Monitoring TCP Applications #01

Deb Allen, 2007-30-10

LTM has built-in application health monitor templates for many TCP-based application protocols (FTP, HTTP, HTTPS, IMAP, LDAP, MSSQL, NNTP, POP3, RADIUS, RTSP, RPC, SASP, SIP, SMB, SMTP, SOAP).

If you need to monitor an application which depends on an upper layer protocol for which there is *not* a built-in monitor template, LTM provides a number of options to build a monitor based on the underlying transport layer protocol-- TCP.

I'll cover each of those options in a separate article, starting here with the built-in "tcp" and "tcp_half_open" monitor types.

Overview: tcp and tcp_half_open

Both monitor types attempt to verify the availability of a service by making a TCP connection on the appropriate port.

There are only a couple of differences between the tcp and the tcp_half_open monitors:

monitor	type	reverse/transparent	connection handling	transact with service?
tcp	ECV	yes (optional)	full open, full close	yes (optional)
tcp_half_open	EAV	no	half open, RST close	no

Both have the same standard monitor configuration options of interval, timeout, and alias address/port (for more on those options, and on reverse & transparent options, see the LTM manual section on Configuring Monitors.)

As you will see below, some of the differences are significant and may dictate which monitor is most appropriate for your application.

Monitor Type "tcp"

The tcp monitor is useful for a couple of different scenarios:
Monitoring services that you can't transact with, but want to verify the availability of the socket and close the connection properly (routers, firewalls);
or
Monitoring services with which you can transact a quick request/response in cleartext after the TCP handshake to verify service availability (telnet is abasic example, but the same concept applies to any other text-based protocol).

How it works

In summary, a monitor of type tcp attempts to send and/or receive specific content over a TCP connection. The check is successful when the server response contains the Receive String value. A tcp monitor may optionally be configured with a Send String value and a Receive String value. If the Send String value is blank and a connection is successfully established, the service is considered up. A blank Receive String value matches any response.

The default tcp monitor, with no Send string or Receive string configured tests a service by establishing a TCP connection with the pool member on the configured service port and then immediately closing the connection without sending any data on the connection. This causes some services such as telnet and ssh to log a connection error, filling up the server logs with unnecessary errors. To eliminate the extraneous logging, you can configure the tcp monitor to send enough data to the service to make it happy, or just use the tcp_half_open monitor. Depending on your monitoring requirements, you may also be able to monitor a service that expects empty connections, such as tcp_echo (by using the default tcp_echo monitor) or daytime (specifying the appropriate alias service port when customizing the tcp monitor template).

Here are the details of a tcp monitor in action, including the option for sending data and evaluating the response:

1. The tcp monitor will perform a normal 3-way TCP handshake.

2. If no Send string is configured, the pool member will be marked UP upon successful completion of the 3-way handshake. If a Send string is configured, it will be sent to the server.

3. If the server fails to respond before the timeout, the pool member is marked DOWN. If the server does respond before the timeout, the server response is compared with the Receive string: If no Receive string is configured, the pool member is marked UP; if a Receive string is configured, and the response contains the Receive string, the pool member will be marked UP. If the response does not contain the Receive string, the pool member will be marked DOWN.

4. If the server resets the connection during the handshake or before an expected response is received, the pool member will be marked DOWN and the connection is torn down immediately. In all other cases, the connection will be closed with a normal 4-way close.

```
handshake successful?
|        |
no       yes
|        |
DOWN     send string configured/sent?
|        |
no       yes
```

```
|         |
UP        server response?
|         |         |
close     no        yes
          |         |
          DOWN      recv string configured?
          |         |         |
          close     no        yes
                    |         |
                    UP        recv string match response?
                    |         |         |
                    close     no        yes
                              |         |
                              DOWN      UP
                              |         |
                              close     close
```

Monitor Type "tcp_half_open"

The tcp_half_open monitor is most widely used for gateway monitoring when you just need to ensure the socket is responding to connection requests and desire the lowest overhead on the monitoring target. For example, a busy router would be less impacted by a half open connection request that is immediately reset than a connection that completes the entire open and close handshake sequence. (Although this approach minimizes the impact of monitoring on the monitoring target, it's important to know that the tcp_half_open monitor uses more of LTM's memory than the tcp monitor does, since the tcp_half_open monitor is an EAV that runs a small script outside of TMM, while the tcp monitor is an ECV internal to TMM.)

Another common use for the tcp_half_open monitor is to prevent the application from spewing a bunch of log messages indicating connections were opened but not used. For example, one consultant recently told me he uses the tcp_half_open monitor to verify sshd is alive and answering without filling up /var/log/secure. Telnet has similar issues with connections on which no data is sent.

It should be noted that some applications cannot gracefully handle the half open connection and subsequent reset, so some testing may be in order before implementing this monitor.

How it works

The tcp_half_open monitor sends a SYN packet to the pool member, and if a SYN-ACK is received from the server in response, the pool member is marked UP.

```
SYN sent
|
SYN/ACK rec'd?
|         |
no        yes
|         |
DOWN**    UP
          |
          RST sent
```

**Not fully functional in some versions:

SOL7362: The BIG-IP tcp_half_open monitor does not mark the service as DOWN after receiving a RST packet from the pool member

More info

LTM manual: Configuring Monitors

Get the Flash Player to see this player.

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

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

F5 Networks
Asia-Pacific
apacinfo@f5.com

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

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