

EMG Management

Franco Roberto Cosenza, Alberto Gonzalez Prieto and Rolf Stadler

KTH - Stockholm, Politecnico di Torino

EMG is an enterprise software for SMS exchange. A management oriented analysis of the product has been performed which led to the formulation of two high level policies and their mappings to low level ones. A policy tries to provide delivery guarantee for accepted messages and the other helps the operator to decide whether to accept a new user in the system. The first policy has been implemented and successful tests have been performed: it is possible to contain the number of messages present in EMG queues using a simple application.

Keywords: EMG, SMS Gateway, user acceptance, self management, throttling adjustment

1 Introduction

The GSM network is estimated to reach about one billion of subscribers by the end of 2003. The SMS service, i.e. the faculty to send a short message to a user terminal (mobile phone), has been added to the services offered by the GSM network since 1992. During April 1999, about one billion of SMS messages have been exchanged in only Europe[1]. Those numbers make clear that the service plays a crucial role in the technological scene both economically and technically.

The SMS service has several application. A typical one is user-to-user messaging (or mobile device to mobile device). Please refer to Figure 1. SMS delivery is carried out by a system of SMS Centers connected through a network of some kind. End users connect their GSM device to their operator's SMSC, providing destination (GSM phone number) and message. The sender SMSC routes the message to the receiver SMSC which, in turn, delivers the message to the receiver end-user.

Relevant services are also the so called Information Systems: news, travel, weather, sport, stock quotes, etc. Essentially, any information that fits into a short message can be delivered by SMS.

1.1 EMG - Enterprise Messaging Gateway.

SMSCs are typically administered by mobile network operators. Each SMSC allows different transport protocols (for instance CIMD2, OIS, SMPP,UCP/EMI). EMG is a software component which provides connectivity between SMSCs which do not understand a common protocol, allowing transparent communication between them. It can be described as an application level router/gateway. Please refer to Figure 2 where EMG is shown in its functionality of protocol converter.

Information Services are usually generated not by GSM/SMS aware systems, rather by computer systems (applications). EMG works also as gateway between applications and SMSCs as shown in Figure 3.

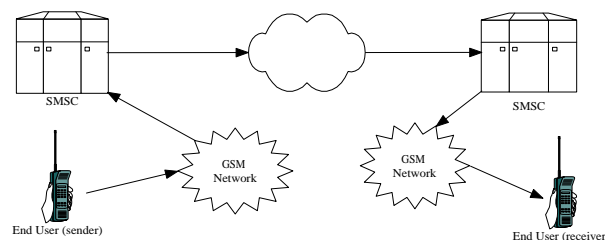


Fig. 1: User-to-user SMS Messaging

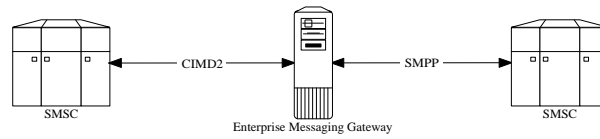


Fig. 2: EMG as a protocol converter

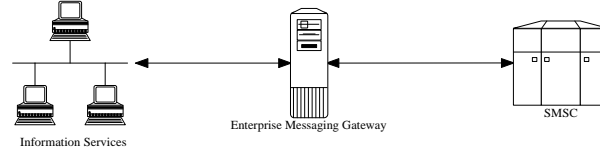


Fig. 3: EMG as a SMS gateway

Applications don't need to be SMSC protocol aware but can use, for example, the standard and widely used HTTP protocol to send SMS messages.

1.2 A business critical environment

With hundreds of messages exchanged per second between SMSCs, a failure in the delivery system can cause crucial economic loss. EMG is capable to handle over 1000 message/second[†]. Although the input entity (SMSC or application) could provide a such throughput, the output entity may or may not. To overcome this problem, EMG stores messages in memory and perform delayed delivery.

Though storing messages for delayed delivery is as an important feature of EMG, it also represents a problem. EMG resources are of course limited (in amount of available memory) and, more important, EMG cannot guarantee delivery for the accepted messages. Accepting messages which may never be delivered or delivered much later may be tolerable in some environments but not in others. Besides, a failure in the EMG can cause the lost of all messages. A problem with the remote entity (non reachability or slowness) can cause the messages to be delivered too late or not to be delivered at all. Estimating an incoming message rate of 10 messages/second, a downtime of only 30 minutes means $30 * 100 = 18000$ messages not being delivered. SMS service providers, stock quotes agencies, automatic payment systems or simple customers willing to send messages to their best friend would be affected.

