3-2021

InfiniBand Network Monitoring: Challenges and Possibilities

Kyle D. Hintze

INFINIBAND NETWORK MONITORING:
CHALLENGES AND POSSIBILITIES

THESIS

Kyle D. Hintze, Captain, USAF

AFIT-ENG-MS-21-M-048

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this document are those of the author and do not reflect the official policy or position of the United States Air Force, the United States Department of Defense or the United States Government.

AFIT-ENG-MS-21-M-048

INFINIBAND NETWORK MONITORING: CHALLENGES AND POSSIBILITIES

THESIS

Presented to the Faculty
Department of Electrical and Computer Engineering
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Computer Engineering

Kyle D. Hintze, B.S.
Captain, USAF

March 2021

AFIT-ENG-MS-21-M-048

INFINIBAND NETWORK MONITORING: CHALLENGES AND POSSIBILITIES

THESIS

Kyle D. Hintze, B.S.
Captain, USAF

Committee Membership:

Scott R. Graham, Ph.D.
Chair

Lt Col Patrick J. Sweeney, Ph.D.
Member

Stephen J. Dunlap, M.S
Member

AFIT-ENG-MS-21-M-048

Abstract

The InfiniBand Architecture is among the leading interconnects supporting High Performance Computing. Capable of providing high bandwidth and low latency, InfiniBand is increasingly found in applications outside the High Performance Computing domain. One of these is critical infrastructure, encompassing almost all essential sectors as the workforce becomes more connected. InfiniBand is not immune to security risks, as prior research has shown that common traffic analyzing tools cannot effectively monitor InfiniBand traffic transmitted between hosts, due to the kernel bypass nature of the IBA in conjunction with Remote Direct Memory Access operations. If Remote Direct Memory Access over Converged Ethernet is used instead, it is possible to restore traffic visibility in novel ways. This research shows that this approach, together with an InfiniBand capable adapter, allows common traffic analysis tools to be used to monitor network traffic without unnecessarily sacrificing the bandwidth and performance of InfiniBand.

AFIT-ENG-MS-21-M-048

This work is dedicated to my wife for all the times you listened to me vent, complain and panic, without your encouragement this would not have been possible.

Acknowledgements

I would like to thank my research advisor Dr. Graham for his constant support and advice during my time at AFIT. You helped me get myself to the end of this journey and made sure that I always stayed grounded and focused on the goals that truly mattered during this program.

I would also like to acknowledge the help of Mr. Steve Dunlap. Your expertise and knowledge of the research process undoubtedly turned my project into a better one and for that I am forever grateful.

Finally, and most importantly I want to thank my wife for the constant support you provided during my masters program. So many nights spent studying, writing, and some spent doubting myself, but you never did. I could not have finished this without you.

Kyle D. Hintze

Table of Contents

Page

List of Figures

Figure Page

List of Tables

List of Acronyms

AOC	Active Optical Cable
API	Application Programming Interface
BTH	Base Transport Header
CA	Channel Adapter
CEA	Cybersecurity Enhancement Act
CLI	Command Line Interface
CPU	Central Processing Unit
CQ	Completion Queue
CRC	Cyclic Redundancy Check
CV	Coefficient of Variation
DMA	Direct Memory Access
DPU	Data Processing Unit
FLOPS	Floating Point Operations Per Second
GID	Global ID
GPU	Graphics Processing Unit
GRH	Global Route Header
GUI	Graphical User Interface
GUID	Globally Unique Identification
HCA	Host Channel Adapter
HPC	High Performance Computing
I/O	Input/Output
IBA	InfiniBand Architecture
IBTA	InfiniBand Trade Association
ICRC	Invariant CRC

IDS	Intrustion Detection System
IP	Internet Protocol
IPoIB	IP over InfiniBand
IPsec	IP Security
IPv6	Internet Protocol version 6
LAN	Local Area Network
LID	Local Identification
LRH	Local Router Header
MAC	Media Access Control
MTU	Maximum Transmission Unit
NAPI	New Application Programming Interface
NIC	Network Interface Card
NIST	National Institute of Standards and Technology
OFED	OpenFabrics Enterprise Distribution
OS	Operating System
OSI	Open Systems Interconnection
OVS	Open Virtual Switch
PCIE	Peripheral Component Interconnect Express
PSN	Packet Sequence Number
QOS	Quality of Service
QP	Quene Pair
RDMA	Remote Direct Memory Access
RoCE	Remote Direct Memory Access over Converged Ethernet
RQ	Receive Queue
SAC	Subnet Administrator Client
SAN	Storage Area Network

