Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 20, 2012

E. McMurry
B. Campbell
Tekelec

June 18, 2012

Diameter Overload Control Requirements
draft-mcmurry-dime-overload-reqs-01

Abstract

When a Diameter server or agent becomes overloaded, it needs to be able to gracefully reduce its load, typically by informing clients to reduce sending traffic for some period of time. Otherwise, it must continue to expend resources parsing and responding to Diameter messages, possibly resulting in congestion collapse. The existing mechanisms provided by Diameter are not sufficient for this purpose. This document describes the limitations of the existing mechanisms, and provides requirements for new overload management mechanisms.

Table of Contents

1. Introduction

When a Diameter [I-D.ietf-dime-rfc3588bis] server or agent becomes overloaded, it needs to be able to gracefully reduce its load, typically by informing clients to reduce sending traffic for some period of time. Otherwise, it must continue to expend resources parsing and responding to Diameter messages, possibly resulting in congestion collapse. The existing mechanisms provided by Diameter are not sufficient for this purpose. This document describes the limitations of the existing mechanisms, and provides requirements for new overload management mechanisms.

This document draws on [RFC5390] and the work done on SIP overload control as well as on overload practices in SS7 networks and studies done by 3GPP.

Diameter is not typically an end-user protocol; rather it is generally used as one component in support of some end-user activity. For example, a WiFi access point might use Diameter to authenticate and authorize user access via 802.11. Overload in a network that uses Diameter applications will likely spill over into the end-user application network. The impact of Diameter overload on the client application (a client application may use the Diameter protocol and other protocols to do its job) is beyond the scope of this document.

This document presents non-normative descriptions of causes of overload along with related scenarios and studies. Finally, it offers a set of normative requirements for an improved overload indication mechanism.

1.1. Causes of Overload

Overload occurs when an element, such as a Diameter server or agent, has insufficient resources to successfully process all of the traffic it is receiving. Resources include all of the capabilities of the element used to process a request, including CPU processing, memory, I/O, and disk resources. It can also include external resources such as a database or DNS server, in which case the CPU, processing, memory, I/O, and disk resources of those elements are effectively part of the logical element processing the request.

Overload can occur for many reasons, including:

Inadequate capacity: When designing Diameter networks, that is, application layer multi-node Diameter deployments, it can be very difficult to predict all scenarios that may cause elevated traffic. It may also be more costly to implement support for some scenarios than a network operator may deem worthwhile. This

results in the likelihood that a Diameter network will not have adequate capacity to handle all situations.

Dependency failures: A Diameter node can become overloaded because a resource on which it is dependent has failed or become overloaded, greatly reducing the logical capacity of the node. In these cases, even minimal traffic might cause the node to go into overload. Examples of such dependency overloads include DNS servers, databases, disks, and network interfaces.

Component failures: A Diameter node can become overloaded when it is a member of a cluster of servers that each share the load of traffic, and one or more of the other members in the cluster fail. In this case, the remaining nodes take over the work of the failed nodes. Normally, capacity planning takes such failures into account, and servers are typically run with enough spare capacity to handle failure of another node. However, unusual failure conditions can cause many nodes to fail at once. This is often the case with software failures, where a bad packet or bad database entry hits the same bug in a set of nodes in a cluster.

Network Initiated Traffic Flood: Issues with the radio access network in a mobile network such as radio overlays with frequent handovers, and operational changes are examples of network events that can precipitate a flood of Diameter signaling traffic, such as an avalanche restart. Failure of a Diameter proxy may also result in a large amount of signaling as connections and sessions are reestablished.

Subscriber Initiated Traffic Flood: Large gatherings of subscribers or events that result in many subscribers interacting with the network in close time proximity can result in Diameter signaling traffic floods. For example, the finale of a large fireworks show could be immediately followed by many subscribers posting messages, pictures, and videos concentrated on one portion of a network. Subscriber devices, such as smartphones, may use aggressive registration strategies that generate unusually high Diameter traffic loads.

DoS attacks: An attacker, wishing to disrupt service in the network, can cause a large amount of traffic to be launched at a target element. This can be done from a central source of traffic or through a distributed DoS attack. In all cases, the volume of traffic well exceeds the capacity of the element, sending the system into overload.

1.2. Effects of Overload

Modern Diameter networks, comprised of application layer multi-node deployments of Diameter elements, may operate at very large transaction volumes. If a Diameter node becomes overloaded, or even worse, fails completely, a large number of messages may be lost very quickly. Even with redundant servers, many messages can be lost in the time it takes for failover to complete. While a Diameter client or agent should be able to retry such requests, an overloaded peer may cause a sudden large increase in the number of transaction transactions needing to be retried, rapidly filling local queues or otherwise contributing to local overload. Therefore Diameter devices need to be able to shed load before critical failures can occur.

Diameter depends heavily on The "Authentication, Authorization, and Accounting (AAA) Transport Profile" [RFC3539], which states assumptions about the scale of AAA services which may be incorrect for current uses of Diameter. In particular, the document suggests that AAA services will typically be low volume and that traffic will typically be application-driven. Section 2.1 of that document uses an example of a 48 port NAS. However, Diameter is commonly used in large-scale mobile data environments, where a typical client could be a packet gateway that serves millions of users, and generates Diameter messages at network-driven rates.