This paper describes an attempt to build a management system for EMG able to solve common problems without human intervention. A policy approach is used, to provide easy management tools which don't need high level technical knowledge to be used.

2 Background

2.1 EMG - Enterprise Messaging Gateway

Enterprise Messaging Gateway (EMG) is a UNIX based messaging platform. EMG provides connectivity between applications, Short Message Service Centers (SMSCs) and other messaging entities. EMG processes high volumes of messages at high speed providing a transparent migration from one protocol to another. In addition, EMG can trigger external applications enabling easy integration with third-party systems. EMG consists of an single server executable and a collection of utilities. Text files are used to store EMG configuration.

The next paragraphs will illustrate the main components of EMG. EMG logic is based on input/output entities called connectors, which will be described first. Refer to Figure 4

[†] on a Linux Red-Hat 7.1 equipped machine, Dell Power-Edge 300, Pentium III 850 MHz, 256 MB RAM

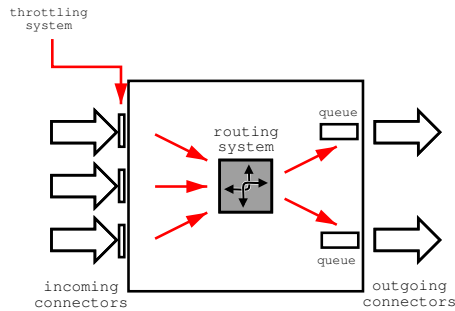


Fig. 4: A simple model for EMG

2.1.1 Connectors

EMG sends and receives messages using *connectors*. Using again a router terminology, connectors can be compared to network interfaces. Messages get in and out only through connectors, as packets get in and out through a network interface. A message is received via a connector, parsed, converted and sent out via another connector.

While each connector is bound to one and only one messaging protocol, connectors are highly configurable. They use TCP-IP connections to remote entities (SMSCs) but can also use system pipes to communicate with other system applications.

There are basically two kind of connectors, *incoming* and *outgoing* connectors. Incoming connectors are used to accept messages unspecified (though authorized) remote entities while outgoing connectors deliver messages only to a predefined entity.

When messages are received they are delivered as fast as possible according to the routing table (see 2.1.2). Incoming message rate, instead, is regulated by a *throttling system* which, operating on the THROUGH-PUT parameter for each incoming connector, can adjust the number of messages per second to be accepted by a connector.

2.1.2 The routing system

Accepted messages are delivered through outgoing connectors. The choice of which connector will take care of the delivery is made by the routing system.

EMG can use different kind of routing:

- static routing: incoming connectors are configured using the ROUTE parameters, which define the outgoing connector used to for delivery.
- source or destination routing: messages are parsed and delivered according the source or destination of the message itself. Note that since the messages are SMS, we are referring to SMS source/destination address, which is generally a phone number and not an IP address.
- keyword routing: messages are routed according message body content.

While the first kind of routing is applied per connector, the remaining options are independent of the connector configuration. It is important to note also that once a message has been routed, there are no tracks of which incoming connector accepted the message.

If a message does not satisfy any of the routing rules, it is placed in a special queue and marked as *orphan*. Orphans are evaluated again whenever the routing informations change after a system refresh or reboot.

To complete the routing offer, load balancing and fail-over are provided. When configured, load balancing splits message among several outgoing connectors in a round robin fashion, i.e. the same number of messages are given to each connector. Fail-over, instead, provides a backup route used in case a connector fails.

<i>Property</i>	
THROUGHPUT	The number of message per second which the user is allowed to send
CREDITS	The maximum number of message which the user is allowed to send
LOCKEDUNTIL	Date until the user is not allowed to send messages.

Tab. 1: User properties

2.1.3 MGP - Messaging Gateway Protocol

SMS are carried through different standard and non standard protocols. Among them CIMD2 and SMPP. In order to overcome the differences of those protocols, EMG provides its own proprietary protocol, MGP.

MGP provides all the options used in the supported protocols and provides also features to monitor and manage EMG. When a message is accepted by EMG, it is translated into MGP format and eventually re-translated into the requested protocol before leaving EMG. Besides, MGP can be used to exchange messages between MGP aware applications.

Because of its monitoring capabilities, EMG is also used by some provided system utilities to monitor and manage EMG. As an example, an SNMP agent is available, which uses MGP to communicate to EMG and to build its MIB. Through MGP is possible to change most of the EMG configuration.

2.1.4 Users

Current versions of EMG (2.x) support two user category, administrators and normal users.

