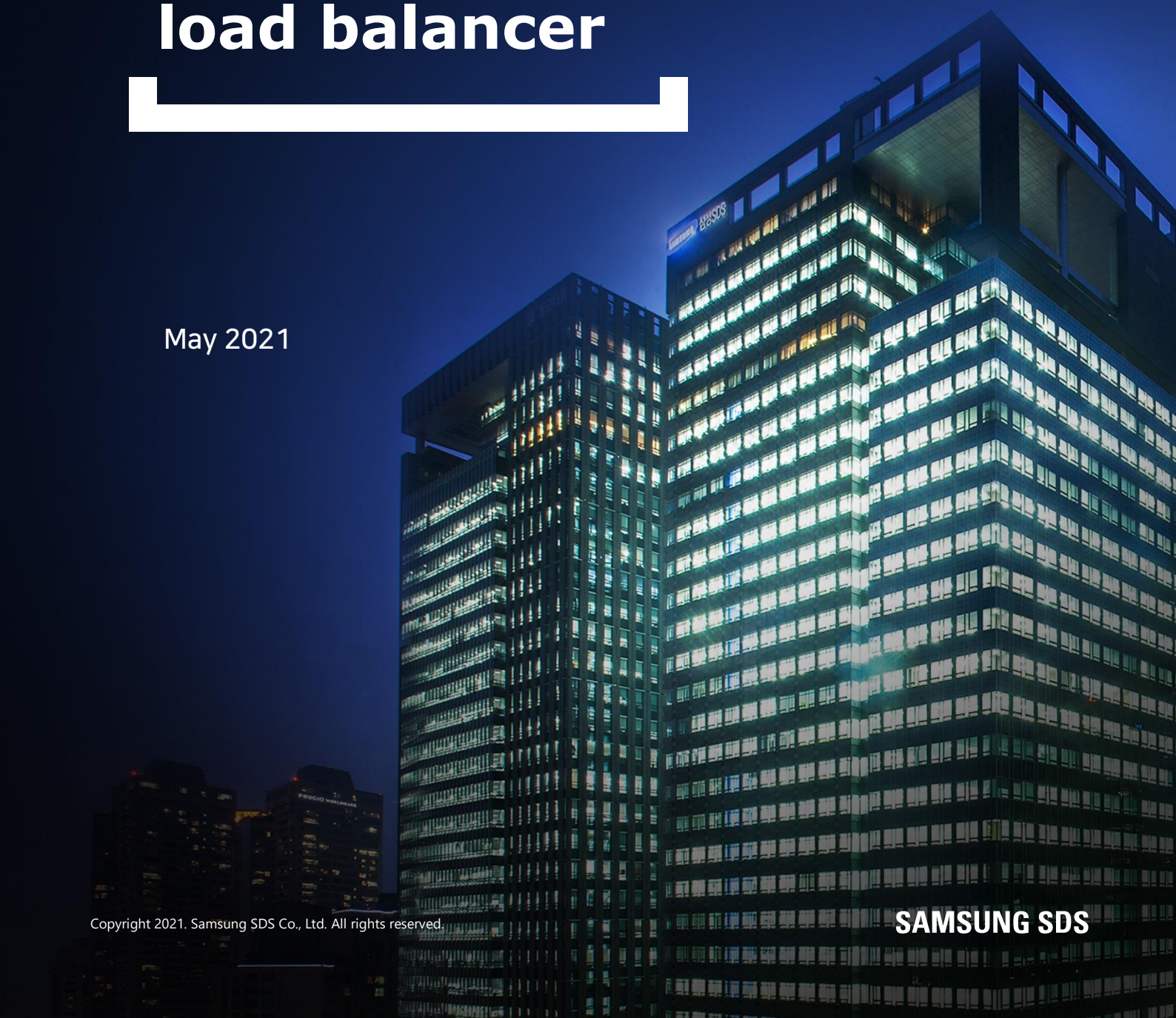
Technical Guide

TCP connection reset case of server connected to load balancer



May 2021

SAMSUNG SDS

Contents

1. Overview

When there is a lot of web traffic, a load balancer is frequently used to distribute multiple servers and handle the traffic and load.

In this document, we are going to look at a case in which connection reset occurs intermittently when connecting to a web server among the failure cases that can occur in an environment using Load Balancer.

