Design and Implementation of Performance Policies for SMS Systems

Alberto Gonzalez Prieto and Rolf Stadler

KTH Royal Institute of Technology, Sweden
{gonzalez, stadler}@imit.kth.se

Abstract. We present a design for policy-based performance management of SMS Systems. The design takes as input the operator's performance goals, which are expressed as policies that can be adjusted at run-time. In our specific design, an SMS administrator can specify the maximum delay for a message and the maximum percentage of messages that can be postponed during periods of congestion. The system attempts to maximize the overall throughput while adhering to the performance policies. It does so by periodically solving a linear optimization problem that takes as input the policies and traffic statistics and computes a new configuration. We show that the computational cost for solving this problem is low, even for large system configurations. We have evaluated the design through extensive simulations in various scenarios. It has proved effective in achieving the administrator's performance goals and fast in adapting to changing network conditions. A prototype has been developed on a commercial SMS platform, which proves the validity of our design.

1 Introduction

The Short Message Service (SMS) is based on out-of-band message delivery, which permits subscribers to send and receive text messages to/from their mobile phones. SMS was introduced in 1992 and, since then, has experienced a remarkable success: 45 billion messages are sent per month [1], which makes SMS to represent about 10% of the revenue of mobile operators [4].

Controlling a SMS system's performance, especially during congestion periods, is a key management task. This work focuses on a design for performance management of SMS systems that (i) dynamically reconfigures, following the manager's policy, a messaging gateway in response to load changes and network conditions, and (ii) allows a manager to dynamically change management policies if needed.

We apply our design to one specific SMS component: the SMS Gateway (SMSG). The SMSG is a key functional block in the SMS architecture. It is responsible for routing messages between different networks and domains.

We take a policy-based approach to performance management for two main reasons. First, the use of policies permits us to *raise the level of abstraction of the interaction* with the managed device[2][9][10]. This is particularly relevant due to the lack of specialists in SMSGs. Second, policies can be used to specify the operation of *automated management systems*. In this paper, we aim at automating a system that controls the performance of an SMSG.

We consider both single-class, as well as multi-class SMS services. A multi-class service provides different performance guarantees to different customers or applications. While service providers currently offer only a single class of service, they are considering the introduction of service differentiation. The rationale for service differentiation comes from new uses of SMS messaging such as emergency alarms and promotional messages, which differ in performance requirements. For instance, alarms require low delays, while promotional messages tolerate higher delays or even losses.

The design in this paper supports two classes of SMS services. The first is the priority service; it guarantees delivery with a maximum delay on the gateway. The second is the non-priority service. Messages using this service may be postponed during congestion: they are stored and will be forwarded when congestion is over. The design in this paper dynamically reconfigures an SMS gateway to provide maximum throughput, while observing the quality of service objectives of maximum delays and maximum percentage of postponed messages for the above classes.

Adapting this design to a single-class or more than two classes is straightforward.

The paper extends our previous work on SMS management [17][18] as follows. First, an earlier design has been extended to support additional quality of service parameters, such as the maximum delay. Second, we studied the computational cost of policy re-evaluation. Third, we present results on the trade-off between postponed messages and system throughput. Fourth, we benchmark the performance of our design against that of an ideal system. Finally, we include our experience with implementing our design on a commercial SMS gateway.

The rest of the paper is organized as follows: section 2 discusses performance policies; section 3 describes the SMS architecture; section 4 describes our design for performance management; in section 5, we evaluate our design through simulations in different scenarios; section 6 discusses our prototype implementation; section 7 presents related work; section 8 contains the conclusions the paper.

2 Performance Policies

Administrators want to specify their performance goals in form of *performance policies*. In general, performance policies are derived from business objectives and SLAs. Such policies include performance goals in form of metrical bounds and utility functions that must be maximized. In the general case, performance policies assume a multi-class service system.

Some examples of performance policies are: (i) maximize the number of processed messages, (ii) maximize the number of served customers, (iii) provide low delays for premium customers, (iv) limit the number of postponed messages for customer A to X%, and (v) provide a minimum throughput to customer B of Y messages/second.