The few differences are visible when using MGP to perform management and monitoring activities. Normal users may operate on a restricted *view* of the queues (the messages sent by themselves) while administrators may operate on the entire queues. Furthermore, normal users may not perform management activities as refreshing the server. For Every user it is possible to define some properties, as shown in table Tab. 1. Users and users property are stored in a separate configuration file or, in the newest versions of EMG, on an external SQL database.

2.2 Policy based management systems

Network management systems are widely used today. SNMP has reached its maturity with version 3, adding security support to what is a *de facto* industrial standard. Besides, other protocols and proprietary systems fill the remaining space in the network management scenario. Nevertheless, the exponential growth of the number of networks and users brings new problems into account. As networks expand, so does their complexity and the effort which must be made in order to manage them. The need of diversification among different users, for example, has arisen, causing a revolution in what was initial created for "equal people". The classical network management approach based on remote configuration, monitoring and access control is too limiting.

Policy based management systems try to help the network operator in its duty simplifying network management and hiding technical details which are not (directly) of interest. To achieve this goal, policy based network management, or in brief PBNM, tries to introduce one or more abstraction layer between the the managed objects and the operator and tries to provide a non technical interface to manage network resources. To be useful, policies must be familiar with business concept so that, for example, an accounting department can use them. In the same time, they must be specific enough to be handled by network devices [2]. Since those two requirements are in contrast, a solution is achieved splitting policies on different abstraction levels. Each level must satisfy different requirements.

In this report we use a two level approach, *high* and *low* level policies. High level policies specify what an operator would use and specify, while low level policies specify what the network device is capable.

It is important that high level policies do not map one to one to low level ones. The more is the abstraction gap between high level and low level policies, the more the PBNM is useful. A good high level policy ignore all details and formulates *what* the management system has to do, not *how* to actually do it. This process is not trivial, but good policies make the task of the network operator much easier.

3 Controlling EMG

Policies applies naturally to network traffic shaping. In that environment there is a need to change network components configuration dynamically and with an high frequency. Many researchers are focused on how to build management systems able to solve this problem.

As stated before in this document, EMG can be compared to an IP router (though working on the application level and where traffic is made of SMS messages) and therefore a similar approach can be used to define a policy system.

3.1 Delivery guarantee versus utilization

EMG is an intermediate system. It accepts SMS messages from a remote entity and delivers them to another one. Messages received on incoming connectors are routed and queued, waiting to be sent through outgoing connectors.

Every outgoing connector builds its own queue, which grows and shrinks according to the number of messages that the outgoing connector can deliver. The number of messages per second that the connector can deliver depends both on EMG performance but also on the performance of the remote entity to which the connector is bound.

When EMG accepts a message, it cannot guarantee for its delivery, unless the sender entity explicitly requests for a DLR, a standard mechanism used by the SMS network to confirm message delivery. This mechanism, anyway, is independent and irrelevant in the EMG context since it is a end-to-end confirmation. After the message is queued, whatever happens to the message is unknown to the sender entity. A failure on the entity bounded to the outgoing connector may cause that the message is never delivered or, in case of a temporary failure, that the message is delivered with unacceptable delay.

Moreover, EMG will accept new messages until resources are available (system memory). Therefore, EMG queues will grow indefinitely until the system will be able to provide the required memory and store the messages. When no memory will be available, EMG will deny incoming connections, not being able to accept any further messages (EMG BLOCK).

In a common EMG configuration, where several incoming connectors accept messages which are relayed through several outgoing connectors, a failure in a single outgoing connector may cause the system to block entirely. If the failure is not detected and fixed by external means, the system will not be able to deliver any message even if the remaining connectors are perfectly working. Furthermore, since EMG is not provided with persistent queues, in the possible case of crash, EMG will loose all the accepted message not yet delivered. It is then important to limit the number of messages present in the queue in order to achieve a higher reliability of the system.

Unfortunately, it is not possible to identify which messages are going to fill the problematic queue before they have entered EMG and have been routed. In fact, because of the non static routing, messages can be sent by whatever user, using whatever incoming connector and still be routed to the same problematic connector. To contain the number of messages we are forced to limit the rate of all incoming messages. A solution is to accept as many messages as EMG can deliver.

If we define ϕ_{in_j} as the number of messages per second accepted by an incoming connector and ϕ_{out_k} as the number of messages per second delivered by an outgoing connector we have:

$$\int \Phi_{in} \geq \int \Phi_{out} \quad (1)$$

where $\Phi_{in} = \sum_j \phi_{in_j}$ and $\Phi_{out} = \sum_k \phi_{out_k}$. Please refer to Figure 5.