1.3. Overload vs. Network Congestion

This document uses the term "overload" to refer to application-layer overload at Diameter nodes. This is distinct from "network congestion", that is, congestion that occurs at the lower networking layers that may impact the delivery of Diameter messages between nodes. The authors recognize that element overload and network congestion are interrelated, and that overload can contribute to network congestion and vice versa.

Network congestion issues are better handled by the transport protocols. Diameter uses TCP and SCTP, both of which include congestion management features. Analysis of whether those features are sufficient for transport level congestion between Diameter nodes, and any work to further mitigate network congestion is out of scope both for this document, and for the work proposed by this document.

1.4. Diameter Applications in a Broader Network

Most elements using Diameter applications do not use Diameter exclusively. It is important to realize that overload of an element can be caused by a number of factors that may be unrelated to the processing of Diameter or Diameter applications.

A element communicating via protocols other than Diameter that is also using a Diameter application needs to be able to signal to Diameter peers that it is experiencing overload regardless of the cause of the overload, since the overload will affect that element's ability to process Diameter transactions. The element may also need to signal this on other protocols depending on its function and the architecture of the network and application it is providing services for. Whether that is necessary can only be decided within the context of that architecture and application. A mechanism for signaling overload with Diameter, which this specification details the requirements for, provides applications the ability to signal their Diameter peers of overload, mitigating that part of the issue. Applications may need to use this, as well as other mechanisms, to solve their broader overload issues. Indicating overload on protocols other than Diameter is out of scope for this document, and for the work proposed by this document.

1.5. Documentation Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The terms "client", "server", "agent", "node", "peer", "upstream", and "downstream" are used as defined in [I-D.ietf-dime-rfc3588bis].

2. Overload Scenarios

Several Diameter deployment scenarios exist that may impact overload management. The following scenarios help motivate the requirements for an overload management mechanism.

These scenarios are by no means exhaustive, and are in general simplified for the sake of clarity. In particular, the authors assume for the sake of clarity that the client sends Diameter requests to the server, and the server sends responses to client, even though Diameter supports bidirectional applications. Each direction in such an application can be modeled separately.

In a large scale deployment, many of the nodes represented in these scenarios would be deployed as clusters of servers. The authors assume that such a cluster is responsible for managing its own internal load balancing and overload management so that it appears as a single Diameter node. That is, other Diameter nodes can treat it as single, monolithic node for the purposes of overload management.

These scenarios do not illustrate the client application. As

mentioned in Section 1, Diameter is not typically an end-user protocol; rather it is generally used in support of some other client application. These scenarios do not consider the impact of Diameter overload on the client application.

2.1. Peer to Peer Scenarios

This section describes Diameter peer-to-peer scenarios. That is, scenarios where a Diameter client talks directly with a Diameter server, without the use of a Diameter agent.

Figure 1 illustrates the simplest possible Diameter relationship. The client and server share a one-to-one peer-to-peer relationship. If the server becomes overloaded, either because the client exceeds the server's capacity, or because the server's capacity is reduced due to some resource dependency, the client needs to reduce the amount of Diameter traffic it sends to the server. Since the client cannot forward requests to another server, it must either queue requests until the server recovers, or itself become overloaded in the context of the client application and other protocols it may also use.

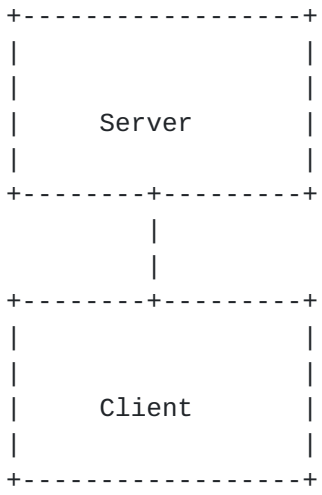

Figure 1: Basic Peer to Peer Scenario

Figure 2 shows a similar scenario, except in this case the client has multiple servers that can handle work for a specific realm and application. If server 1 becomes overloaded, the client can forward traffic to server 2. Assuming server 2 has sufficient reserve capacity to handle the forwarded traffic, the client should be able to continue serving client application protocol users. If server 1 is approaching overload, but can still handle some number of new

request, it needs to be able to instruct the client to forward a subset of its traffic to server 2.

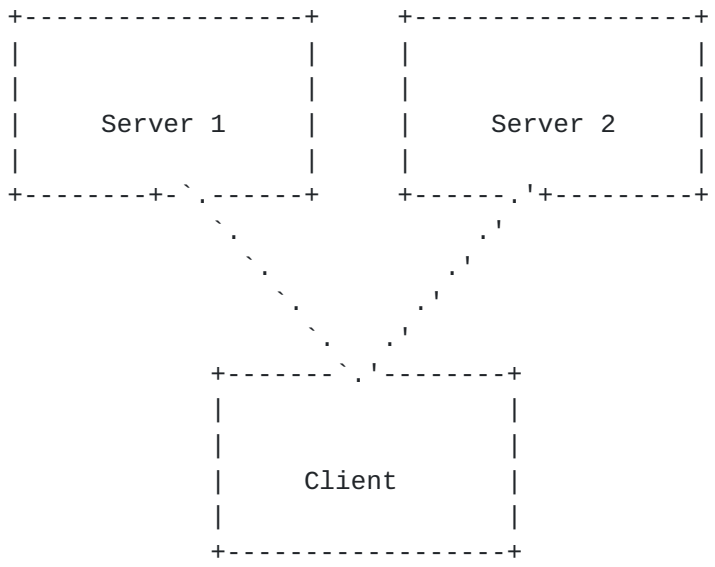

Figure 2: Multiple Server Peer to Peer Scenario

Figure 3 illustrates a peer-to-peer scenario with multiple Diameter realm and application combinations. In this example, server 2 can handle work for both applications. Each application might have different resource dependencies. For example, a server might need to access one database for application A, and another for application B. This creates a possibility that Server 2 could become overloaded for application A but not for application B, in which case the client would need to divert some part of its application A requests to server 1, but should not divert any application B requests. This requires server 2 to be able to distinguish between applications when it indicates an overload condition to the client.

