Verifying Packet Loss Detection Tools

ABSTRACT

This report describes a procedure to verify the correct operation of tools intended to detect and report packet loss.

INTRODUCTION

When using UDP-based protocols, a certain amount of loss is a fact of life. Most UDP loss happens at one or more of three specific points: switch (egress queue), NIC (packet buffers), or socket buffer.

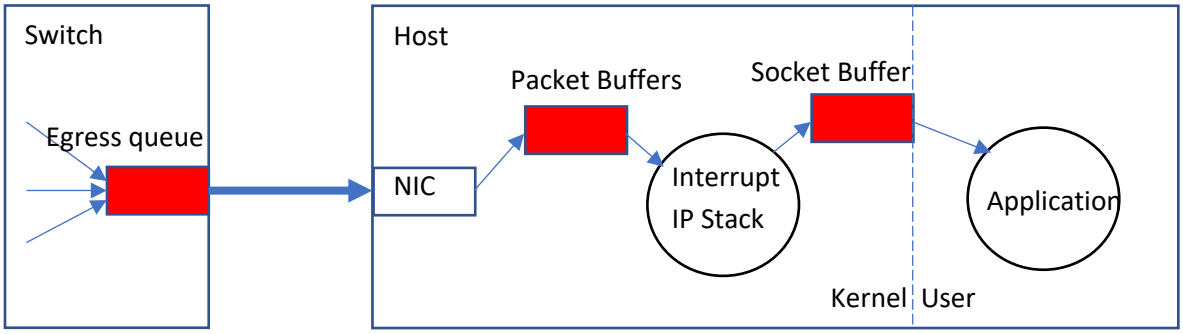

Figure 1: UDP loss points

Switch loss: A switch can lose packets in its egress queue if multiple up-stream senders are bursting data at a combined rate greater than the Ethernet link. For short bursts, the egress queue simply holds the packets until they can be drained to the receiving host. But if the incoming bursts are long enough (typically less than a second), the egress queue can overflow, dropping packets. This is known as switch port oversubscription.

NIC loss: The host NIC works by receiving packets and transferring them into host memory by DMA. The NIC itself has a fixed number of receive descriptors which basically contain pointers to packet buffers in host memory. When packets are available, the NIC interrupts the host CPU and the driver passes the packets through the IP stack and into the socket buffer(s) for delivery to the applications. However, there are circumstances where the Kernel can get overloaded, so that it cannot process received packets as fast as they are coming in. If the packet buffers are all full and the NIC receives additional packets, those packets will be dropped by the NIC.

Socket buffer loss: The application (an Ultra Messaging context thread, for example) is responsible for reading UDP datagrams out of the socket buffer. If the application is not able to run fast enough, it might be unable to process datagrams as quickly as they are entering the socket buffer. If this speed mismatch lasts long enough, the socket buffer will become full and the kernel will drop newly received datagrams.

In almost all well-designed, well-provisioned systems, a small number of UDP packet drops can be expected in the switch's egress queue. Ten packets per hour might be considered acceptable by many users; others will work to drive the lost packet rates even lower (it can be done). High rates of switch loss, or loss in the NIC or socket buffers, generally represent a pathological overload condition which needs to be diagnosed and treated. Even if the higher-level protocols are able to recover this "pathological" lost data, these undesirable forms of loss cause latency, extra CPU load, can be a warning of larger problems in the near future - unrecoverable loss.

There are a variety of treatment methodologies, depending on the type of loss. A detailed discussion of those treatments is beyond the scope of this document. However, an accurate diagnosis of the location of the loss is critical to selecting the proper treatment. The tools used to detect and report loss must work reliably. Unfortunately, there are many cases where the typical tools do not work properly.

For example, the "`netstat -s`" command is commonly used on Linux systems to detect socket buffer loss (UDP receive errors). However, although Windows systems do have the "`netstat -s`", it does not detect and report socket buffer overflow for Windows versions prior to version 7. See Appendix 1: Common Tools for more information.

The main purpose of this document is to provide a procedure to verify that your loss detection/reporting tools really do work. The general process is to force each form of loss, and verify that the corresponding tool detects and reports that loss.

