F5 Technical Brief

Message-Based Load Balancing

Using F5 solutions to address the challenges of scaling Diameter, RADIUS, and message-oriented protocols.

by Lori MacVittie
Technical Marketing Manager, Application Services

Contents

Introduction

Load balancing has been integral to successfully providing high availability and nearly transparent scalability to web-based applications for almost two decades. Used at first to simply scale web-oriented protocols such as HTTP, once the integration of web-based applications with other key data center protocols became more prevalent, load balancing became more of an imperative for those protocols as well.

Unlike HTTP, which is synchronous and stateless, some protocols—such as Diameter, RADIUS, and Session Initiation Protocol (SIP)—are not only asynchronous but also do not adhere to a single request-reply communication sequence. This makes it more difficult to distribute those protocols because most load balancing systems are designed to operate best in a synchronous messaging environment in which a single request is made and responded to before another is processed.

Some of the most demanding environments require scalability solutions for such protocols. Service providers and financial institutions routinely process multi-gigabits of data in intervals that are unprecedented even on the most highly trafficked websites. And, they demand high levels of availability while simultaneously maintaining strenuous performance criteria. For example, SIP is one of the most commonly used signaling protocols in the world of service providers, but its inherent nature and its reliance on authentication protocols such as Diameter, RADIUS, and LDAP make it very difficult to scale efficiently.

There are several challenges associated with scaling the protocols common to service-provider offerings, and all of them must be addressed to support the scalability and stringent performance requirements of this demanding market.

Challenges

Asynchronous Messaging

Traditional web-based protocols—HTTP, FTP, and SMTP—are all synchronous: A client sends a message and expects a response. (See Figure 1.) In most cases, no other requests can be made until a response has been received, which enforces a strict synchronous communication paradigm. This adherence to a request-reply message pattern, although unique for each protocol, makes load balancing such traffic fairly easy and technically similar in implementation.

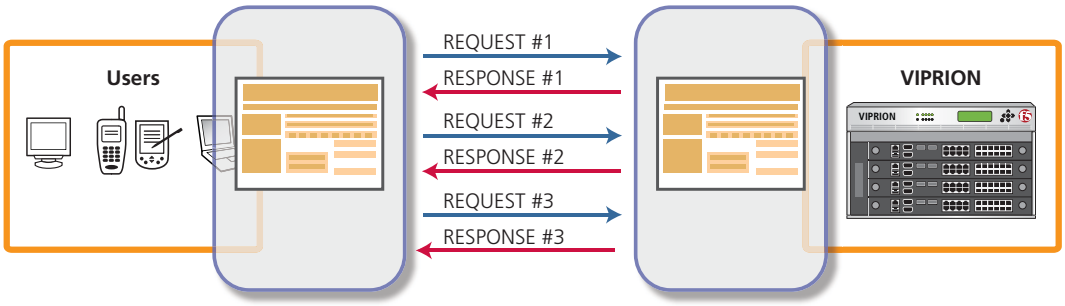

Figure 1: Example of a traditional request-reply load balancing exchange

Protocols such as Diameter and SIP also maintain the one-to-one (1:1) relationship in which there is always a matching reply for every request, but unlike traditional web-based protocols, they do not need to maintain a strict synchronous exchange. Multiple requests may be sent before receiving a reply. This makes intermediaries based on traditional protocols like HTTP unable to handle load balancing responsibilities because they cannot process more than one request at a time and are limited to load balancing on a per-connection basis.

The result is that a Diameter client, for example, may send one or more requests over the same connection without waiting for a response to the first request. It is also possible that responses will be sent to the client in a different order than they were received. (See Figure 2.) This requires that Diameter servers—as well as any intermediary managing Diameter traffic—must be able to handle this process. However, most load balancing systems cannot because they assume a traditional request-reply protocol processing paradigm in which every request is followed by a reply that adheres to the order in which requests were received.

