SIP Server Overload Problem Statement

Victor Pascual
Tekelec
Berlin, Germany
victor@iptel.org

Abstract— **As the use of Session Initiation Protocol (SIP) servers as Next Generation Networks (NGN) telecommunication level devices increase, the need for effective overload control mechanisms is essential. Overload occurs when SIP servers have insufficient resources to handle all SIP messages they receive. This situation not only reduce the performance of a server but can also lead to a complete failure of the service it provides.**

The current overload control mechanism of SIP (the *503 Service Unavailable* response) is unable to prevent congestion collapse and may spread the overload condition throughout the network. Whilst work to address this topic is underway within Standards Development Organizations (SDO) as well in the research community, it's still in its infancy.

This document reviews the SIP server overload problem statement.

Index Terms— **SIP, overload, problem statement**

I. Introduction

Overloads generate calling rates much greater than the predictable daily profile to which the network can be economically dimensioned. Operators of traditional PSTNs have long recognized the need for providing overload controls to prevent the associated processing resources from being swamped. For instance, VIII shows the range of calling rate measurements taken from BTs network. From there, it can be observed that overload can exceed 64 times the systematic peak calling rate for six 15 minutes periods a year. While, during such an overload one might expect a large proportion of call attempts to fail, however, it would be unacceptable for the service to fail completely due to processing overload. In particular, emergency traffic and other important streams should be guaranteed at any time under any circumstance.

With the maturity of telecommunications networks, current fixed network, mobile network and the Internet are moving towards the convergence on to an IP-based network.

The Session Initiation Protocol (SIP) is an application-layer signaling protocol standardized by IETF for creating, modifying, and terminating multimedia sessions in the Internet.
SIP is capable of running on Transport Control Protocol (TCP), User Datagram Protocol (UDP), or Stream Control Transmission Protocol (SCTP) which are in turn carried over IP. SIP is actually the center of efforts for the previously mentioned telecommunications convergence. Indeed, major standards bodies including 3GPP, ITU, and ETSI have all adopted SIP as the core signaling protocol for Next Generation Networks (NGN) predominately based on the IP Multimedia Subsystem (IMS) architecture where SIP servers constitute the core components and are responsible for processing and routing signaling traffic.

An overload can lead to two types of congestion. One is a network congestion in which packets are lost in the IP layer. The other is a server overload in which a load is concentrated at a particular server. As the result, the server will be overloaded with a consequent degrade on the service quality, such as throughput and call setup delay. This document focuses the server congestion.

Server congestion is not a new problem. This type of congestion is observed in the PSTN where telephone traffic sometimes is concentrated at a specific telephone exchange. In such a scenario, server overload control is significantly helped by the hierarchical nature of network implementation. NGNs, however, usually have much flatter control architectures with large, uncertain and complex peer interactions. In the IP-based network, it also is expected that traffic created by SIP users is concentrated at a particular SIP server causing the server congestion. Overload is said to occur if a SIP server does not have sufficient resources to process all incoming SIP messages. These resources may include CPU processing capacity, memory, network bandwidth, input/output, or disk resources.

The SIP protocol provides a limited built-in mechanism for overload control through its *503 Service Unavailable* response code. However, since the cost of rejecting a SIP session usually cannot be ignored compared to the cost of serving a session, this mechanism cannot prevent overload of a SIP server and it cannot prevent congestion collapse. When a SIP server has to reject a large amount of arriving sessions, its performance collapses and, in addition, it may spread the overload condition throughout the network— this are the key observations that distinguishes the SIP server overload problem from others.

With sharp demand already seen in PSTN networks, the need for an automatic means of minimizing the effect of server overload in SIP signaling networks is paramount; specially in the service-level assured world of the telecommunications operators, where the user experience requires more than "best efforts".