2. The issue

In this case, clients are connected to the web server through a load balancer and the server access was unavailable at a specific time.

- A server connection error occurred.
- The client used a server that has other functions to access the web server.

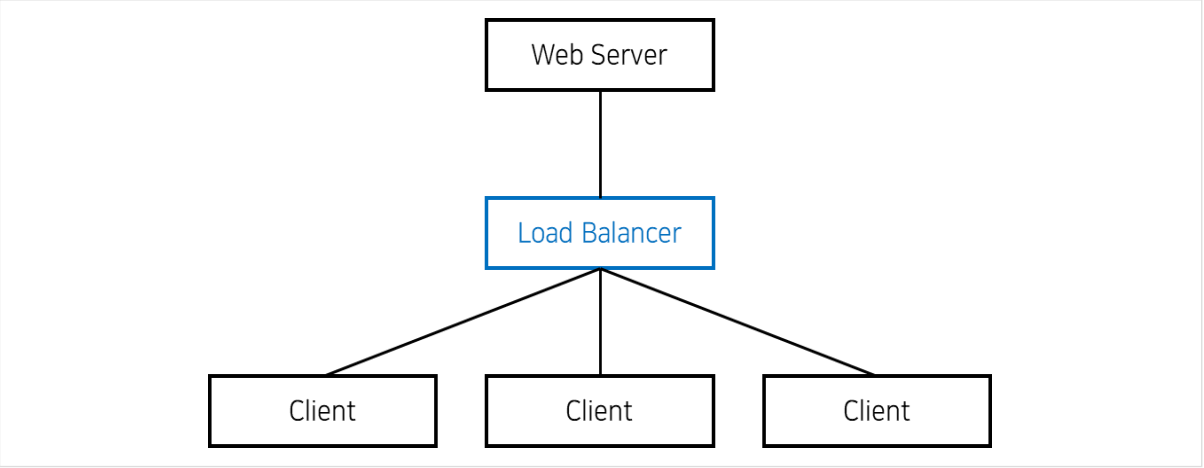


Figure 1. System configuration to connect to web server

3. Tcpdump analysis

Let's summarize the results of analyzing the tcpdump files collected in the event of a problem. In general, a normal TCP 3-way handshaking operates as shown in the figure below.

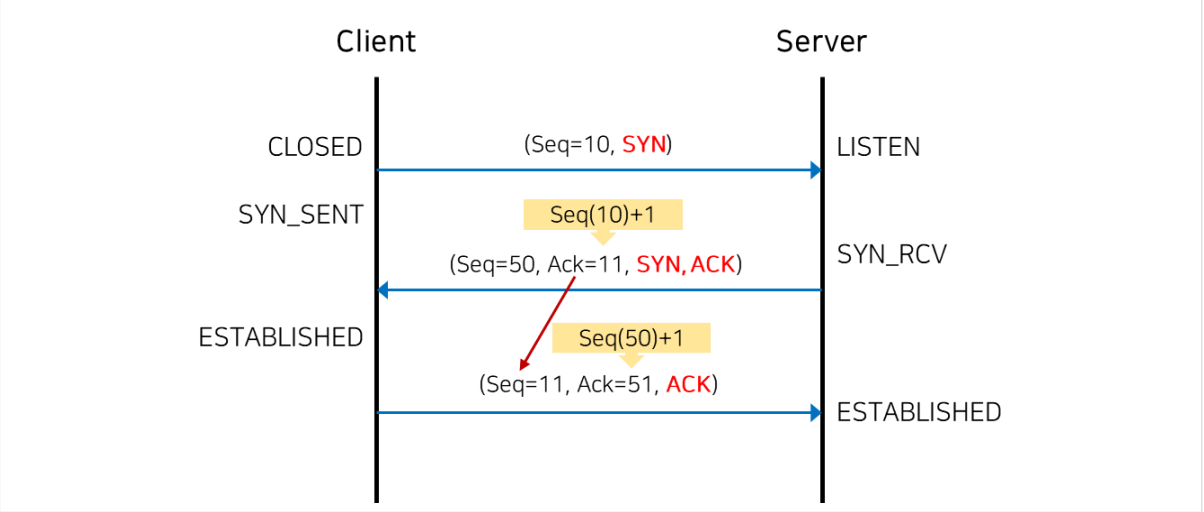


Figure 2. Normal TCP 3-way handshaking

However, when a connection error in the web server occurred as a result of tcpdump analysis, it was confirmed that the client sent an TCP RST packet when attempting to connect to the same port again after a normal termination of the TCP session.

