Troubleshooting

Troubleshooting is the art of taking a problem, gathering information about it, analyzing it, and finally solving it. While some problems are inherently "harder" than others, the same basic approach can be taken for every problem.

Using a fixed approach ensures that critical steps are not left out, and that troubleshooting becomes a repeatable process. Not just fixing.

While fixing a problem is one of the major parts of troubleshooting, there are other parts that cannot be neglected: documenting the problem (and fix) and performing a root cause analysis (RCA).

Documenting the problem (and the fix) can help in the future when another (or possibly the same) administrator is faced with the same, or a similar, problem. Performing a root cause analysis can help in preventing similar problems in the future.

Using the scientific method, A good schema to follow when troubleshooting is the scientific method:

1. Clearly define the issue

Take a step and view the larger picture, then clearly define the actual problem. Most problems that get reported are symptoms of another problem, not the actual problem. For example, a user might call about a problem signing into a machine. While this is a problem for the user, the actual problem can be a forgotten password, an incorrectly configured machine, a networking issue, or something else entirely. Further investigation is needed to determine what the cause of this symptom is.

A typical action during this process is attempting to recreate the issue, or observing the issue as it happens.

2. Collect information

The next step is collecting as much (relevant) information as possible. This information can come from a wide variety of sources: reading log files, information displayed on screen or in a GUI, follow-up questions for the original reporter, etc.

During this step, "relevant information" is still a broad term; information from seemingly unconnected subsystems can turn out to be useful.

On the other hand, gathering too much information is also unnecessary and might even be counterproductive, as all information will need to be viewed, assessed, and interpreted.

3. Form a hypothesis

After looking at all gathered information, and the symptoms observed/reported, it is time to form a hypothesis about the cause of the problem.

Sometimes this can be easy; for example, when a user has forgotten his password. Other times, it can be harder;

4. Test the hypothesis

With an initial hypothesis formed, it can be tested for validity. How this testing happens depends on the problem and the hypothesis. For example, when the hypothesis for a login problem states, "The network connection between the workstation and the KDC is being interrupted by a firewall," the testing will be different from a hypothesis for a spontaneously rebooting server including a faulty UPS. If, after testing, the hypothesis still appears valid, an attempt at fixing the problem can be made. If the hypothesis is found to be invalid, a new hypothesis will need to be formed, possibly using extra information found during the testing of the previous hypothesis.

5. Fixing the problem

If a hypothesis was not found to be invalid, an attempt can be made to fix the problem.

During this stage, it is vital to only change one variable at a time, documenting all changes made, and testing every change individually. Keeping backups of any changed configuration files and reverting to those backups if a change was found to be ineffective, is also crucial. Modifying multiple configurations at once typically, only leads to further issues, not fixes. After any change, the entire configuration will need to be tested to see if the issue has been resolved, reverting any changes, and possibly forming a new hypothesis, if a change has not resolved the issue.

6. Rinse & repeat

If the proposed fixes did not actually resolve the issue, the process will need to be restarted from the top. This time, any new information discovered during this cycle can be added to the mix to form a new hypothesis.

Storage Troubleshooting

How to Check Disk Type:

cat /sys/block/sda/queue/rotational

You should get output 1 for HDD disks and 0 for a SSD disks.

Definition:

Writes/sec – write operations rate.
Reads/sec – read operations rate.
Busy time – the % of the elapsed time when your disk drive was busy in servicing write or read requests.
Queue length – the number of requests on the disk that are in the queue.

sda0, sda1 are the partitions of the hard drive (sda) attached to your machine.

dm-0 & dm-1 are the Logical volume managers' logical volumes you would have created while installing or configuring your machine. If you're using either LVM or encrypted volumes, you'll see dm-X devices.