This document examines and summarizes results from selected papers. While a **related work analysis is out of the scope** of this document, the present is aimed to provide a discussion on the SIP server overload control **problem statement** definition and invites other opinions and comment. In Section II definitions and abbreviations are included for the sake of understanding. In Section III the interaction between transport and application layers is described. A SIP server can be overloaded for many reasons, such as emergency-induced call volume or flash crowds generated by TV programs. In Section IV possible causes for overload are presented. Different categories of SIP server overload are described in Section V. Section VI describes the impact of overload on user behavior, server resources and the potential implications on other servers. In Section VII existing overload control mechanisms are described stressing the limitations of the current mechanisms. Finally, Section VIII summarizes the document and describes open issues that will need to be solved in future work.

II. Definitions and abbreviations

For the purpose of the present document, the following definitions and abbreviations apply.

A. Definitions

- Admission control: mechanism that accepts or rejects SIP requests on the basis of system load state of processing resource.
- Effective throughput: rate of admitted (and successfully completed) requests per second.
- Engineered throughput: number of requests the system should be able to deal with under normal operational conditions.

B. Abbreviations

CPU	Central Processing Unit
ETSI	European Telecommunications Standards Institute
ITU	International Telecommunication Union
IMS	IP Multimedia Susbsytem
3GPP	3rd Generation Partnership Project
DNS	Domain Name System
SIP	Session Initiation Protocol
SIP-T	SIP for Telephones
SIP-I	SIP with encapsulated ISUP
ISUP	ISDN User Part
BICC	Bearer Independent Call Control
DoS	Denial of Service
DDoS	Distributed Denial of Service
IP	Internet Protocol
IM	Instant Messaging
SLA	Service Level Agreement
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
SCTP	Stream Control Transmission Protocol
NGN	Next Generation Network
VoIP	Voice over IP
PSTN	Public Switched Telephone Network
ISDN	Integrated Services Digital Network

III. interaction between transport and application layers

SIP is capable of running on top of both unreliable and reliable transport protocols. This section summarizes the interaction between the application (SIP) and the transport layers.

Despite RFC3261 only mandates the implementation of Transport Control Protocol (TCP) and User Datagram Protocol (UDP) for SIP transport, emerging carrier grade implementations are also including the Stream Control Transmission Protocol (SCTP) in order to overcome some of the limitations of TCP.

A. SIP over unreliable transport protocols

Under congestion, SIP message discard or packet losses in a network might occur. SIP detects this failure by a time out and retransmits the failed message. Unreliable transport protocols simply forward the message from/to the IP layer and the application layer is responsible for detecting and recovering from the failure. RFC3261 defines retransmission procedures to improve the reliability of transmitting SIP messages. It defines two retransmission types- one is for the INVITE transaction (used for common call set-up) and the other is for non-INVITE transaction, which is extensible to most of the SIP extensions enabling new applications (e.g. IM, presence, etc.)

1) INVITE transaction retransmission

In the INVITE transaction, the client retransmits the original request at intervals of 0.5, 1.0, 2.0, 4.0, 8.0 and 16.0 seconds. After 32 seconds without any response, the client transaction ceases retransmission. If a provisional response is received, this time could be extended up to 3 minutes.

2) Non-INVITE transaction retransmission

The non-INVITE requests are retransmitted at intervals of 0.5, 1.0, 2.0, 4.0, 4.0, 4.0, 4.0, 4.0 and 4.0 seconds. After 32 seconds in total, the retransmission is ceased.

Although these retransmissions improve the message reliability, they increase the load applied to a SIP server and may affect the SIP signaling performance. Specially during overload conditions.

B. SIP over reliable transport protocols

In case of TCP or SCTP, SIP does not retransmit SIP messages. Transport-layer flow control protects from packet loss. However, the flow control makes the SIP message transmission delay large. If the SIP queue is full and a SIP message is received, there can be two scenarios. One is that the received message is discarded at the SIP queue, just like UDP. The other is that the received message waits until the SIP queue becomes free. Since SIP is a real-time protocol, it can be assumed that SIP messages that encounter the sending buffer full are usually discarded.

IV. Causes of sip server overload

A SIP server is said to be overloaded if one or more of its resources is having a value above some maximal limits. Going above these limits can be caused for several reasons (for instance, when it is offered more traffic that its designed

capacity) and it can degrade the system performance and even lead to a complete failure.

SIP server overload can occur for many reasons. The following subsections explain potential sources for signaling peaks in SIP networks.