On the other hand, it's possible that the servers host many applications. If server 2 becomes overloaded for all applications, it would be undesirable for it to have to notify the client separately for each application. Therefore it also needs a way to indicate that it is overloaded for all possible applications.

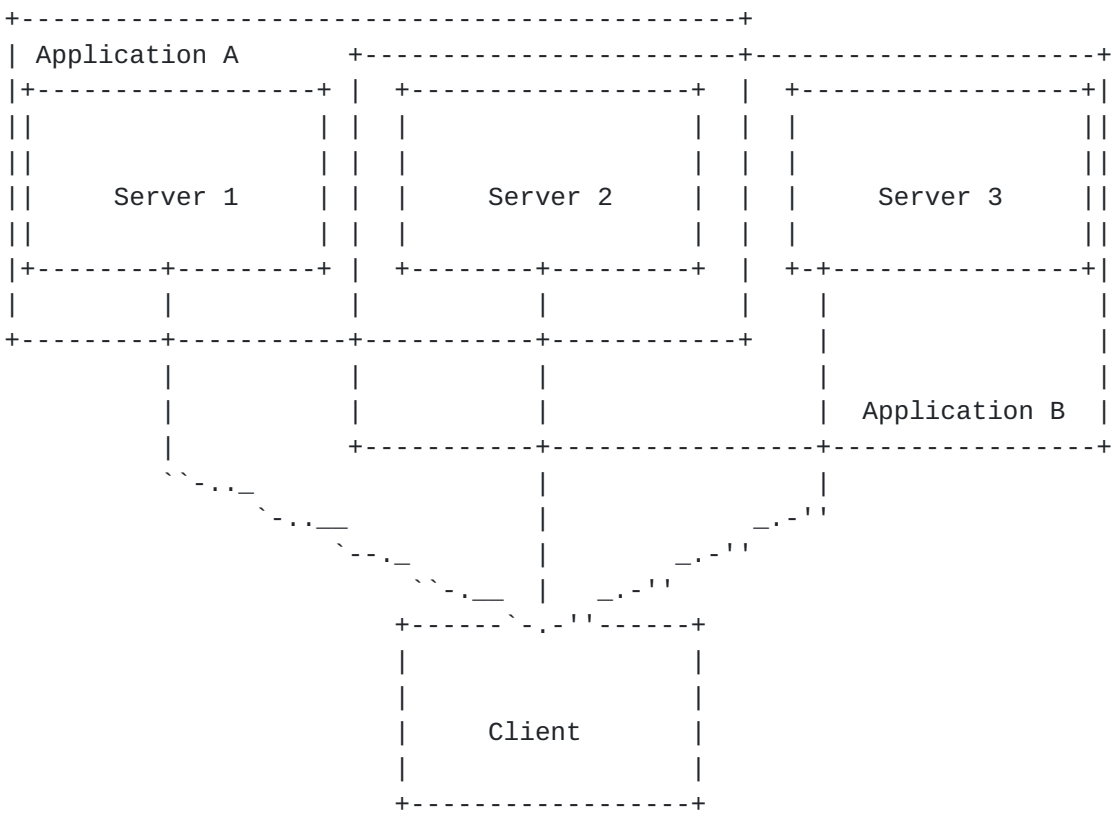

Figure 3: Multiple Application Peer to Peer Scenario

2.2. Agent Scenarios

This section describes scenarios that include a Diameter agent, either in the form of a Diameter relay or Diameter proxy. These scenarios do not consider Diameter redirect agents, since they are more readily modeled as end-servers.

Figure 4 illustrates a simple Diameter agent scenario with a single client, agent, and server. In this case, overload can occur at the server, at the agent, or both. But in most cases, client behavior is the same whether overload occurs at the server or at the agent. From the client's perspective, server overload and agent overload is the same thing.

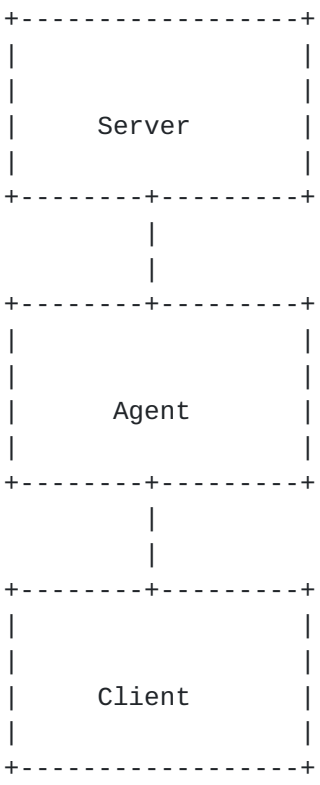

Figure 4: Basic Agent Scenario

Figure 5 shows an agent scenario with multiple servers. If server 1 becomes overloaded, but server 2 has sufficient reserve capacity, the agent may be able to transparently divert some or all Diameter requests originally bound for server 1 to server 2.

