Remoting issue

Jenkins remoting may fail to maintain connection between master and slave, and in such case can report a cryptic stacktrace in build console / jenkins log. This page gives some guidance on collecting adequate information to help diagnose the problem.

- **What your users saw**: normally, people who are seeing failed builds are the first one to notice the problem, as they see exceptions like the following show up in the build log. Is it seeing abnormal termination ? Is it seeing EOF like this?

```
java.io.IOException: Unexpected termination of the channel
        at hudson.remoting.SynchronousCommandTransport$ReaderThread.run(SynchronousCommandTransport.java:50)
Caused by: java.io.EOFException
```

- **Failure mode on the master**: On master, the details of the connection loss is captured into the slave log file, and not the build log. So you'll have to go find archived slave launch logs under `$JENKINS_HOME/logs/slaves/*/slave.log*`, which records 10 more recent connection logs to that slave. This information is also captured in the support bundle from Support Core Plugin under `/nodes/slave/*/launchLog/*`. If you look at the bottom of them, you can see how the last 10 connections to the slave has ended.
- **Failure mode on the client**: Collect slave node jenkins-*.out.log and jenkins-*.err.log. This is where stdout and stderr from a JNLP slaves go. See what error is reported in there. Could be "java.net.SocketTimeoutException: Read timed out", as well as OutOfMemoryException, HotSpot JVM crash dump, etc.
- Determine which side is failing first. "read time out" is not fatal as far as TCP is concerned, but a TCP reset is. This let us know which side is detecting the problem

Disable ping thread and get to the root cause

One of the common cause of the connection loss is the forced connection shutdown by the Ping Thread. When a ping thread detects that it's not getting a reply back in 4 minutes, it proceeds to terminate the connection to prevent an infinite hang. This will leave a message like the following in the slave launch log:

```
Ping failed. Terminating
Nov 01, 2014 1:22:35 PM hudson.slaves.ChannelPinger$1 onDead
INFO: Ping failed. Terminating the channel.
```

If you've identified that this is how your connection is lost, then you now need to dig deeper and understand what's causing the ping reply to be so late. Doing this diagnosis usually requires that you disable the ping thread, so that you can cause a slave to hang and exhibit the problem without getting killed. Then use tools like `jmap` and `jstack` to obtain the diagnostic information of the slave JVM. See here and here for how to use those tools.

If you see that one side is hanging on write to the network while the other side is hanging on read from the network, it indicates a network communication problem. For example, if you remove the network cable to the slave, TCP packets will fail to get there in timely fashion, and ping timeout happens before TCP times out.

Pay particular attention to what `SynchronousCommandTransport$ReaderThread.run` is doing, as this is the thread that's responsible for reading remoting requests that arrive from the network and put the tasks into other threads' queue. If this thread is blocking on anything other than reading from the network, then it should be filed as a bug.

SSH slaves

SSH Slaves plugin v1.9 contains improvements to help diagnose the connection loss problem. Users who are seeing a problem is encouraged to upgrade to this version.

To better understand the failure mode of an SSH slave connection, it is helpful to learn the layering of protocols in SSH.

- At the base level, SSH connection is a TCP connection. This is referred as "SSH connection."
- Next up, there is an SSH session. An SSH session is a data stream inside a SSH connection. In Jenkins context, a single SSH session inside a SSH connection is connected to the slave process and controls this process.
- Next up, there is a Jenkins remoting channel. This Jenkins protocol implements the general communication framework between two Java processes (Jenkins master and the slave.)

When a connection to an SSH slave is lost, the first thing you need to figure out is which layerthe failure originated from. The following three are the major sources:

- Ping timeout. See a section above for more discussion about ping problem.
- Loss of SSH socket connection by a TCP problem. For example, if SSHD or firewall in-between sends you a TCP RESET packet to abort the TCP connection, it fails in this way. SSH channels and remoting runs inside SSH connection, so when an SSH connection is lost, they are lost as well. We think this is primarily a network problem, but we are still gathering various failure modes that fall into this category. Look for "Socket connection to SSH server was lost" message, which indicates that the failure is from the socket layer. The following is an example of this failure mode, when I "kill -9" SSHD on the slave:

```
ERROR: Connection terminated
java.io.IOException: Unexpected termination of the channel
        at hudson.remoting.SynchronousCommandTransport$ReaderThread.run(SynchronousCommandTransport.java:50)
Caused by: java.io.EOFException
        at java.io.ObjectInputStream$PeekInputStream.readFully(ObjectInputStream.java:2298)
        at java.io.ObjectInputStream$BlockDataInputStream.readShort(ObjectInputStream.java:2767)
        at java.io.ObjectInputStream.readStreamHeader(ObjectInputStream.java:798)
        at java.io.ObjectInputStream.<init>(ObjectInputStream.java:298)
        at hudson.remoting.ObjectInputStreamEx.<init>(ObjectInputStreamEx.java:40)
        at hudson.remoting.AbstractSynchronousByteArrayCommandTransport.read
(AbstractSynchronousByteArrayCommandTransport.java:34)
        at hudson.remoting.SynchronousCommandTransport$ReaderThread.run(SynchronousCommandTransport.java:48)
ERROR: Socket connection to SSH server was lost
java.io.IOException: Cannot read full block, EOF reached.
        at com.trilead.ssh2.crypto.cipher.CipherInputStream.getBlock(CipherInputStream.java:81)
        at com.trilead.ssh2.crypto.cipher.CipherInputStream.read(CipherInputStream.java:108)
        at com.trilead.ssh2.transport.TransportConnection.receiveMessage(TransportConnection.java:232)
        at com.trilead.ssh2.transport.TransportManager.receiveLoop(TransportManager.java:685)
        at com.trilead.ssh2.transport.TransportManager$1.run(TransportManager.java:481)
        at java.lang.Thread.run(Thread.java:722)
Slave JVM has not reported exit code before the socket was lost
[11/04/14 19:00:50] [SSH] Connection closed.
```

