SAND2015-9378R

Comparing Ring-buffer–based Packet capture solutions

Introduction

Traditional packet-capture solutions using commodity hardware incur a large amount of overhead as packets are copied multiple times by the operating system. This overhead slows sensor systems to a point where they are unable to keep up with high bandwidth traffic, resulting in dropped packets. Incomplete packet capture files hinder network monitoring and incident response efforts. While costly commercial hardware exists to capture high bandwidth traffic, several software-based approaches exist to improve packet capture performance using commodity hardware.

Packet Capture Options

Netmap

Netmap is a community-driven software framework that started as a research project at the University of Pisa. It uses a kernel module and modified NIC drivers in an attempt to speed up packet capture. The kernel module implements a ring buffer. A ring buffer is pre-allocated memory that, once filled, is over-written from the beginning in a circular ring-like fashion. Netmap has its own API for network applications to pull data from the ring buffer, and also includes a modified version of libpcap for supporting legacy applications. 10G support through Netmap is only available for Intel NICs.

Netmap is free and open-source, with compatibility for Linux and Free-BSD. Documentation is extremely limited, though community support is available through a mailing list.

PF_Ring

PF_RING is made by ntop, a small company based in Italy. Like Netmap, PF_RING is a framework that supports up to 10Gbps packet capture with Intel cards by using a kernel module and modified NIC drivers. It also makes use of a ring buffer and requires applications to use its API or a modified version of libpcap. PF_RING is only supported on Linux machines with Intel NICs.

There are two versions of PF_RING: a “Vanilla” open-source version, and a ZC (“Zero Copy”) version. PF_RING ZC allows the application to completely bypass the host operating system’s network processing, resulting in much better performance. The ZC version requires closed-source binaries and a per-MAC address license fee of €249.95, though it may be used freely in a demo mode for up to five minutes at a time.

Documentation is available, but parts may be outdated or of varying quality. Ntop offers additional support for their products for €90 per hour.

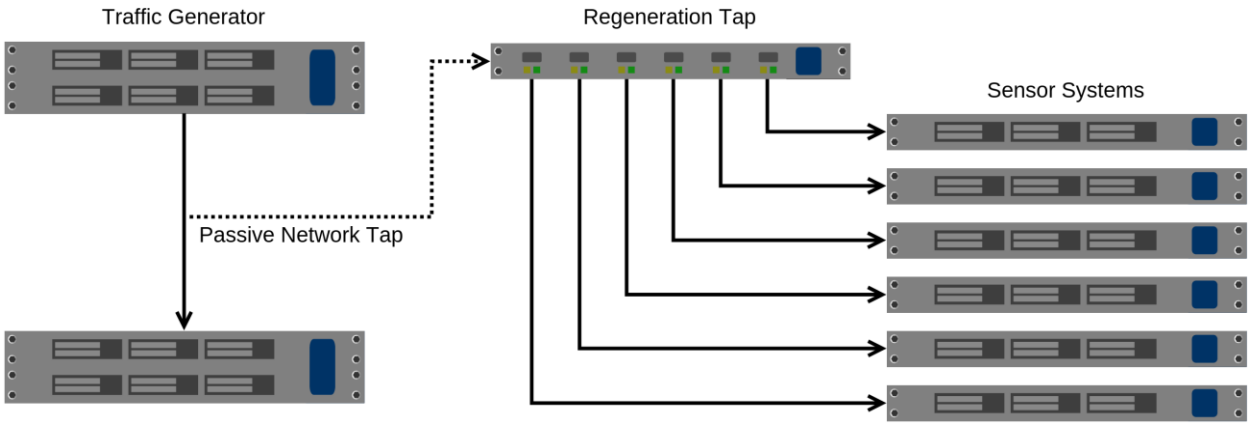
Sniffer10G

Myricom, a US-based network hardware and software company, offers a packet capture solution called Sniffer10G for their own network cards. Like the other options, Sniffer10G provides a ring buffer and reduces the number of times a packet is copied in memory. Sniffer10G is supported on Linux, FreeBSD, and Windows. The software is closed-source and requires a license fee (cost varies slightly by vendor). The license fee is per network card, rather than per MAC address, resulting in potential savings when using a dual-interface network card. A one-week trial license is available upon request. Myricom provides thorough documentation and supports both the hardware and the software.

