Controlling Overload in Networks of SIP Servers

Volker Hilt, Indra Widjaja
Bell Labs/Alcatel-Lucent
volkerh@bell-labs.com, iwidjaja@bell-labs.com
Outline

Motivation

SIP Background

Performance Evaluation of SIP
  ▪ Congestion Collapse
  ▪ Cascading Effects of Overload
  ▪ 503 (Service Unavailable) Response Code

SIP Overload Control
  ▪ Local Overload Control
  ▪ Distributed Overload Control

Conclusion
Motivation

The Session Initiation Protocol (SIP) has become the main signaling protocol for multimedia sessions in the Internet and IP-based telephony networks.

- Many telephony service providers are deploying SIP-based networks.

SIP server networks can be exposed to a signaling load that is much higher than its processing capacity. Sources of overload include:

- Call floods: emergencies, TV commercials, ...
- Decrease in processing capacity: component failures, ...
- Avalanche restart: recovery from failures, misconfiguration, ...

Over-provisioning SIP servers is not a viable solution.

- Example: British Telecom reported incidents with an incoming load of 64 times the systematic quarter hour peak rate.
- Dimensioning for overload is not economical and often not possible.

A solution for overload control in SIP networks is needed.
SIP Background
The SIP Trapezoid

Alice

UAC

INVITE
100 Trying
180 Ringing
200 OK
ACK

INVITE
100 Trying
180 Ringing
200 OK
ACK

Session in Progress

BYE
200 OK

Bob

UAS

INVITE
180 Ringing
200 OK
ACK

BYE
200 OK

Proxy

SIP

atlanta.com

Media

biloxy.com
Simulation Model

**Discrete-event Simulator**

- Implements the full SIP transaction state machine.
- Each server has a finite input buffer (1000msg) and a message processor.
- Simulator has been verified in the IETF overload design team and with a SIP proxy.
- Goodput is measured in the number of calls per second terminated.

**Topologies**

- SIP server topology used in most simulations
  - Edge servers are connected to UAs.
  - Core server are connected to edge servers.

![SIP Server Topology](image)
Performance Evaluation of SIP Congestion Collapse

Scenario
SIP servers do not use overload control.
- Messages are dropped when input buffer is full.

Server capacity: ~160 cps

Observations
1. Goodput collapses when offered load exceeds server capacity.
2. Collapse persists even if load is reduced.
3. Collapse is resolved only after the offered load is further reduced.

SIP overload behavior is regenerative.
⇒ Congestion collapse
Performance Evaluation of SIP
Cascading Effects of Overload

Observations

1. Core server 1 becomes unresponsive as load exceeds its capacity.
2. Edge servers start to retransmit messages to core 1 and eventually become unresponsive as well.
3. Core 2 starts to retransmit messages to edge servers and becomes unresponsive.

⇒ Overload can spread through a network of SIP servers
503 Response Code
Background

SIP provides the 503 (Service Unavailable) response for overload control.
503 (Service Unavailable) response is returned by a server that is temporarily unable to process a request due to temporary overloading or maintenance.
The receiver of a 503 response can
- retry the rejected request at an alternative proxy or
- reject the request back to the UAC.
The 503 response can contain a Retry-After header.
- The Retry-After header prevents the receiver of a 503 response from forwarding requests to the overloaded server for a number of seconds.
503 Response Code
Performance Evaluation

Scenario
503 responses used for overload control.
- Server returns 503 responses
  - when input buffer reaches high watermark (800 msg)
  - until input buffer drops below low watermark (600 msg).
- Retry-After: 10 sec

Server capacity: S1: 120cps, S2: 160cps

Observations
S1 sends 503 responses as load is increased beyond its capacity.
- All traffic is now forwarded to S2, which also sends 503 responses.
  Utilization of S2 oscillates at a period of 10 seconds.
- 10 seconds do allow S2 to recover. S2 is hit by full traffic when 503 expires.
  S1 never recovers.

⇒ 503 responses can lead to traffic oscillation and congestion collapse.
SIP Overload Control

Server monitors resource usage and determines the load level that can be accepted without reaching overload.
Feedback is conveyed to an entity that adjusts load.

Local Overload Control
Servers locally reject messages that exceed their processing capacity.
- Rejecting messages is less expensive than processing them and stops retransmissions.

Hop-by-Hop Overload Control
Feedback is sent to next upstream neighbor.
Neighbor adjusts the number of messages forwarded.

End-to-End Overload Control
Feedback is accumulated across a path from source to destination, taking the minimum of all feedback values.
Ingress server adjusts load on a per-destination-basis.
Performance Evaluation
Local Overload Control

Scenarios
- Case 1: no overload control
- Case 2: watermark-based - rejected requests are not retried at other core server.
- Case 3: watermark-based - rejected requests are retried at other core server.
- Case 4: occupancy algorithm (OCC) - rejected requests are not retried.

Observations
1. Watermark-based OC allows a server to reach a high goodput followed by a collapse.
2. Retrying requests at an alternative server decreases performance.
3. OCC provides a gentle decline in goodput and maintains a low delay.

⇒ Local overload control can alleviate overload.
Performance Evaluation
Distributed Overload Control

Scenarios
Occupancy algorithm to determine desired load.
Feedback is conveyed upstream in a SIP response header.
Topology with 4 servers on a path.

Observations
1. Hop-by-hop overload control provides a high goodput and significantly outperforms local overload control.
2. End-to-end overload control outperforms hop-by-hop overload control but comes with a higher complexity.
   ⇒ Distributed overload control can prevent a congestion collapse
Conclusion

Our comprehensive study of SIP server overload revealed key problems.

SIP servers are vulnerable to congestion collapse.

The existing overload control mechanism of SIP is considered harmful.

The SIP protocol can be protected against overload.

Local overload control provides as simple remedy for light cases of overload.

Distributed overload control protects SIP servers against congestion collapse.

SIP Overload Control has been taken on by the SIPPING WG in the IETF.

A design team has been formed to evaluate overload control mechanisms.

A requirements draft and design considerations draft are SIPPING WG items.