SD	Standard Deviation
SM	Subnet Manager
SMA	Subnet Managment Agent
SoC	System on Chip
SQ	Send Queue
TCA	Target Channel Adapter
TCP/IP	Transmission Control Protocol/Internet Protocol
VCRC	Variant CRC
WQ	Work Queue
WQE	Work Queue Element

INFINIBAND NETWORK MONITORING: CHALLENGES AND POSSIBILITIES

I. Introduction

1.1 Background and Motivation

While the capabilities of modern processors continue to improve via optimizations of current architectures or the introductions of new ones, other computing technologies have not been able to keep pace. In particular the majority of industry-standard Input/Output (I/O) bus systems can not keep up with the raw power of today's modern processors [1]. A possible solution to this deficit came from the InfiniBand Trade Association (IBTA), and the resulting InfiniBand Architecture (IBA) interconnect technology. Boasting higher bandwidth and lower memory latency compared to the Ethernet protocol, the IBA stands as a powerful technology with promising capabilities. According to the most recent ranking by Top 500 (a list that tracks the 500 most powerful supercomputers in the world), seven out of the top 10 supercomputers on Earth are utilizing the IBA [2].

In an information-driven society, network communications are key, providing one of the main avenues of operations for critical infrastructure. As InfiniBand becomes adopted it becomes necessary to evaluate security issues in the IBA, as seen in the following research efforts [3, 4, 5, 6, 7]. This research effort focused on expanding the monitoring capability of the IBA. In particular, this research evaluated the capability of common traffic analyzers on the Remote Direct Memory Access over Converged Ethernet (RoCE) protocol created by the IBTA and how effectively they can capture InfiniBand network traffic. The analysis of the case studies presented is intended to guide future research into securing an InfiniBand-based network.

1.2 Problem Statement

Increasing computing speeds are likely to lead to applications of InfiniBand beyond the High Performance Computing (HPC) domain. As seen in other widely used technologies, it is somewhat inevitable that the IBA will be subjected to attacks by malicious cyber actors. A key component to mitigating these attacks is the capability to passively monitor the network traffic to detect such attacks. While some research has been conducted on monitoring high speed networks, it is limited, particularly with regards to the IBA [8, 9, 10]. Most is geared towards the HPC domain [3, 4, 5, 6, 11]. This research expands on prior work that investigated whether certain network security systems were effective on InfiniBand programs. Included now are network monitoring tools with the goal of finding a complete network security system that provides security and performance. In particular this study evaluates whether Mellanox's BlueField Data Processing Unit (DPU) SmartNIC is capable of providing the support needed for an effective security environment. An analysis of this work may help guide future research in securing an InfiniBand network.

1.3 Research Objectives

The research presented in this thesis evaluates key elements in monitoring an InfiniBand network, possible tools and underlying factors that affect their performance, and suggests possible solutions. Three case studies were performed to determine if modern network monitoring tools can be effectively applied to InfiniBand. Nvidia Mellanox adapters were explored to establish their monitoring capabilities and limitations. The research objectives for this thesis are outlined below.

- Understand the key parameters of the IBA, in particular its communication model and packet generation and how this affects traffic monitoring.

- Research IBA applications in typical environments and how trends in computing suggest its rapid adoption in other industries.

- Establish how the National Institute of Standards and Technology (NIST) framework, in particular the Identify and Detect stages can be applied as best practices regarding InfiniBand monitoring.

- Integrate modern, open-source network monitoring tools into an Infiniband network and evaluate their performance and effectiveness in meeting criteria set for monitoring an InfiniBand network.

- Define a network monitoring solution based on the results and explain how it can be used to monitor InfiniBand network traffic.

