Chapter 9

INFINIBAND NETWORK MONITORING: CHALLENGES AND POSSIBILITIES

Kyle Hintze, Scott Graham, Stephen Dunlap and Patrick Sweeney

Abstract The InfiniBand architecture is among the leading interconnects that support high performance computing. The high bandwidth and low latency provided by InfiniBand are increasing its applications outside the high performance computing domain. One of the important application domains is the critical infrastructure.

However, InfiniBand is not immune to security risks. Previous research has shown that common traffic analysis tools cannot effectively monitor InfiniBand traffic transmitted between hosts. This is due to the kernel bypass nature of the InfiniBand architecture and remote direct memory access operations. However, if the Remote Direct Memory Access over Converged Ethernet (RoCE) protocol is employed, it is possible to restore traffic visibility in novel ways. This research demonstrates that the approach, coupled with an InfiniBand-capable adapter, enables common traffic analysis tools to be used to monitor InfiniBand network traffic without sacrificing bandwidth and performance.

Keywords: InfiniBand architecture, network security, network monitoring

1. Introduction

While the capabilities of modern computer processors continue to improve by optimizing current architectures or introducing new architectures, other computing technologies have been unable to keep pace. In particular, the majority of industry-standard input/output bus systems cannot keep up with the raw power of modern computer processors [3].

A promising solution is the InfiniBand architecture interconnect technology [4]. InfiniBand offers higher bandwidth and lower memory latency than Ethernet, and is a powerful technology with promising capabilities. According to the most recent Top 500 ranking [18], which tracks the 500 most powerful supercomputers in the world, seven of the top ten

J. Staggs and S. Shenoi (Eds.): Critical Infrastructure Protection XV, IFIP AICT 636, pp. 187–208, 2022.
https://doi.org/10.1007/978-3-030-93511-5_9

supercomputers utilize the InfiniBand architecture. Network communications are key to the operation of critical infrastructure assets. As InfiniBand is adopted in the critical infrastructure, serious evaluations of its security issues are vital [6, 7, 15, 17, 19].

This research focuses on expanding the monitoring capability of the InfiniBand architecture. In particular, it evaluates the efficacy of common traffic analyzers at capturing and monitoring Remote Direct Memory Access (RDMA) over Converged Ethernet (RoCE) protocol traffic in InfiniBand networks. Several case studies are considered and the results are intended to guide future research focused on securing InfiniBand networks.

2. InfiniBand Architecture

InfiniBand is a network protocol similar to Ethernet that is quickly becoming the standard for high performance computing clusters and data centers. While it is similar to the Ethernet protocol in several ways, InfiniBand was designed to handle higher network bandwidths with significantly reduced memory latency. This came as a direct response to the inability of traditional input/output systems to provide the speeds required to keep up with advancements in modern computing technology. At a high level, by treating input/output as communications, using point-to-point connections and transferring information between hosts and devices via messages instead of memory operations, the InfiniBand architecture is able to achieve the performance desired by the modern computing industry [5]. Figure 1 shows a high-level view of a generic InfiniBand network.

2.1 InfiniBand Hardware

An InfiniBand network has many of the components and connections found in Ethernet networks. Network interface cards (NICs) connect to workstations and processors handle certain workloads related to network traffic. Fundamentally, InfiniBand is an interconnect that enables multiple processors, switches and other devices to communicate with each another. In the InfiniBand architecture, channel adapters, switches and subnet managers play crucial roles that differentiate InfiniBand networks from their Ethernet counterparts.

- **Channel Adapter:** A channel adapter (CA) connects InfiniBand to other devices. A channel adapter can be a host channel adapter (HCA) or a target channel adapter (TCA). Both types of channel adapters generate and consume packets. A host channel adapter supports the functions specified by InfiniBand verbs (described

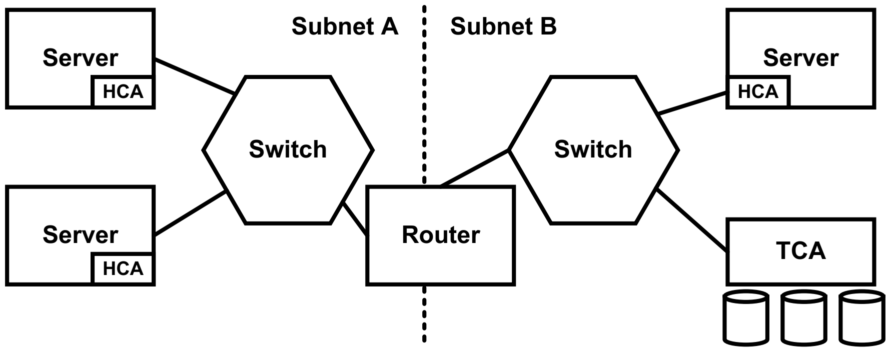

Figure 1. Generic InfiniBand network.

later) whereas a target channel adapter uses an implementation-dependent interface to the transport layer.

What sets a channel adapter apart from a normal interconnect interface (e.g., in Ethernet) is its ability to serve as a programmable direct memory access engine. This enables direct memory access operations to be made locally on hardware and independent of the central processing unit (CPU). Additionally, to identify devices in a network, a channel adapter is assigned a local identifier (LID) by the subnet manager and a globally unique identifier (GUID) by the manufacturer, analogous to the interface identifiers and media access control (MAC) addresses used in Ethernet networks [4].

Channel adapters communicate via work queues that comprise multiple sub-queues [4]. A work queue is initiated by the client (sending network interface card) and the traffic to be sent is placed in a sub-queue. After this is done, the channel adapter processes the received information from the sub-queue and sends it to the requesting device (receiving network interface card). After the information is received, the receiving network interface card returns a status response to the sending network interface card via a completion queue. Multiple queues can exist at a time, enabling a client to conduct other activities while transactions are being processed by channel adapters [11].