Comparison	PF_RING		Netmap	Sniffer10G
	Vanilla	ZC		
Compatible NIC	Intel		Intel	Myricom
Open Source	Yes	No	Yes	No
Compatible Operating Systems	Linux		Linux, FreeBSD	Linux, FreeBSD, Windows
License	Free	Per MAC address	Free	Per NIC
Trial	N/A	Five minute demo mode	N/A	One week trial license
Documentation	Somewhat outdated, not well maintained		Very little official documentation	Thorough documentation
Maintenance/Support	Small company		Community	Corporation

Network Setup

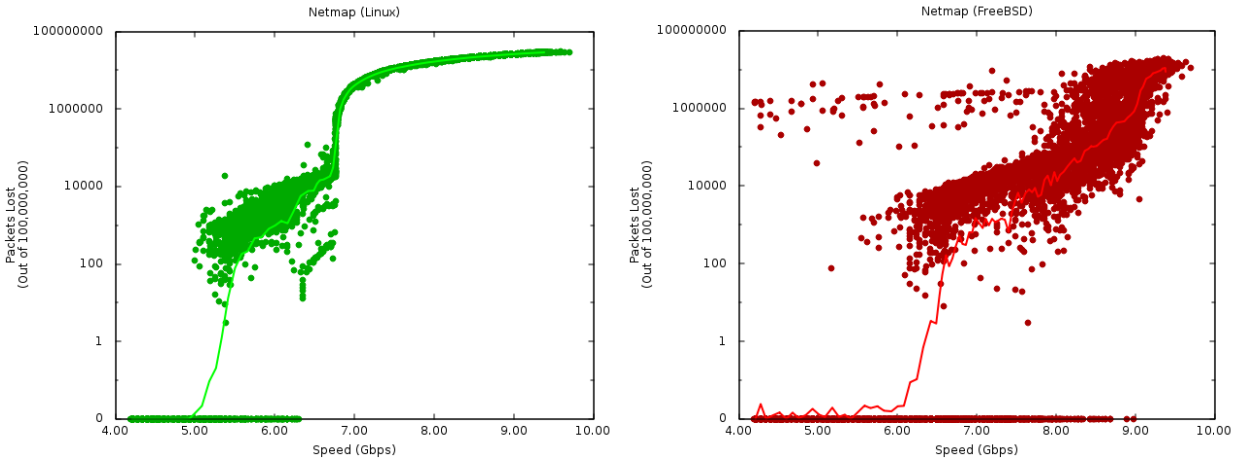
Network SetupNetwork traffic was generated by a Linux machine running zsend, a test utility that is included with PF_RING ZC which can be configured to generate packets of varying size and send them over the network at up to line rate. We used a passive network tap to send the packets to a regeneration tap, which sent the network stream to several network sensors simultaneously. Each packet capture solution includes a utility to count packets received at the application layer. By using a regeneration tap we ensure that all sensors are receiving the same packets at the network interface and can make a valid comparison of the performance of different packet capture solutions.



Results

Netmap (FreeBSD vs. Linux)

