Introduction to Diameter

Get the next generation AAA protocol

Jeffrey Liu
Steven Jiang
Hicks Lin

January 24, 2006

Get an overview of the Diameter protocol, the next generation Authentication, Authorization, and Accounting (AAA) protocol. See how it works and gain the ground knowledge for exploring it in more detail.

Introduction

The Diameter protocol was derived from the RADIUS protocol with a lot of improvements in different aspects, and is generally believed to be the next generation Authentication, Authorization, and Accounting (AAA) protocol. The Diameter protocol was widely used in the IMS architecture for IMS entities to exchange AAA-related information. Because the IMS system might be the next big thing in the telecom industry, we believe a clear understanding of the Diameter protocol is necessary for understanding the essence of the IMS architecture. This article offers an overview of Diameter and how it works. For developers interested in how AAA in IMS works, or who want to implement Diameter applications, this article is a good starting page.

With the emergence of new technologies and applications such as wireless networks and mobile IPs, the requirements for authentication and authorization have greatly increased, and access control mechanisms are more complex than ever. The existing RADIUS (Remote Authentication Dial-In User Service) protocol can be insufficient to cope with these new requirements; what's needed is a new protocol that is capable of fulfilling new access control features while keeping the flexibility for further extension. This is where the Diameter protocol comes into play.

Please note that this article provides an overview of Diameter and does not cover all the protocol details. If you want to go further and implement the Diameter base protocol, refer to RFC3588 in Related topics for more details. So, because this article mainly addresses the base protocol, Diameter will refer to the Diameter Base Protocol.

AAA and Diameter

Before immersing ourselves in protocol details, let's see what drives the requirement for AAA protocol. In the old days, people tried to dial into their Internet Service providers (ISPs) by
providing their ID and password to an access server, which then authenticated the user before granting Internet access. In most cases, a user’s credential information is not stored directly in the access server, but in a more secure location such as a Lightweight Directory Access Protocol (LDAP) server behind a boundary firewall. Therefore, a standardized protocol is required between the access server and the user information repository in order to exchange authentication-, authorization-, and accounting-related information. The RADIUS protocol was designed to provide a simple, but efficient, way to deliver such AAA capability.

As with the evolution of network applications and protocols, new requirements and mechanisms are required to authenticate users. These requirements are summarized in RFC2989 (see Related topics), which includes such topics as failover, security, and audit ability. Although there are some subsidiary protocols intended to extend the capability of the RADIUS protocol, a more extensible and general protocol was expected. The Diameter protocol was then derived from that of RADIUS, and designed to be a general framework for future AAA applications.

The Diameter protocol is not a brand-new one for AAA, but rather, as its name implies, is an enhanced version of the RADIUS protocol. It includes numerous enhancements in all aspects, such as error handling and message delivery reliability. It extracts the essence of the AAA protocol from RADIUS and defines a set of messages that are general enough to be the core of the Diameter Base protocol. The various applications that require AAA functions can define their own extensions on top of the Diameter base protocol, and can benefit from the general capabilities provided by the Diameter base protocol. **Figure 1** illustrates the relationship between the Diameter base protocol and various Diameter applications.

**Figure 1. The relationship of the Diameter base protocol and Diameter applications**

![Diagram](image)

**Diameter nodes and agents**

Diameter is designed as a Peer-To-Peer architecture, and every host who implements the Diameter protocol can act as either a client or a server depending on network deployment. So the term **Diameter node** is used to refer to a Diameter client, a Diameter server, or a Diameter agent, which we will introduce later. The Diameter node that receives the user connection request will act as the Diameter client. In most cases, a Diameter client will be a Network Access Server. After collecting user credentials, such as username and password, it will send an access request message to one Diameter node serving the request. For simplicity, we assume it is the Diameter server. The Diameter server authenticates the user based on the information provided. If the authentication process succeeds, the user’s access privileges are included in the response message and sent back to the corresponding Diameter client. Otherwise, an access reject message is sent.
Although the architecture just described looks like a traditional client-server architecture, a node acting as the Diameter server for some requests might actually act as a Diameter client in some situations; the Diameter protocol is actually peer-to-peer-based architecture in a more generic sense. Besides, a special Diameter node called Diameter agent is clearly defined in Diameter. Typically, there are three kinds of Diameter agents:

**Relay Agent**

A Relay Agent is used to forward a message to the appropriate destination, depending on the information contained in the message. The Relay Agent is advantageous because it can aggregate requests from different realms (or regions) to a specific realm, which eliminates the burdensome configurations of network access servers for every Diameter server change.