Once you have verified which of your loss detection tools work properly, you should implement some form of continuous monitoring and storage of that information. If you get loss in a production system and sample the counters after the fact, you don't know if the non-zero counters are from the event that just happened, or if they are old. Sample the tools every 5 or 10 minutes - you'll be glad you did.

Much of the information contained in this document has been learned through experience with users. Please contribute to the quality of this document by sharing your experiences when performing this procedure. In particular, we would like to know if anybody has successfully detected socket buffer loss in Windows versions before 7, and switch loss in Cisco 10-Gig NEXUS switches. Please send all of your experiences to sford@informatica.com.

BEFORE YOU START

The procedure makes use of the "mtools" programs "msend" and "mdump". These are available free of charge at https://community.informatica.com/solutions/informatica_mtools where you will find pre-built packages for a variety of platforms, plus source code for both tools and documentation. The tools make use of high-rate multicast streams, so please make sure that your network can handle that kind of traffic.

1. Download the mtools package to each machine on which you want to verify the loss diagnosis tools. Note that these tools are updated periodically, so a fresh download should be done. There is no "installation procedure" for the tools (other than decompressing them). They are provided as simple command-line executables. Also note that for most platforms, only a 32-bit executable is provided. But for Windows, both a 32-bit and a 64-bit is provided. This is because 32-bit programs have very poor UDP send performance on 64-bit Windows 2003. (Win 2008 appears to fix this.)
2. Contact your network administrator and arrange a time that they can work with you. They will need to allocate a multicast group (address) for you, and tell you which network to use. During the procedure, they will also need to monitor the switch port statistics and tell you if output drops happen.
3. Contact your host system administrator and arrange to be able to reconfigure the host's NIC for flow control ON and OFF. During the procedure, you will need to change that setting more than once. (On some Microsoft systems, the NIC driver does not have a flow control option. Often, simply downloading the latest driver fixes this.)
4. Determine which tools are supposed to detect and report loss at each of the points: switch, NIC, and socket buffer. See Appendix 1: Common Tools for suggested tools.
5. Gain access to two machines: the sender and the receiver. The receiver machine will be the focus for verifying the loss detection tools. Determine the IP addresses of the interfaces.

In the procedure that follows, you will use the mtools "msend" and "mdump" commands. The "mdump" command should be run on the machine on which you wish to verify the loss detection tools. The "msend" tool should be run on a separate machine ("sender machine"), usually of the same operating system as the system under test.

TEST 1: BASELINE

This test run verifies that the hardware and operating system are able to handle the traffic generated by mtools. This test should run without any loss. If loss is detected during this phase, then it will be very difficult to properly verify your loss detection tools.

Prep: make sure NIC flow control is turned OFF.

1. Use your loss detection tools to get current counts of all 3 forms of loss (switch, NIC, socket buffer). Note those values.
2. On the system under test:
   `mdump -q` *MCAST_ADDR* `12000` *INTFC_ADDR*
3. On the sender machine:
   `msend -5` *MCAST_ADDR* `12000 15` *INTFC_ADDR*
4. When the "msend" completes, use your loss detection tools to get new counts of all 3 forms of loss. Compare these values with those taken in step 1.

RESULTS:

The test should run for just a few seconds. The "mdump" command should report zero loss. The loss detection tools should also report zero loss at each of the three loss points.

DISCUSSION:

The "msend -5" tool generates 50,000 messages of 800 bytes each in a single intense burst. This should generate close to a full gigabit of bandwidth, but only for a few seconds. A network utilization measurement tool which averages over a longer period of time, like 5 seconds, will not report that level of network load. A properly configured network and receiving system should have no trouble receiving all messages.

Note the use of port "12000" - this was chosen somewhat arbitrarily. Feel free to change it if it conflicts with existing port usage. Also, the multicast TTL is set to "15". One might think that "1" would be a safer setting, but it turns out that TTL=1 often places a heavy load on switches. See https://www.informatica.com/downloads/1568_high_perf_messaging_wp/Topics-in-High-Performance-Messaging.htm#TTL-1-AND-CISCO-CPU-USAGE for additional detail.