A. Poor Capacity Planning

SIP networks need to be designed with sufficient numbers of servers, hardware, disks, etc. in order to meet the needs of the subscribers they are expected to serve. If this work is not done properly, the network may have insufficient capacity to handle even predictable usages.

B. Capacity Reduction

SIP server overload can be caused by reducing the available capacity. This may be caused by network equipment failures (e.g. the loss of a SIP server) or other kind of failures. These events happen very rapidly and it is difficult for the network to shed load in these circumstances.

1) Dependency Failures

A SIP element can become overloaded because a resource on which it is dependent has failed or become overloaded. In this case, even minimal traffic might cause the server to go into overload. Examples of such dependency overloads include DNS servers, databases, disks and network interfaces.

2) Internal Failures

Local failures could block the server from serving SIP requests. For example, software errors might deplete the available server memory (in a similar manner as a memory DoS attack).

3) External Failures

A SIP element can become overloaded when it is a member of a cluster of servers sharing the load, and one or more of the other members in the cluster fail. In this case, the remaining elements take over the work of the failed elements.

C. Avalanche Restart

This happens when a large number of clients all simultaneously attempt to connect the network at the same time. Avalanche restart can be caused by several events.

1) Reboots after a Blackout

Once the power is restored after a failure in a large metropolitan area, all the SIP user agents simultaneously power on and begin booting. They will all then connect to the network and register at the very same time, causing a flood of a registration attempts.

2) Failure of a large network connection

In this scenario there is a failure in a network device like the access router for a large enterprise. When the connectivity is restored clients will register all within a short period of time.

3) Failure of a SIP server

When a SIP server fails, if clients had all connected to the server with a connection-oriented protocol (e.g. TCP or SCTP), its failure will be detected followed by re-connection and re-registration to another server.

D. Flash crowds

A flash crowd occurs when an extremely large number of users all attempt to simultaneously make a call. This sudden increase in the number of calls may occur for many reasons, including:

1) Media stimulated events

Televotes for TV shows can generate high calling rates to particular small ranges of numbers. Such events can have a very rapid onset, with the calling rate increasing at a rate of 4 k calls per second per second over 6 seconds as observed in parts of BTs network. Often these events are known about in advance, so steps can be taken to prepare the overload controls. Also they are usually focused on a small range of destinations and some existing mechanisms might help to prevent the overload situation.

However, SIP signaling presents many unpredictable factors (unlike the Erlang traffic model) impacting the message size as well as the sending rate. IMS applications are tight-coupled with the SIP signaling, especially using XML for application data encapsulation. This changes the signaling traffic and hence dramatically increases the number of SIP signaling messages together with their sizes. For instance, presence and IM, which are two typical IMS applications, follow uncertain traffic models and generate messages which are relatively large (can be up to 1M bytes).

A recent example in China (2007) has set a record with SMS-based voting—high SMS traffic of 2 million voting SMS for favorite singer during a 15-minute interval.

2) Special dates/events

Special dates/events like New Year's may stimulate high calling rates to a large number of destinations. Even if it is known in advance, these events result in diffuse overloads with no specific destinations that can be used to target the anomalous load.

3) Disasters

Disasters may stimulate overload. Some times focused on a few numbers (emergency services and information lines) and others to a larger number of destinations. The former case is similar to media stimulated events, albeit with much less time to prepare the network. The latter is similar to the special dates/events scenario since there are no specific destinations. Furthermore, in such scenario the operator network may itself be damaged (e.g. by flooding) reducing capacity at the very time that it is exposed to these additional loads.

E. Non-legitimate traffic

Excess of useless traffic can cause an overload situation.

1) Unintended traffic

Software errors or misconfiguration can cause devices to generate and send unintentionally a higher amount of the traffic that they usually generate.