In most cases, the client does not have detailed knowledge of the Diameter topology upstream of the agent. If the agent uses dynamic discovery to find eligible servers, the set of eligible servers may not be enumerable from the perspective of the client. Therefore, in most cases the agent needs to deal with any upstream overload issues in a way that is transparent to the client. If one server notifies the agent that it has become overloaded, the notification should not be passed back to the client in a way where the client could mistakenly perceive the agent itself as being overloaded. If the set of all possible destinations upstream of the agent no longer has sufficient capacity for incoming load, the agent itself becomes effectively overloaded.

On the other hand, there are cases where the client needs to be able to select a particular server from behind an agent. For example, if a Diameter request is part of a multiple-round-trip authentication, or is otherwise part of a Diameter "session", it may have a

DestinationHost AVP that requires the request to be served by server 1. Therefore the agent may need to inform a client that a particular upstream server is overloaded or otherwise unavailable. Note that there can be many ways a server can be specified, which may have different implications (e.g. by IP address, by host name, etc).

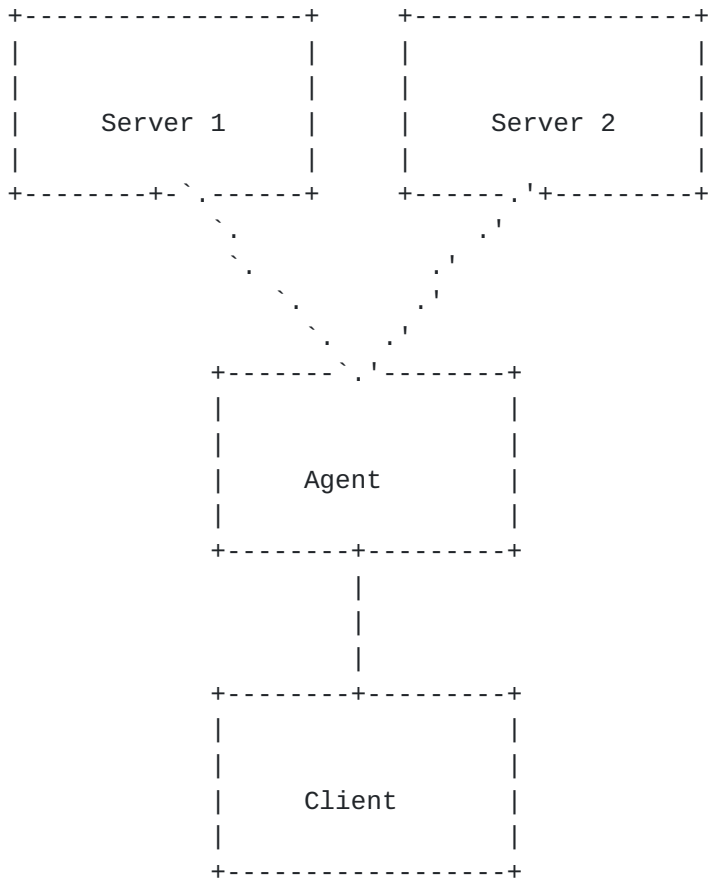

Figure 5: Multiple Server Agent Scenario

Figure 6 shows a scenario where an agent routes requests to a set of servers for more than one Diameter realm and application. In this scenario, if server 1 becomes overloaded or unavailable, the agent may effectively operate at reduced capacity for application A, but at full capacity for application B. Therefore, the agent needs to be able to report that it is overloaded for one application, but not for another.