In pursuit of these objectives, the questions to be answered by this thesis are:

- Are network monitoring tools, built for lower speed Ethernet networks, effective on higher speed IBA-based networks?

- Do the network monitoring tools impose any negative impacts on an InfiniBand network?

- Are there advantages to monitoring the network from the InfiniBand Network Interface Card (NIC)?

- What are the overall implications and requirements to effectively monitor an InfiniBand network?

1.4 Thesis Organization

The organization of this thesis is outlined as follows. Chapter II introduces the IBA, its main concepts and features, and the components that enable its high speed network capabilities. Technologies relevant to the IBA or used in this thesis are discussed. The NIST framework is discussed and how it applies to monitoring an InfiniBand network that will later be used to define a network monitoring framework. Additionally, related work in IBA security is introduced as a basis for this thesis.

Chapter III is organized into three case studies that enumerate challenges to effectively monitor InfiniBand network traffic. The test-bed, network configuration, and monitoring tools used are introduced and discussed. The motivation behind each case study is listed and explained and the experimental set-up and execution is discussed.

Chapter IV analyzes the results of each case study and the challenges presented by each configuration. The issues arising from each experiment are shown and discussed in detail. Additionally, Chapter IV explores a potential network monitoring solution based on the observations. It defines the desired capabilities the solution must possess and possible hardware to implement them. After an analysis of the proposed solutions capabilities, it observes that combining Ethernet traffic with RoCE operations, in conjunction with an InfiniBand capable NIC allows for effective network monitoring at high speeds using the InfiniBand protocol.

Finally, Chapter V summarizes the work. It suggests routes for future research in InfiniBand security and how they can contribute to the protocol. A discussion of the benefits of this research and any drawbacks and challenges encountered are also presented.

II. Background and Related Work

2.1 Overview

This chapter presents background information about the IBA and discusses relevant technologies associated with this research effort. It begins by reviewing the NIST Cybersecurity Framework, in particular Identify and Detect, and how these definitions will help in laying the groundwork for a network monitoring solution for the IBA. Next, the IBA is presented together with a discussion of the components and concepts of the protocol which are essential to this research. Next, several technologies used during experimentation are discussed. Finally, relevant research in InfiniBand security is highlighted to show other ares of interest that have been explored and how they support the goals of this work.

2.2 Cyber Security Framework

Through the Cybersecurity Enhancement Act (CEA) of 2014, the National Institute of Standards and Technology (NIST) was given the task to identify and develop cybersecurity risk frameworks for voluntary use by critical infrastructure owners and operators in the United States [12]. Instead of a one-size-fits-all approach to managing cybersecurity risk, the NIST framework outlines a flexible approach to protecting critical infrastructure. This approach is useful to large private entities and government organizations working to protect their assets, as well as individual researchers developing ways to better secure a protocol like InfiniBand.

The NIST framework consists of three categories; Profiles, Implementation Tiers, and the Core. The Framework Implementation Tiers provide context on how an organization views cybersecurity risk and the processes in place to manage that risk. Consisting of four tiers, this framework describes an increasing degree of rigor and sophistication in cybersecurity risk management practices [12]. A framework profile enables organizations to establish a road map for reducing cybersecurity risk that is well aligned with the organizations' goals. Split between Current and Target profiles, the former indicates cybersecurity outcomes currently achieved, while the later indicates outcomes still needed to achieve desired cybersecurity risk management goals [12]. Finally, the framework core (the primary focus of this research), has four areas: functions, categories, subcategories, and informative references. Functions organize cybersecurity activities at their highest level. There are five defined Functions, with *Identity* and *Detect* being the primary functions explored in this research [12]:

- **Identify** - Develop an organizational understanding to manage cybersecurity risk to systems, people, assets, data, capabilities.
- **Protect** - Develop and implement appropriate safeguards to ensure delivery of critical services.
- **Detect** - Develop and implement appropriate activities to identify the occurrence of a cybersecurity event.
- **Respond** - Develop and implement appropriate activities to take action regarding a detected cybersecurity incident.
- **Recover** - Develop and implement appropriate activities to maintain plans for resilience and to restore any capabilities or services that were impaired due to a cybersecurity incident.