Performance policies are given as input to a management system, which maps them into executable functions to achieve the administrator's goals [10][11].

The design for a management system presented in this paper considers an SMS gateway that supports two service classes, priority and non-priority. It supports the following policies: (1) *Maximize the system's throughput* in messages per second.

(2) *Limit the postponement of non-priority messages* to a configurable maximum percentage. (3) *Limit the maximum delay of priority messages* to a configurable value.

Note that an alternative to postponing a message is to drop it. This is not an option for emergency alarms. However, this might be an attractive solution for handling promotional messaging. Our design supports both alternatives and the results we present hold for both of them.

3 SMS Architecture

Figure 1 positions our work within the network architecture for SMS deployment in the GSM context.

The SMSC acts as a store-and-forward system for short messages. Upon receiving an SMS, it queries the HLR database to get the location of the addressee of the message. With this information, the SMSC determines the servicing base station for the addressee and delivers the message to the terminal of the receiver. The SMSC receives messages from two different parties: mobile terminals and SMS gateways.

The SMS Gateway (SMSG) is the functional block in the SMS architecture that interconnects the wireless network to others, such as other mobile operator's network or TCP/IP networks. The gateway's administrator agrees to a traffic profile with the operators of its neighboring SMSCs/SMSGs, typically in the form of a maximum rate.

We use a model for an SMSG that is similar to an IP router. It consists of incoming ports, a routing engine and outgoing ports. Incoming ports receive the messages the gateway has to deliver. On reception, the message is routed to the appropriate outgoing port. After that, the message may need to be converted to a protocol understood by the receiving network. This conversion phase is not considered in this work.

Each outgoing port of the gateway has an associated queue. This permits the gateway to cope with brief periods of congestion. However, longer periods of congestion require control mechanisms.

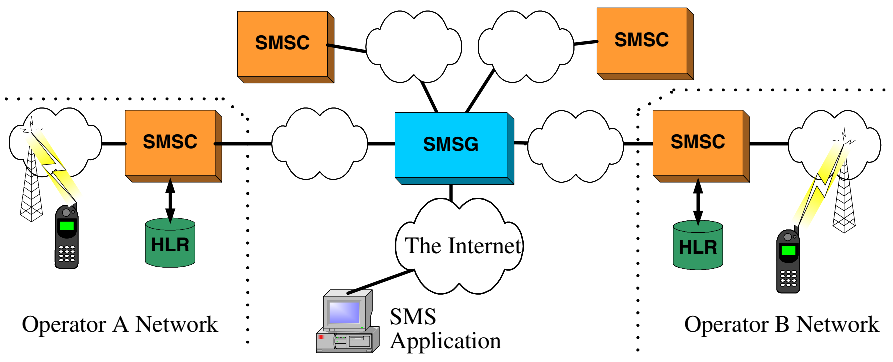


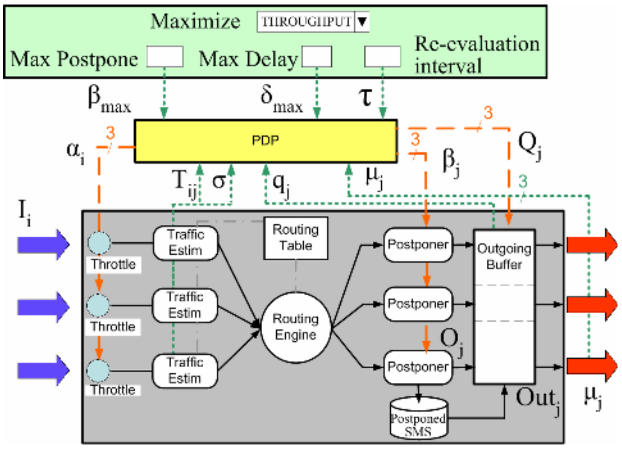
Fig. 1. Functional Architecture for SMS. This work focuses on the SMS Gateway Performance Management.

The SMSCs and SMSGs form an overlay network on top of an IP network. The overlay links are created on top of TCP connections. Therefore, ports in a SMSG are software ports, not hardware ones.