Check Disk Statistics:

```
# iostat -dx
# iostat -dx 1 3          // 1 second interval, 3 times //

# mpstat                  //  To get CPU utilization information, type in following command.
# mpstat -P ALL       // To monitor individual processor performance.
```

MAP dm -X to LV:

```
# for x in /dev/mapper/*; do echo "$(realpath $x) -> $x"; done;
```

Top Command:

```
# us - user cpu time (or) % CPU time spent in user space
# sy - system cpu time (or) % CPU time spent in kernel space
# ni - user nice cpu time (or) % CPU time spent on low priority processes
# id - idle cpu time (or) % CPU time spent idle
# wa - io wait cpu time (or) % CPU time spent in wait (on disk)
# hi - hardware irq (or) % CPU time spent servicing/handling hardware interrupts
# si - software irq (or) % CPU time spent servicing/handling software interrupts
# st - steal time % CPU time in involuntary wait by virtual CPU while hypervisor is servicing another processor (or) % CPU time stolen from a virtual machine.
```

What is Zombie:

Zombie (Z): The kernel maintains various data structures in memory to keep track of processes. A process may create several child processes, and they may exit while the parent is still around. However, these data structures must be kept around until the parent obtains the status of the child processes. Such terminated processes whose data structures are still around are called zombies.

Linux Network Troubleshooting:

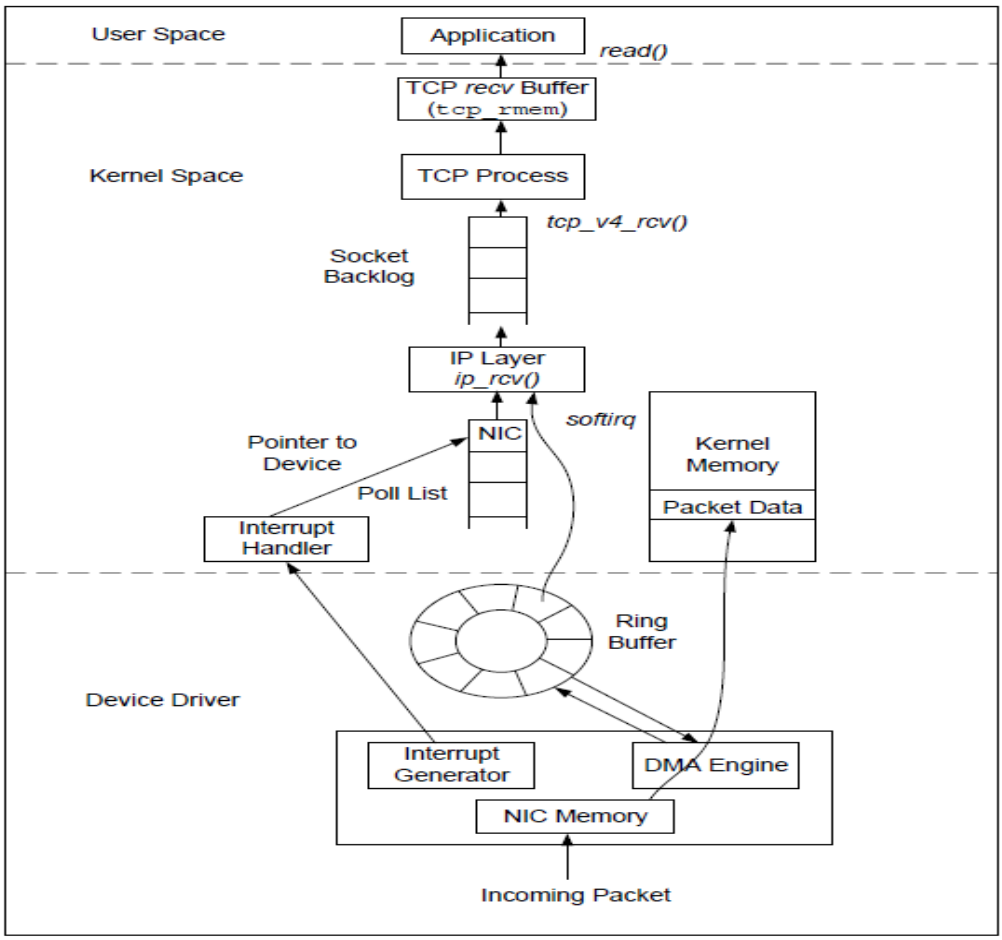

Fig. 1: Packet Reception

- ✓ qdisc/txqueuelen value is the packet sending queue. If the output queue is full, the attempt to enqueue a packet generates a local congestion event, which is propagated upward to the TCP layer.
- ✓ sk_buff is managing all the queue to and from ring_buffer.
- ✓ ring_buffer give and take packet from NIC DMA.