These functions allow an organization to manage its cybersecurity risk and offer avenues to learn from and improve upon cybersecurity practices currently employed.

2.2.1 Network Monitoring Requirements

Using the two functions, *Identify* and *Detect* from the NIST framework and supplemental documentation, together with a requirement that no proposed solution to monitoring InfiniBand may impose negative effects on the network, a network monitoring framework can be outlined, as listed below [12, 13]:

- **Identify** - This function lays a foundation for identifying possible risks to an organization and performing risk management to help mitigate them [12]. While the true meaning of identify in regards to the NIST framework does not completely fit with this research's aim, it nevertheless suggests a basis for what the monitoring solution should be aware of, namely the threats to the InfiniBand network; what they might look like, who may be trying to compromise it, and what steps can be taken to help mitigate the risk posed by these malicious actors.

  In an ideal world, a monitoring solution would be able to capture 100% of all network traffic and be capable of processing this data to identify important information about a network. While having all network traffic could give one insight to the happenings within a network, only a small portion of network traffic is needed to determine how a particular network is organized. In practice even a small percentage, perhaps 1% of network traffic can reveal important information [13]. Therefore, this research specifies that a monitoring tool capture at least 10% of all network traffic in a specified time frame to meet this requirement.

- **Detect** - This function guides efforts to implement appropriate activities to identify cybersecurity events [12]. Detection systems are focused on identifying possible incidents, logging information about them, attempting to stop them, and reporting them to security administrators. Most of the supporting documents regarding the *detect* function do not label a general threshold, such as a percentage of network traffic captured, to be considered meeting the *detect* requirement. Instead, it is more organization dependent, determined by the users and what they consider detrimental to their organization. For this research, considering the high volume of data being transmitted in a given experiment, and how some attacks could be hidden by large volumes of network data, the threshold for meeting the *detect* function will be set at 80% of all traffic sent in a set time frame.

- **Negative Impacts to Network** - Not only should the solution be able to know what the attacks to the InfiniBand network could be and be able to detect them, but the solution must not inadvertently degrade the network. In this research, packets can be dropped either by the InfiniBand hardware or monitoring tools themselves. What is important to determine is if packets lost are due to issues specific to the hardware or monitoring tools themselves, or if the tools are also causing packets to be lost before reaching the intended destination (i.e. the tools themselves are degrading network performance). InfiniBand is designed to achieve very high bandwidths, therefore a viable monitoring solution must not cause significant bandwidth loss.

2.3 The InfiniBand Architecture

InfiniBand is a network protocol, providing networking services much like Ethernet, that is quickly becoming the standard for many HPC clusters and data centers. As mentioned in Chapter 1, seven out of the top ten supercomputers in the world are now using the IBA as the core interconnect [2]. Even more telling is the growth of InfiniBand further down the performance spectrum, where the IBA has increased its share of the Top 500 Supercomputer interconnects from 28% in 2019 to 31% in 2020 [2], as shown in Table 1.

Table 1. Top Supercomputer Interconnects

Year	Interconnect:	Counter	Share (%)
2020	Ethernet	254	50.8
	InfiniBand	155	31.0
	OmniPath	47	9.4
	Custom	37	7.4
	Proprietary	6	1.2
	Myrinet	1	0.2
2019	Ethernet	259	51.8
	InfiniBand	140	28.0
	OmniPath	50	10.0
	Custom	45	9.0
	Proprietary	5	1.0
	Myrinet	1	0.2

At first glance, the IBAs small increase in interconnect share may not seem significant. However, if the share of performance by interconnect is observed instead, it can be seen that InfiniBand holds 40% of the total max Floating Point Operations Per Second (FLOPS), as seen in Table 2. Over the last four years, InfiniBand has increased its share of the performance by 13%.