- **Switch:** As in the case of Ethernet, an InfiniBand network switch is responsible for forwarding decisions, acting as the fundamental routing component for intra-subnet routing [4]. Data is forwarded from one channel adapter to another based on the data link layer addresses. Forwarding decisions are made based on channel adapter local identifiers (analogous to Ethernet MAC addresses) and based on the forwarding table of the switch, which is configured by the subnet manager at startup. InfiniBand switches also allow for unicast and multicast packet forwarding, enabling the network to support Internet Protocol (IP) applications.

- **Subnet Manager:** A subnet manager is responsible for configuring and managing all switches, routers and channel adapters in a subnet [4, 10]. Multiple subnet managers can exist in an InfiniBand network, one assuming the role of the master and the other serving as a fallback in the event of master subnet manager failure. The master subnet manager communicates with every switch, channel adapter and slave subnet manager to ensure that all routing and forwarding tables are correct.

  The master subnet manager has four responsibilities: (i) discovering the subnet topology, (ii) configuring each channel adapter port with local identifiers, globally unique identifiers, subnet prefixes and partition keys, (iii) configuring each switch with a local identifier, subnet prefix and forwarding database and (iv) maintaining the end node and service databases for the subnet and providing a globally unique identifier to local identifier resolution service.

2.2 InfiniBand Software Architecture

The InfiniBand architecture is compatible with all the major operating systems. The architecture is abstracted away from user space to enable consumers to interact with InfiniBand without any knowledge about the processes executing in kernel space.

The InfiniBand software stack can be divided into the hardware, kernel and application levels [11]. The hardware level comprises the physical network components (i.e., input/output), which transmit electromagnetic signals along copper or fiber waveguides. Connections are made at the hardware level between multiple host devices in a variety of configurations to provide network communications. As with Ethernet communications, InfiniBand network traffic enters and exits through these access points before moving into the host machine.

In kernel space, physical components (host channel adapters) are controlled by input/output drivers to enable user space applications to directly control the hardware. An application is executed in user space through which the device driver maps to an operation [1]. In doing so, a user can leverage an array of InfiniBand capabilities.

In the next level of kernel space, core kernel modules provide the main InfiniBand services. Important services such as the verbs application programming interface and subnet administrator client reside in this level [11]. Indeed, these services distinguish InfiniBand from Ethernet.

Finally, upper layer protocols reside in the top level of kernel space. These protocols enable user applications to leverage the InfiniBand architecture.

2.3 InfiniBand Transport Services

While InfiniBand offers many transport services, this research focuses primarily on its use of the RoCE and IP over InfiniBand (IPoIB) protocols. In an Ethernet network, packets traverse the protocol stack, which involves the host operating system kernel. The kernel processes packets and determines where to send them, consuming many CPU clock cycles in the worst case, leading to lower throughput.

The InfiniBand architecture avoids this limitation by using remote direct memory access, the access of memory from one machine to another without direct CPU involvement. This means that the operating system kernel is bypassed, enabling data transfers to be done by applications directly from user space. User applications can move data directly between virtual memory on different network nodes without operating system intervention [9]. In the case of remote direct memory access, the CPU initiates the communications channel, after which the user application and hardware that perform message passing take control. Throughout the process, verbs are used to convey requests to the hardware.

This study focuses on the RoCE v1 protocol, which replaces the physical and data link layers of the InfiniBand protocol stack with Ethernet [9]. RoCE, which provides the same speed and low latency as remote direct memory access, comes in two versions: (i) RoCE v1 that supports communications between two hosts in the same Ethernet broadcast (link layer protocol) and (ii) RoCE v2 that enables packets to be routed outside a local area network (network layer protocol).

Since RoCE employs Ethernet as its link layer protocol, it supports the use of the IP over InfiniBand protocol [12]. This upper layer protocol implements a network interface using the InfiniBand architecture. It encapsulates IP datagrams over an InfiniBand transport service [15]. After the appropriate kernel modules are loaded, the service can be enabled using standard Linux tools such as `ifconfig` and `ip`. The tools provide standard IP addresses to the chosen interfaces. All applications configured to use the IP over InfiniBand protocol traverse the Transport Control Protocol/Internet Protocol (TCP/IP) stack in the kernel [15].

3. Related Work

Due to the use of InfiniBand in high performance computing environments, early research was mainly directed at increasing bandwidth and reducing latency. Until recently, limited research focused on InfiniBand security.

Dandapanthula et al. [2] investigated the scalable monitoring and analysis of InfiniBand networks. They developed the INAM tool for monitoring an InfiniBand cluster in real time and querying subnet management entities. A web interface was provided for visualizing network performance and the communications patterns of target applications. The INAM tool provided a foundation for developing monitoring capabilities for InfiniBand networks.

Mireles et al. [15] demonstrated that network packets crafted using InfiniBand verbs could not be handled by standard networking monitoring tools. Unlike Ethernet, InfiniBand traffic uses verb semantics to describe operations between a host channel adapter and a consumer (receiver of network traffic) [4, 9]. Traffic crafted with verbs bypasses the operating system kernel to achieve high bandwidth and low latency via remote direct memory access operations. However, this prevents modern traffic analyzers from capturing and analyzing InfiniBand traffic because it bypasses the TCP/IP stack in the kernel. Mireles and colleagues concluded that hardware offloads are key to securing InfiniBand networks.

Lee et al. [7] focused on security enhancements to the InfiniBand architecture. Their research highlighted the promising features provided by InfiniBand for clusters and system area networks, but the lack of security features meant that InfiniBand networks could be exploited. The most serious vulnerability involved network traffic authentication based on the presence of plaintext keys in packets. Lee and colleagues proposed a new authentication mechanism that treated the Invariant Circular Redundancy Check (ICRC) field as an authentication tag, a solution that is compatible with the current InfiniBand specification. Experiments revealed that the new tag enhanced InfiniBand authentication capabilities with marginal performance overhead.