A typical port configuration for a large operator has a small number of ports (< 10) with message rates in the order of some tens messages per second. For small operators, configurations often consist of a large number of ports (~20) and lower message rates (< 10 msg/sec).

4 System Design

Functional Architecture for Performance Management. Figure 2 presents the functional architecture of our design. It permits the SMS system to achieve the administrator's performance goals, while adapting dynamically to changes in the load pattern.



β_{max}	Maximum percentage of non-priority messages that can be postponed
δ_{max}	Maximum allowable delay for priority messages
α_i	Acceptance rate at port i
β_j	Percentage of postponed non-priority messages on outgoing port j
I_i	Traffic offered to port i
T_{ij}	Traffic matrix
O_j	Traffic rate sent to outgoing port j
Out_j	Traffic rate sent by outgoing port j
μ_j	Service rate of port j
σ	Percentage of priority messages
q_j	Occupancy of outgoing port j queue
Q_j	Capacity of outgoing port j queue
τ	Policy Reevaluation interval

Fig. 2. Functional Architecture for SMS Gateway Management. The management interface is on top. The PDP block is responsible for the dynamic configuration of the gateway. The bottom block represents the gateway.

The *throttles* in the incoming ports are responsible for limiting the acceptance rates. The *traffic estimators* estimate (i) the traffic matrix, and (ii) the percentage of priority messages. These estimations are used for re-computing the gateway configuration. The *routing engine* decides the outgoing port for each message. It takes this decision based on the information stored in the *routing table*. The *postponer* is responsible for postponing messages, if needed.

The administrator specifies her performance policies for the gateway through the management interface depicted on the upper part of figure 2. In this paper, we consider the policy of maximizing the overall throughput, while observing a maximum

Maximize:

$$\sum_j O_j \qquad \text{(Eq. 1)}$$

Subject to:

$$\sum_i \alpha_i T_{ij}(1-(1-\sigma)\beta_{\max}) \le \mu_j \qquad \forall j \quad \text{(Eq. 2)}$$

$$O_j \le \sum_i \alpha_i T_{ij} \qquad \forall j \quad \text{(Eq. 4)}$$

$$\alpha_{\max} \ge I_i \ge \alpha_i \ge 0 \qquad \forall i \quad \text{(Eq. 3)}$$

$$O_j \le \mu_j \qquad \forall j \quad \text{(Eq. 5)}$$

The decision variables are α_i.

The values for β_j are determined by:

$$\beta_j = \max\left(\frac{\sum_i \alpha_i T_{ij} - \mu_j}{(1-\sigma)\sum_i \alpha_i T_{ij}}, 0 \right) \qquad \forall j \quad \text{(Eq. 6)}$$

Fig. 3. Optimization Problem to determine the gateway configuration

delay in seconds for priority messages ($\delta_{\max}$) and a maximum percentage of non-priority messages that can be postponed ($\beta_{\max}$).

We use three mechanisms to achieve the administrator's performance goals for the SMS system. The first controls the *acceptance rate* in the incoming port. In our design, the acceptance rate can be set per port. Note that reducing the acceptance rate on a specific incoming port results in reducing the load on all outgoing ports. The specific values depend on the traffic matrix.

The second mechanism *postpones***,** if needed, some of the non-priority messages routed to congested ports. This mechanism permits having a higher overall throughput at the cost of postponing messages.

These two mechanisms allow the system's administrator to control a trade-off: achieving high system throughput vs. postponing a low percentage of messages. $\beta_{\max}$ is the management parameter that controls this trade-off. It defines the maximum percentage of non-priority messages that can be postponed and takes values from 0% (no messages postponed) to 100% (highest throughput).

The third mechanism sets the *buffer capacity* in the outgoing ports, which controls the maximum queuing delay. In practice, the queuing delay dominates the overall delay of a message passing through the gateway (up to several seconds during congestion periods), while other sources of delay are comparatively small (well below one second).

The PDP (Policy Decision Point). This is the main block of the architecture: it is responsible for dynamically adapting the gateway configuration to achieve the performance goals. The PDP evaluates the performance policies and periodically re-calculates the optimal configuration for the gateway.

