



**MSF Technical Report
MSF-TR-ARCH-007-FINAL**

NGN Control Plane Overload and its Management

Authors: British Telecommunications plc
Ian Jenkins
ian.wg.jenkins@bt.com

Acknowledgements:
James McEachern, Nortel
Tom Taylor, Nortel
Naseem Khan, Verizon

www.msforum.org
February 2006

Abstract:

This white paper examines the types of overload scenarios encountered within traditional networks and concludes that similar traffic behaviour, requiring overload management, will be required within Next Generation Networks (NGNs). It looks at the principals of overload protection within real-time communications servers and the types of overload management methods that are available to optimise server performance. This white paper notes that the distributed computing nature of NGN architectures such as the 3GPP IMS, has an increased detrimental impact on overload behaviour and looks at the overload management capability of the main protocols that are commonly used. It finally comments on the work in progress within ETSI TISPAN to address the issues discussed.

Disclaimer:

The following is a technical report of the MultiService Forum. The information in this publication is believed to be accurate as of its publication date. Such information is subject to change without notice and the MultiService Forum is not responsible for any errors or omissions. The MultiService Forum does not assume any responsibility to update or correct any information in this publication. Notwithstanding anything contained herein to the contrary, neither the MultiService Forum nor the publisher make any representation or warranty, expressed or implied, concerning the completeness, accuracy, or applicability of any information contained in this publication. No liability of any kind whether based on theories of tort, contract, warranty, strict liability or otherwise, shall be assumed or incurred by the MultiService Forum, its member companies, or the publisher as a result of reliance upon or use by any party of any information contained in this publication. All liability for any implied or express warranty of merchantability or fitness for a particular purpose, or any other warranty, is hereby disclaimed.

For additional information contact:

MultiService Forum
39355 California Street, Suite 307, Fremont, CA 94538
+1 (510) 608-5922
+1 (510) 608-5917 (fax)
info@msforum.org
<http://www.msforum.org>
Copyright © MultiService Forum 2005

1 Abbreviations

IN.....	Intelligent Network
PSTN.....	Public Switched Telephone Network
NGN	Next Generation Network
IETF	Internet Engineering Task Force
SIP	Session Initiation Protocol
HTTP	Hyper-Text Transfer Protocol
DNS.....	Domain Name System
COPS	Common Open Policy Service
SNMP	Simple Network Management Protocol
SMPP	Short Message Peer-to-Peer
SAML	Security Assertion Mark-up Language
LDAP	Lightweight Directory Access Protocol
SOAP	Simple Object Access Protocol
INAP	Intelligent Network Application Protocol

2 Introduction

Operators of traditional PSTNs and INs have long recognized the need for providing overload controls to prevent the associated processing resources from being swamped. Such situations are provoked by unusually high demands during events such as disasters and mass media stimulation of user response.

As the IT world of computing and data networking converges with that of telecommunications, the management of transaction peaks has been largely ignored. Is there still a need for such techniques amongst the technologies of the NGN or is this a symptom of NGN immaturity? The Internet has proved the scalability of its technology and its cost effectiveness whilst it can support similar types of service as an NGN. A trawl of the IETF Working Groups suggests that much thought has been, and is being, given to the management of bandwidth and router congestion in the bearer plane (c.f. RFC3124 'The Congestion Manager'), whilst there is no work addressing the management of host or server processing overload in the control plane. This suggests that the Internet community sees no need for overload control at the control plane, perhaps viewing it as an 'edge intelligence' rather than a 'network intelligence' problem to be solved.

Another difference between the technologies born in the Internet community compared to the telco world is that of the intended quality of service to be supported. The Internet world is a loose federation of network operators providing a 'best efforts' service over shared data network infrastructure. In the Internet, session set-up times and server responses can vary wildly and high usage peaks can provide abnormal and unexpected behaviour. In the computing world, as demand increases, the user experience is of slower or delayed service. As transaction demand outstrips processing, queues overflow and messages are discarded, causing unexpected service response or failure. Traditional telcos however, operate self-contained and highly managed, real-time networks in order to provide a deterministic quality of service to their users. If resources

are not available, the user is positively rejected (e.g. telephone calls receive a supervisory tone or announcement, such as 'congestion'). If processing loads start to cause servers to produce delays approaching their service quality limit, load is gracefully and proportionally rejected to maintain the quality of the successful events.