4. Experimental Setup and Case Studies

This research sought to evaluate the ability of common network traffic analyzers to monitor the RoCE protocol in InfiniBand networks. Three experimental case studies were conducted to observe the capabilities of monitoring tools in various configurations and identify an approach for capturing the maximum amount of network traffic. This section de-

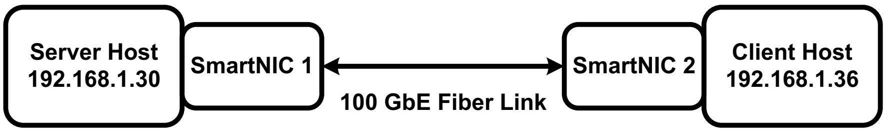

Figure 2. Network configuration of RoCE 100 Gbps with ConnectX-5 adapters.

scribes the experimental setup and network monitoring tools employed, the data collection metrics and the three case studies.

4.1 Experimental Setup

The experimental setup incorporated host workstations, network interface cards and virtual multilayer switches:

- **Host Workstations:** Two host workstations were employed in the experiments. Each workstation was powered by an Intel Xeon Silver 4114 processor (2.2 GHz, 10 cores and 20 logical processors) with 126 GB RAM. The workstations ran Ubuntu 18.04 LTS 64-bit, kernel version 5.0.4-56-generic.

- **Network Interface Cards:** Each workstation was installed with an Nvidia Mellanox Bluefield DPU Programmable SmartNIC [14]. The SmartNICs incorporated 16 ARMv8 A72 cores and 16 GB RAM. Each SmartNIC used a ConnectX-5 adapter as the host channel adapter to provide the physical network interface. Additionally, each SmartNIC ran Ubuntu 18.04 LTS 64-bit.

  Figure 2 shows the network configuration used in the experiments. The two ConnectX-5 adapters were connected in a “back-to-back” manner using a 100 Gbps active optical cable. A switch was not required for this configuration. Interconnect traffic adhered to the RoCE protocol.

  A SmartNIC has two modes of operation. In the default separated host mode, the host workstation and SmartNIC operating systems act as separated entities, communicating with each other or with the network via the ConnectX-5 module of the SmartNIC. This research employed the SmartNIC mode in which a host workstation communicates with the network only through the SmartNIC ARM cores [13].

- **Open vSwitches:** Each SmartNIC has an Open vSwitch application installed as part of its operating system. This multi-layer software switch provides security, monitoring functionality, quality of service and automated control. However, its principal purpose is to provide a switching stack for hardware virtualization environments [8]. In the experiments, Open vSwitch enabled network traffic from the host workstation to be directed through the ARM cores of the SmartNIC and then out to the wire. This provided an opportunity to directly monitor and manipulate traffic in an out-of-band manner, along with other capabilities supported by a software stack.

In addition to acting as a virtual switch, Open vSwitch provided the high bandwidth attributed to the InfiniBand architecture via a hardware offload. In a traditional network stack, all the incoming packets are processed by the operating system kernel. This is very CPU intensive and has low bandwidth because the CPU has to inspect each packet before forwarding it to its destination. By leveraging the Open vSwitch hardware offload, the ConnectX-5 adapter freed up the host workstation CPU and achieved higher bandwidth.

In the configuration, a packet reached the Open vSwitch daemon and kernel module within user and kernel space, respectively. Open vSwitch then made the decision to offload all subsequent packets to the InfiniBand hardware.

4.2 Network Monitoring Tools

InfiniBand network traffic was monitored using Wireshark/`tshark`, `tcpdump` and `ntopng`. These open-source tools are commonly employed for network traffic monitoring.

A network monitoring tool uses a packet capture library (such as `libpcap`) to capture packets from live network devices or files. In general, a packet capture library polls for suitable devices, gains control of the devices and proceeds to filter and capture the incoming network packets. In addition to `libpcap`, the `ntopng` tool also uses the PF_RING library, a newer network socket that dramatically improves packet capture speed. New versions of PF_RING provide packet capture speeds exceeding 10 Gbps (up to 100 Gbps is possible) on multiple network adapters with low packet loss.

PF_RING polls packets from the SmartNIC using the Linux NAPI, which copies the packets from the hardware to a circular buffer. The incoming packets are then distributed to multiple rings simultaneously, drastically improving the packet capture speed and reducing packet loss [16].

The iPerf tool was employed for traffic generation and bandwidth testing. The tool reports multiple metrics related to network health, including bandwidth, latency, jitter and datagram loss. In the simplest setup, a server is created that opens up a socket to receive traffic and a client is configured to send traffic at a specified interval and bandwidth.

4.3 Data Collection Metrics

The following data collection metrics were employed in the experiments to analyze the effectiveness of the monitoring tools:

- **Bandwidth:** The average bandwidth in each experimental trial determines how much TCP/IP traffic was sent from a client to a server, and the (potential) negative impact of a monitoring tool on the network.
- **Packets Received:** This metric measures the number of packets received by a monitoring tool.
- **Packets Dropped:** This metric measures the number of packets dropped by a monitoring tool.
- **Data Consistency:** Data consistency is measured using the mean, standard deviation and coefficient of variation (CV) of the experimental trials.

Additionally, a baseline was set using the Linux utility `ip` on the server host machine. The `ip` utility reports many statistics, but the experiments only used the packets received and packets dropped statistics. While the utility does drop some packets, it captures 99% of all TCP/IP traffic that enters and leaves a client host machine. This enabled the collection of statistics pertaining to the numbers of packets dropped by the monitoring tools due to their inability to handle high traffic flows.

4.4 Case Study 1

The first case study, involving host-based monitoring with hardware offload enabled, was designed to evaluate the efficacy of applying network monitoring tools in InfiniBand networks. The experiments used the Wireshark and `tcpdump` tools.