The PDP calculates the values for α_i (acceptance rates) and β_j (fraction of non-priority postponed messages). This computation maximizes, for the steady state, the overall throughput, while keeping the postponed non-priority messages below $\beta_{\max}$ and the maximum delay for priority messages below $\delta_{\max}$.

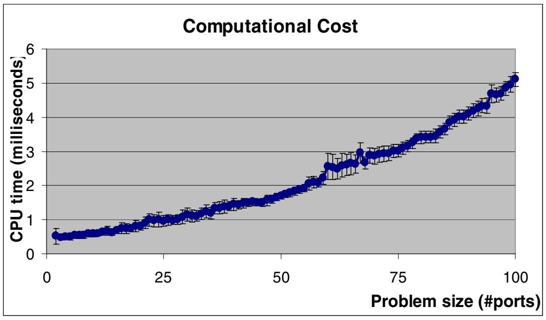


Fig. 4. Computation Cost of the Simplex Algorithm to re-compute the gateway configuration on the PDP

This computation re-evaluates the performance policies every τ seconds as follows. If the occupancy of any queue j is larger that $\delta_{max} * \mu_j$, which means that the maximum delay policy is being broken, then α_i will be set to 0 for all i's. Otherwise, the PDP predicts for each queue the future occupancy after the next τ seconds, based on the traffic estimates. If any queue is expected to overflow, then the PDP computes a new gateway configuration by solving the linear optimization problem discussed below. Otherwise, if no queue is expected to overflow, the gateway will be configured to $\alpha_i=\alpha_{max}$ for all i's and $\beta_j=0$ for all j's. This means that incoming traffic will be accepted at the maximum rate and no messages will be postponed.

Figure 3 shows the optimization problem that the PDP solves. The objective function (eq. 1) refers to the overall throughput that is to be maximized. The first constraint (eq. 2) states that traffic sent to an outgoing port is limited by the port's service rate. The second constraint (eq. 3) indicates that an incoming port cannot receive more traffic than what it is offered. Equation 6 implies that $\beta_j \leq \beta_{max}$. For a detailed discussion, see [16]. This problem is solved using the well-known Simplex algorithm [5]. Simplex will always find the global solution for all instances of our problem [16].

We have evaluated the computational cost of the Simplex algorithm for re-computing the gateway configuration on the PDP in function of the problem size, which is the number of incoming and outgoing ports. The PDP has been written in C++ and uses the COIN library implementation of the Simplex algorithm [6]. The experiments have been run on an Intel Pentium 1.6 Ghz with 512 MB of RAM with Windows XP Professional 2002 and Cygwin [7]. For a detailed description of the experiments see [16].

Figure 4 shows the results of our evaluation. As expected, the execution times increase with the problem size. However, the algorithm is very efficient in computational terms: it determines the global maximum within a few milliseconds of CPU time, permitting hundreds of policy evaluations per second, even for large configurations. We conclude that performance wise, our design is feasible to realize using current technology.

5 Evaluation Through Simulation

We have evaluated our design through extensive simulation. For this purpose, we have developed a simulator for an SMS gateway that allows us to exercise our design. For details on the simulator implementation, see [16].