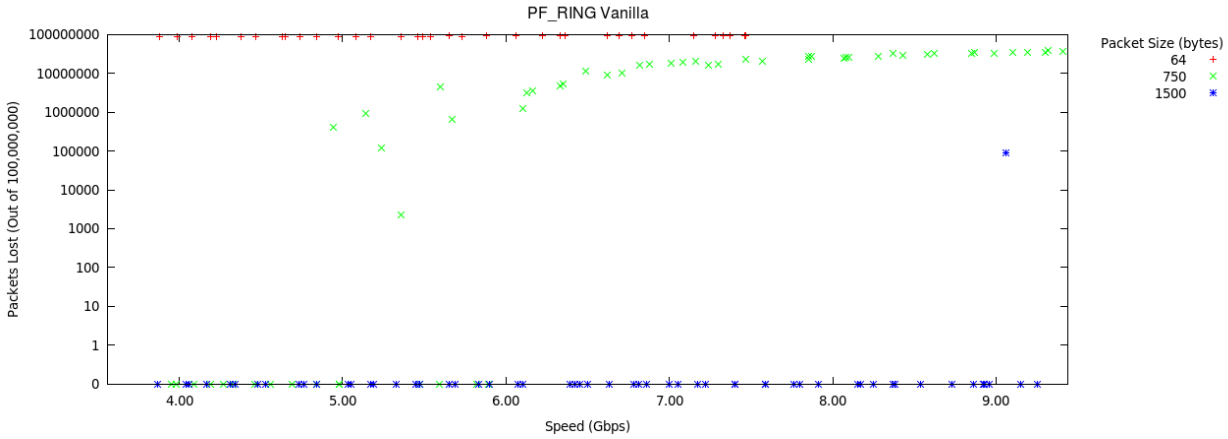
Because Netmap runs on both Linux and FreeBSD, we wanted to compare its performance on each operating system. We deployed Netmap on two machines with identical hardware, with one running Ubuntu 14.04 and the other running FreeBSD 9.3. 64-byte packets were sent across the network in groups of 100,000,000 at various speeds and we kept track of how many packets were lost in each group.



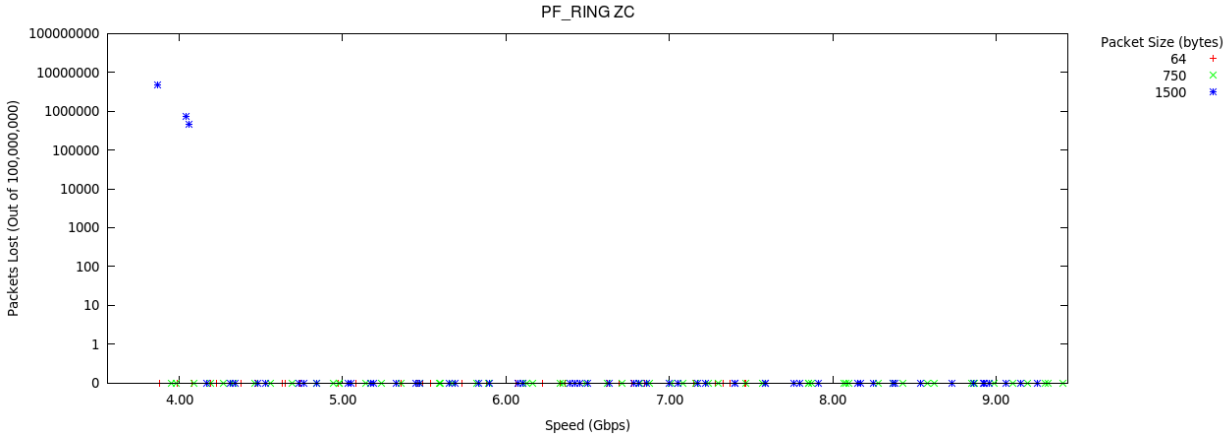
Netmap running on FreeBSD dropped zero packets in more groups; however Netmap on Linux shows far more consistent results.