If we could put

$$\Phi_{in} = \Phi_{out} \quad (2)$$

we could solve the problem of the indefinitely growing queues but it would affect the global system utilization, penalizing those messages which would have run through the system smoothly. In fact, Φ_{out} depends on Φ_{in} , according to (1). Limiting the number of messages getting into the system means also limiting the number of messages that get through the system. A trade-off is needed, based on which quality of service is requested and on the risk that the customer accepts to take.

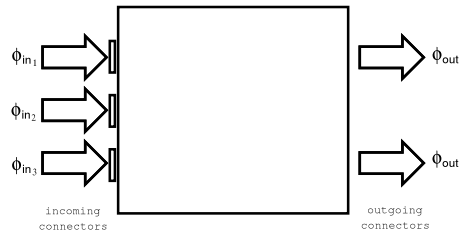


Fig. 5: EMG message flow

Even if we just showed the problem with some level of detail, the network operator is not concerned with that. What he needs is a tool that provides him with the capability to easily adjust the EMG *policy*. He is not interested in how to implement the solution but he is interested in reaching his goal. He needs also to quantify his decisions on an higher level, maybe on a economical one.

A solution to the problem is a *selector*, which gives the network operator the possibility to choose between maximum utilization of the system, maximum delivery or a "middle" choice, in order to tune the system according one's requisites. In those environments in which the EMG blocking is not considered to be big deal, as example when EMG is used for mass message delivery, the operator may choose maximum utilization and maybe monitor EMG with SNMP to notice eventual problems; in environments where the time of delivery is more important, as when the SMS are used for notifying alarm circumstances and then more concerned about delivery guarantee, the operator may choose maximum delivery. It is important to emphasize that the operator is unaware of how the policy would be mapped in the actual system implementation: all the implementation details are completely hidden.

- *Delivery versus utilization policy*
- Delivery can be expressed with a parameter (γ), where $-1 \leq \gamma \leq 1$
- $\gamma = -1$, maximum system utilization
- $\gamma = 1$, maximum delivery guarantee

3.2 User acceptance

When a new user has to be added to the system, the operator asks him self different questions, among which

- How much resources do I have left?
- How much resources will the new user need?
- How good can I serve him (and, consequently, how much can I get paid?)

Currently, except for the second question, the operator has no direct answer.

EMG users are all served *best effort*. It is possible to permit or deny access to single connectors for each user. Though this mechanism allows a sort of differentiation among users, it does not provide any indication or measure for available EMG resources. Accepting a new user in EMG is today a completely arbitrary process: no checks are performed to see if user's (and operator's) requirements are fulfilled. Besides, EMG can apply limitations to single users both on the number of messages per second and on the maximum number of message accepted but it has no internal mean to assure any quality of service.

A user expresses his requirements in the form of a *request*, which, depending on the system conditions, can be accepted or denied. The quality of service that EMG can provide can be measured either in throughput or delivery reliability. An example of user request could be expressed in this form:

- medium throughput
- low delivery guarantee

EMG Management

- 99% of the time

Beyond the problem of mapping the *qualitative* properties to *quantitative* ones, the operator needs an answer to the big question: *Can the user be accepted into the system?*

The answer is not easy, since EMG configuration may change dynamically, faster or slower according to the value of γ , making the operator duty more difficult absolute.

4 Policy refinement

The policies suggested in the previous chapter are high level policies. They hide all the implementation details to the operator. In order to enforce a policy, the refinement process, has to be performed. Refining a policy means to map the high level policy to simpler, more technical and less abstract ones, which can be easily understood and implemented by a device. Much research is focused on providing policy based management systems as general as possible but the process of refinement from high level policies to low level ones is still have to be done manually. The more is the gap between high level policies and low level policies, the more difficult is to automate the mapping process.

Low level policies must have enough details to be understood by the managed network objects while high level policies are often expressed using *qualifiers* rather than *quantifiers*. Terms as `low throughput` can be easily understood by a human interpreter but not by a computer system.

4.1 Delivery guarantee versus utilization

As described in section 3.1, we are concerned in avoiding EMG block and in avoiding delayed delivery of messages. Since the problem arise when a queue size exceed a certain limit, the low level policies can be described with a

```
if(event) do actions
```

pattern.

In this case the event manifests when remote entity is slow and it is not able to accept the traffic generated by EMG. This will also happen if for some reason the network link itself gets slow. The event can be summarized in having a connector queue growing indefinitely, until the the situation gets normal again.