**Proxy Agent**

A Proxy Agent can also be used to forward messages, but unlike a Relay Agent, a Proxy Agent can modify the message content and, therefore, provide value-added services, enforce rules on different messages, or perform administrative tasks for a specific realm. Figure 2 shows how a Proxy Agent is used to forward a message to another domain. If the Proxy Agent will not modify the content of the original request, a Relay Agent in this scenario would be sufficient.

**Figure 2. The Diameter Proxy Agent**

![Diagram of the Diameter Proxy Agent](image)

**Redirect Agent**

A Redirect Agent acts as a centralized configuration repository for other Diameter nodes. When it receives a message, it checks its routing table, and returns a response message along with redirection information to its original sender. This would be very useful for other Diameter nodes because they won't need to keep a list of routing entries locally and can look up a Redirect Agent when needed. Figure 3 illustrates how a Redirect Agent works. The scenario in Figure 3 below is basically identical to the one in Figure 2, but this time the Proxy Agent is not aware of the address of the contacting Diameter node within example.com. Therefore, it looks up the information in the Redirect Agent of its own realm to get the address.
In addition to these agents, there is a special agent called Translation Agent. The responsibility of this agent, as you might have guessed, is to convert a message from one AAA protocol to another. The Translation Agent is helpful for a company or a service provider to integrate the user database of two application domains, while keeping their original AAA protocols. Another situation is that a company wants to migrate to Diameter protocol, but the migration consists of many phases. The Translation Agent could provide the backward capability for a smooth migration. Figure 4 shows how one agent translates the RADIUS protocol into the Diameter protocol, but, of course, other kinds of protocol translation (for example, Diameter to RADIUS, Diameter to TACACS+) are also possible.

A Diameter message is the base unit to send a command or deliver a notification to other Diameter nodes. For different purposes, Diameter protocol has defined several types of Diameter messages, which are identified by their command code. For example, an Accounting-Request message recognizes that the message carries accounting-related information, while a Capability-Exchange-Request message recognizes that the message carries capability information of the Diameter node sending the message.
Because the message exchange style of Diameter is synchronous, each message has its corresponding counterpart, which shares the same command code. In both previous examples, the receiver of an Accounting-Request message prepares an Account-Answer message and sends it to the original sender.

The command code is used to identify the intention of a message, but the actual data is carried by a set of Attribute-Value-Pairs (AVPs). The Diameter protocol has predefined a set of common attributes and imposes each attribute with a corresponding semantic. These AVPs carry the detail of AAA as well as routing, security, and capability information between two Diameter nodes. In addition, each AVP is associated with an AVP Data Format, which is defined within the Diameter protocol (for example, OctetString, Integer32), so the value of each attribute must follow the data format.

Figure 5 illustrates the relationship between Diameter messages and their AVPs. For more details on each part of Figure 5, see Chapters 3 and Chapter 4 of the Diameter base protocol.

### Figure 5. The Diameter Packet Format

![Diameter Packet Format Diagram]

Table 1 lists all messages defined in the Diameter base protocol:

### Table 1. Messages defined in the Diameter base protocol

<table>
<thead>
<tr>
<th>Message name</th>
<th>Abbreviation</th>
<th>Command code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abort-Session-Request</td>
<td>ASR</td>
<td>274</td>
</tr>
<tr>
<td>Abort-Session-Answer</td>
<td>ASA</td>
<td>274</td>
</tr>
<tr>
<td>Accounting-Request</td>
<td>ACR</td>
<td>271</td>
</tr>
<tr>
<td>Accounting-Answer</td>
<td>ACA</td>
<td>271</td>
</tr>
<tr>
<td>Capabilities-Exchanging-Request</td>
<td>CER</td>
<td>257</td>
</tr>
<tr>
<td>Capabilities-Exchanging-Answer</td>
<td>CEA</td>
<td>257</td>
</tr>
<tr>
<td>Device-Watchdog-Request</td>
<td>DWR</td>
<td>280</td>
</tr>
<tr>
<td>Device-Watchdog-Answer</td>
<td>DWA</td>
<td>280</td>
</tr>
<tr>
<td>Disconnect-Peer-Request</td>
<td>DPR</td>
<td>282</td>
</tr>
</tbody>
</table>
Peer Discovery

Before Diameter, system administrators had to manually configure the Network Access Server with the location of its AAA server, so that when a user came in, the device could send a request to the correct address. Such configuration efforts could, however, be tedious for a complex network deployment. One notable enhancement in Diameter is its peer discovery capability. Besides manual configuration, which must be supported by all Diameter nodes, two options -- SRVLOC and DNS -- may be supported for dynamic peer discovery. The concept here is that it is required for Diameter server, or Diameter agent, to broadcast which applications they support, along with the provided security level.

Diameter clients can then depend on the desired Diameter application, security level, and realm info to look up suitable first-hop Diameter nodes to which they can forward Diameter messages.

For a Diameter node, the discovered peer location as well as routing configuration will be stored locally using two Diameter tables:

**Peer Table.** A Peer Table is mainly used to store the host address of known Diameter nodes. Other information, like whether a Diameter node was looked up dynamically, its status, and security-related information of that node, is also included in this table.

**Peer Routing Table.** There are four important columns of this table requiring extra attention for message routing. The first two are the *Realm Name* and *Application Name*, which act as the selection criteria for message routing. The third is the action to be taken for the target message, which could be PROXY, RELAY, REDIRECT, or LOCAL. LOCAL means the message should be processed locally instead of forwarding to other nodes.

The last one is a reference to an entry in the Peer Table, used to determine the actual host address of the destination. Note that a Peer Routing Table will always contain a default entry for messages not meeting the routing criteria.

**Connection and session**

After an appropriate peer has been discovered, the next step is to establish a connection with that peer. A connection is a physical link between two Diameter nodes. It is mandatory for the Diameter protocol to run over either TCP or SCTP. Compared with UDP, used in RADIUS, these two protocols provide more reliable transportation, which is critical for applications exchanging accounting-related information.

Given that Diameter is basically a peer-to-peer architecture; there could be more than one connection established for a particular node. Diameter protocol explicitly defines that a Diameter
node must establish a connection with two peers per realm at a minimum, which act as primary and secondary contacts. Of course, additional connections could be established when necessary.

Compared with a connection, a session is a logical connection between two Diameter nodes, and can cross multiple connections. A session is actually the concept of a sequence of activities within a timeframe, and refers to the interactions between a Diameter client and a Diameter server in a given period. Each session in Diameter is associated with a client-generated Session-Id that is globally and eternally unique. The Session-Id is then used to identify a particular session during further communication.

**Figure 6** shows the concept of a connection and session:

**Figure 6. Session and connection in Diameter**

![Session and connection in Diameter](image)

**Session initiation**

Like most client-server communication models, a Diameter session starts by issuing a request message from the client to the server. In Diameter context, a Diameter client will send an auth-request message containing a unique Session-Id to a Diameter server (or a Diameter proxy if message forwarding is required). Note that the AVPs to be used for authentication and authorization are application-specific, and that they are not defined in the base protocol.

After accepting the auth-request message, the Diameter server may include an Authorization-Lifetime AVP in the response messaging. This AVP is used to indicate the amount of time in seconds that the Diameter client needs to be re-authorized. After the timeout and acceptable Auth-Grace-Period have passed, the Diameter server will remove the session from its session list and release all resources allocated for the session.

**The session**

During the session, a Diameter server might initiate a re-authentication or re-authorization request. With prepaid service, this type of request is used to check whether the user is still engaging the service, and if not, the server removes the session to avoid further charging.