However NGN architectures aimed at delivering telco strength services have adopted and onward developed many of the protocols that have originated from the Internet world where quality of service levels are not guaranteed and unpredictable delays and 'abnormal ends' can be periodically expected. What is more, it is outside the remit of the associated standards bodies, such as the IETF, to harden these Internet protocols for use in a telco environment, although the IETF does retain change authority over any protocol extensions that may be proposed.

NGNs rely on real-time functions such as authentication, location determination, presence information, user registration, real time pricing, bandwidth management etc. These functions will generally be distributed over a number of servers. The protocols used between these servers could comprise any, or all, of the following: SIP, HTTP(S), Radius, Diameter, DNS, COPS, SNMP, SMPP, SAML, LDAP, Parlay, Java, SOAP, H.248, SIP-I, INAP etc. Although some of these protocols (e.g. SIP and HTTP) do provide response or status codes that could be used to indicate processing overload, none explicitly specifies an overload control mechanism. The same conclusion also seems to apply to the following non-IETF protocols: H.323, SOAP, SAML, SMPP, and Parlay.

3 Overview of Traffic Behaviour in Telephone Networks

Telephone networks were originally only designed around providing bearer capacity between different nodes on a network in order to meet a given probability of blocking during the average busy hour. However, as technology increasingly relied on common computing resources for call processing in both central office switches and intelligent network service control points, it soon became clear that purely dimensioning these components for the busy hour was insufficient.

Telephony traffic patterns have changed as services that support direct response to broadcast advertising and TV audience participation became more common. These produce highly focused traffic on a small group of telephone numbers terminating on specific groups of answering locations, such as call centres and automated announcement equipment. Also disaster events such as flood, earthquake, plane crashes or terrorist attacks produce both highly focused traffic patterns, as people phone emergency services and information lines and more regionally focused traffic, as people call each other to check if they are OK. In both of these cases, the caller behaviour and the associated traffic patterns significantly alter as the target service or network destination becomes unavailable through being busy or through network congestion. In these cases, repeat calls sharply increase within seconds. Call control servers can experience demand increasing to 20 – 30 times the normal busy hour design limit, with a rise time of 10s of seconds. Such patterns rule out reliance on a manual network management response. Even successful calls to services such a 'televoting' pose significant processing problems, as average call hold times can be as

low as 3-4 seconds and traffic surges peak and die typically during a 5 minute window around the display of televote access numbers on television.

Whilst it is essential that the call processing servers in such an environment do not break down under the load, it is also important to note that users making standard telephone calls, and on NGNs multi-media sessions, do not expect mass media campaigns to interfere with their normal service experience. In addition, regulators typically require that emergency and government services are not affected by these events.

4 Principals of Server Overload Management

Modern call processing servers have protection mechanisms built into themselves to prevent them from crashing under traffic peaks that are outside of their normal operating limits. These involve some form of input message, 'front end' that assesses the request demand against the available processor capacity. One simple mechanism is to monitor message queue lengths and to reject an increasing proportion of new service requests as the queue length extends past various thresholds. However as the load increases, the processing expended on rejecting new requests rises so that less processing is available for successful calls. This mechanism usually provokes even more demand as users repeat their request for service. As the demand increases beyond normal processing maximum, the server starts spending more and more processing time in rejecting requests so that the total successfully processed requests declines (fig. 1).