TEST 2: SOCKET BUFFER LOSS

This test run verifies that the socket buffer loss detection tool works. On Unix and Windows version 7 and beyond, this consists of the "netstat -s" command, and the counter is "UDP receive errors". Pre-7 Windows users are out of luck. This test should have significant loss reported by both the "mdump" command and the loss detection tool.

Prep: make sure NIC flow control is turned OFF.

1. Use your loss detection tools to get current counts of all 3 forms of loss. Note those values.
2. On the system under test:
   `mdump -q -p1000/5` *MCAST_ADDR* `12000` *INTFC_ADDR*
3. On the sender machine:
   `msend -5 -s2200` *MCAST_ADDR* `12000 15` *INTFC_ADDR*
4. When the "msend" completes, it may take up to 5 more seconds for the "mdump" command to also complete. Use your loss detection tools to get new counts of all 3 forms of loss. Compare these values with those taken in step 1.

RESULTS:

The test should run for about 7 seconds. The "mdump" and socket buffer detection tools should report significant loss, typically around to 90%. The NIC and switch tools should report no loss.

DISCUSSION:

The "-p1000/5" option causes the "mdump" command to pause for 1000 milliseconds between each of the first 5 messages read. Since the sending machine is sending packets at full speed, this will fill the socket buffer to overflowing very quickly. After those first 5 messages, "mdump" reads the rest of the messages at full speed. The "-s2200" option on "msend" makes it pause for 2200 milliseconds before sending "mdump" a "stat" command to report the loss statistics. This gives "mdump" time to read 2 messages, making room in the socket buffer for that "stat" command.

TEST 3: NIC LOSS

This test run verifies that the NIC loss detection tool works. The correct tool varies by the version of the OS and the driver, and some OS/drivers do not expose NIC loss statistics at all. On some Unix systems, the command is "ifconfig INTFC_NAME" has a counter for "receiver overrun". Some Unix systems use "netstat -Si INTFC_NAME". Some Unix systems use "ethtool". Some Unix systems have NIC receiver discards/overrun available in the "/proc" file system. Some Unix and Windows systems have a utility for reporting NIC statistics supplied by the NIC vendor or system integrator.

Prep: make sure NIC flow control is turned OFF.

1. Use your loss detection tools to get current counts of all 3 forms of loss. Note those values.
2. On the system under test run 15 copies of "mdump":
   `mdump -q` *MCAST_ADDR* `12000` *INTFC_ADDR*
3. On the sender machine:
   `msend -5` *MCAST_ADDR* `12000 15` *INTFC_ADDR*
4. When the "msend" completes, use your loss detection tools to get new counts of all 3 forms of loss. Compare these values with those taken in step 1.

RESULTS:

The test should run for just a few seconds. The "mdump" command and the NIC loss tool should report significant loss. If by chance "mdump" reports no loss, try again with more copies of "mdump" running.

The socket and switch detection tools should report zero loss.

DISCUSSION:

Running 15 copies of "mdump" will overload the kernel. Each message received will have to be replicated 15 times in software by the kernel, and will be copied to the 15 sockets, and then 15 processes need to be woken up. Even on a fast CPUs, this takes significantly more time than "msend" takes sending messages. Thus, the kernel falls behind, the NIC fills its packet buffers, and then drops packets.

TEST 4: SWITCH LOSS

This test run verifies that the switch loss detection tool/method works. The vast majority of switches on the market will accurately count drops at the output port, but it can sometimes be a challenge to determine which counter is the right one. Typically the switch does not count this kind of packet drop as an "error", since it is technically the application's fault that the drops have to happen (switch port oversubscription). The counter is normally called "output drops" or something like that. There is a report that new 10-Gig Cisco NEXUS switches might not properly report output drops.

Prep: turn NIC flow control is turned ON. <-- Look! A change! :-)

1. Use your loss detection tools to get current counts of all 3 forms of loss. Note those values.
2. On the system under test run **30 copies** of "mdump":
   `mdump -q` *MCAST_ADDR* `12000` *INTFC_ADDR*
3. On the sender machine:
   `msend -5` *MCAST_ADDR* `12000 15` *INTFC_ADDR*
4. When the "msend" completes, use your loss detection tools to get new counts of all 3 forms of loss. Compare these values with those taken in step 1.