Small, Medium and Large Packet Summary

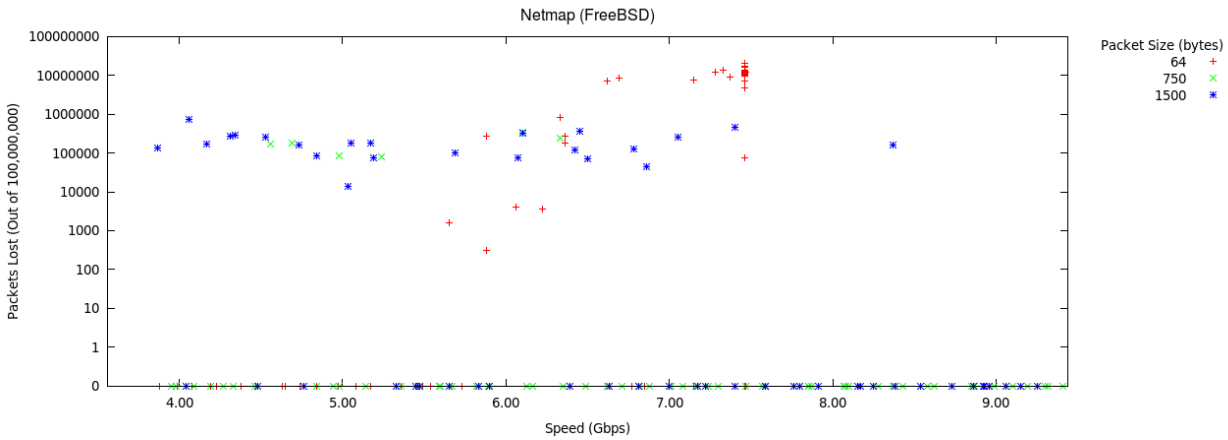
We sent packets of 64, 750, and 1500 bytes across the network at varying speeds in groups of 100,000,000 and captured them on five systems at once. Because processing is done on a per-packet basis, smaller packets cause a higher burden on the sensor system than larger packets received at the same network speed. It is expected that small packets will be dropped more often than larger ones at a given speed. (Note that at the time this test was performed we were unable to consistently send 64-byte packets at 10Gbps.)



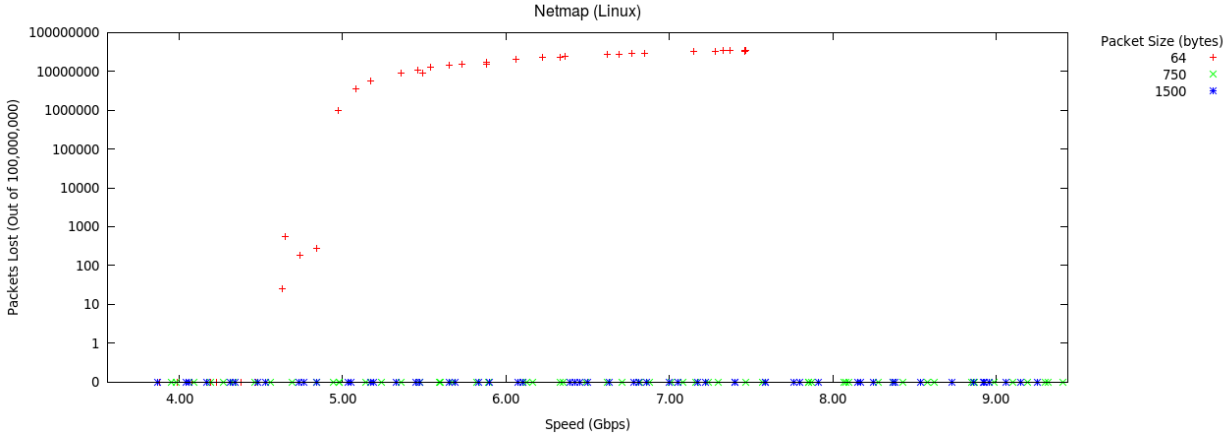
PF_RING Vanilla did poorly with the small- and medium-sized packets, but handled the large packets well.