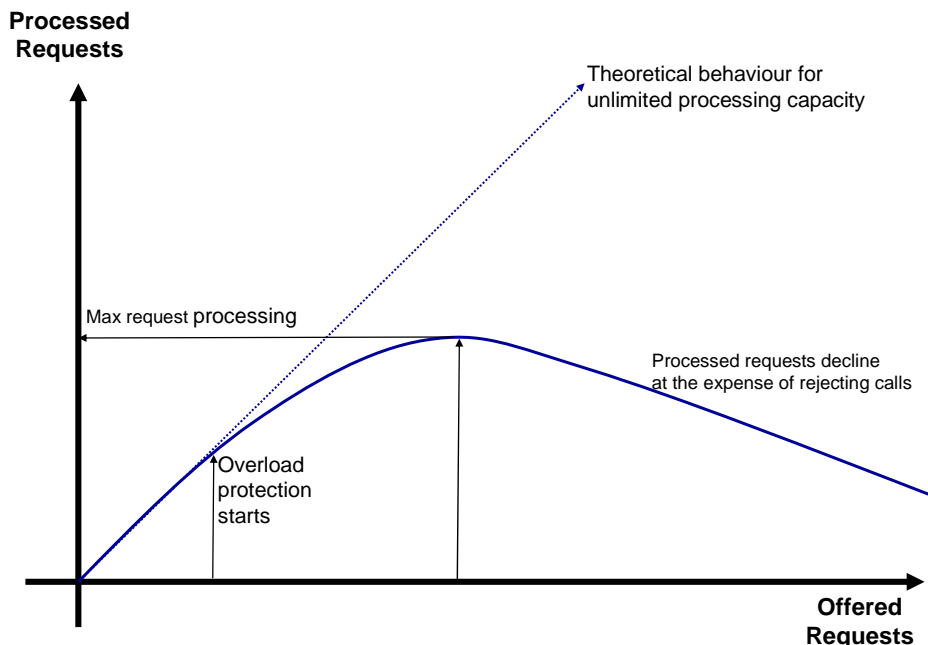


Figure 1: Characteristics of a Server using Request Rejects to Protect Against Processor Overload in Order to Maintain System Response Times.

This mechanism on its own is indiscriminate as no analysis is made regarding the importance of the rejected call (e.g. emergency calls should be treated differently to

other calls) or whether a rejected call had any more chance of success than any of the calls accepted by this mechanism (e.g. a request associated with a focused overload is less likely to succeed than one that is not associated with a focused overload). The ideal is to apply controls that cause the clients to restrict demand only to the overloaded services or destinations, e.g. only calls to overloaded call centres are affected. This leaves capacity at the server for processing requests that are likely to complete successfully.

Another desirable characteristic is to be able to mark requests associated with emergency and government services as having priority. To discriminate against normal requests in favour of these priority services is straightforward in that the restriction mechanism in the target server can choose not to discard priority requests. This can be detected by specialist fields in the signalling or by the service address (e.g. emergency number). Whilst this works well in the face of overloads caused by requests to non-priority services, this does nothing for server overloads caused by requests to emergency services such as is encountered during a common disaster. To cope with this, a second pass analysis is required to establish if the priority calls need further restriction.

In some cases server overload comes from a small number of clients, in perhaps even a single client. In those cases, feedback to the client, either implicit or explicit, can readily enhance overload protection.

A logical overview of a typical server overload protection mechanism is shown in figure 2. Here the demand from incoming requests is monitored against the processing available to the serving application. An Overload Detection function detects when the processing available to the application passes various thresholds, instructing the Reject function to mark a proportion of new requests as needing rejection. If the protocol supports a reject message, the server Application will inform the source; if not the request is simply discarded.

If the protocol used between the client and the server supports a reject message, the client can use this to assess the overload control state of the server itself (Fig 3). By monitoring the rate of rejects from the server, a client can choose to apply a restriction to the message rate it sends in order to adapt to the server's load state. In addition, the client Monitor function can analyse patterns in the reject messages to establish if requests to a particular service or services is producing an overload. Therefore, this method not only responds to the overload protection mechanisms in the server front end but also to the server application rejecting the requests due to the end resource being unavailable or busy (e.g. all the call centre lines are engaged). This has the added advantage of applying restrictions to new requests for an individual service based on both the state of the end point and the server load, thereby limiting the load expended on unsuccessful requests and leaving more processing for other services.