Under typical conditions, an InfiniBand application using remote direct memory access bypass the operating system kernel and would, therefore, be hidden from network monitors. Nvidia Mellanox adapters support the use of custom vendor tools such as the Offloaded Traffic Sniffer to capture TCP/IP packets on desired interfaces [15]. However, because

Ethernet was used as the link layer protocol in this research (along with the RoCE protocol for direct memory access), network packets could be captured without using custom tools on Nvidia Mellanox and other vendor-specific hardware. In general, capturing network traffic using remote direct memory access or RoCE is the same. Both use direct memory access operations; the main difference is that either Ethernet or InfiniBand is used as the link layer protocol for network communications. The main benefit of RoCE is that a custom tool is not needed to monitor incoming traffic. Any common network monitoring tool can see TCP/IP traffic on the network interface card interface with RoCE, a benefit that stems from using Ethernet as the underlying link layer protocol.

Case Study 1 employed InfiniBand/Ethernet 100 GbE with the SmartNIC ConnectX-5 adapter network configuration. iPerf was used to send TCP/IP packets and report the average bandwidth. A total of 50 trials were performed, five trials for each of five bandwidth levels (1, 3, 5, 10 and 25 Gbps) for each of the two monitoring tools. The trial durations were 120 seconds and the average bandwidth was reported at one second intervals throughout the trials. Hardware offload was enabled in the case study. The goal was to determine whether the monitoring tools could capture TCP/IP traffic at bandwidths of 1 Gbps or higher because large volumes of packets are dropped at these bandwidths.

The experiments in Case Study 1 involved the following steps with each monitoring tool:

- **Step 1:** Configure the server and client host machines to enable IPoIB so that network traffic can be sent on Layer 3.
- **Step 2:** Run iPerf receiver on the server host at the desired bandwidth level.
- **Step 3:** Initiate the network monitoring tool on the server host and specify the interface for packet capture.
- **Step 4:** Run iPerf sender on the client host to send TCP/IP packets to the receiver.
- **Step 5:** Terminate the network monitoring tool after 120 seconds of capture.
- **Step 6:** Record the 120 samples captured during the trial.
- **Step 7:** Repeat Steps 2 through 6 for the remaining bandwidth levels.

After all the trials were completed, the baseline numbers of received and dropped packets were compared against the numbers of received and dropped packets in Case Study 1. The number of dropped packets reported by the `ip` tool was subtracted from the number of dropped packets reported by each monitoring tool.

Next, the variation of the data collected during each bandwidth level trial was computed. A statistical test of the mean, standard deviation and coefficient of variation (CV) of the dropped packets was conducted to evaluate data consistency. In this case study as well as the other two case studies, a coefficient of variation less than 10% was assumed to indicate that the collected data was consistent across all the bandwidth level trials.

4.5 Case Study 2

The second case study, involving SmartNIC monitoring with hardware offload disabled, was designed to evaluate the throughputs of the network monitoring tools for InfiniBand applications at the SmartNIC itself. In the case study, a degree of anonymity was gained because network traffic was captured from the network interface card hardware instead of the host workstation. The monitoring tools were configured as in Case Study 1. However, because the SmartNIC only had a command line interface, the experiments were performed using `tshark`, the command line equivalent of Wireshark.

The `ntopng` tool was selected to evaluate the efficacy of a flow-based traffic analyzer, a capability that arose from the SmartNIC having the Open vSwitch application in its Linux installation. In the experiments, `ntopng` collected NetFlow records that were configured on and transmitted by Open vSwitch running on the Smart NIC. NetFlow is a network protocol for collecting IP traffic information and monitoring network flows. The Open vSwitch application on the SmartNIC was configured to send NetFlow records to a collector (`ntopng`) running on the server host machine. The `ntopng` tool analyzed the flow and updated network statistics for observation and analysis on a web browser on the client host machine.

Case Study 2 employed InfiniBand/Ethernet 100 GbE with the SmartNIC ConnectX-5 adapter network configuration. iPerf was used to to send TCP/IP packets and report the average bandwidth. A total of 45 trials were performed, five trials at each of three bandwidth levels (1, 3 and 5 Gbps) for each of the three monitoring tools; the maximum bandwidth was 5 Gbps instead of 25 Gbps because TCP/IP traffic could not be captured by the SmartNIC with hardware offload enabled. The trial durations were 120 seconds and the average bandwidth was reported at one second intervals throughout the trial.

All the packets captured on the SmartNIC were related to RoCE operations. This is because only the first packet reached the Open vSwitch daemon, all subsequent packets were instructed to be offloaded to the ConnectX-5 network interface card. The destinations of all the packets (server host workstation) were observed. TCP/IP traffic was observed in a parallel packet capture on the server host. This is why hardware offloading had to be disabled to see traffic on the SmartNIC in the Case Study 2.

The experiments in Case Study 2 involved the following steps with each of the two monitoring tools:

- **Step 1:** Configure the server and client host machines to enable IPoIB so that network traffic can be sent on Layer 3.
- **Step 2:** Run iPerf receiver on the server host at the desired bandwidth level.
- **Step 3:** Initiate the network monitoring tool on the SmartNIC and specify the interface for packet capture.
- **Step 4:** Run iPerf sender on the client host to send TCP/IP packets to the receiver.
- **Step 5:** Terminate the network monitoring tool after 120 seconds of capture.
- **Step 6:** Record the 120 samples captured during the trial.
- **Step 7:** Repeat Steps 2 through 6 for the remaining bandwidth levels.

As in Case Study 1, the numbers of dropped packets reported by the `ip` utility were subtracted from the numbers of dropped packets reported by the network monitoring tools. This was done to correct for the packets that were not dropped by the monitoring tools themselves. Statistical tests of the mean, standard deviation and coefficient of variation (CV) of the dropped packets were conducted to evaluate data consistency. A coefficient of variation less than 10% was assumed to indicate that the collected data was consistent across all the bandwidth level trials.