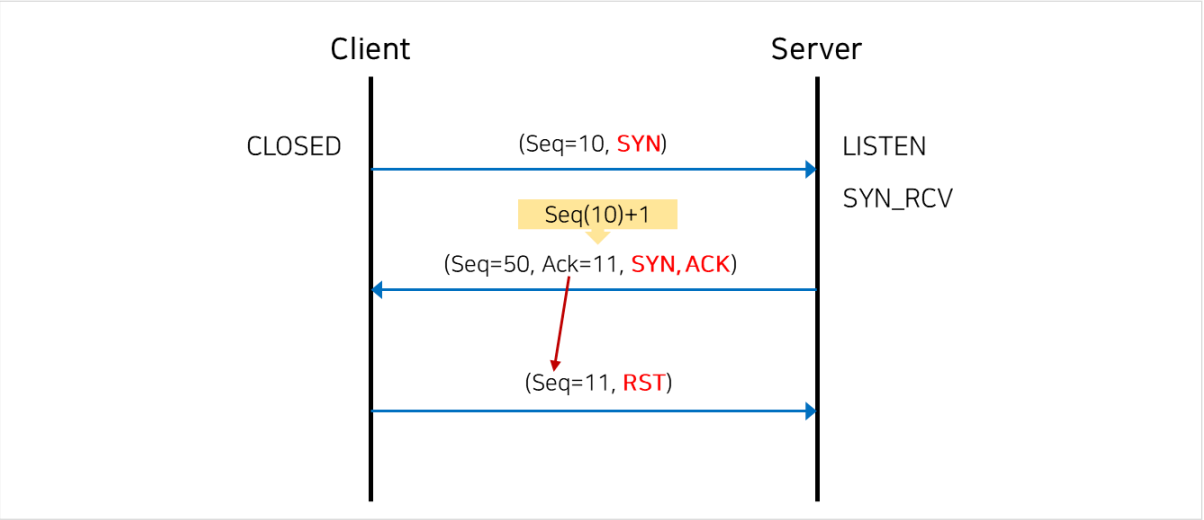


Figure 3. TCP Handshaking when connection error occurs

4. Tcpdump analysis result in detail

A Normal TCP session connection and termination between the client and the web server is shown in the figure below.

As you can see, the last sequence number sent by the web server to the client is 2805574211. After the connection is closed, the web server records this last sequence number, waits in TIME_WAIT state, and disappears after 60 seconds, which is the default value of TIME_WAIT in Linux.