2) Denial of Service (DoS) attacks

The goal of a DoS attack is to disrupt service in the network. This can be done from a central source of traffic or through a distributed DoS (DDoS) attack and could be achieved by flooding a SIP server with a high number of useless requests (flooding attacks) or sending a number of requests through a server which will receive no answer from the final destinations (memory attacks).

a) Flooding attacks

An attacker generates a large number of SIP requests. Even if these requests end up being dropped by the server, the server will first have to parse and process them before deciding to either accept and forward, reject or drop them. Such attacks can misuse a large portion of the CPU and reduce the capacity for processing legitimate traffic.

b) Memory attacks

In this scenario the memory depletion is not caused by a high call arrival rate but due to excessive transaction delays. The attacker sends valid SIP requests that are forwarded by the server to destinations that do not answer properly. With each forwarded request the server will maintain some transaction state (from 32 seconds up to 3 minutes), before it can delete the state information. In addition, stateful servers could retransmit the request and hence increase the signaling load.

Non-legitimate traffic can be disguised as legitimate traffic so distinguishing between a DoS attack or a sudden surge in traffic due to some event is not always possible.

V. Categories of sip server overload

SIP server overload can be grouped into two main categories: server to server overload or client to server overload.

A. Server to server overload

In this scenario a relatively small number of upstream servers are sending a large amount of traffic to the same receiving server, putting it into overload.

B. Client to server overload

This type of overload occurs when a large number of clients overload the next hop server directly.

VI. Impact of overload

During periods of overload, the effective throughput of a SIP server can be significantly degraded. In fact, overload may lead to a situation in which the throughput drops down to a small fraction of the engineered throughput, exceeding customer tolerance of long set-up delays while blocking internal system resources as well as inducing a cascade effect which will overload other servers. In extreme overload situations it might even cause failures of the elements that are trying to process the traffic and lead to service discontinuity.

A. Impact on user behavior

User persistence can lead to a number of repeat attempts when service requests are rejected. This results in an increased number of requests which worsen he overload situation. Therefore, it is desired to maintain the effective throughput as high as possible subject to keeping response time small enough to preclude customers (and protocol retransmission mechanisms) repeating service requests.

B. Impact on resources

Resources include all of the capabilities of the SIP server used to process a request. There are two main resources that may get overloaded at a SIP server, CPU processing and memory. Other resources including I/O, and disk resources might also get overloaded.

1) CPU

CPU resources are used for parsing incoming messages and executing service specific tasks which might include validating the request format, writing logging information or evaluating a user's profile.

During overload, the effective CPU resources go down, since much of the capacity is spent just rejecting requests or treating load that it cannot actually process.

2) Memory

A SIP server can act either at stateful or stateless mode. When acting in stateful mode, which is the most common scenario, a SIP server needs to keep some state information describing on-going transactions/sessions for a certain period of time. If no protective measures are taken, all of the memory available to the server might be occupied in overload situations. In such a case the server would no longer be able to serve new calls.

3) Other Resources

Besides these two main resources several other resources are essential to the proper working of a SIP server. This includes the number of busy ports, ISDN trunks and disk space.

However, resources beyond CPU and memory are out the scope of this document.

C. Impact on other servers

As SIP has the feature of being able to select another server if service is lost at the current one, the overload or failure of a SIP server might cause even more load on the remaining servers. Unfortunately, the impact of overload on other servers and services can be difficult to predict.

VII. Overload control

The goal of SIP overload control is targeted to maximize the successful call setup rate while keeping the amount of used resources at the SIP server at predictable levels.

Reducing the load on the SIP server can be realized by means of the internal overload control by either dropping incoming requests or rejecting them. However, because dropping or rejecting requests takes processing effort (cost in terms of CPU usage), effective throughput at an overloaded resource must eventually fall as the load offered to it is increased; and ultimately it will spend all its time dropping or rejecting requests. To prevent this, it is necessary to reduce the offered

load to the level at which its effective throughput is maximized. This is achieved by means of the external overload control.

A. Internal overload control

Internal overload control is implemented locally on a SIP server. All resources that can get into processor overload have a function that can detect processor overload and an admission function that drops or rejects just enough incoming demand to maximize successful completion of admitted sessions. Such adaptive internal overload control is, in fact, the approach taken by most telcos' PSTN call processors.

a) Explicit request rejection

Basically, the overloaded server rejects a service request by sending an explicit response indicating that the request was rejected due to processing overload.
Figure 1 shows that the amount of resources used for serving requests is much higher than that for rejecting them.

b) Request drop

In this case, the overloaded server does not reject a service request but drops it instead. Figure 1 suggests that dropping incoming requests, consumes slightly less CPU at the SIP server than rejecting them. However, messages that get dropped due to overload can be retransmitted and hence increase the offered load for the already overloaded server. Therefore, the dropping approach will actually be more costly in terms of CPU usage at the end.