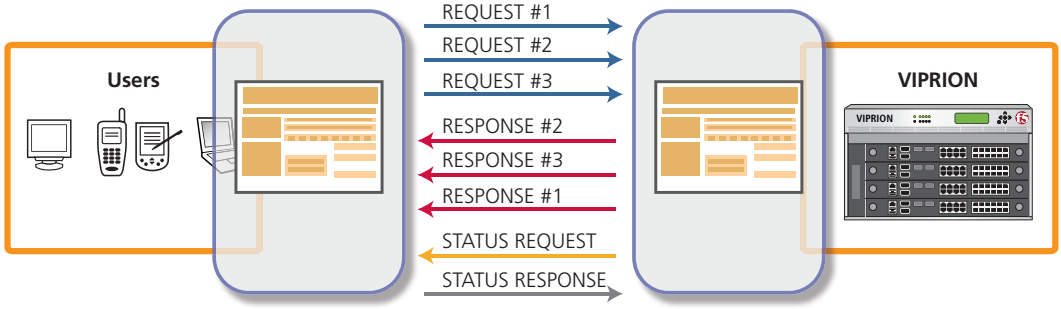

Figure 2: Example of a message-based protocol load balancing exchange

Further complicating support for these protocols is the lack of distinction between client and server roles. In a traditional communication exchange, the client sends requests and the server sends responses. SIP, Diameter, and related protocols erase those lines and allow the server to act essentially as a client, sending requests and waiting for responses. This bidirectional communication exchange is not present in

any traditional request-reply protocols and thus is not a behavior typically supported by load balancing solutions that focus on handling HTTP and similar protocols.

Long-Lived Sessions

In traditional load balancing solutions, load balancing is accomplished at layer 4 (TCP/UDP) on a per-session or connection basis. All requests received over the same session are load balanced to the same server. When communications are complete, the session is terminated and is available for use by another client.

This behavior is not acceptable for some protocols, particularly those associated with service provider and telecommunications implementations. Protocols such as SIP, Diameter, and RADIUS establish longer-lived sessions over which associated protocols and messages will be sent, each potentially requiring processing by a *different* server. For example, authentication and authorization services are often handled by Diameter but are tunneled through SIP. When the default load balancing behavior assumes all requests sent over the same TCP/UDP connection should be sent to the same server, it does not allow for the processing and subsequent load balancing of individual messages sent over the same session. This means traditional load balancing mechanisms are incapable of supporting the scalability and availability requirements of such protocols.

The alternative to long-lived sessions is to require that each request be made over a new and *separate* session. Yet, this is impractical because the overhead associated with establishing connections on a per-protocol basis would degrade performance and negatively impact availability. Any load balancing solution supporting protocols such as SIP, Diameter, and RADIUS must be capable of applying load balancing decisions on a per-message basis over the same session.

Message-Oriented Communication

In a traditional request-reply protocol load balancing scenario, each request can be directed to a specific server based on a variety of parameters, such as the content or request type. This behavior is also desirable in message-oriented communication, but it is typically more difficult to support because of the need to scale intermediaries to open and maintain multiple connections to different servers. Traditionally load balancing maintains a 1:1 ratio between requests and server-side connections, but in a message-oriented protocol such as SIP or Diameter, there is a need to maintain a one-to-many (1:N) ratio instead.

Once the problem of maintaining 1:N ratios is addressed, it is then necessary to attend to the lack of clearly defined message boundaries inherent in message-oriented communications. HTTP, for example, clearly differentiates between messages using specific headers and syntactical control. Diameter, by contrast, is less specific, and it requires an intermediary to distinguish between messages based on recognition of the specific request type and length. This means intermediaries must be able to parse and interpret the data associated with message-oriented protocols before load balancing decisions can be made.

Once a message is extracted from the session, it is then necessary to examine some part of the content to determine the type of the request. In the case of Diameter, this is usually achieved by examining an attribute–value pair (AVP) code representing different requests. AVP codes indicate a variety of specific services related to authentication and authorization of services. Diameter is load balanced on AVP codes even though the requests are sent over the same long-lived session. Because the sessions are long-lived, the load balancing solution must be able to maintain connections to multiple servers (hence the 1:N ratio) and correctly route requests based on the AVP codes contained within the Diameter messages.