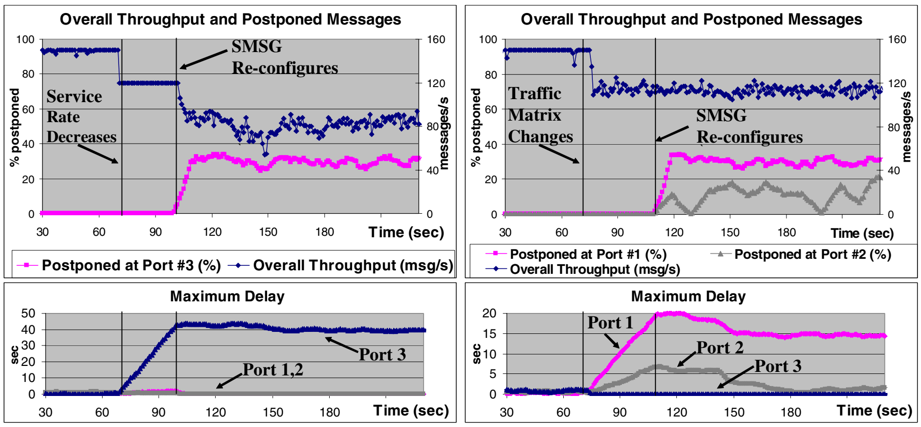


Fig. 5. Evaluation through Simulation. (Left) Service Rate Decrease: μ_3=20msg/sec, β_{max} = 30%, δ_{max} = 50sec, uniform traffic matrix. (Right) Traffic Matrix Change: β_{max} = 30%, δ_{max} = 20sec, non-uniform traffic matrix.

We present simulation results for two scenarios: *service rate decrease in one port*, and *traffic matrix change*. The analysis for a third scenario (*service rate increase in one* port) can be found in [16]. The scenarios share the following characteristics: (i) the port configuration of the gateway consists of three incoming and three outgoing ports; (ii) under normal conditions, the service rate of each outgoing port is 50 msg/sec, and the acceptance rate of each incoming port is 50 msg/sec. The chosen port configuration and rates correspond to a typical configuration for a large operator. (iii) 10% of the messages use the priority service; (iv) the PDP re-calculates the optimal configuration every ten seconds.

The offered load used in the scenarios is based on traces from a commercial SMSG. These traces exhibit low average message rates. To simulate scenarios that are representative for large operator scenarios, we superimposed several of those traces to achieve an average offered load of 50 msg/sec.

We used three different traffic matrices in our experiments. For the first one, each incoming port distributes its traffic roughly evenly among the outgoing ports, each outgoing port receiving about one third. We call this matrix a *uniform matrix*. The second matrix is a *non-uniform matrix*, where the traffic of an incoming port is split unevenly among the outgoing ports. None of these matrices causes congestion under normal conditions. For the third matrix, the *congested matrix*, each incoming port, sends 50% of its traffic to outgoing port 1, 40% to port 2, and 10% to port 3. This causes congestion in outgoing ports 1 and 2.

In the following descriptions, throughput figures refer to 1-second averages; statistics on postponed messages refer to 10-seconds averages; maximum delays are instant values.

Service Rate Decrease in Port 3. In this scenario, we analyze the behavior of our system when the service rate of outgoing port 3 slows down. The experiment starts with all outgoing ports serving at 50 msg/sec. At time=70 sec, outgoing port 3 slows down to 20 msg/sec following a step function, which causes congestion, since the

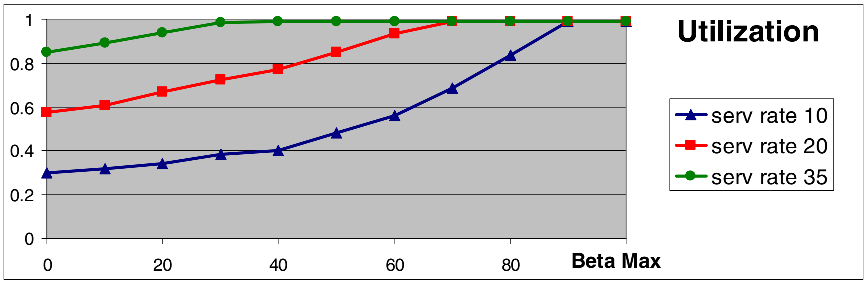


Fig. 6. Controlling the Tradeoff between Achieving Higher Overall Throughput vs. Postponing Fewer Messages

offered load to this port is higher than its service rate. For this experiment, β_{max}=30% and δ_{max}=50 secs. The traffic matrix is uniform.

Figure 5 (left) shows that the system re-configures the gateway 30 seconds after the service rate decreases. This reaction time depends on δ_{max}, μ_j, and the traffic matrix. The dependency on δ_{max} is linear. Higher values of δ_{max} permit the system to cope with longer congestion periods without throttling or postponing messages.