Figure 2 shows the effect of messages being dropped in the case the receiving SIP server is either overloaded or the network is lossy.

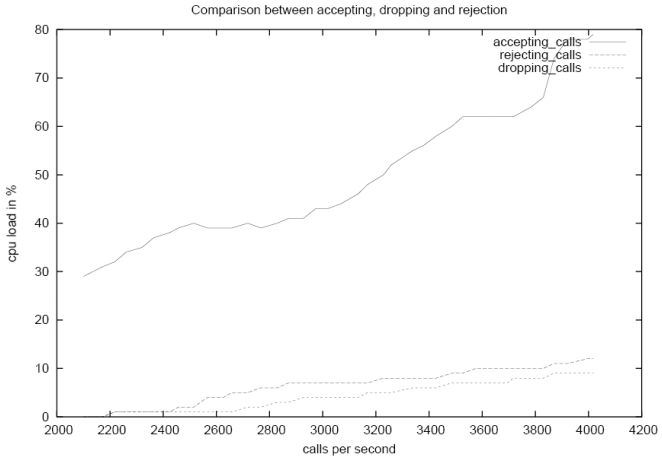

Figure 1— Comparison between accepting, dropping and rejecting requests. Source: Sisalem, D. and J. Floroiu; *Protecting VoIP Service Against DoS Using Overload Control*

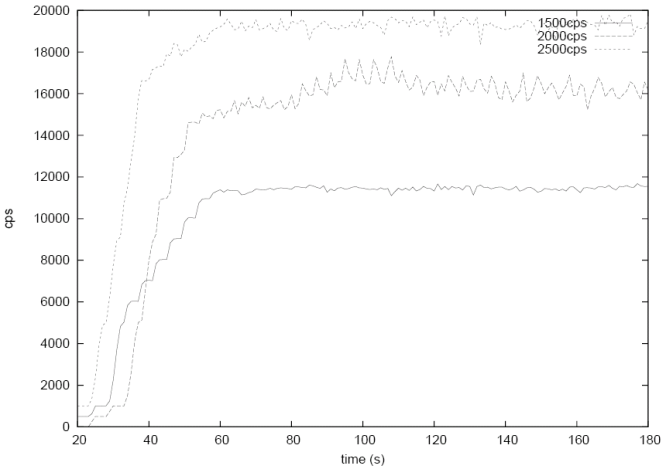

Figure 2— Number of retransmissions in the case where all requests are dropped. Source: Sisalem, D., Floroiu, J. and M. Liisberg; *VoIP Overload, a Senders Burden*

B. External overload control

As shown, internal (local, receiver-based) overload control techniques can provide a simple remedy for light cases of overload; however, as previously mentioned it is ineffective to treat higher amounts of load.

The goal of the external (distributed, feedback-based) control is to use an explicit overload signal to request a reduction on the offered load. This enables a SIP server to adjust the offered load to a level to match the resources capacity, whatever the capacity may be, and however many demand sources are causing the overload.

In this ideal situation, there would be no message retransmission due to timeout or message drop and no extra processing cost due to rejection. The server CPU power can be fully utilized to deliver its maximum session service capacity.