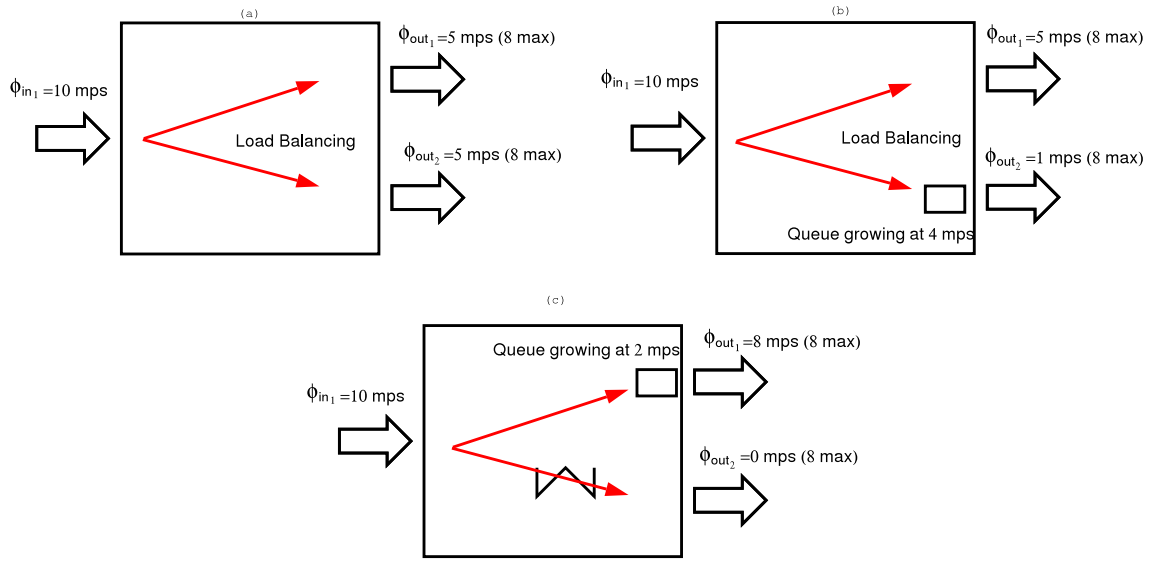
When EMG is running load balancing on the interested connector, we can apply an action based on deactivating load balancing which will try to solve the problem without affecting the global system utilization. Because of its *neutral* property, this action can be used whatever is the value of γ . If the problem will not be solved, the system will go further applying other actions affecting, this time, delivery guarantee and system utilization. For *positive* values of γ we will slow down customers; for *negative* values of γ we will drop undeliverable messages.

4.1.1 Deactivating load balancing

As mentioned in section 2.1.2, EMG can use load-balancing among several connectors. Deactivating load-balancing can help to solve the problem without affecting the utilization of other connectors. When two connectors are configured with load balancing and one of them fails or is too slow, deactivating load balancing means that no messages will be routed anymore to the slow connector. All messages will be routed to the other, healthy, connector. If the healthy connector is fast enough to deliver the new rate of messages, the problem will be solved. If the connector is not as fast as required, the total number of messages present in the queues will grow anyway, but probably slowly.

As an example, let's suppose to have an EMG configuration with three connectors, an incoming one and two outgoing ones, configured with load-balancing, as showed in Figure 6. This is a very simple configuration which may seem to reductive but it will help to explain the idea. Nevertheless, EMG is often used as a traffic balancer and such configuration is not atypical in real world production environments.

All incoming messages will be equally split among the two outgoing connectors. Let's also suppose that each outgoing connector has an initial capacity of delivering 8 messages per second each (depending on the remote entity capacity) and that the incoming connector accepts 10 messages per second. In normal condition, 5 messages per second will be delivered to each outgoing connector (a). If one of the outgoing


Fig. 6: Deactivating load balancing

connectors rate gets slow, down to 1 message per second, a queue will be built at the rate of 4 messages per second. The other connector will not be affected, since it is still receiving 5 messages per second which is smaller of the 8 that it can deliver (b). If we deactivate load balancing, all of the 10 messages per second will be routed to the healthy connector. It will not be able to deliver all of them and will build a queue, which will only grow at a rate of 2 messages per second (c). The problem is not solved but, in case of temporary failures, the system will recover without any actions on the number of incoming messages.

4.1.2 Slow down customers

To solve our problem we can operate on the number of messages per second accepted from EMG. If the number of incoming messages is higher of the number of outgoing messages, a queue will be built somewhere. The solution is to reduce the number of incoming messages so that we get close to what described Formula (2).