Table 2. Top Supercomputer Interconnects Performance

Year	Interconnect:	GFlops	Share (%)
2020	InfiniBand	971927068	40.0
	Ethernet	475356880	19.6
	Proprietary	472942300	19.5
	Custom	321955166	13.3
	Omnipath	184605368	7.6
	Myrinet	1975070	< 0.1

While similar to the Ethernet protocol in several ways, InfiniBand was designed to handle higher network bandwidth at a significantly reduced memory latency. This came as a direct response to the inability of traditional I/O systems to provide the bandwidth required to keep up with the improvements in modern computing technology. By treating I/O as communications, using point-to-point connections and transferring information between hosts and devices through messages instead of memory operations, IBA is able to achieve higher performance [14]. A high-level view of a generic InfiniBand network is shown in Figure 1.

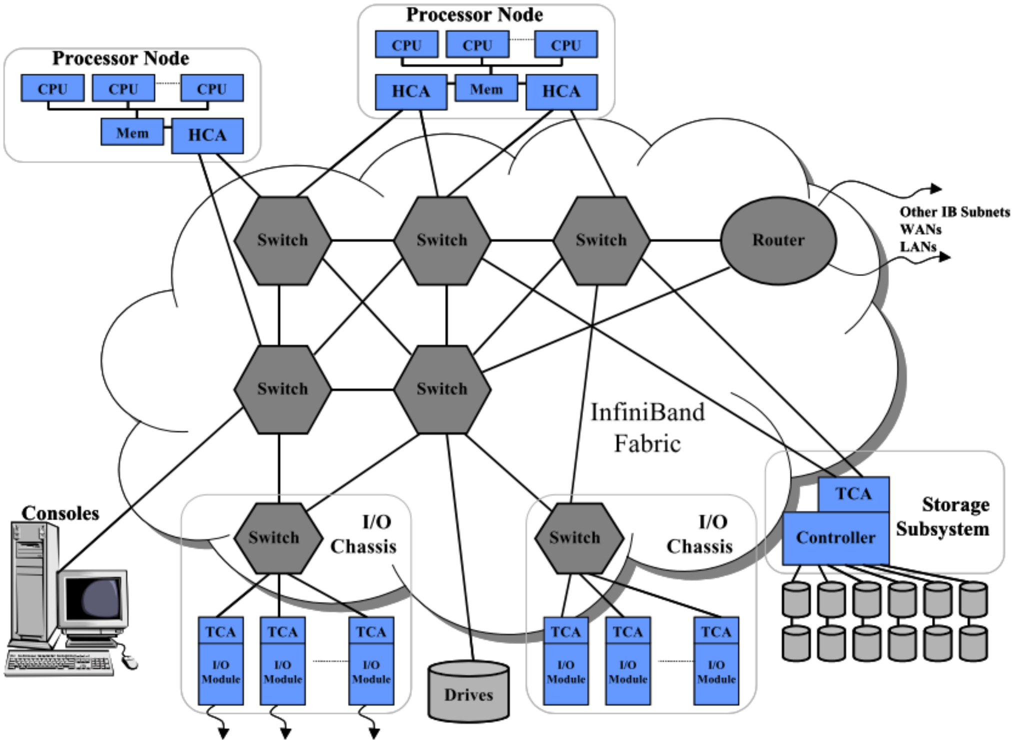


Figure 1. InfiniBand System Fabric

2.3.1 InfiniBand Hardware

An InfiniBand network consists of many of the same components and connections found in Ethernet. A NIC connects to workstations, and processors handle certain network traffic workloads. Fundamentally, it is an interconnect, allowing for multiple processors, switches, and other devices to communicate with one another. In particular, in the IBA, the Channel Adapter (CA), the switch, and the Subnet Manager (SM) play crucial roles, differentiating an InfiniBand network from its Ethernet counterpart.

2.3.1.1 Channel Adapters

The CA connects InfiniBand to other devices and comes in two forms, a Host Channel Adapter (HCA) or a Target Channel Adapter (TCA). Both forms of CAs are able to generate and consume packets. An HCA allows for functions specified by InfiniBand Verbs (described later) while a TCA uses an implementation dependent interface to the transport layer. A visual representation of these differences is shown in Figure 2 [15].