After the system re-configures the gateway, there is a short *transient period* of about 12 seconds. We consider the system stable again when all the outgoing queues are empty, except those of the congested ports. The time to empty the queues depends on β_{max} and on the traffic matrix. It is longer for higher values of β_{max}, since more traffic can be sent to the outgoing ports.

In this scenario, the average throughput in steady state is 87 msg/sec, and 29% of the non-priority messages are postponed.

Traffic Matrix Change. In this scenario, we analyze the behavior of our system, in reaction to a change of the traffic matrix. All outgoing ports serve at 50 msg/sec. The experiment starts with the non-uniform traffic matrix. At this stage, there is no congestion in any outgoing port. At time=70 secs, the traffic matrix changes to the *congestion matrix* following a step function. This change causes congestion in outgoing ports 1 and 2. For this experiment, β_{max}=30% and δ_{max}=20 seconds.

Figure 5 (right) shows that the system re-configures 40 seconds after the traffic matrix changes.

Note that the gateway re-configuration has a marginal effect on the overall throughput. The reason is that only limited throttling is applied, and the postponement mechanism copes with the congestion almost entirely. In outgoing port 1, postponing β_{max} of the non-priority messages reduces the traffic so that it is slightly higher than the service rate. In outgoing port 2, the values for β_2 are well below β_{max}, which is enough to address congestion in this port.

In this scenario, the average throughput in steady state is 113 msg/sec, and 31% (8%) of the non-priority messages are postponed in ports 1 (port 2).

Minimizing Postponed Messages vs Maximizing Throughput. As previously stated, β_{max} controls the trade-off between (i) minimizing the number of postponed non-priority messages and (ii) maximizing the overall throughput. Higher values of

β_{max} allow the system to reach higher throughputs. Lower values of β_{max} result in lower throughputs. Next, we analyze this trade-off.

We have run a number of 'service rate decrease' experiments. Figure 6 shows the system utilization (overall throughput divided by the sum of the service rates) as a function of β_{max}. Each line in the graph represents a different service rate for port 3. The statistics we present for each experiment are 140-second averages in steady state.

Our results show that the throughput in steady state depends on β_{max}. This dependency is not linear. The derivative of this function increases with β_{max}, until the system is fully utilized.

The throughput also depends on the traffic matrix. The experiments included in [16] show that the throughput is higher in the case of the non-uniform matrix than for the uniform matrix. This is because the non-uniform matrix permits the system to discriminate better among the sources of messages routed to the congested port. In other words, incoming ports with a small traffic contribution to the congested ports do not need to reduce their acceptance rates significantly.

Benchmarking Against an Ideal System. An ideal system always achieves the administrator's performance goals for a given policy re-evaluation interval: it (i) keeps the maximum delay at δ_{max}, (ii) never postpones more than β_{max} non-priority messages in a given re-evaluation interval, (iii) ensures that the average traffic sent to an outgoing port never exceeds its service rate in a given re-evaluation interval, and (iv) always achieves the maximum overall throughput.

In contrast to an ideal system, which has complete knowledge of the traffic statistics at any time, a real system or our simulated system has to estimate or predict them. Since the estimation/prediction process is prone to errors, a real system generally performs worse than the ideal one. In our case, the system sometimes breaks the performance constraints. As a consequence, the obtained throughput can occasionally be higher than in the ideal system. For the same achieved performance constraints, the throughput of an ideal system is higher than that of our system.

We compared our design with the ideal system with respect to the overall throughput, the rate of postponed messages, and the maximum delay in steady state. For doing this, we have run a number of 'service rate decrease' experiments. Each of them has a different combination of β_{max} and service rate values. The statistics we present for each experiment are 140-second averages in steady state.

In all the experiments we conducted with our design, the constraint on *maximum delay* has never been broken. We explain this by the fact that, when the arrival of a new message in the output queue would break the δ_{max} constraint, a non-priority message from this buffer is postponed.

Our experiments (included in [16]) show that the simulated system tends to slightly outperform the ideal one in terms of *overall throughput*. This is possible since our system occasionally breaks the constraints, caused by inaccurate predictions of the traffic matrix. For most of our experiments, the performance of our design is within 1.5 % of that of the ideal system.