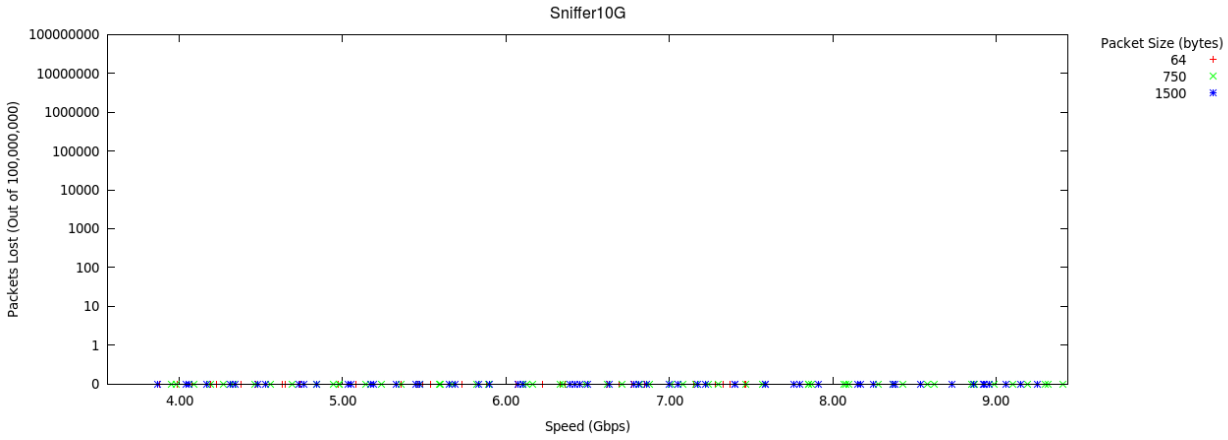
Strangely, PF_RING ZC dropped some larger packets at low speeds.



Netmap on FreeBSD performed inconsistently with dropped packets all over the map.



On Linux, Netmap dropped more 64-byte packets as the speed increased, but had no problem with larger packet sizes at all speeds.