4.6 Case Study 3

The third case study, involving SmartNIC monitoring with hardware offload enabled, was designed to evaluate the throughputs of the network monitoring tools at the SmartNIC with the maximum bandwidth. The monitoring tools and their configurations were similar to those in Case Study 2, except that Open vSwitch was configured to allow for hardware offload. Hardware offload enables network traffic to be handled by the host channel adapter instead of having to traverse the host operating system kernel, reducing CPU intensive operations and improving bandwidth.

With hardware offload enabled, the full bandwidth of the ConnectX-5 adapter and InfiniBand applications can be achieved, and the efficacy of the network monitoring tools for InfiniBand applications can be evaluated. Case Study 3 employed InfiniBand/Ethernet 100 GbE with the SmartNIC ConnectX-5 adapter network configuration. Once again, iPerf was used to send TCP/IP traffic and report the average bandwidth. However, since Case Study 3 only used the `ntopng` tool, a total of 25 trials were performed, five trials at each of the five bandwidth levels (1, 3, 5, 10 and 25 Gbps). Only `ntopng` was used because the two previous case studies revealed that Wireshark and `tcpdump` were incapable of handling high bandwidths.

The experiments in Case Study 3 involved the following steps with the `ntopng` monitoring tool:

- **Step 1:** Configure the server and client host machines to enable IPoIB that allows network traffic to be sent on Layer 3.
- **Step 2:** Run iPerf receiver on the server host at the desired bandwidth level.
- **Step 3:** Initiate the `ntopng` network monitoring tool on the SmartNIC and specify the interface for packet capture.
- **Step 4:** Run iPerf sender on the client host to send TCP/IP packets to the receiver.
- **Step 5:** Terminate the `ntopng` network monitoring tool after 120 seconds of capture.
- **Step 6:** Record the 120 samples captured during the trial.
- **Step 7:** Repeat Steps 2 through 6 for the remaining bandwidth levels.

5. Results

This section presents the results of the three case studies and their implications with regard to traffic monitoring in InfiniBand networks.

Table 1. Baseline packet losses reported by the `ip` tool.

Case Study	**Bandwidth**	**Packets Received**	**Packets Dropped**	**Percent Dropped**
Case Study 1	1 Gbps	10,414,492	0	0.0%
	3 Gbps	31,242,709	153	0.1%
	5 Gbps	52,070,648	193	0.0%
	10 Gbps	104,137,417	5,212	0.5%
	25 Gbps	225,989,529	10,987	0.5%
Case Study 2	1 Gbps	10,414,521	0	0.0%
	3 Gbps	31,242,631	148	0.1%
	5 Gbps	52,070,535	199	0.0%
Case Study 3	1 Gbps	10,414,537	0	0.0%
	3 Gbps	31,242,595	177	0.1%
	5 Gbps	52,070,499	201	0.0%
	10 Gbps	104,136,650	5,170	0.5%
	25 Gbps	260,115,117	11,086	0.4%

Before starting the experiments, a baseline of packets dropped by the operating system kernel was determined using the Linux tool `ip` running on the server host machine. Regardless of the network monitoring tool used, some packets will be dropped by the operating systems of the server hosts or by the hardware interfaces that receive packets. While many factors contribute to packets being dropped, these commonly occur due to hardware issues (e.g., faulty cables or hardware incapable of routing effectively), software problems or insufficient bandwidth, among others. Therefore, at the beginning of each case study, a test was conducted using iPerf at each bandwidth level used in the case study. To normalize the comparisons of monitoring tools, the number of packets dropped in the bandwidth test was subtracted from the total number of dropped packets reported by each monitoring tool.

Table 2 shows the baseline packet losses reported by the `ip` tool. The results show that for every bandwidth level, less than 1% of the packets during the test were dropped, which is well within the acceptable standards. Applying this data to the network monitoring tool results in the case studies makes it possible to make better claims about the effects of using the monitoring tools, especially if they impose negative effects on the network. While monitoring tools are expected to drop some packets for any number of reasons, it would be highly undesirable if the tools were to degrade the network.

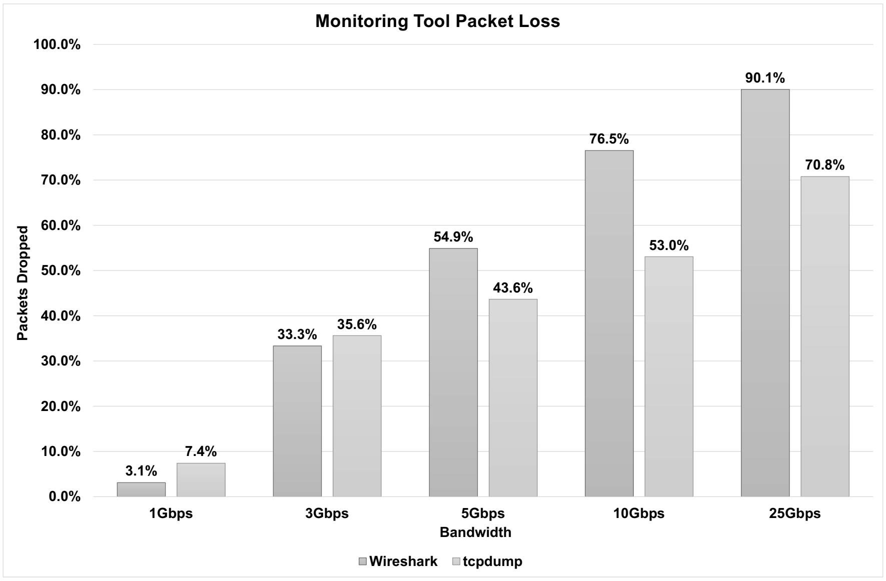

Figure 3. Dropped packets on the host workstation at various bandwidths.

5.1 Case Study 1 Results

The first case study was designed to explore the capabilities of common network monitoring tools in an InfiniBand network. The Wireshark and `tcpdump` tools were used to capture packets at the server host machine (`192.168.1.30`). In all the experiments, the two tools were able to capture TCP/IP traffic at the host workstations.

