Contents

[1. High level path a packet takes from arrival to socket receive buffer is as follows: 1](#_Toc183981573)

Linux Network Driver Flow

# High level path a packet takes from arrival to socket receive buffer is as follows:

1. Driver is loaded and initialized.

1. Packet arrives at the NIC from the network.
2. Packet is copied (via DMA) to a ring buffer in kernel memory.
3. Hardware interrupt is generated to let the system know a packet is in memory.
4. Driver calls into [NAPI](http://www.linuxfoundation.org/collaborate/workgroups/networking/napi) to start a poll loop if one was not running already.
5. ksoftirqd processes run on each CPU on the system. They are registered at boot time. The ksoftirqd processes pull packets off the ring buffer by calling the NAPI poll function that the device driver registered during initialization.
6. Memory regions in the ring buffer that have had network data written to them are mapped.
7. Data that was DMA’d into memory is passed up the networking layer as an ‘skb’ for more processing.
8. Incoming network data frames are distributed among multiple CPUs if packet steering is enabled or if the NIC has multiple receive queues.
9. Network data frames are handed to the protocol layers from the queues.
10. Protocol layers process data.
11. Data is added to receive buffers attached to sockets by protocol layers.

This entire flow will be examined in detail in the following sections.

The protocol layers examined below are the IP and UDP protocol layers. Much of the information presented will serve as a reference for other protocol layers, as well.

## Detailed Look

**UPDATE** We’ve released the counterpart to this post: [Monitoring and Tuning the Linux Networking Stack: Sending Data](https://packagecloud.io/blog/monitoring-tuning-linux-networking-stack-sending-data/).

**UPDATE** Take a look at [the Illustrated Guide to Monitoring and Tuning the Linux Networking Stack: Receiving Data](https://packagecloud.io/blog/illustrated-guide-monitoring-tuning-linux-networking-stack-receiving-data/), which adds some diagrams for the information presented below.

This blog post will be examining the Linux kernel version 3.13.0 with links to code on GitHub and code snippets throughout this post.

Understanding exactly how packets are received in the Linux kernel is very involved. We’ll need to closely examine and understand how a network driver works, so that parts of the network stack later are more clear.

This blog post will look at the igb network driver. This driver is used for a relatively common server NIC, the Intel Ethernet Controller I350. So, let’s start by understanding how the igb network driver works.

# References: -

1. <https://blog.packagecloud.io/monitoring-tuning-linux-networking-stack-receiving-data/>
2. <https://blog.packagecloud.io/monitoring-tuning-linux-networking-stack-sending-data/>
3. <https://blog.packagecloud.io/illustrated-guide-monitoring-tuning-linux-networking-stack-receiving-data/>
4. Very good detailed: -

https://blog.packagecloud.io/monitoring-tuning-linux-networking-stack-receiving-data/