulated weighted content access delay, 3425, compared to replication of the whole content 3636. Now we consider the following interesting scenarios:

- By relaxing the constraint on the budget for SPs and assuming that the available space for each RCS is greater or equal to the size of the content pool then the optimal solution is to make a local copy of all the contents in each RCS.

- In case that each chunk can be replicated only to one of the RCSs in the system, we can get a lower bound on the acceptable performance of the system.

Thus we have the upper and lower bounds for the performance of any decentralized content replication strategies for the mentioned scenario.

5. Conclusion

A chunk based cooperative content replication strategy was introduced and formulated as an Integer programming problem. Practical limitations of such a system like limited storage size, limited bandwidth and limited budget are introduced as constraints in the problem. Effects of the contact duration of the users to the access points in the system is reflected in the problem. The global optimal solution of this centralized problem was calculated. Lower and upper bounds on the performance of any decentralized chunk based content replication strategy is introduced. As a future work such strategies will be studied.

6. Acknowledgment

Payam Amani is a member of the Lund Center for Control of Complex Engineering Systems (LCCC) supported by Swedish Research Council.

References

Androutsellis-Theotokis, S. and D. Spinellis (2004). "A survey of peer-to-peer content distribution technologies". *ACM Computing Surveys (CSUR)* **36**:4, pp. 335–371.

Bo, C., Z. F. Li, and W. Can (2013). "Research on chunking algorithms of data de-duplication". In: *Proceedings of the 2012 International Conference on Communication, Electronics and Automation Engineering*. Springer, pp. 1019–1025.

Borst, S., V. Gupta, and A. Walid (2010). "Distributed caching algorithms for content distribution networks". In: *Proceedings of the 2010 IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, pp. 1–9.

Laoutaris, N., V. Zissimopoulos, and I. Stavrakakis (2005). "On the optimization of storage capacity allocation for content distribution". *Computer Networks* **47**:3, pp. 409–428.

Mohan, C. (2001). "Caching technologies for web applications". In: *Proceedings of the 27th International Conference on Very Large Data Bases*. VLDB '01. Morgan Kaufmann Publishers Inc., pp. 726–726.