However, further analysis revealed that increasing the bandwidth reduced the ability to capture all the traffic. What started out as trivial at 1 Gbps became overwhelming at the maximum bandwidth. Figure 3 shows that both monitoring tools begin to lose their effectiveness rapidly beyond a bandwidth of 3 Gbps. Notably, Wireshark begins to drop off dramatically at 5 Gbps and is incapable of keeping up with traffic at higher bandwidths. The `tcpdump` performs better than Wireshark above 3 Gbps, but it still drops a large percentage of packets. A monitoring tool drops packets when its packet capture library is slow, when packets are copied between user and kernel space and/or when the buffer space allocated by the operating system kernel fills up quickly.

Next, the statistics of the dropped packets at the various bandwidth levels were used to provide estimates of collection accuracy. A total of 50 trials were performed, with five trials at each of the five bandwidth levels

Table 2. Case Study 1: Packet loss – data variance.

Monitoring Tool	Bandwidth	Mean of Packets Dropped	SD of Packets Dropped	CV of Packets Dropped
Wireshark	1 Gbps	3.1%	493.25	2.4%
	3 Gbps	33.3%	1,079.03	0.2%
	5 Gbps	54.9%	5,450.47	0.2%
	10 Gbps	76.5%	2,488.52	0.1%
	25 Gbps	90.0%	46,663.91	0.0%
`tcpdump`	1 Gbps	7.4%	372.61	0.8%
	3 Gbps	35.5%	5,240.27	0.7%
	5 Gbps	43.6%	9,643.42	0.7%
	10 Gbps	53.0%	169,169.98	4.7%
	25 Gbps	70.8%	229,920.93	2.9%

for each of the two monitoring tools. The mean and standard deviation (SD) of the dropped packets at each bandwidth level were computed. The coefficient of variation (CV) was then computed by dividing the standard deviation by the mean.

Table 2 shows that the coefficients of variation were all under 5%. The coefficient of variation is a measure of the standard deviation relative to the mean, which provides a dimensionless measure of the spread of the collected data. The results imply that the data collected in Case Study 1 is generally consistent across all the trials and, barring possible outliers, additional runs would likely produce similar results.

Two statements can made based on the overall packet losses of the monitoring tools and the variations in the numbers of dropped packets. First, network traffic created using Ethernet and RoCE operations can be captured using common network monitoring tools; also, both monitoring tools are capable of receiving the network traffic. Second, at bandwidths above 1 Gbps, the monitoring tools are ineffective at capturing all the network traffic. Wireshark and `tcpdump` begin to drop well over 30% of the incoming packets starting at 3 Gbps and the packet dropping only becomes worse with increasing bandwidth. In summary, while traffic capture is possible, common network monitoring tools are ineffective at high bandwidths.

5.2 Case Study 2 Results

The second case study was designed to determine if the monitoring tools used to capture InfiniBand traffic on the host workstation could

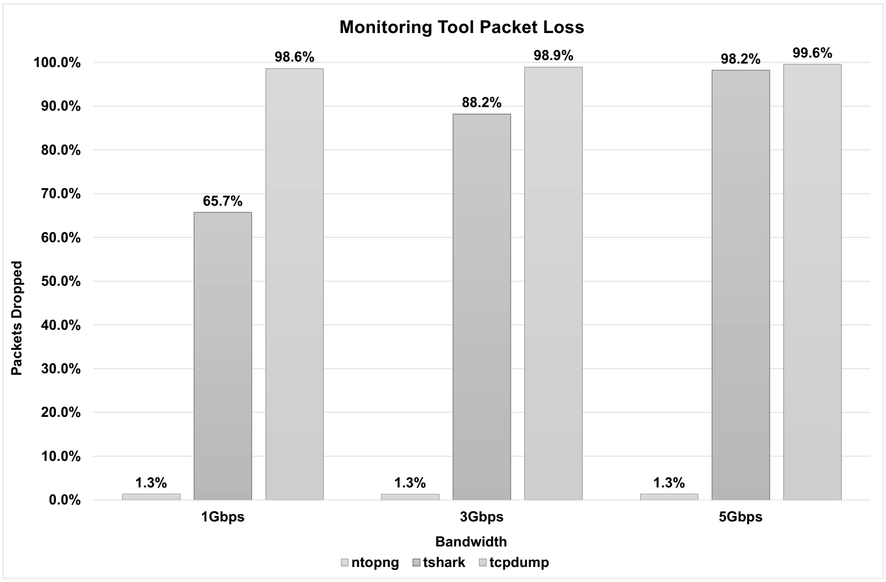

Figure 4. Dropped packets on the SmartNIC at various bandwidths.

capture traffic on the SmartNIC. TCP/IP traffic could not be captured on the SmartNIC when hardware offload was enabled. Therefore, the hardware offload feature was disabled in the experiments.

Figure 4 shows that `tshark` and `tcpdump` were unable to capture all the traffic. Specifically, `tshark` dropped a minimum of 65.7% of packets and `tcpdump` essentially dropped all the packets. Interestingly, `ntopng` incurred minimal packet loss. The `ntopng` tool also enables live captures of packets at specified time intervals.

In order to confirm that the packet loss counts were accurate, the numbers of packets reported by the monitoring tools were compared with those reported by the Linux tool `ethtool`, which provides numerous statistics about traffic received at an interface. The comparisons confirmed that the packet loss counts of the monitoring tools were accurate. The minimal packet loss incurred by `ntopng` was confirmed by examining the PCAP files.

As in Case Study 1, the dropped packet statistics at various bandwidth levels were compared. A total of 45 trials were performed, five trials at each of the three bandwidth levels for each of the three monitoring tools. Note that trials were not performed at the 10 and 25 Gbps levels due to bandwidth limitations imposed by the hardware offload

Table 3. Case Study 2: Packet loss – data variance.