Unfortunately it is not possible to identify an appropriate sub set of incoming connector causing (indirectly) the problem. We are forced to lower the total number of incoming messages per second Φ_{in} operating on each of the incoming connectors, instead of performing a finer tweaking.

Using the *throttling system* (refer to 2.1.1) we can control the maximum value reached by Φ_{in} . In fact, since each incoming connector can accept no more messages than indicated by the THROUGHPUT parameter, we can affirm

$$\Phi_{in} \leq MAXRATE \quad (3)$$

where

$$MAXRATE = \sum_j THROUGHPUT_j \quad (4)$$

is the sum of the maximum admissible number of messages accepted by all incoming connectors.

It sounds reasonable to set $MAXRATE$ at a value which is close to Φ_{out} . Anyway, since we want out policy to be flexible, we should set $MAXRATE$ depending on the value of γ defined in section 3.1. A possible solution is illustrated in Figure 7. This is a very simple and probably not optimal solution which satisfies three conditions:

- If $\gamma = 0$, $MAXRATE$ is not changed from its original state
- If $\gamma = 0.5$, $MAXRATE = \Phi_{out}$. The system accepts as many messages as it can delivery

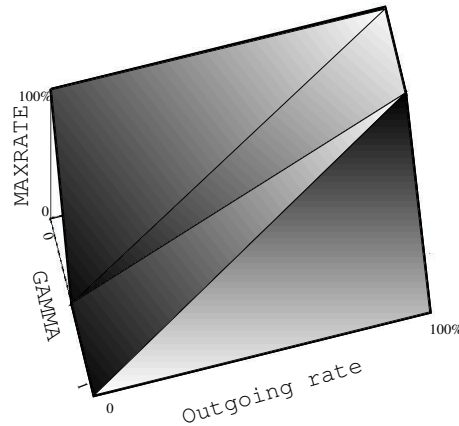


Fig. 7: New value of MAXRATE

- If $\gamma = 1$, $MAXRATE = 0$. The system stops accepting messages.

It should be remembered that the value of $MAXRATE$ is computed when the event manifests and after the abnormal condition has been resolved, $MAXRATE$ should be set to its original value (or to a similar, appropriate one). The return to the *normal condition* should be detected when the value of the queue size has reached a *lower threshold*, to avoid unwanted oscillations. In the case in which the system will not become normal because of an ineffective value of $MAXRATE$, $MAXRATE$ should be computed again. There are different strategies to detect a persistent critical state but we suggest to redefine proportionally the `QUEUE_THRESHOLD` causing the event. The new threshold can be set augmenting the old one of a certain percentage. If the queue exceeds the threshold the value of $MAXRATE$ will be computed again and so on, until the system will get into its normal state and the threshold will be put to the original value.

Having many incoming connectors, $MAXRATE$ value must be repartitioned (equally or not equally) among all the connectors. It must be noticed that in case the remote outgoing entity will suffer a long time failure, the system may be running at a very low speed for a long time, while if the failures can be classified as transient, the system will adapt itself automatically keeping the queue under its limit.

4.1.3 Drop undeliverable messages

Dropping message is a quite drastic solution which in certain cases can be acceptable anyway. Some kind of application requires almost real time SMS delivery. For example, banks sending stock quotes variations to their customers via SMS will absolutely not be interested of keeping EMG resources busy with *old* informations. The system maybe used by customers with different requirements, some may want that their messages are never discarded, even if their delivery time increases considerably while others may want their messages not to be sent anymore if the delivery time exceeds some limit. Besides, messages can also be discarded based on some user priority.

When γ gets negative values, we can drop messages. Dropping messages augments the number of messages that EMG can accept, providing a fake idea of throughput. In the same time, it lowers the delivery guarantee because it increases the probability that a message will not be delivered. According to the value of γ , we can apply different policies

- Drop all messages
- Drop messages randomly
- Drop the latest accepted messages (or the first)
- Drop low priority user messages
- Drop messages for which the maximum delivery time has expired

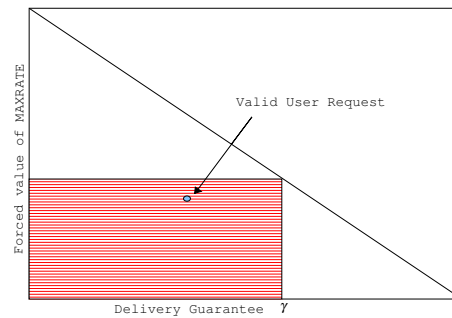


Fig. 8: MAXRATE versus delivery guarantee

4.2 User Acceptance

Accepting a user into the system can be described with a boolean function $f(\gamma, t, d, p)$. where γ is the value defined in section 3.1, r is requested throughput, d is the requested delivery guarantee, p is the percentage of time during which the request has to be fulfilled.