Also, the Origin-State-Id AVP is used for the inference of exceptional session closure. The sender of a request will include this AVP, and because it is required for the value of this AVP to be monotonically increased, the receiver of the request can easily infer that the previous session was closed, either because the access device was abnormally shutdown or because of some
other exceptional situation. The request receiver can then remove the session from its list, free the resources, and possibly notify its upper stream Diameter servers if it acts as a proxy server.

**Session termination**

Session termination messages are only used in the context of authentication and authorization, and only when the session state was maintained. For accounting services, an accounting stop message is used instead.

A session termination message can be initiated by either the Diameter client or the Diameter server. When a session is deemed to be closed, the Diameter client sends a Session-Termination-Request message to the Diameter server. The *Termination-Clause* AVP is included in this request telling the Diameter server the reason why the session should be closed. Alternately, if the Diameter server detects that the session should be closed -- perhaps because the user runs out of credit or just for administrative purposes -- the Diameter server sends an Abort-Session-Request message to the Diameter client. However, depending upon different policy or usage scenarios, the Diameter client might decide not to close the session even when receiving a session termination message from the server, and let the user keep using its service.

**AAA in Diameter**

**Authentication and Authorization**

As we mentioned, the Diameter protocol isn't bound to a specific application running on top of it. It focuses on general message exchanging features. Because authentication and authorization mechanisms vary among applications, the Diameter base protocol doesn't define command codes and AVPs specific to authentication and authorization. It is the responsibility of Diameter applications to define their own messages and corresponding attributes based on the application's characteristics.

For example, the *AA-Request* message is used to carry authentication and authorization information in the NAS application, while in the SIP application (see Related topics) the message is called *User-Authorization-Request*.

**Accounting**

Unlike authentication and authorization, the behavior and message to be exchanged for accounting is clearly defined. Accounting in Diameter essentially follows a server directed model, which means the device that generates accounting records follows the direction of an authorization server.

Based on the user profile or any business condition, a Diameter server informs the corresponding Diameter client as to what behavior is expected, such as how often the accounting record should be sent from client to server, or if the accounting record should be generated continuously within an accounting session.

Generally speaking, depending on the service to be provided, there are two kinds of accounting records: For one-time invocation-based services, the *EVENT_RECORD* is used. However, if the service will be provided in a measurable period, the accounting record types *START_RECORD*,
INTERIM_RECORD, and STOP_RECORD could be used to mark the start, update, and end of a session.

To prevent duplicated accounting records, each accounting message is associated with a Session-Id AVP along with an Accounting-Record-Number AVP. As this combination can uniquely identify an accounting record, a Diameter node acting as a Diameter agent can use this information to detect duplicated accounting messages being sent to the Diameter server, thereby avoiding unnecessary processing for the Diameter server. This situation might come from temporary network problems or client shutdowns. Also, it is required that the Diameter client keep a local cache of outgoing accounting messages until an acknowledgement message arrives.

**Error handling**

Errors in the Diameter fall into two categories: protocol errors and application errors. Protocol errors refer to something being wrong with the underlying protocol used to carry Diameter messages, perhaps incorrect routing information or temporary network failure.

Applications errors, on the other hand, result from the failure of the Diameter protocol itself, and there are plenty of sources that will cause application errors. For example, when a mandatory AVP is missing in a particular Diameter command, a DIAMETER_MISSING_AVP error code is returned. Every response message in Diameter will carry a Result-Code AVP, and the receiver of a response message can check this AVP to see if the previous message was successfully processed.

To support early connection failure detection, the Diameter protocol defined a Device-Watchdog-Request message. When two connected Diameter nodes don't exchange messages for a certain length of time, this message is sent from either of these nodes to detect possible network problems. The discussion of algorithms to detect transport failures is beyond the scope of this article, but if you are interested in this topic, you can refer to the AAA Transport Profile (see Related topics) for more information.