Monitoring Tool	**Bandwidth**	**Mean of Packets Dropped**	**SD of Packets Dropped**	**CV of Packets Dropped**
`tshark`	1 Gbps	65.7%	2,043.18	0.4%
	3 Gbps	88.2%	19,554.85	0.9%
	5 Gbps	98.2%	90,484.81	2.8%
`tcpdump`	1 Gbps	98.6%	10,687.54	1.5%
	3 Gbps	98.9%	31,744.95	1.3%
	5 Gbps	98.2%	16,916.16	0.5%
`ntopng`	1 Gbps	1.3%	492.94	5.1%
	3 Gbps	1.3%	1,231.38	4.1%
	5 Gbps	1.4%	1,050.55	2.1%

feature. Table 3 shows that the coefficients of variation were all under 6%. Thus, it is safe to conclude that the collected data was consistent across all the trials and, barring possible outliers, additional trials would likely produce similar results.

Two statements can made based on the overall packet losses of the monitoring tools and the variations in the numbers of dropped packets. First, the hardware offloading feature of the SmartNIC is not available when using the network monitoring tools; therefore, the ability to see individual TCP/IP packet data is lost. Second, network traffic created using Ethernet and RoCE operations can be captured using the network monitoring tools on the SmartNIC with hardware offload is disabled, albeit only at lower bandwidths.

Figure 4 and Table 3 show that `tcpdump` and `ntopng` are capable of receiving network traffic. However, `tshark` and `tcpdump` are not effective at capturing traffic. In fact, Figure 4 shows that `tcpdump` and `tshark` drop more than 99% of the network packets at the maximum bandwidth. Thus, another method is required to monitor network traffic.

Fortunately, the `ntopng` results are promising. At each bandwidth level, only a small percentage of packets were dropped – a little over 1%. Note that dropped implies that the packets were not captured and processed by the network interface. The consistency and efficacy of the `ntopng` tool demonstrate that its flow-based traffic monitoring capability is a promising alternative for InfiniBand networks.

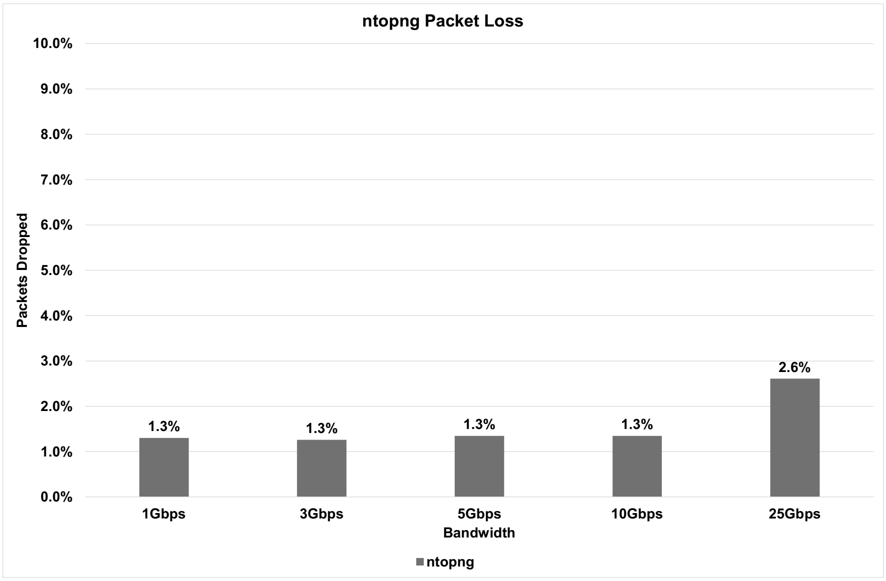

Figure 5. Dropped packets at various bandwidths at the host server using `ntopng`.

5.3 Case Study 3 Results

The third case study was designed to determine if the promising results obtained using `ntopng` to monitor InfiniBand traffic on the SmartNIC in Case Study 2 would persist at higher bandwidths. In the previous case studies, all the other monitoring tools dropped significant percentages of packets as bandwidth increased. This is likely due to the packet capture library being slow, packets being copied between user and kernel space and/or buffer space in the operating system kernel overflowing quickly.

Case Study 2 demonstrated that `ntopng`, which is a flow-based monitoring tool, captured InfiniBand traffic very effectively up to 5 Gbps. In Case Study 3, `ntopng` executed on the server host workstation while the flow data was configured on and sent from the SmartNIC via Open vSwitch. Thus, it seemed possible that hardware offload could be leveraged, even at high bandwidths, to monitor traffic at the SmartNIC without losing the ability to see packet data.

The experiments revealed that, not only were TCP/IP packets visible and captured at all bandwidths by `ntopng`, but very few packets were dropped. Figure 5 shows `ntopng` only dropped between 1.3% to 2.6% of network packets at bandwidths from 1 Gbps up to 25 Gbps. To

Table 4. Case Study 3: Packet loss – data variance.

Monitoring Tool	Bandwidth	Mean of Packets Dropped	SD of Packets Dropped	CV of Packets Dropped
`ntopng`	1 Gbps	1.3%	466.00	5.0%
	3 Gbps	1.3%	1,040.50	3.7%
	5 Gbps	1.4%	347.24	0.7%
	10 Gbps	1.4%	317.27	0.3%
	25 Gbps	2.6%	1,765.02	0.4%

confirm that the packet loss counts were accurate, the numbers of packets reported by `ntopng` were compared with those reported by `ethtool` running on the host server. The comparisons confirmed that `ntopng` was seeing all the generated traffic and that its reported metrics were accurate.

The dropped packet statistics at various bandwidth levels were also compared. A total of 25 trials were performed, five at each of the five bandwidth levels. Table 2 shows that the coefficients of variation were all under 5%. Thus, it is safe to conclude that the collected data was consistent across all the trials and, barring possible outliers, additional trials would likely produce similar results.