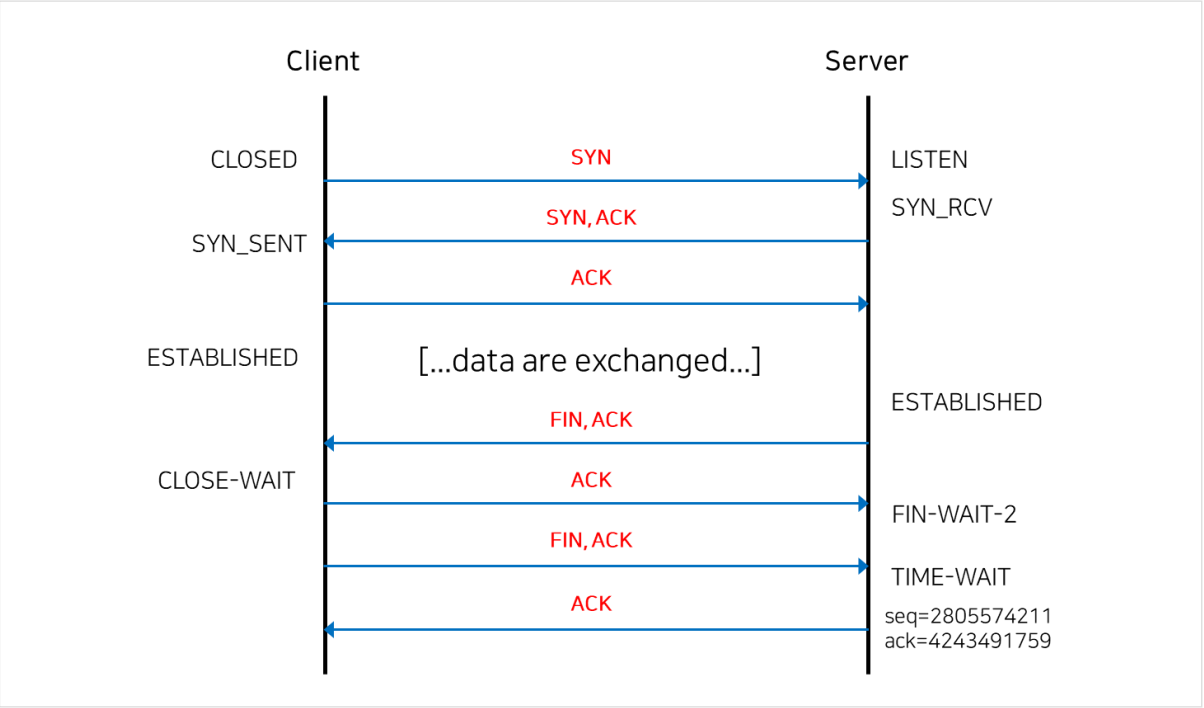


Figure 4. Normal TCP session connection and termination between client and server

When a connection reset issue occurs in the web server, the disconnected session of the web server waits for 60 seconds in the TIME_WAIT state as shown in the figure above. The client therefore can try to connect to the session in the TIME_WAIT state using the same port and new sequence number.
In particular, since we use the Load Balancer IP address when going through the Load Balancer, increasing the number of sessions can increase the likelihood of having the same client ports.

Because the web server is still synchronized on the previous connection and assumes that the client did not receive the last packet, it will resend the same sequence and acknowledgment the number from the previous connection. The client recognizes that the packet does not match what it sent and sends an RST packet to reestablish the connection. Because of this, a connection error occurs when the client tries to connect to the server.

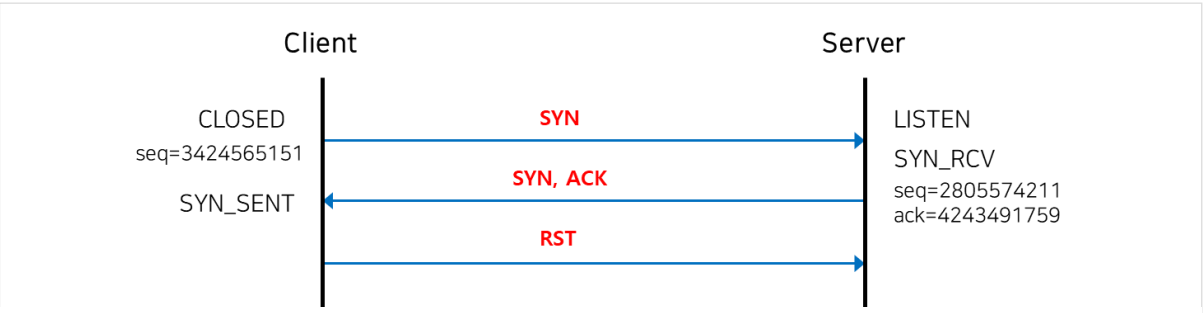


Figure 5. TCP Handshaking when connection error occurs

5. Solutions

You can enable the TCP timestamp parameter to ensure that the server that received the SYN on the second attempt to connect with the new sequence number is actually for a new connection and establishes the connection normally.

The kernel parameters related to timestamps of the client and server were being used by default as the following:

```
net.ipv4.tcp_timestamps = 1 (enable, default)
net.ipv4.tcp_tw_recycle = 0 (disable, default)
net.ipv4.tcp_tw_reuse = 0 (disable, default)
```

Figure 6. TCP parameter settings in kernel

The timestamp parameter was enabled on the client and web server but disabled in Load Balancer, so both the client and web server communicated with the parameter became disabled. This means that the timestamp parameter must be enabled on all connected devices to communicate with the active value.

Finally, you can fix the issue by enabling the timestamp parameter in Load Balancer. When linking between the server and client using Load Balancer, checking the timestamp parameter setting is also required.