Solutions

Message-Based Load Balancing

The primary requirement to solve the challenges associated with scaling message-oriented protocols such as SIP and Diameter is the ability to extract individual messages out of a single, shared TCP connection. This means that a high-availability solution must be able to inspect application layer data in order to split out individual messages from the TCP connection and distribute them appropriately—including maintaining persistence.

Because so many protocols in the telecommunications arena are message-oriented, F5 has architected a foundation of message-based load balancing (MBLB) that is easily extended via new profiles and iRules™. MBLB capabilities provide the core support for scaling protocols such as SIP, Diameter, LDAP, and RADIUS. These capabilities enable BIG-IP® Local Traffic Manager™ (LTM) to perform the disaggregation of messages from a single TCP connection and handle asynchronous message types.

MBLB makes it possible for BIG-IP LTM to implement specific protocol support in a turnkey manner while still enabling customer- and environment-specific customization through iRules. F5 already supports several message-oriented protocols such as SIP, and it now also supports scaling of Diameter.

F5 Support for Diameter

Base Diameter (RFC 3588) protocol support has been provided on BIG-IP LTM in the past strictly via iRules, but there is now more turnkey support via a new profile option. The Diameter profile enables customers to specify load balancing of Diameter messages based on a single AVP code.

The Diameter profile configuration on BIG-IP LTM allows for messages with specific AVP codes to be directed to a specific server (or pool of servers) that is responsible for handling the requests. (See Figure 3.) The specified AVP code is used to keep Diameter messages persistent on the same server so that clients can maintain state across a long-lived session. At the same time, BIG-IP LTM maintains availability by ensuring no individual server is ever overwhelmed with requests and sessions. Further customization or the use of additional AVP codes to determine routing and persistence can be accomplished using iRules.

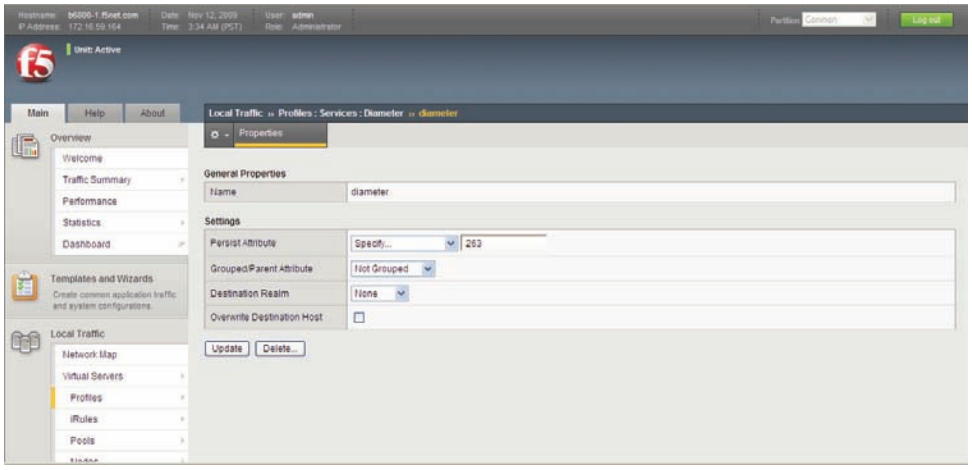

Figure 3: Diameter profile configuration options

The F5 solution to Diameter load balancing also provides high availability and fast failover through a session mirror across an active or standby pair of BIG-IP LTM devices. Mirroring sessions across the pairs of BIG-IP LTM devices ensures that in the event of a catastrophic failure, all sessions will be seamlessly transferred from the primary to the secondary BIG-IP LTM device to maintain availability.