In summary, network traffic created by RoCE operations even at high bandwidths is effectively captured by `ntopng` on the SmartNIC. Specifically, hardware offload could be enabled on the SmartNIC, allowing for high network speeds; also, packet data was observable in the captured files. Second, the results in Figure 5 and Table 4 demonstrate that `ntopng` and its packet capture library implementation in conjunction with the Nvidia Mellanox hardware drastically reduced packet loss. The packet loss measured for `ntopng` at the maximum bandwidth was reduced by more than 95% compared with Wireshark and `tcpdump` in Case Study 1.

6. Conclusions

This research has attempted to evaluate the capabilities of modern traffic monitoring tools (Wireshark/`tshark`, `tcpdump` and `ntopng`) for monitoring InfiniBand networks. The experimental results demonstrate that the monitoring tools can capture InfiniBand traffic, but are limited to lower bandwidths due to inefficiencies in their packet capture libraries and/or limited buffer space in the operating system kernels. The experimental results also reveal that positioning the monitoring tools on an

InfiniBand network interface incurred significant packet loss except at low bandwidths. In contrast, the `ntopng` flow-based monitoring tool, due to its efficient packet capture library, had very low packet loss at much higher bandwidths. The `ntopng` incurred minor packet loss even at the maximum bandwidth of 25 Gbps, rendering it a viable option for monitoring InfiniBand networks.

The case study results indicate that options exist for monitoring InfiniBand networks. However, additional research is needed to determine the optimal configurations to achieve low-cost, efficient and reliable InfiniBand network monitoring solutions.

The views expressed in this chapter are those of the authors, and do not reflect the official policy or position of the U.S. Air Force, U.S. Department of Defense or U.S. Government. This document has been approved for public release, distribution unlimited (Case #88ABW-2020-3829).

References

[1] J. Corbet, A. Rubini and G. Kroah-Hartman, *Linux Device Drivers*, O'Reilly Media, Sebastopol, California, 2005.

[2] N. Dandapanthula, H. Subramoni, J. Vienne, K. Kandalla, S. Sur, D. Panda and R. Brightwell, INAM – A scalable InfiniBand network analysis and monitoring tool, in *Euro-Par 2011: Parallel Processing Workshops, Part II*, M. Alexander, P. D'Ambra, A. Belloum, G. Bosilca, M. Cannataro, M. Danelutto, B. Di Martino, M. Gerndt, E. Jeannot, R. Namyst, J. Roman, S. Scott, J. Traff, G. Vallee and J. Weidendorfer (Eds.), Springer, Berlin Heidelberg, Germany, pp. 166–177, 2011.

[3] J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, San Francisco, California, 2011.

[4] InfiniBand Trade Association, InfiniBand Architecture Specification, Volume 1, Release 1.4, Beaverton, Oregon, 2020.

[5] H. Jin, T. Cortes and R. Buyya (Eds.), *High Performance Mass Storage and Parallel I/O: Technologies and Applications*, Wiley-IEEE Press, New York, 2001.

[6] M. Lee and E. Kim, A comprehensive framework for enhancing security in the InfiniBand architecture, *IEEE Transactions on Parallel and Distributed Systems*, vol. 18(10), pp. 1393–1406, 2007.

[7] M. Lee, E. Kim and M. Yousif, Security enhancement in the InfiniBand architecture, *Proceedings of the Nineteenth IEEE International Parallel and Distributed Processing Symposium*, 2005.

[8] Linux Foundation, What is Open vSwitch? San Francisco, California (`docs.openvswitch.org/en/latest/intro/what-is-ovs`), 2016.

[9] P. MacArthur and R. Russell, A performance study to guide RDMA programming decisions, *Proceedings of the Fourteenth IEEE International Conference on High Performance Computing and Communications and the Ninth IEEE International Conference on Embedded Software and Systems*, pp. 778–785, 2012.

[10] Mellanox Technologies, Introduction to InfiniBand, White Paper, Document No. 2003WP, Santa Clara, California (`www.mellanox.com/pdf/whitepapers/IB_Intro_WP_190.pdf`), 2003.

[11] Mellanox Technologies, InfiniBand Software and Protocols Enable Seamless Off-the-Shelf Applications Deployment, White Paper, Sunnyvale, California (`www.mellanox.com/pdf/whitepapers/WP_2007_IB_Software_and_Protocols.pdf`), 2007.

[12] Mellanox Technologies, RDMA Aware Networks Programming User Manual, Rev. 1.7, Sunnyvale, California (`www.mellanox.com/related-docs/prod_software/RDMA_Aware_Programming_user_manual.pdf`), 2015.

[13] Mellanox Technologies, BlueField SmartNIC Modes, Sunnyvale, California (`community.mellanox.com/s/article/BlueField-SmartNIC-Modes`), 2019.

[14] Mellanox Technologies, Nvidia Mellanox BlueField SmartNIC for InfiniBand and Ethernet, Sunnyvale, California (`www.mellanox.com/files/doc-2020/pb-bluefield-vpi-smart-nic.pdf`), 2020.

[15] L. Mireles, S. Graham, S. Dunlap, P. Sweeney and M. Dallmeyer, Securing an InfiniBand network and its effect on performance, in *Critical Infrastructure Protection XIV*, J. Staggs and S. Shenoi (Eds.), Springer, Cham, Switzerland, pp. 157–179, 2020.

[16] ntop, Vanilla PF_RING, Pisa, Italy (`www.ntop.org/guides/pf_ring/vanilla.html`), 2018.

[17] D. Schmitt, S. Graham, P. Sweeney and R. Mills, Vulnerability assessment of InfiniBand networking, in *Critical Infrastructure Protection XIII*, J. Staggs and S. Shenoi (Eds.), Springer, Cham, Switzerland, pp. 179–205, 2019.

[18] E. Strohmaier, J. Dongarra, H. Simon and M. Meuer, Top 500 The List, Prometeus, Sinsheim, Germany, 2020.

[19] K. Subedi, D. Dasgupta and B. Chen, Security analysis of InfiniBand protocol implementations, *Proceedings of the IEEE Symposium Series on Computational Intelligence*, 2016.