We will not describe how to map *qualitative* values to *quantitative* ones and we will assume that the requests is already in a form which can be numerically described.

The requested throughput is the minimum value that the THROUGHPUT property will have once the user is added to the system (refer to Table 1).

In general to assure throughput guarantee, the sum of all user THROUGHPUTs should not exceed the *MAXRATE* defined in Formula 4. In other words, we should not "sell" more throughput than we have available.

If introducing the new user causes the sum of all user throughputs to exceed *MAXRATE*, than the requested throughput will not be assured in the moment in which every user is sending at the maximum allowed rate (This rule can be further specified keeping in mind that not all users are allowed to use all connectors). Besides, *MAXRATE* may change dynamically, as described in section 4.1.2, depending on γ .

As we have seen previously the delivery guarantee (γ) and the forced value of *MAXRATE* in case of events are in contrast. High values of γ correspond to low values of *MAXRATE* and vice versa. Let assume then, as an example, that the forced value of *MAXRATE* and γ are bound by a direct proportional low. Though it can be risky to make this assumption since the value of *MAXRATE*, as seen in the previous section, may depend on other factors, it will not make any difference in our reasoning. Let's fix a value for γ , a minimal guarantee of delivery. The shaded region of the graph in Figure 8 represents the *guarantee area* for EMG: Each point of the region satisfies two conditions:

- The delivery guarantee is less than the one that EMG is providing;
- The value of *MAXRATE* is smaller than the one that EMG will provide in case of event (during normal operation, the actual utilization will be equal or bigger).

Consequently, each user request that maps to one of those points can be accepted in the system, since the system can provide both qualities or better one with no restriction in the time.

If the user request does not fit in the shaded region, the request has still some chances to be accept according to the value of p . High values of p will require that the distance by the point individuated by the request and the *guarantee area* is small. Vice versa, low values of p allow the request to be farther away from the area. Anyway, it is not easy to provide a valid representation and measure of p since EMG behavior (and the position of EMG in the area) depends on many external factors; even if we are able to constrain its behavior so that we can get a minimum utilization/delivery, it is not possible to go any further without analyzing other elements as, for example, the kind and burstiness of the incoming traffic, reliability of remote outgoing entities.

EMG Management

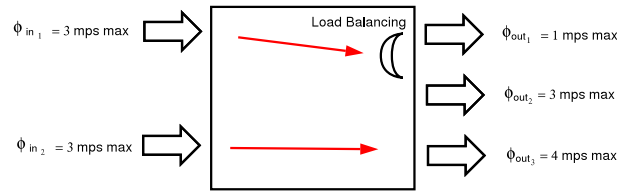


Fig. 9: EMG testbed

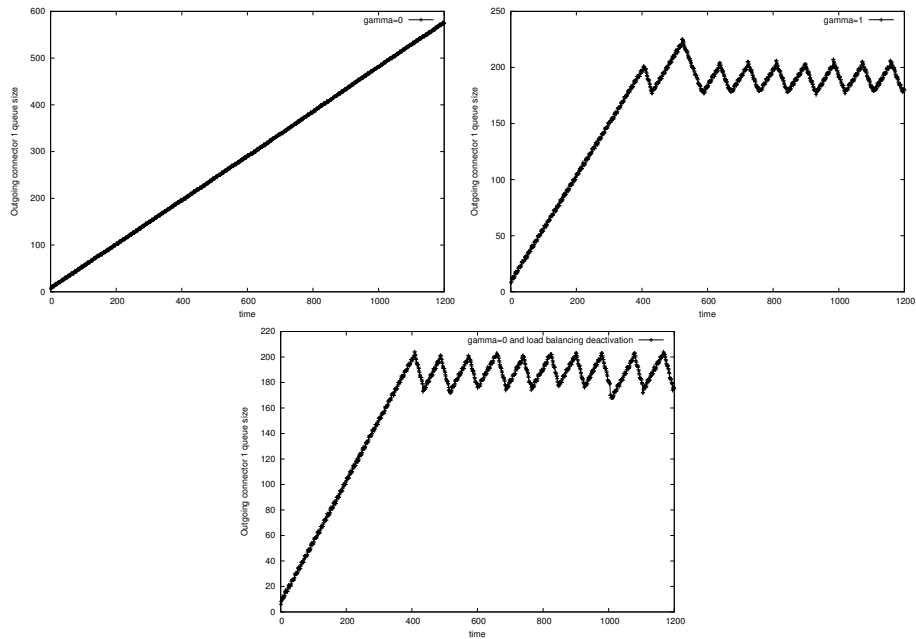


Fig. 10: Connector 1 queue analysis

5 Test Scenario

To verify the validity of the policy described in this paper, we realized a simple application communicating with EMG through MGP, able to read queue sizes, to read the number of messages per second accepted or delivered by each connector and to change the value of the THROUGHPUT parameter for each incoming connector.