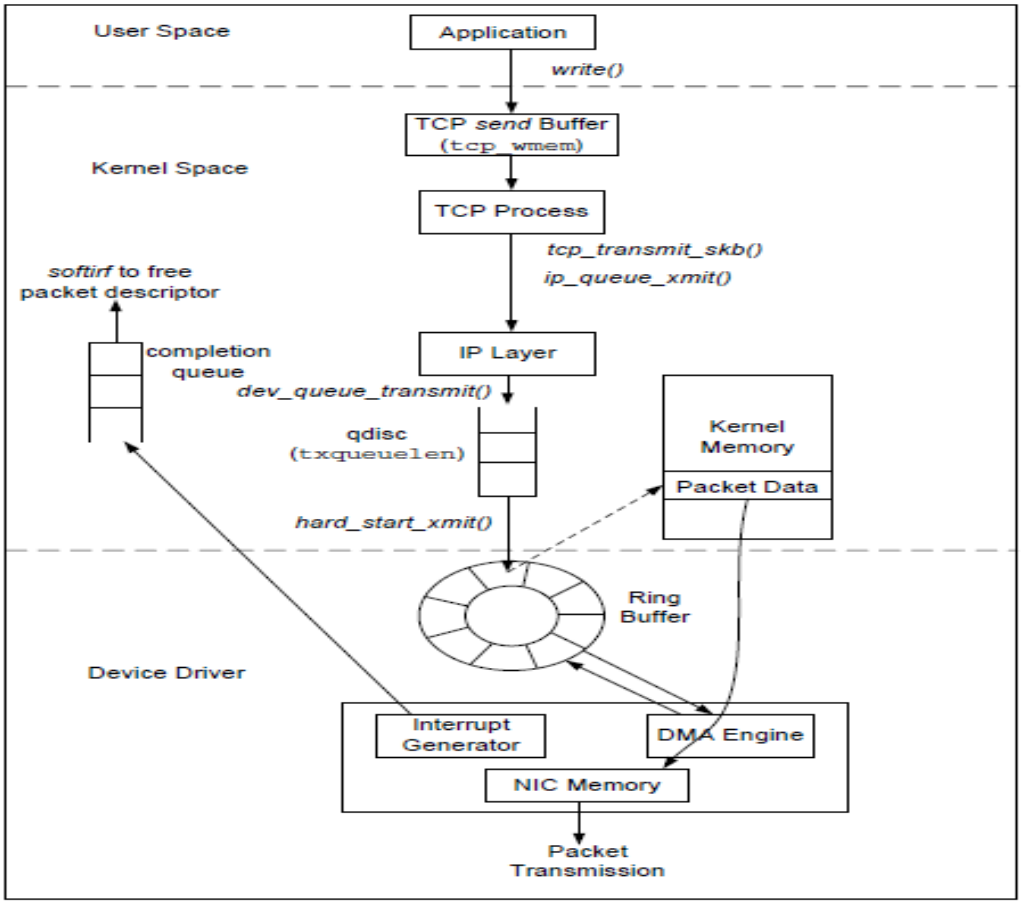

Fig. 2: Packet Transmission

- ✓ **TCP receive buffer size from IP Layer managing by softirq:**

[root@node1 ipv4]# cat /proc/sys/net/ipv4/tcp_rmem
4096 87380 6291456

- ✓ **TCP send buffer size to IP Layer managing by qdisk:**

[root@node1 ipv4]# cat /proc/sys/net/ipv4/tcp_wmem
4096 16384 4194304

```
# ifconfig
# ifconfig enp0s3 | grep RX
# ifconfig enp0s3 | grep TX
# ethtool -g eth0
# ethtool -c eth0
```

```
# ethtool -G enp0s3 tx 512

# ethtool -G enp0s3 rx 512

# ethtool -S enp0s3 To display drop counters
# ethtool enp0s3
# ethtool -h
# grep -n eth0 /var/log/messages
# ping -c 1 -W 3 172.25.250.254
# echo $?
0

# ifconfig
# ethtool -S enp0s3
# sar -n EDEV
# netstat -s
# dropwatch -l kas
```