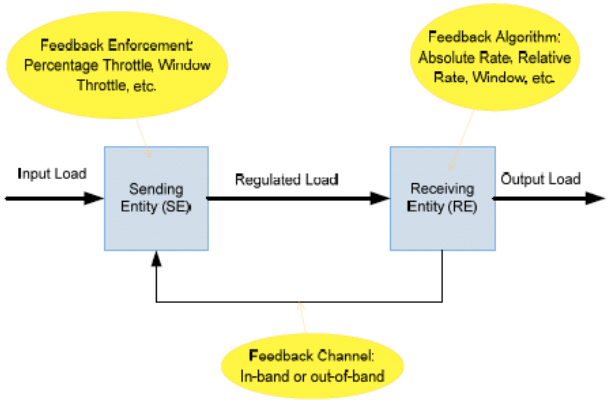

Figure 3— External overload control. Source: Shen, C., Schulzrinne, H. and E. Nahum; *Session Initiation Protocol (SIP) Server Overload Control: Design and Evaluation*

Through a feedback channel (which is usually hop-by-hop but could also be end-to-end), the receiving entity notifies the sending entity the amount of load that is acceptable. There are three main components in the model: feedback algorithm execution at the receiving server, feedback communication from receiving server to the sending node, and feedback

enforcement at the sending node. The following are four different types of overload control feedback algorithm.

a) Rate-based Overload Control

The key idea is to limit the request rate (requests per second) at which the sending node is allowed to forward to the SIP server. Each sending node could be assigned a different rate.

b) Loss-based Overload Control

This enables a SIP server to ask a sending node to reduce the number of requests it would normally send by a percentage. An advantage of using a percentage value is that the receiving server does not need to track the set of sending nodes or the request rate it receives from each sending node. It is sufficient to monitor the overall system utilization.

c) Window-based Overload Control

Here each sender maintains an overload window that limits the number of messages that can be in transit without being confirmed. Window-based overload control is inherently self-limiting; i.e. the sending node will stop sending traffic if it does not receive any feedback from an overloaded server.

d) On/Off Overload Control

This type of feedback enables a SIP server to turn the traffic it is receiving from a sending node either on or off.
Unfortunately, the On/Off approach results in a stop and go traffic behavior at the overloaded server which would lead to an oscillative and instable over all network behavior.

C. Existing SIP overload control mechanisms

Without overload control, messages that cannot be processed by the server are simply dropped. As mentioned, simple drop causes the corresponding SIP timers to fire, and further amplifies the overload situation.

SIP provides very basic support for overload. It defines the *503 Service Unavailable* response, which is sent by an element that is overloaded to inform an upstream element that it is overloaded. The objective is to provide a mechanism to move the work of the overloaded server to another server so that the request can be processed. The Retry-After header field, when present, is meant to allow a server to tell an upstream element to back off for a period of time, so that the overloaded server can work through its backlog of work.

To some extent the existing SIP 503 Service Unavailable mechanism with the “Retry-after” header is a basic form of the feedback mechanism and represents an on/off overload control approach.

D. Limitations with existing SIP overload control mechanisms

At the surface, the 503 mechanisms seems to be workable. Unfortunately, this mechanism is suboptimal for managing overload and a number of drawbacks and limitations have been identified.

a) Load Amplification

The principal problem with the 503 mechanism is that it tends to substantially amplify the load in the network when the network is overloaded, causing further escalation of the problem and introducing of congestive collapse. The 503 mechanism works well when a single element is overloaded. But when the problem is overall network load, the 503 mechanism actually generates more messages and more work for all servers, ultimately resulting in the rejection of the request anyway.

b) Underutilization

There are also examples of deployments where the network capacity is greatly reduced as a consequence of the overload mechanism. For example, a 503 response from a single server will make the sending entity believe that an entire cluster is overloaded.

c) Overload as a binary state

The Retry-After mechanism allows a server to tell a sending node to stop sending traffic for a period of time. The work that would have otherwise been sent to that server is instead sent to another server. The mechanism is an all-or-nothing technique, also known as *the On/Off Retry-After problem*. It interprets the overload as a binary state and does not recognize the fact that there are several degrees of overload. A server can turn off all traffic towards it, or none. There is nothing in between. This tends to cause highly oscillatory behavior under even mild overload.
It is important to observe that this problem is only observed for servers where there are a small number of sending nodes sending a large amount of traffic. If a server is accessed by a large number of clients, each of which sends a small amount of traffic, the 503 mechanism with Retry-After is quite effective when utilized with a subset of the clients. This is because spreading the 503 out amongst the clients has the effect of providing the server more fine-grained controls on the amount of traffic it receives.

d) Ambiguous Usages

RFC 3261 is unclear on the scope and do not provide any guidelines for the 503 retry-after duration. Hence the specific instances under which a server is to send a 503 are ambiguous.