How It Works

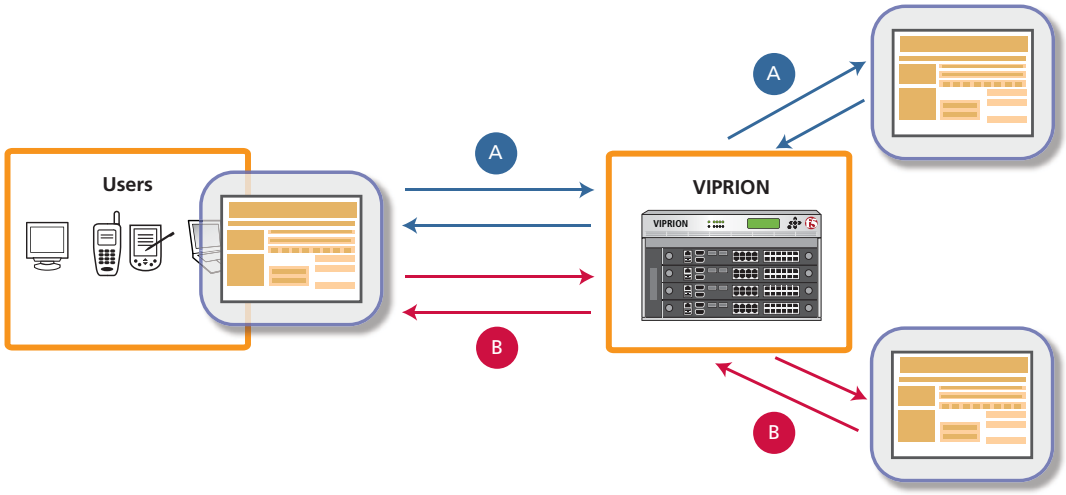

Figure 4: Flow of messages in a Diameter load balancing configuration

BIG-IP LTM is configured with a virtual IP address (proxy) to which the Diameter profile is applied. When a client initiates a session (A) (see Figure 4), the AVP code specified in the Diameter profile configuration is extracted from the message and mapped to a server using the virtual server's configured load balancing algorithm.

Subsequent messages received over that connection with the same AVP code will be load balanced to the same server. This is the same behavior as is expected with traditional application protocols and load balancing mechanisms involving persistence. Where Diameter and message-based load balancing diverge from traditional load balancing solutions is that messages received over the same connection but containing a different AVP code (B) will be processed in the same way as the original. BIG-IP LTM extracts the code and stores it to map all subsequent messages over that session with *that* AVP code to the same server.

Traditional load balancing solutions generally expect only one value per client on which to execute persistence-based routing. Because BIG-IP LTM leverages TMOS®, the extensible foundation of the F5 BIG-IP platform, it has the fundamental understanding of how to extract and act upon message-level information rather than just connection information. It recognizes that each message may have its own persistence key and may require routing over different server-side connections based on that key.

The choice of server during the initial connection can be determined through the use of standard load balancing algorithms or can be customized using iRules. If more than one AVP code is necessary for purposes of routing requests, then iRules can be used to provide the flexibility necessary.

Conclusion

The ability to load balance some protocols requires a deep understanding of the way in which the applications that use those protocols behave. Protocols that are both asynchronous and communicate bidirectionally are challenging to scale for most load balancing solutions because they do not have the ability to extract and route requests at a message level and are inherently tied by their architecture to load balancing based on connections.

F5 TMOS architecture provides the means by which F5 is able to quickly implement support for message-based protocols such as Diameter, RADIUS, and LDAP. TMOS enables BIG-IP Local Traffic Manager to extract and act upon message-level information, providing turnkey scalability and high availability while maintaining the ability to extend and adapt that functionality through iRules.

F5 Networks, Inc. 401 Elliott Avenue West, Seattle, WA 98119 888-882-4447 www.f5.com

F5 Networks, Inc.
Corporate Headquarters
info@f5.com

F5 Networks
Asia-Pacific
info.asia@f5.com

F5 Networks Ltd.
Europe/Middle-East/Africa
emeainfo@f5.com

F5 Networks
Japan K.K.
f5j-info@f5.com

CS10059 1109