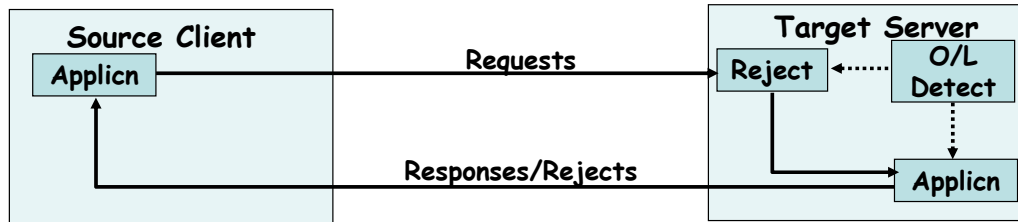


Fig.2 Overload Protection in Target Server Only

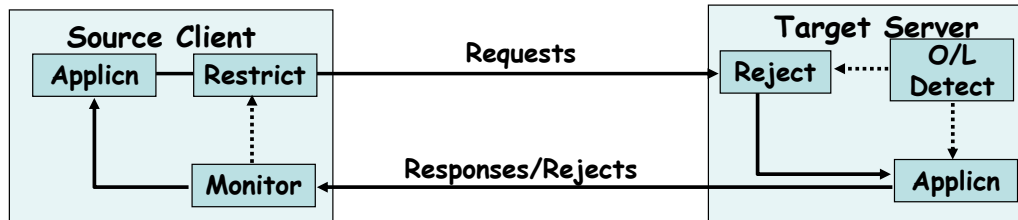


Fig.3 Overload Restriction in Client Driven by Monitoring Target Rejected Responses

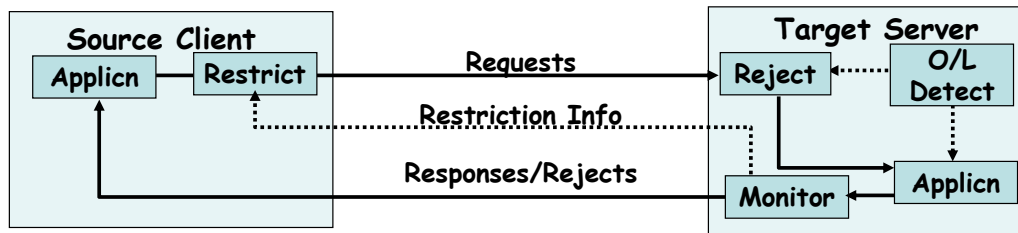


Fig.4 Overload Restriction in Client Driven Explicitly from the Target

Whilst this seems ideal there are two drawbacks with this approach. The first is that it only works with protocols that have a suitable reject message. The second is that the model of the client Monitor function that assesses the state of the server / end points becomes less precise in a network where there is a moderate number of clients to the same server. For example, a call agent may serve an endpoint on an access gateway that is the subject of a focused overload control. Calls may be directed to that terminating call agent (i.e. the server) from many other originating call agents (i.e. the clients). With Monitor functions in all the clients, it is difficult for them to assess what the optimum send rate should be as the information is diluted amongst all the clients.

Another approach is to move the Monitor function to the server (fig4). This means that one Monitor function sees all of the responses from the server Application, but now needs a mechanism to pass restriction information to the client(s). This is the same technique often used in intelligent networks where, within the INAP signalling the Service Control Point (server) has the ability to apply Call Gapping, a mechanism for restricting the rate at which new session requests can be made to numbers, number ranges or complete services at the Service Selection Points (clients). This approach has the advantage of knowing all the information about the server condition and what is causing it to overload. But not every protocol has a restriction mechanism built in to its message set. As the server is aware of the complete overload state, the restriction decisions can be optimised. For the numbers of components within the control plane of a carrier scale network this should work well. However, where there are a very large

number of clients, e.g. ½ million individual Integrated Access Devices or SIP terminals parented on one Call Agent, communicating a restrict message to such a large number is difficult without some broadcast method and defining a restriction value that is meaningful at non-aggregated traffic points is difficult to imagine. This implies that in this situation, proxy functions are required to gather together enough client sources so as to get a sufficient concentration of service requests to be able to perform some meaningful statistical restriction. However, it should also be noted that a server based monitor function requires additional processing resources in the server to collect, analyze, and feedback overload information. As overload increases, the amount of additional processing required increases correspondingly which can lower the processing maximum shown in figure 1 and if the source clients do not respond to the restriction messages, there can be a steeper decline in processed requests as the offered load increases than without the overload control being present. Therefore the mechanisms used for realising the server based monitor must be engineered to have an acceptable performance impact for the overload control benefit it delivers.