Our experiments (included in [16]) show that the system tends to slightly break the constraint on *postponed non-priority messages*. In most cases, it is not more than 2% above β_{max}. The reasons for this are inaccurate predictions of the traffic matrix and burstiness of the offered load. For a more in-depth discussion, see [16].

6 Prototype Implementation

A prototype of our architecture has been implemented on a commercial SMSG: the Enterprise Messaging Gateway (EMG), version 3.0 [3]. The prototype runs on an Intel Pentium 850 Mhz with 384 Mb of RAM with Linux 2.4 (Debian).

Next, we present our experience with the prototype implementation. Specifically, we discuss how it differs from the simulated model.

In the prototype, the acceptance rate is enforced by controlling the TCP connection with the sender. The SMSG rejects or closes TCP connections to enforce the acceptance rates in each port.

In the simulated model of the gateway, the acceptance and service rates are enforced for control intervals of 0.01 seconds. In a real system, the control interval is generally larger, permitting small bursts of messages. The effect of such bursts is outlined in [16].

In the current version of the prototype, the traffic estimator is the processing bottleneck of the system. The implementation of the traffic estimator is based on analyzing the routing logs generated by the EMG and stored in a database (mysql 4.0 [8]) on the same machine. Currently, it takes between two to four seconds to retrieve and process the data required by the estimators. While the performance of this block limits the time between policy evaluations, an effective re-evaluation period of ten seconds can be achieved.

The reconfiguration of the EMG is not instantaneous as assumed in the simulations. It takes about one second for the EMG to read the new configuration and to apply it.

7 Related Work

Performance and congestion management in routing engines has been extensively studied in the context of IP routers [15]. Our work differs from that work in both the problem space and the solution space. First, congestion management for IP routers considers physical networks. In contrast, an SMSG is a node in an overlay network, where the service rate of outgoing ports can vary, depending on the state of (i) neighboring SMS systems and (ii) the links that connect them. The overlay links are created on top of TCP-IP networks. Therefore, the links' performance is that of a TCP connection.

Second, the approaches to congestion management in IP networks often focus on per-flow end-to-end feedback. Flow-based mechanisms are not relevant in the SMS context, since an SMS message fits into a single packet. In addition, currently, it is not possible to provide congestion-related feedback to the SMS sources. Therefore, such mechanisms are not applicable directly. They would require a major change in the SMS architecture, which is unlikely in the short or medium term.

Congestion control has also been studied by the ATM community. Two main lines were studied [12]. One of the lines was *rate-based control*, which is based on end-to-end control mechanisms. Such approaches are limited by the lack of support to end-to-end feedback.

The other line was *credit-based control* [13], which is based on link by link back-pressure mechanisms, as our design is. However, there are two main differences with respect to our work. First, it makes use of per virtual-circuit (VC) control. This allows reducing selectively the rates of the VCs that traverse the congested port, without affecting others. This is not possible for us due to the lack of flows or VCs. TCP congestion mechanisms as RED [14] also benefit from selective reductions of TCP-flows rates. A second difference is a consequence of having per-VC control. These approaches aim at avoiding losses and do not consider postponing/dropping packets.

8 Discussion

In this paper, we presented a policy-based design for congestion management of SMS systems. The design has been evaluated through extensive simulation studies, out of which we described in detail two scenarios: service rate decrease and traffic matrix change.

The results from our experiments are that the system performs remarkably close the administrator's performance goals. First, the overall throughput is within 1.5 % of that of the ideal system. Second, the maximum delay constraint for priority messages is always met. Third, while the system has a tendency to postpone slightly more messages than the given objective, the achieved rate is (in absolute terms) 2% above the given upper bound in most experiments. In addition, our experiments show that the system adapts fast to variations in service rate and traffic matrix.

The simulation studies in [16] suggest that the system is not very sensitive to the traffic characteristics of the offered load. In [16], we present the results for the same scenarios shown in this paper, but using Poisson sources instead of SMS traces. In both cases, the measured performance values, in terms of throughput, postponement rates and delays, are within a few percentage points. We explain this by the fact that the incoming traffic is shaped by the throttles in the incoming ports.