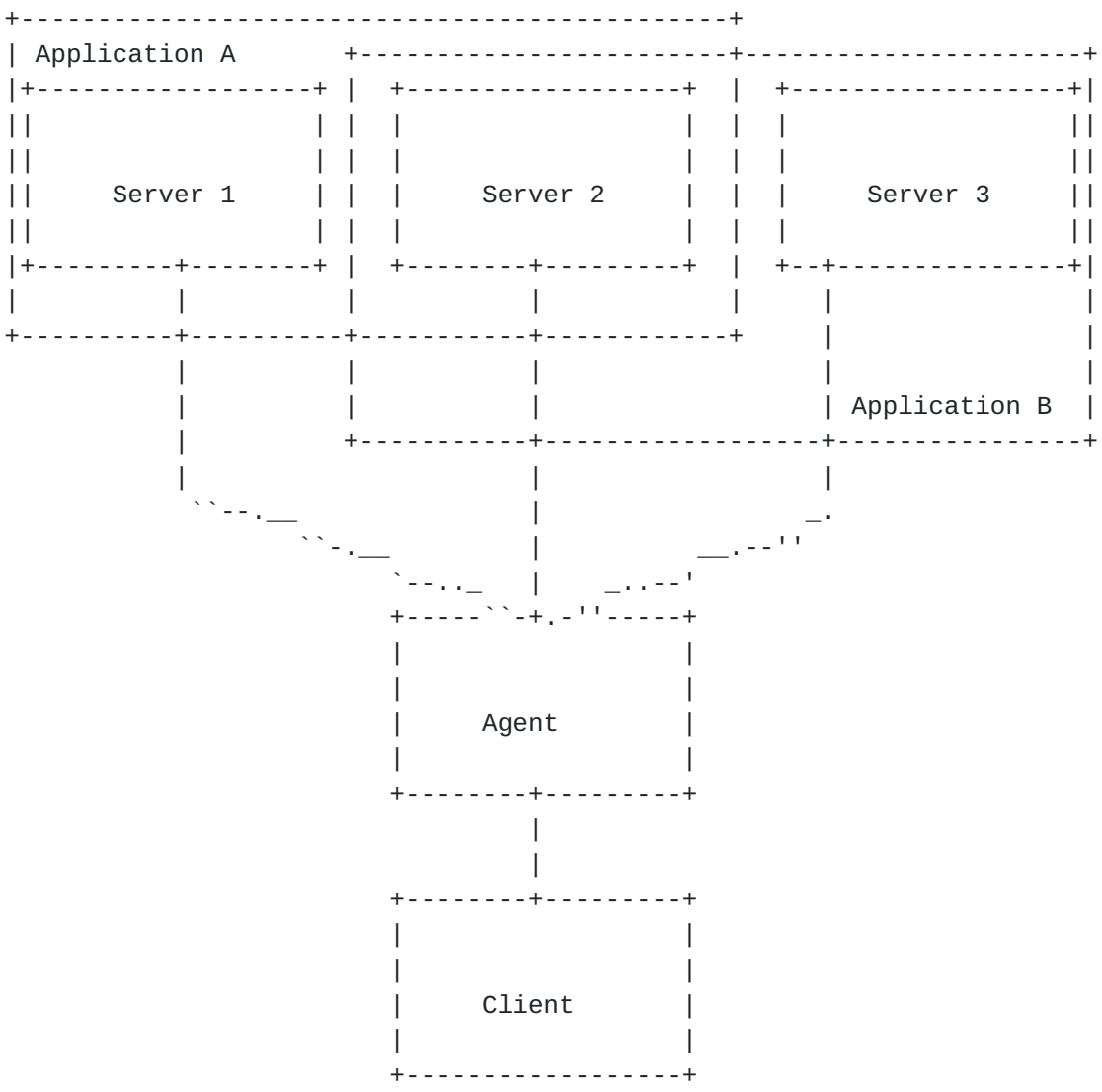

Figure 6: Multiple Application Agent Scenario

3. Existing Mechanisms

Diameter offers both implicit and explicit mechanisms for a Diameter node to learn that a peer is overloaded or unreachable. The implicit mechanism is simply the lack of responses to requests. If a client fails to receive a response in a certain time period, it assumes the upstream peer is unavailable, or overloaded to the point of effective unavailability. The watchdog mechanism [RFC3539] ensures that a certain rate of transaction responses occur even when there is otherwise little or no other Diameter traffic.

The explicit mechanism involves specific protocol error responses, where an agent or server can tell a downstream peer that it is either

too busy to handle a request (DIAMETER_TOO_BUSY) or unable to route a request to an upstream destination (DIAMETER_UNABLE_TO_DELIVER), perhaps because that destination itself is overloaded to the point of unavailability.

Once a Diameter node learns that an upstream peer has become overloaded via one of these mechanisms, it can then attempt to take action to reduce the load. This usually means forwarding traffic to an alternate destination, if available. If no alternate destination is available, the node must either reduce the number of messages it originates (in the case of a client) or inform the client to reduce traffic (in the case of an agent.)

Diameter requires the use of a congestion-managed transport layer, currently TCP or SCTP, to mitigate network congestion. It is expected that these transports manage network congestion and that issues with transport (e.g. congestion propagation and window management) are managed at that level. But even with a congestion-managed transport, a Diameter node can become overloaded at the Diameter protocol or application layers due to the causes described in Section 1.1 and congestion managed transports do not provide facilities (and are at the wrong level) to handle server overload. Transport level congestion management is also not sufficient to address overload in cases of multi-hop and multi-destination signaling.

4. Issues with the Current Mechanisms

The currently available Diameter mechanisms for indicating an overload condition are not adequate to avoid service outages due to overload. This may, in turn, contribute to broader congestion collapse due to unresponsive Diameter nodes causing application or transport layer retransmissions. In particular, they do not allow a Diameter agent or server to shed load as it approaches overload. At best, a node can only indicate that it needs to entirely stop receiving requests, i.e. that it has effectively failed. Even that is problematic due to the inability to indicate durational validity on the transient errors available in the base Diameter protocol. Diameter offers no mechanism to allow a node to indicate different overload states for different categories of messages, for example, if it is overloaded for one Diameter application but not another.

4.1. Problems with Implicit Mechanism

The implicit mechanism doesn't allow an agent or server to inform the client of a problem until it is effectively too late to do anything about it. The client does not know to take action until the upstream

node has effectively failed. A Diameter node has no opportunity to shed load early to avoid collapse in the first place.

Additionally, the implicit mechanism cannot distinguish between overload of a Diameter node and network congestion. Diameter treats the failure to receive an answer as a transport failure.

4.2. Problems with Explicit Mechanisms

The Diameter specification is ambiguous on how a client should handle receipt of a DIAMETER_TOO_BUSY response. The base specification [I-D.ietf-dime-rfc3588bis] indicates that the sending client should attempt to send the request to a different peer. It makes no suggestion that a the receipt of a DIAMETER_TOO_BUSY response should affect future Diameter messages in any way.