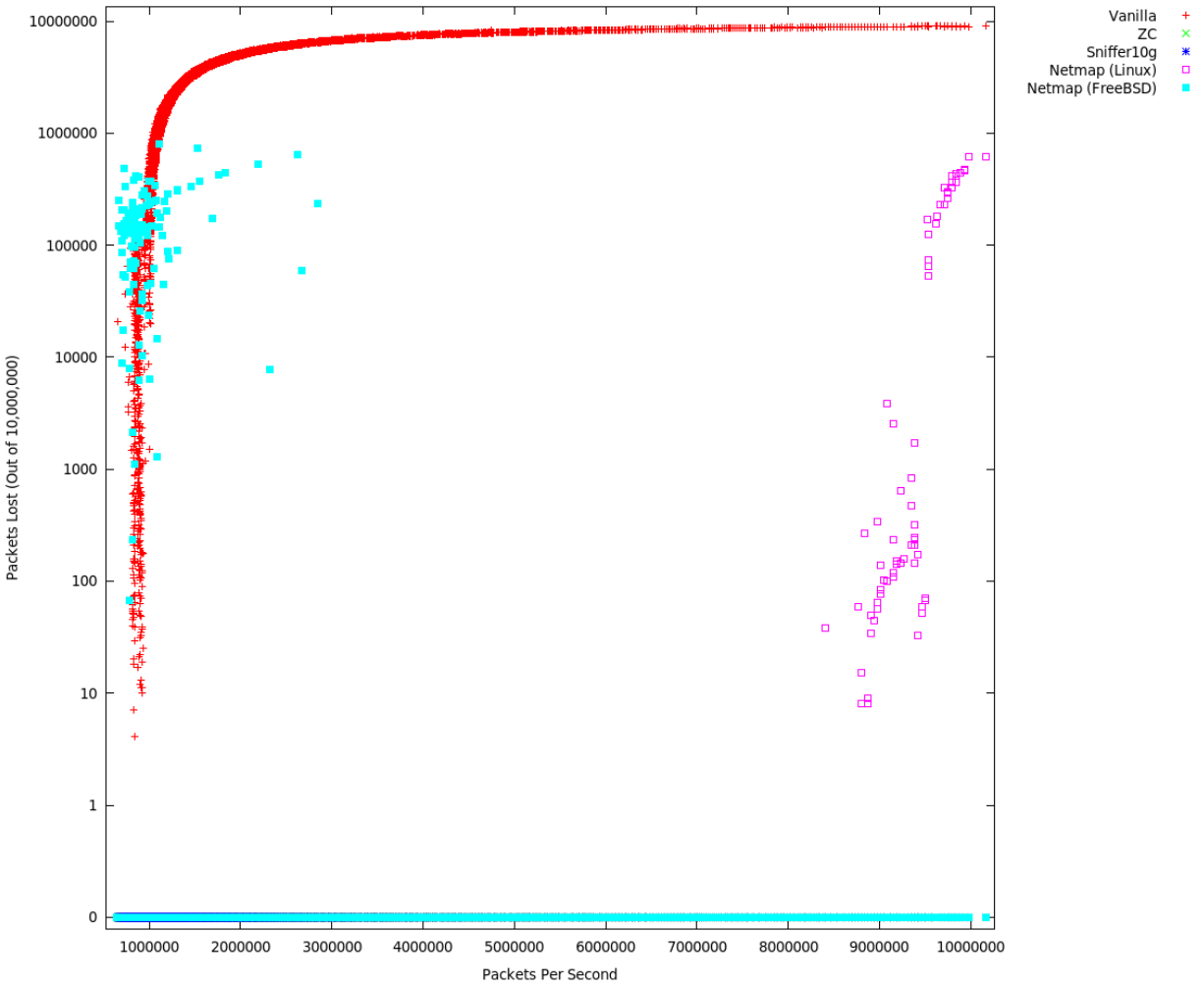


Zero packets were dropped by Sniffer10G; perfect score!

Full Spectrum

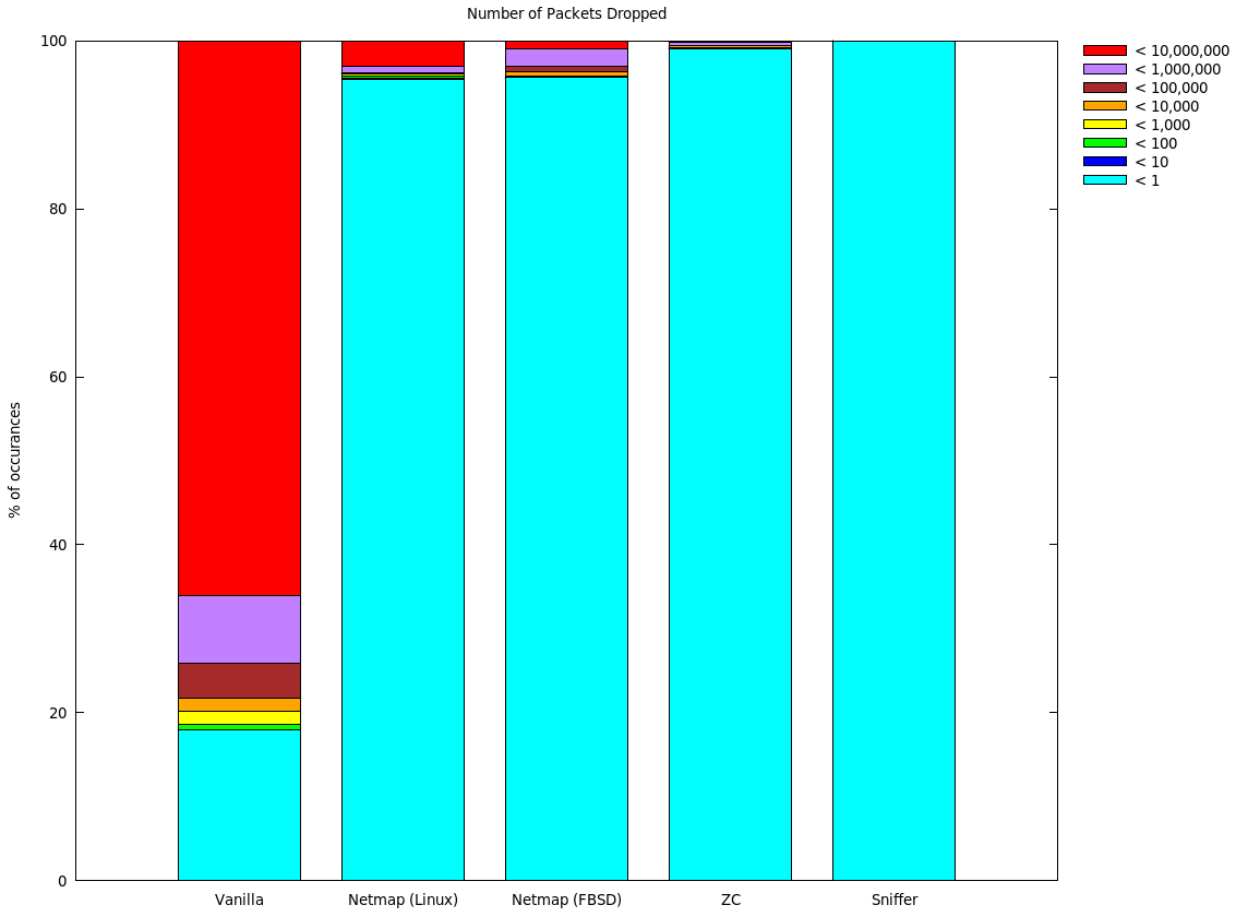
Packets of varying size (64 – 1500 bytes) were sent at varying speeds (4 – 10 Gbps) in groups of 10,000,000*. This graph shows how many packets were dropped from each group by the different packet capture solutions at varying packets per second.