For a server based Monitor function with a backward restriction communication mechanism, originating load can be managed even if the communications protocol between the client and the server does not support reject messages as any message discards within the server can be processed by its Monitor function.

5 Implication for NGN Architectures

Much of the work on NGN architectures is based on the work of the 3rd Generation Partnership Project (3GPP) for mobile NGNs. 3GPP has defined the IP Multimedia Subsystem (IMS) as its logical architecture for the NGN control plane. Whilst logical, early implementations and the way the logical functions operate with each other, particularly in user roaming scenarios, more than suggests that the IMS will be implemented as a complex web of distributed inter-server communications.

Overload control in PSTNs is significantly helped by the hierarchical nature of network implementation. NGNs, however, usually have much flatter control architectures. Whilst they define a set of components that have a hierarchical relationship, from the perspective of managing a distributed server estate, the interactions consist of complex and largely peer interactions.

Trying to maintain such a complex set of capacity relationships in a state of variable capacity growth will be difficult in itself. In real network deployments, ensuring correct processing ability will be problematic as the various component capacities will vary as software updates are loaded or as components are lost through unplanned outages. If future user behaviours follow past experience, un-planned outages are likely to be provoked at a time of high calling rates; just the time when a step decrease in processing capacity is most detrimental. With the sharp demand rise times already seen in PSTN networks, the need for an automatic means of minimising the effect of server overload is paramount. It is also required if the user experience is to be more than 'best efforts' and operator revenues are to be maximised.

An NGN control plane solution will consist of many interoperating types of server (e.g. call agents, SIP proxies, application servers, bandwidth managers, profile servers, etc). This situation is further complicated as each server type will consist of a number of separate servers or server clusters, probably in a n+1 geographically resilient configuration. The impact of overload on one server causing, at worst server outage and at best reduced performance, on all other servers and services can be difficult to predict. As many of the communication protocols used in NGNs have the feature of being able to select another server if service is lost at the current one, the nightmare scenario is that an overload in one server which is detected as failure by the clients causes them to switch demand elsewhere, producing a knock on effect through the control plane. This has potential for a major impact on all services. Therefore, in a real-time, distributed computing architecture, as used in NGNs, a mechanism for managing the demands on server nodes, in order to prevent uncontrolled demand flows, is essential for maintaining service levels and customer experience. Whilst co-operative mechanisms that provide dynamic, graceful and optimal load management is required to maximise the effectiveness of the network, this does not preclude the need for an autonomous protection mechanisms at each node in case any co-operating node misbehaves.

It should also be noted that in some areas co-operative overload techniques are not appropriate. For example overload control between SIP terminals and call agents raises issues of scalability, implementation and trust. The trust issue is of particular relevance where the choice and operating environment of a SIP node is outside the control of the network operator such as SIP clients provided by the end user or SIP nodes used in other interconnecting networks. For these untrusted nodes, functionality is required in 'edge' nodes so that they can autonomously protect the core components from misbehaviour such as DoS attacks and malformed signalling messages. However, even with these edge protection measures, some autonomous protection mechanisms may be required at each node to deal with threats such as distributed denial of service (DDoS) attacks, or "trusted" nodes that have been compromised in some way. In the MSF architecture the Session Border Control at the access network edge or at the interconnection of the network core with peer networks generally limits the scope of overload control. However, at the inter network operator level there may be a sufficient level of trust to also implement co-operative overload control.

Fortunately, some control mechanisms do exist in the protocol soup used by NGNs. H.248 has packages (.10 & .11) for managing demand on media gateways. This does not have any discrimination as to the service type and only copes with terminating demand to the gateway, not with originating demand to a call agent. However, it should be noted that the .10 package requires some companion definition of nodal behaviour to achieve a deterministic behaviour and inter-operability and the .11 package has implementation complexity issues. These issues will be the subject of a further work item within the MSF.