The Diameter protocol shares the same semantics of error code definition as the HTTP protocol. The return status of messages can be easily identified by checking the first digit of the return code:

1. 1xxx: means the request can't be satisfied and additional information is required for the service to be granted.
2. 2xxx: means the request was processed successfully.
3. 3xxx: means there was a protocol error when transmitting a Diameter message. Generally, a Diameter proxy should try to fix this problem by either routing the message to another Diameter server, or by keeping the message in a local cache and sending it again later.
4. 4xxx: means the requested message cannot be satisfied at the moment, but it might work in the future. An example is a server that temporarily lacks physical storage space to handle any incoming requests.
5. 5xxx: means there was an application error when the server was processing the request message. The sender should not try to send the same message again. Instead, the sender will have to determine the cause of the application error by checking the error code, and then fix the problem.
Besides the Return-Code AVP, the message sender can also check other AVPs that carry additional information for error handling. The Error-Message AVP carries human readable error messages and can be used to determine the actual cause. The Error-Reporting-Host AVP contains the identity of the host generating the Result-Code. This AVP is very helpful for troubleshooting to spot the location of a problem. The Failed-AVP contains the group of AVPs that caused the exception.

After an error has been detected, the sending node forwards all pending messages to an alternative Diameter node. This process is called FailOver. A pending message is a message that has been sent, but hasn’t received its corresponding answer yet. It is required for each Diameter node to keep a local copy of its outgoing messages. The Hop-to-Hop Identification within each message is used to reference out-going messages for each target peer. However, this process may cause a Diameter node to receive an identical message more than one time. As a result, the Diameter node must use the combination of End-to-End Identification message header and Original-Host AVP to uniquely identify a message coming from a specific Diameter node.

Table 2 summarizes some significant changes between the Diameter and RADIUS protocols:

<table>
<thead>
<tr>
<th></th>
<th>Diameter</th>
<th>RADIUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transportation Protocol</td>
<td>Connection-Oriented Protocols (TCP and SCTP)</td>
<td>Connectionless Protocol (UDP)</td>
</tr>
<tr>
<td>Security</td>
<td>Hop-to-Hop, End-to-End</td>
<td>Hop-to-Hop</td>
</tr>
<tr>
<td>Agent Support</td>
<td>Relay, Proxy, Redirect, Translation</td>
<td>Implicit support, which means the agent behaviors might be implemented in a RADIUS server</td>
</tr>
<tr>
<td>Capabilities Negotiation</td>
<td>Negotiate supported applications and security level</td>
<td>Don’t support</td>
</tr>
<tr>
<td>Peer Discovery</td>
<td>Static configuration and dynamic lookup</td>
<td>Static configuration</td>
</tr>
<tr>
<td>Server Initiated Message</td>
<td>Supported. for example, re-authentication message, Session termination</td>
<td>Don’t support</td>
</tr>
<tr>
<td>Maximum Attribute Data Size</td>
<td>16,777,215 octets</td>
<td>255 octets</td>
</tr>
<tr>
<td>Vendor-specific Support</td>
<td>Support both vendor-specific messages and attributes</td>
<td>Support vendor-specific attributes only</td>
</tr>
</tbody>
</table>

**In summary**

In this article, we have gone through the Diameter base protocol from many aspects, including the role and responsibility of different Diameter nodes, the anatomy of a Diameter message, and how messages are sent and received. You have also been given a brief overview of how AAA and Error Handling are achieved in Diameter.

After reading this article, you should have a sense of how the Diameter protocol works and have the base knowledge for exploring it in more detail.

In addition to SIP, Diameter is the other core protocol used in the IP Multimedia Subsystem (IMS) architecture, both in the service plane and the control plane. IMS defines a set of reference points...
between different IMS entities and some of them use Diameter as the underlying protocol to exchange subscription-, presence-, and billing-related messages. For example, the Sh reference point in IMS defined a set of Diameter messages for subscription and notification purposes.

As IMS continues to evolve, we believe there will be more Diameter applications to come, as well as Diameter-related implementations.
Related topics

- **RFC3588**: The Diameter Base Protocol.
- **RFC4005**: Diameter Network Access Server Application.
- **RFC3539**: Authentication, Authorization and Accounting (AAA) Transport Profile.
- **Diameter official Web site**: Find more resources on Diameter.