VIII. Conclusions

Over time, the PSTN overload controls have been finely tuned to maximize network efficiency when the network is subjected to overloads. However, SIP has only limited capabilities to control network overloads with the use of a 503 retry message, which indicates that a network element is unable to process requests. A number of drawbacks and limitations have been identified with this mechanism.

The SIP server overload problem is interesting since the cost of rejecting a request is not negligible compared to the cost of

serving it. Also the various SIP timers lead to many retransmissions in overload which amplify the situation.

SIP servers need to incorporate mechanisms that would deal with the overload condition in a manner that would not lead to a complete service interruption. These mechanisms, irrespective of the overloaded resource's capacity and the number of sources generating the overload, should:

- Use dynamic parameter setting so that they take into consideration the cause of the overload as well the nature and state of the overloaded resource in its reaction to overload
- Keep response times and blocking probability low and enable the server to serve at a meaningful throughput under all circumstances— i.e. stabilizing the behavior the server during overload conditions and preventing a complete collapse of the service.
- Be aware of different importance levels of messages — be able to know which types of service requests may drop/reject and which may not is essential to enforce SLAs and regulation requirements. It is also important to note that users making standard calls do not expect mass media campaigns to interfere in their normal service experience.
- Be applied to servers using any transport protocol and to protect all kinds of server resources
- Prevent forwarding traffic to other servers that might be overloaded themselves and
- Work even if not all servers in the network support it — hence it could be introduced without requiring other servers to support overload mechanisms.

While internal overload detection and control is necessary to ensure element protection, when load increases beyond engineered limits, it is more efficient to throttle requests as the source rather than at the overloaded element. The external overload control might be used to limit offered load to an overloaded server. Anyhow, the internal overload control shall be effective enough to ensure server protection under DoS attacks.

References

For the purposes of this document, the following references apply:
Note: *While any hyperlinks included in this clause are valid at the time of writing this document, their long term validity cannot be guaranteed.*

[1] Whitehead MJ and Williams PM, *Adaptive Network Overload Controls*, BT Technology Journal, Vol. 20, No.3, July 2002

[2] Sisalem, D. and J. Floroiu; *Protecting VoIP Service Against DoS Using Overload Control*

[3] Sisalem, D., Floroiu, J. and M. Liisberg; *VoIP Overload, a Senders Burden*

[4] Shen, C., Schulzrinne, H. and E. Nahum; *Session Initiation Protocol (SIP) Server Overload Control: Design and Evaluation*

[5] V. Hilt; *Design Considerations for Session Initiation Protocol (SIP) Overload Control*, IETF draft-ietf-sipping-overload-design (Work in Progress), January 2009 http://tools.ietf.org/html/draft-ietf-sipping-overload-design

[6] Hilt, V. and I. Widjaja; *Controlling Overload in Networks of SIP Servers*

[7] Hilt, V. and I. Widjaja; *Session Initiation Protocol (SIP) Overload Control*, IETF draft-hilt-sipping-overload (Work in Progress), July 2009 http://tools.ietf.org/html/draft-hilt-sipping-overload

[8] M. Ohta; *Overload Control in a SIP Signaling Network*

[9] ETSI; *Architecture for Control of Processing Overload*, ETSI DTR/ TISPAN-02026_NGN, 2006

[10] MSF; *NGN Control Plane Overload and its Management*, MSF Technical Report, February 2006

[11] Noel, E. and C. Johnson; *Initial Simulation Results that analyze SIP based VoIP Networks under overload*

[12] M. Ohta; *Effects of Interaction between Transport and Application Layers on SIP signaling performance*

[13] Li, B., Wang, D. and S. Zhang; *Policy Based SIP signaling Management in IMS*

[14] J. Rosenberg; *Requirements for management of Overload in the Session Initiation Protocol*, IETF RFC 5390, December 2008 http://tools.ietf.org/html/rfc5390

[15] Zhang, Y., Zhang, Z., Zhang F. and Y. Li; *A new overload Control Algorithm of NGN Service Gateway*