RESULTS:

The test should run for just a few seconds. The "mdump" command and the switch loss tool should report significant loss. The socket and NIC detection tools should report zero loss.

DISCUSSION:

As with test 3, running 15 copies of "mdump" will overload the kernel. However, with NIC flow control turned on, the NIC will signal the switch to slow down its sending rate to prevent NIC loss. Instead, the switch egress queue will fill and drop packets.

A network administrator should be able to quickly examine the proper switch port and tell you that there are output drops. In this case, the "tool" is sometimes the telephone or an IM screen to ask the administrator to check. One obvious problem with this approach is that if loss is suspected in a production system, it is often too late to catch it "in the act". Whenever possible, we recommend that the network administration group implement some form of periodic sampling and archiving of critical systems' output ports. See Appendix 1: Common Tools for suggestions.

SOMETIMES, LIFE ISN'T FAIR

So, let's say that you discover that one of the loss detection tools doesn't work in your environment. Worse yet, what if *two* tools don't work! If you have loss, how can you treat it if you can't even identify where it is happening?

If only one tool doesn't work, then you can use the process of elimination. For example, if a pre-7 Windows machine is getting packet loss, and the NIC and switch report no loss (and you've proven that those tools work properly), then you can safely assume that the loss is in the socket buffer. But what if you DO see loss in the NIC or switch? Can you assume that there is no loss in the socket buffer? No, you can't. But at least you can try to treat the NIC or switch loss. Once you eliminate those sources of loss, any remaining loss would be in the socket buffer.

Ok, so what if both socket buffer and NIC loss are not directly detectable? All you have is switch loss detection. Are you stuck? Not necessarily. Although I generally recommend configuring NICs with flow control OFF, this would be a situation where it makes sense to have it ON. This basically moves NIC loss into the switch. If you have a loss event and the switch shows no loss, then you can be pretty sure it is in the socket buffer. But what if the switch *does* show loss? How do you know if it is because of port oversubscription v.s. kernel overload? You can't. But, if you can reproduce the loss condition, you can experiment by turning flow control back OFF. Reproduce the loss and check the switch again. If the switch still reports loss, you know it is real switch loss (port oversubscription). If not, then it is NIC loss (kernel overload).

I always prefer to have all three tools working properly since I don't like to make assumptions. But, you know the old saying: sometimes life isn't fair. We do the best we can with the tools we have.

APPENDIX 1: COMMON TOOLS

As described, these tools sometimes work, sometimes not. For Linux and Solaris, the socket buffer tools seem to be very reliable (we have never seen it not work correctly). However, we have seen many cases where the NIC tool does not work.

Linux

Socket buffer loss tool:

```
netstat -s
```

(look for the UDP field "packet receive errors")

NIC loss tool:

```
ifconfig eth0
```

<--- *replace eth0 with correct NIC name*

(look for "RX packets ... overruns:")

Alternate NIC loss tool; must be run as root:

```
ethtool -s eth0
```

<--- *replace eth0 with correct NIC name*

(proper field name varies by NIC and driver)

Solaris-10

Socket buffer loss tool:

```
kstat | grep udpInOverflows
```

NIC loss tool:

```
kstat -n bge0 | grep norcvbuf
```

<--- *replace bge0 with correct NIC name*

Windows

Socket buffer loss tool; Windows-7 (and beyond?):

```
netstat -s
```

(look for the UDP field "receive errors")

NIC loss tool varies by the NIC and driver and is often a separate executable.

Cisco switch loss:

A possible Unix command that a network administrator could use is:

```
snmpwalk -v 1 -c public SWITCH_ADDR IF-MIB::ifOutDiscards
```

Note that the above community string ("public") is probably not enabled; the network administrator will know the appropriate value. Ideally, the network administrator would run that command every 5 or 10 minutes, logging to a file, with a time stamp. If this log file could be shared read-only to the project groups, they can time-correlate any unusual application event with loss reported by the switch.