The Authentication, Authorization, and Accounting (AAA) Transport Profile [RFC3539] recommends that a AAA node that receives a "Busy" response failover all remaining requests to a different agent or server. But while the Diameter base specification explicitly depends on RFC3539 to define transport behavior, it does not refer to RFC3539 in the description of behavior on receipt of DIAMETER_TOO_BUSY. There's a strong likelihood that at least some implementations will continue to send Diameter requests to an upstream peer even after receiving a DIAMETER_TOO_BUSY error.

BCP 41 [RFC2914] describes, among other things, how end-to-end application behavior can help avoid congestion collapse. In particular, an application should avoid sending messages that will never be delivered or processed. The DIAMETER_TOO_BUSY behavior as described in the Diameter base specification fails at this, since if an upstream node becomes overloaded, a client attempts each request, and does not discover the need to failover the request until the initial attempt fails.

The situation is improved if implementations follow the [RFC3539] recommendation and keep state about upstream peer overload. But even then, the Diameter specification offers no guidance on how long a client should wait before retrying the overloaded destination. If an agent or server supports multiple realms and/or applications, DIAMETER_TOO_BUSY only offers no way to indicate that it is overloaded for one application but not another. A DIAMETER_TOO_BUSY error can only indicate overload at a "whole server" scope.

Agent processing of a DIAMETER_TOO_BUSY response is also problematic as described in the base specification. DIAMETER_TOO_BUSY is defined as a protocol error. If an agent receives a protocol error, it may either handle it locally or it may forward the response back towards

the downstream peer. (The Diameter specification is inconsistent about whether a protocol error MAY or SHOULD be handled by an agent, rather than forwarded downstream.) If a downstream peer receives the DIAMETER_TOO_BUSY response, it may stop sending all requests to the agent for some period of time, even though the agent may still be able to deliver requests to other upstream peers.

DIAMETER_UNABLE_TO_DELIVER also has no mechanisms for specifying the scope or cause of the failure, or the durational validity.

5. Diameter Overload Case Studies

5.1. Overload in Mobile Data Networks

As the number of Third Generation (3G) and Long Term Evolution (LTE) enabled smartphone devices continue to expand in mobility networks, there have been situations where high signaling traffic load led to overload events at the Diameter-based Home Location Registries (HLR) and/or Home Subscriber Servers (HSS). The root causes of the HLR congestion events were manifold but included hardware failure and procedural errors. The result was high signaling traffic load on the HLR and HSS.

The 3GPP standards specification[need citation] for the end-to-end signaling call flows in 3G and LTE, from the end user device traversing through the radio and the core networks to the HLR/HSS, did not have an equivalent load control mechanism which is provided in the more traditional SS7 elements in GSM [need citation]. The capabilities specified in the 3GPP standards do not adequately address the abnormal condition where excessively high signaling traffic load situations are experienced.

Smartphones contribute much more heavily to the continuation of a registration surge due to their very aggressive registration algorithms. The aggressive smartphone logic is designed to:

a. always have voice and data registration, and

b. constantly try to be on 3G data (and thus on 3G voice) for their added benefits.

Non-smartphones typically have logic to wait for a time period after registering successfully on voice and data.

The smartphone aggressive registration is problematic in two ways:

- first by generating excessive signaling load towards the HLR that is ten times that from a non-smartphone,

- and second by causing continual registration attempts when a network failure affects registrations through the 3G data network.

5.2. 3GPP Study on Core Network Overload

A study in 3GPP SA2 on core network overload has produced the technical report [TR23.843]. This enumerates several causes of overload in mobile core networks including portions that are signaled using Diameter. This document is a work in progress and is not complete. However, it is useful for pointing out scenarios and the general need for an overload control mechanism for Diameter.

It is common for mobile networks to employ more than one radio technology and to do so in an overlay fashion with multiple technologies present in the same location (such as GSM, UMTS or CDMA along with LTE). This presents opportunities for traffic storms when issues occur on one overlay and not another as all devices that had been on the overlay with issues switch. This causes a large amount of Diameter traffic as locations and policies are updated.

Another scenario called out by this study is a flood of registration and mobility management events caused by some element in the core network failing. This flood of traffic from end nodes falls under the network initiated traffic flood category. There is likely to also be traffic resulting directly from the component failure in this case.

Subscriber initiated traffic floods are also indicated in this study as an overload mechanism where a large number of mobile devices attempting to access services at the same time, such as in response to an entertainment event or a catastrophic event.

While this study is concerned with the broader effects of these scenarios on wireless networks and their elements, they have implications specifically for Diameter signaling. One of the goals of this document is to provide guidance for a core mechanism that can be used to mitigate the scenarios called out by this study.

6. Solution Requirements

This section proposes requirements for an improved mechanism to control Diameter overload, with the goals of improving the issues described in Section 4 and supporting the scenarios described in Section 2

REQ 1: The overload mechanism MUST provide a communication method for Diameter nodes to exchange overload information.

REQ 2: The overload mechanism MUST be useable with any existing or future Diameter application. It MUST NOT require specification changes for existing Diameter applications.

REQ 3: The overload mechanism MUST limit the impact of overload on the overall useful throughput of a Diameter server, even when the incoming load on the network is far in excess of its capacity. The overall useful throughput under load is the ultimate measure of the value of an overload control mechanism.

REQ 4: Diameter allows requests to be sent from either side of a connection and either side of a connection may have need to provide its overload status. The mechanism MUST allow each side of a connection to independently inform the other of its overload status.