*Out of 6629 groups of packets, there were 42 instances where no sensor received all 10,000,000 packets. In every case three or more sensors received the same number of packets, so it was assumed that the missing packets were not received and forwarded by the regeneration tap to the sensors. The number of packets lost in these cases was adjusted to account for this.



Using the same data as above, the following chart shows how many packets were dropped out of each group of 10,000,000 packets sent to each of the packet capture solutions, broken down into bins.

Packets Lost	Vanilla	ZC	Sniffer	Netmap (Linux)	Netmap (FBSD)
0	1189	6568	6629	6331	6340
1 - 9	3	0	0	3	0
10 - 99	39	1	0	17	1
100 - 999	107	4	0	21	15
1000 - 9999	101	17	0	3	29
10000 - 99999	279	4	0	3	42
100000 - 999999	531	25	0	48	140
1000000 - 9999999	4380	10	0	203	62



Total Loss

Total Lost of 66,299,414,066 Packets		
Vanilla	22997115756	(34.6868%)
ZC	27160855	(0.0410%)
Sniffer	0	(0.0000%)
Netmap (Linux)	579541673	(0.8741%)
Netmap (FBSD)	127007949	(0.1916%)

Conclusions

During our tests, Myricom's Sniffer10G achieved the best performance by far. During all of our tests, Sniffer10G did not drop a single packet. PF_RING ZC also performed quite well, whereas PF_RING Vanilla achieved the poorest rate of packet loss. Netmap running on FreeBSD performed erratically, dropping a wide variety of packets even when a test is repeated with the same parameters. Netmap running on Linux dropped more packets overall than on FreeBSD, but was much more consistent with the packets that it did receive. This makes it easier to estimate how many packets will be dropped in a given situation.

The synthetic tests that we performed do not conform with real-world traffic and in many cases represent a worst-case scenario. Though PF_RING Vanilla and Netmap on Linux dropped a high number of packets under heavy loads, these tools gave very consistent results and may perform well with real-world traffic at lower bandwidths.

Having an idea of the expected traffic load can be helpful in determining which traffic capturing system to deploy. If any amount of packet loss is unacceptable, Sniffer10G stands out as a clear choice in our results. For cases where budget is a constraint and some loss is acceptable, Netmap would possibly be adequate.