Our managed EMG server has a simple configuration, two incoming connectors and three outgoing connectors. Please refer to Figure 9. Two of the outgoing connectors are running load balancing. The incoming connectors are initially accepting 3 messages per second each (produced by an external source) and the outgoing connectors are, respectively, delivering 1,3 and 4 messages per second (forced by the receiving entity, not shown in figure). The system is using static routing which will not affect the validity of the result but will make things easier. The first incoming connector is routed through the two outgoing connectors with load balancing and the other is routed to the last outgoing one. The system runs for 3600 seconds and the values of the queue sizes and message rates are logged every second. The queue limit not to be exceeded has been fixed to 200 messages.

5.1 Results

In Figure 10 is shown the queue behavior for the first of the outgoing connectors running load balancing when the policy manager is running with two different values of γ , $\gamma = 0, \gamma = 1$. Since connector 1 is not able to deliver the number of messages available the queue starts to grow. For $\gamma = 0$ no action is taken and then the queue persists growing (a). For $\gamma = 1$, when the queue exceeds the value of 200, the number of

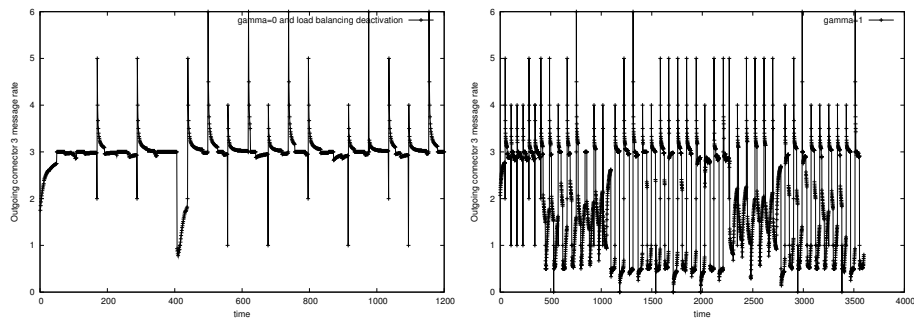


Fig. 11: Throughput analysis for outgoing connector 3

incoming messages accepted is forced to 0 allowing the queue to shrink to a value lower than 200 (b). The queue size is stable around the desired value.

The same result is obtained deactivating load balancing, as shown in (c). This result has been obtained because connector 2 was able to deliver the exceeding messages. In other situations, deactivating load balancing would not have been enough. It is interesting to look at Picture 11 where it is shown connector 3 throughput. When only load balancing is deactivated, the connector delivers at the maximum speed available. When *MAXRATE* is modified ($\gamma = 1$), the connector performs worse.

5.2 Comments

The system has proved to be successful, being able to contain the queue size around the desired value. Unfortunately EMG had not been originally designed to be a managed system: it reacts quite slowly to changes in THROUGHPUT or it doesn't provide tools to remote administer routing.

EMG and MGP are continuously being developed, opening the way to further policy based management capabilities.

6 Conclusion and future work

We have provide a management system which enhances the state of the art of the EMG server. It provides an important tool for the network manager who is now able to choose between different EMG behaviors without any having any low level knowledge of EMG mechanism. It is now also possible to make a basic evaluation of the quality of service provided by EMG.

A better representation of *MAXRATE* should be elaborated. The example presented in this report performs its duty but is not proved to be optimal at all. Furthermore, the value of *MAXRATE* has equally been shared among incoming connectors, while better implementation could use more sophisticated algorithms.

Dropping undeliverable messages has not been yet implemented because but it presents slightly more programming complexity.

References

- [1] GSM World - The website of the GSM Association, <http://www.gsmworld.com>
- [2] Verma D.C., "Policy-Based Networking, Architecture and Algorithms", 2001, New Riders, ISBN 1-57870-226-7
- [3] Nordic Messaging Technologies, EMG documentation, <http://www.nordicmessaging.se/support/docs/emg/>