REQ 5: Diameter allows nodes to determine their peers via dynamic discovery or manual configuration. The mechanism MUST work consistently without regard to how peers are determined.

REQ 6: The mechanism designers SHOULD seek to minimize the amount of new configuration required in order to work. For example, it is better to allow peers to advertise or negotiate support for the mechanism, rather than to require this knowledge to be configured at each node.

REQ 7: The overload mechanism MUST ensure that the system remains stable. When the offered load drops from above the overall capacity of the network to below the overall capacity, the throughput MUST stabilize and become equal to the offered load.

REQ 8: The mechanism MUST allow nodes to shed load without introducing oscillations. Note that this requirement implies a need for supporting nodes to be able to distinguish current overload information from stale information, and to make decisions using the most currently available information.

REQ 9: The mechanism MUST function across fully loaded as well as quiescent transport connections. This is partially derived from the requirements for stability and hysteresis control above.

REQ 10: Consumers of overload state indications MUST be able to determine when the overload condition improves or ends.

REQ 11: The overload mechanism MUST be scalable. That is, it MUST be able to operate in different sized networks.

REQ 12: When a single network node fails, goes into overload, or suffers from reduced processing capacity, the mechanism MUST make it possible to limit the impact of this on other nodes in the network. This helps to prevent a small-scale failure from becoming a widespread outage.

REQ 13: The mechanism MUST NOT introduce substantial additional work for node in an overloaded state. For example, a requirement for an overloaded node to send overload information every time it received a new request would introduce substantial work. Existing messaging is likely to have the characteristic of increasing as an overload condition approaches, allowing for the possibility of increased feedback for information piggybacked on it.

REQ 14: Some scenarios that result in overload involve a rapid increase of traffic with little time between normal levels and overload inducing levels. The mechanism SHOULD provide for increased feedback when traffic levels increase. The mechanism MUST NOT do this in such a way that it increases the number of messages while at high loads.

REQ 15: The mechanism MUST NOT interfere with the congestion control mechanisms of underlying transport protocols. For example, a mechanism that opened additional TCP connections when the network is congested would reduce the effectiveness of the underlying congestion control mechanisms.

REQ 16: The mechanism MUST operate without malfunction in an environment with a mix of nodes that do, and nodes that do not, support the mechanism.

REQ 17: In a mixed environment with nodes that support the overload control mechanism and that do not, the mechanism MUST NOT result in less useful throughput than would have resulted if it were not present. It SHOULD result in less severe congestion in this environment.

REQ 18: In a mixed environment of nodes that support the overload control mechanism and that do not, users and operators of nodes that do not support the mechanism MUST NOT benefit from the mechanism more than users and operators of nodes that support the mechanism.

REQ 19: It MUST be possible to use the mechanism between nodes in different realms and in different administrative domains.

REQ 20: Any explicit overload indication MUST distinguish between actual overload, as opposed to other, non-overload related failures.

REQ 21: In cases where a network node fails, is so overloaded that it cannot process messages, or cannot communicate due to a network failure, it may not be able to provide explicit indications of the nature of the failure or its levels of congestion. The mechanism MUST properly function in these cases.

REQ 22: The mechanism MUST provide a way for an node to throttle the amount of traffic it receives from an peer node. This throttling SHOULD be graded so that it can be applied gradually as offered load increases. Overload is not a binary state; there may be degrees of overload.

REQ 23: The mechanism MUST enable a supporting node to minimize the chance that retries due to an overloaded or failed node result in additional traffic to other overloaded nodes, or cause additional nodes to become overloaded. Moreover, the mechanism SHOULD provide unambiguous directions to clients on when they should retry a request and when they should not considering the various causes of overload such as avalanche restart.

REQ 24: The mechanism MUST provide sufficient information to enable a load balancing node to divert messages that are rejected or otherwise throttled by an overloaded upstream node to other upstream nodes that are the most likely to have sufficient capacity to process them.

REQ 25: The mechanism MUST provide a mechanism for indicating load levels even when not in an overloaded condition, to assist nodes making decisions to prevent overload conditions from occurring.

REQ 26: The specification for the overload mechanism SHOULD offer guidance on which message types might be desirable to send or process over others during times of overload, based on Diameter-specific considerations. For example, it may be more beneficial to process messages for existing sessions ahead of new sessions.

REQ 27: The mechanism MUST NOT prevent a node from prioritizing requests based on any local policy, so that certain requests are given preferential treatment, given additional retransmission, or processed ahead of others.

REQ 28: The overload mechanism MUST NOT provide new vulnerabilities to malicious attack, or increase the severity of any existing vulnerabilities. This includes vulnerabilities to DoS and DDoS attacks as well as replay and man-in-the middle attacks.

REQ 29: The mechanism MUST provide a means to match an overload indication with the node that originated it. In particular, the mechanism MUST allow a node to distinguish between overload at a next-hop peer from overload at a node upstream of the peer. For example, in Figure 5, the client must not mistake overload at server 1 for overload at the agent, whether or not the agent supports the mechanism.(see REQ 4).

REQ 30: The mechanism MUST NOT depend on being deployed in environments where all Diameter nodes are completely trusted. It SHOULD operate as effectively as possible in environments where other nodes are malicious; this includes preventing malicious nodes from obtaining more than a fair share of service. Note that this does not imply any responsibility on the mechanism to detect, or take countermeasures against, malicious nodes.