We showed that the computational cost of the policy evaluation, which is performed periodically, is low, even for large system configurations. Policy evaluations can be run on standard microprocessors in the order of milliseconds.

The prototype demonstrates the feasibility of implementing our design on a commercial platform.

Our design facilitates the management of messaging gateways. Compared to today's practices, where administrators often manipulate individual message queues, our design raises the level of abstraction in that an administrator specifies performance goals, and the system adapts its configuration to network conditions. This permits any administrator with a basic understanding of performance metrics to control a gateway, without the need for detailed knowledge of the device internals.

In this paper, we have considered two classes of SMS services. The adaptation of our design to a single-class or more than two classes is straightforward.

We have studied a specific objective function, the overall throughput. Extending our design to alternative objective functions involves the modification of the PDP.

Acknowledgments

We thank Roberto Cosenza from Infoflex Connect AB for his information about SMS systems and their management in commercial environments and for implementing the prototype. We also thank Ulf Brännlund from KTH for his advice on the optimization problem in this paper. This work has been supported in part by VINNOVA under the project Policy-Based Network Management.

References

[1] S. Coulombe and G. Grassel, “Multimedia Adaptation for the Multimedia Messaging Service”, IEEE Communications, Vol. 42, No.7, July 2004

[2] M. J. Masullo, S. B. *Calo, “Policy management: an architecture and approach”*. Proc. of IEEE Workshop on Sys. Management, UCLA, Cal., April 1993

[3] Nordic Messaging, www.nordicmessaging.se, August 2005

[4] GSM Association, www.gsmworld.com, May 2005

[5] G.B. Dantzig, “Maximization of linear function of variables subject to linear inequalities”, in T.C. Koopmans, editor, “Activity Analysis of Production and Allocation”, pages 339-347, 1951

[6] Computational Infrastructure for Operations Reseach, http://www.coin-or.org/index.html, July 2005

[7] Cygwin, http://cygwin.com, August 2005

[8] MySQL, http://www.mysql.com, May 2005

[9] A. Polirakis, R.Boutaba, “The Meta-Policy Information Base”, IEEE Network, special issue on Policy-Based Networks, Vol.16, No. 2, pp. 40-48 2002

[10] D. Verma, “Simplifying Network Administration Using Policy-Based Management”, IEEE Network, special issue on Policy-Based Networks, Vol.16, No. 2, pp. 20-26 2002

[11] J. Moffett, M. Sloman, “Policy Hierarchies for Distributed Systems Management”, IEEE Journal on Selected Areas in Communications, Vol.11, No. 9, pp 1404-1414, Dec 1993

[12] D. Cavendish, M. Gerla, S. Mascolo, “A Control Theoretic Approach to Congestion Control in Packet Networks”, IEEE/ACM Transactions on Networking, Vol. 12, No. 5, October 2004

[13] H.T. Kung, R. Morris, “Credit-Based Flow Control for ATM Networks”, IEEE Network Magazine, pp. 40-48, March-April 1995.

[14] Floyd S., Jacobson, V., “Random Early Detection gateways for Congestion Avoidance”, IEEE/ACM Transactions on Networking, Vol.1 No.4, pp. 397-413, August 1993.

[15] A. Mankin, K. Ramakrishnan, “RFC 1254- Gateway Congestion Control Survey”

[16] A. Gonzalez Prieto, R.Stadler, “Policy-based Performance Management for SMS gateways”, Technical Report, KTH Royal Institute of Technology, August 2005

[17] A. Gonzalez Prieto, R.Stadler, "Evaluating a Congestion Management Architecture for SMS Gateways", 9th IFIP/IEEE International Symposium on Integrated Network Management (IM 2005), Nice, France, May 15-19, 2005

[18] A. Gonzalez Prieto, R. Cosenza, and R. Stadler, “Policy-based Congestion Management for an SMS Gateway”, IEEE 5th International Workshop on Policies for Distributed Systems and Networks (POLICY 2004), Yorktown Heights, New York, June 7-9, 2004