- Abrupt loss of slave JVM process. If the slave experiences a violent JVM crash (such as segfault, kill -9, GC limit), then the slave process will quit, and the remoting channel is lost, but SSH channel and SSH connection stays intact. SSH channel will report back any dying message from the slave process (such as hotspot crash dump that it produces to stdout), and it will also subsequently report the exit code of the slave process. Jenkins master will wait up to 3 seconds for this to come in, and it'll close down the SSH connection. The following example shows one such failure, where the slave has segfaulted. Look for the "Slave JVM has terminated" message, which indicates this failure mode:

```
#
# A fatal error has been detected by the Java Runtime Environment:
#
#  SIGSEGV (0xb) at pc=0x00007fbae76a861b, pid=5998, tid=140440723117824
#
# JRE version: 6.0_33-b33
# Java VM: OpenJDK 64-Bit Server VM (23.25-b01 mixed mode linux-amd64 compressed oops)
# Derivative: IcedTea6 1.13.5
# Distribution: Ubuntu 12.04 LTS, package 6b33-1.13.5-1ubuntu0.12.04
# Problematic frame:
# C  [jna8661775961556997745.tmp+0xa61b]  Java_com_sun_jna_Native_setInt+0x9b
#
# Failed to write core dump. Core dumps have been disabled. To enable core dumping, try "ulimit -c unlimited"
before starting Java again
#
# An error report file with more information is saved as:
# /tmp/slave1/hs_err_pid5998.log
#
# If you would like to submit a bug report, please include
# instructions how to reproduce the bug and visit:
#   https://bugs.launchpad.net/ubuntu/+source/openjdk-6/
# The crash happened outside the Java Virtual Machine in native code.
# See problematic frame for where to report the bug.
#
ERROR: Connection terminated
java.io.IOException: Unexpected termination of the channel
        at hudson.remoting.SynchronousCommandTransport$ReaderThread.run(SynchronousCommandTransport.java:50)
Caused by: java.io.EOFException
        at java.io.ObjectInputStream$PeekInputStream.readFully(ObjectInputStream.java:2298)
        at java.io.ObjectInputStream$BlockDataInputStream.readShort(ObjectInputStream.java:2767)
        at java.io.ObjectInputStream.readStreamHeader(ObjectInputStream.java:798)
        at java.io.ObjectInputStream.<init>(ObjectInputStream.java:298)
        at hudson.remoting.ObjectInputStreamEx.<init>(ObjectInputStreamEx.java:40)
        at hudson.remoting.AbstractSynchronousByteArrayCommandTransport.read
(AbstractSynchronousByteArrayCommandTransport.java:34)
        at hudson.remoting.SynchronousCommandTransport$ReaderThread.run(SynchronousCommandTransport.java:48)
Slave JVM has terminated. Exit signal=ABRT
[11/04/14 18:51:53] [SSH] Connection closed.
```

The following example shows "kill -9" to the slave process. Note the "Exit signal=KILL" in the message:

```
Slave successfully connected and online
ERROR: Connection terminated
java.io.IOException: Unexpected termination of the channel
        at hudson.remoting.SynchronousCommandTransport$ReaderThread.run(SynchronousCommandTransport.java:50)
Caused by: java.io.EOFException
        at java.io.ObjectInputStream$PeekInputStream.readFully(ObjectInputStream.java:2298)
        at java.io.ObjectInputStream$BlockDataInputStream.readShort(ObjectInputStream.java:2767)
        at java.io.ObjectInputStream.readStreamHeader(ObjectInputStream.java:798)
        at java.io.ObjectInputStream.<init>(ObjectInputStream.java:298)
        at hudson.remoting.ObjectInputStreamEx.<init>(ObjectInputStreamEx.java:40)
        at hudson.remoting.AbstractSynchronousByteArrayCommandTransport.read
(AbstractSynchronousByteArrayCommandTransport.java:34)
        at hudson.remoting.SynchronousCommandTransport$ReaderThread.run(SynchronousCommandTransport.java:48)
Slave JVM has terminated. Exit signal=KILL
[11/04/14 18:52:52] [SSH] Connection closed.
```


There was a bug in remoting that prevented stdout from the slave process from getting captured into the master. Kohsuke hasn't determined exactly when this regression was introduced, but he thinks it's around remoting 2.38 (which appeared in Jenkins 1.560.) You can tell whether your connection is affected by this bug by looking for the message "Evacuated stdout" in slave launch log. If you do not see this message, you are affected. This problem was resolved in remoting 2.48 (which will appear in Jenkins 1.590.) If you are running affected versions of Jenkins, you will not see the hotspot crash dump message.

Custom launcher

If you use the option "Launch agent via execution of command on the master", then be aware that the input and output to this command will be used for the communication between the central server and the remote slaves. Do not pollute the output stream with additional output and do not consume anything from the input stream.

Usually, the launcher command will be a shell script on the server that does some preparation and then uses ssh to execute slave.jar on the remote slave.

In the preparation, avoid additional ssh invocations to perform preparations on the remote slave, because ssh will consume bytes from the input stream, even if the remote command does not. See this stackoverflow answer for details and workarounds. If the problem you are seeing is java.io. StreamCorruptedException: invalid stream header: 0BDAACED (or another number), this is most likely your problem.