REQ 31: It MUST be possible for a supporting node to make authorization decisions about what information will be sent to peer nodes based on the identity of those nodes. This allows a domain administrator who considers the load of their nodes to be sensitive information to restrict access to that information. Of course, in such cases, there is no expectation that the overload mechanism itself will help prevent overload from that peer node.

REQ 32: The mechanism MUST NOT interfere with any Diameter compliant method that a node may use to protect itself from overload from non-supporting nodes, or from denial of service attacks.

REQ 33: There are multiple situations where a Diameter node may be overloaded for some purposes but not others. For example, this can happen to an agent or server that supports multiple applications, or when a server depends on multiple external resources, some of which may become overloaded while others are fully available. The mechanism MUST allow Diameter nodes to indicate overload with sufficient granularity to allow clients to take action based on the overloaded resources without forcing available capacity to go unused. The mechanism MUST support specification of overload information with granularities of at least "Diameter node", "realm", "Diameter application", and "Diameter session", and SHOULD allow extensibility for others to be added in the future.

REQ 34: The mechanism MUST provide a method for extending the information communicated and the algorithms used for overload control.

7. IANA Considerations

This document makes no requests of IANA.

8. Security Considerations

A Diameter overload control mechanism is primarily concerned with the load and overload related behavior of nodes in a Diameter network, and the information used to affect that behavior. Load and overload information is shared between nodes and directly affects the behavior and thus is potentially vulnerable to a number of methods of attack.

Load and overload information may also be sensitive from both business and network protection viewpoints. Operators of Diameter equipment want to control visibility to load and overload information to keep it from being used for competitive intelligence or for targeting attacks. It is also important that the Diameter overload control mechanism not introduce any way in which any other information carried by Diameter is sent inappropriately.

This document includes requirements intended to mitigate the effects

of attacks and to protect the information used by the mechanism.

8.1. Access Control

To control the visibility of load and overload information, sending should be subject to some form of authentication and authorization of the receiver. It is also important to the receivers that they are confident the load and overload information they receive is from a legitimate source. Note that this implies a certain amount of configurability on the nodes supporting the Diameter overload control mechanism.

8.2. Denial-of-Service Attacks

An overload control mechanism provides a very attractive target for denial-of-service attacks. A small number of messages may affect a large service disruption by falsely reporting overload conditions. Alternately, attacking servers nearing, or in, overload may also be facilitated by disrupting their overload indications, potentially preventing them from mitigating their overload condition.

A design goal for the Diameter overload control mechanism is to minimize or eliminate the possibility of using the mechanism for this type of attack.

As the intent of some denial-of-service attacks is to induce overload conditions, an effective overload control mechanism should help to mitigate the effects of an such an attack.

8.3. Replay Attacks

An attacker that has managed to obtain some messages from the overload control mechanism may attempt to affect the behavior of nodes supporting the mechanism by sending those messages at potentially inopportune times. In addition to time shifting, replay attacks may send messages to other nodes as well (target shifting).

A design goal for the Diameter overload control mechanism is to minimize or eliminate the possibility of causing disruption by using a replay attack on the Diameter overload control mechanism.

8.4. Man-in-the-Middle Attacks

By inserting themselves in between two nodes supporting the Diameter overload control mechanism, an attacker may potentially both access and alter the information sent between those nodes. This can be used for information gathering for business intelligence and attack targeting, as well as direct attacks.

A design goal for the Diameter overload control mechanism is to minimize or eliminate the possibility of causing disruption man-in-the-middle attacks on the Diameter overload control mechanism. A transport using TLS and/or IPSEC may be desirable for this.

8.5. Compromised Hosts

A compromised host that supports the Diameter overload control mechanism could be used for information gathering as well as for sending malicious information to any Diameter node that would normally accept information from it. While is is beyond the scope of the Diameter overload control mechanism to mitigate any operational interruption to the compromised host, a reasonable design goal is to minimize the impact that a compromised host can have on other nodes through the use of the Diameter overload control mechanism. Of course, a compromised host could be used to cause damage in a number of other ways. This is out of scope for a Diameter overload control mechanism.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[I-D.ietf-dime-rfc3588bis] Fajardo, V., Arkko, J., Loughney, J., and G. Zorn, "Diameter Base Protocol", draft-ietf-dime-rfc3588bis-33 (work in progress), May 2012.

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, September 2000.

[RFC3539] Aboba, B. and J. Wood, "Authentication, Authorization and Accounting (AAA) Transport Profile", RFC 3539, June 2003.

9.2. Informative References

[RFC5390] Rosenberg, J., "Requirements for Management of Overload in the Session Initiation Protocol", RFC 5390, December 2008.

[TR23.843] 3GPP, "Study on Core Network Overload Solutions", TR 23.843 0.4.0, April 2011.

Appendix A. Contributors

Significant contributions to this document were made by Adam Roach and Eric Noel.

Appendix B. Acknowledgements

Review of, and contributions to, this specification by Martin Dolly, Carolyn Johnson, Jianrong Wang, Imtiaz Shaikh, and Robert Sparks were most appreciated. We would like to thank them for their time and expertise.

Authors' Addresses

Eric McMurry
Tekelec
17210 Campbell Rd.
Suite 250
Dallas, TX 75252
US

Email: emcmurry@estacado.net

Ben Campbell
Tekelec
17210 Campbell Rd.
Suite 250
Dallas, TX 75252
US

Email: ben@nostrum.com