SIP is one of the prime NGN protocols, but whilst it has no overload controls it does have rejection messages that could be harnessed. However, nothing in the standards specifies how these should be used. A SIP feature also exists for a SIP Invite to be rejected with a 'Retry After n secs' Message but has no clear definition for its use. An

originating client / proxy can try elsewhere with its requests until the retry timer expires. This mechanism seems to be best suited for taking a server out of use rather than for overload management. If Retry After were used as an overload control, it would produce a rather undesirable effect as it would control demand by setting the server availability to either on or off. This would produce demand surges with a minimum granularity of one second. This characteristic for demand flow could even produce more extreme user 'retry session setup' behaviour. In addition, this mechanism could create additional server load (generating the Retry After messages) when the server was already overloaded.

SIP-I/SIP-T carries C7 ISUP signalling which in itself contains overload control messages between the server and client, but this link by link control is lost when the signalling is passed through 'ISUP unaware' SIP servers (e.g. Session Border Controllers, pure SIP proxies, or an S-CSCF).

Whilst the Diameter protocol can respond to requests with a Server Too Busy message, no client behaviour is defined to allow for limiting of requests and Radius cannot even signal this! Parlay can signal load levels, but there is no definition as to what to do with them and SOAP has a faultcode = 'server' to indicate busy, but again there is no defined client behaviour to use this information. Therefore, judging from the lack of ability for these protocols to cope with overload management, it appears that to-date, little thought, let alone design effort, has been applied to overload management of NGNs, even though these protocols are the very arteries of a NGN control plane.

6 Work in ETSI TISPAN

The lack of overload management in NGNs is being addressed by the European Standards body ETSI within the architecture study group of its TISPAN project. The traditional approach for solving overload management issues has to been to define rate limiting control fields in messages that are specific to an individual application control protocol, such as has been done for ISUP, INAP and AIN in the past. The overload control group have taken a different approach to that of the past by:

- firstly defining a generic, robust and flexible overload control architecture with well defined behaviours, defining the overload detection function within the server and the request restriction function in the client

and

- secondly specifying a separated common overload control protocol that can operate between the client and the server. (The TISPAN protocols group have yet to define how this overload management protocol will be realised.)

This approach has a number of advantages:-

1. It defines common overload management behaviour across multiple control plane components, even when they communicate with different protocols.
2. It avoids having to redesign existing protocols and get them ratified in their associated standards groups.

3. It allows suppliers to provide overload management as a 'bolt on' extra for their products.
4. As an overload management protocol could be maliciously spoofed to perpetrate a denial of service attack, it allows a separate overload management flow to be secured via its own encrypted channel if required, without impacting the performance of the main control protocol.

The architecture provides for differential marking of the target service that is the cause of an overload in demand, so as to gracefully *back-off* requests to that service allowing other service requests to be processed by the server.

The overload control work in TISPAN is a valuable step toward providing overload control in NGN networks. Application of this approach to some overload problems, particularly within the control plane core, will be relatively straightforward. In the case of some NGN scenarios, (e.g. untrusted clients, or flexibly interconnected meshes of servers) the application of the TISPAN work will require further study.

In addition to work on overload control, the TISPAN protocols group have also been defining how SIP will carry priority markings to give precedence to session establishment for emergency services.

7 Conclusion

If major carriers are to use NGNs for both new applications and to support existing services (such as PSTN with its attended characteristics), an overload management mechanism is required. The approach common to NGN architectures such as MSF, 3GPP, and ETSI TISPAN is to distribute various functions onto different server groups. This exacerbates the effect of server overload on the rest of the control plane components that use it and their associated network services. Generally, the issues of overload management are not sufficiently addressed within NGN design, with attention being focused on the functional rather than the non-functional issues. Whilst work to address this is underway within ETSI TISPAN, it is in its infancy. Furthermore, the importance of this area of work in supporting the overall success of carrier scale NGNs is still to be widely appreciated.