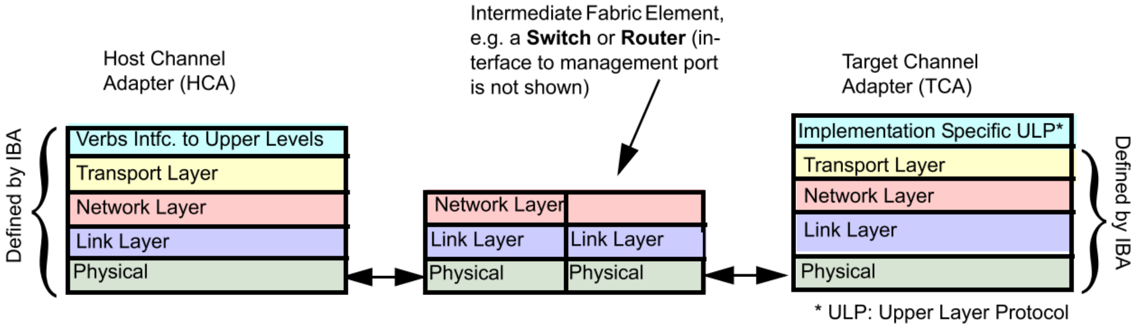

Host Channel Adapter (HCA)
Intermediate Fabric Element, e.g. a Switch or Router (interface to management port is not shown)
Target Channel Adapter (TCA)
Defined by IBA
Verbs Intfc. to Upper Levels
Transport Layer
Network Layer
Link Layer
Physical
Network Layer
Link Layer
Link Layer
Physical
Physical
Implementation Specific ULP*
Transport Layer
Network Layer
Link Layer
Physical
Defined by IBA
* ULP: Upper Layer Protocol

Figure 2. IBA Architecture Layers: HCA vs. TCA

What sets a CA apart from a normal interconnect interface, like that of Ethernet, is its capability of being a programmable Direct Memory Access (DMA) engine. This allows DMA operations to be made locally, on hardware and independent of the Central Processing Unit (CPU). Additionally, to identify devices within the network, each CA is assigned a Local Identification (LID) by the SM and a Globally Unique Identification (GUID) by the manufacturer, directly analogous to Interface IDs and Media Access Control (MAC) addresses used in the Ethernet protocol [15].

All CAs communicate via Work Queues (WQs), consisting of multiple sub-queues [15]. WQs are initiated by the client, (sending NIC) where the traffic to be sent is placed in the queue. After this, the CA processes the received information from within the Send Queue (SQ) and then sends the information to the requesting device (receiving NIC). Once received the receiving NIC returns a status response to the sending NIC through a Completion Queue (CQ). Multiple queues can exist at any given time, allowing a client to continue other activities while transactions are processed by the CAs [16].

2.3.1.2 Switch

Similar to Ethernet, a switch in an InfiniBand network is responsible for forwarding decisions, acting as the fundamental routing component for intra-subnet routing [15]. Data is forwarded from one CA to another based on addresses at the data link layer. Forwarding decisions are made based upon CA LIDs (analogous to MAC addresses in Ethernet) and the switch's forwarding table, which is configured by the SM at startup. IBA switches also allow for forwarding packets via unicast and multicast, enabling support for Internet Protocol (IP) applications.

2.3.1.3 Subnet Manager

A key component in an InfiniBand network, the Subnet Manager (SM) is responsible for configuring and managing all switches, routers and CAs within the subnet [11, 15]. Multiple SMs can exist within the network, with one holding the role of master SM; other SMs will be held in a fallback role in case of SM failure. The SM communicates with every switch, CA and Subnet Managment Agent (SMA) to ensure that all routing and forwarding tables are correct. The master SM is responsible for the following actions:

- Discover the subnet topology.
- Configure each CA port with LIDs, GUIDs, subnet prefixs, and Partition keys.
- Configure each switch with a LID, subnet prefix, and forwarding database.
- Maintain the endnode and service databases for the subnet and provide a GUID to LID resolution service.

2.3.2 InfiniBand Software Architecture

The IBA is compatible with all major Operating Systems (OSs). The IBA is abstracted away from user space to allow for consumers to interact with InfiniBand without knowledge of the processes ongoing in kernel space. At a high level, the InfiniBand Software Stack can be separated into the hardware, kernel and application level as represented in an IP scenario in Figure 3 [16].

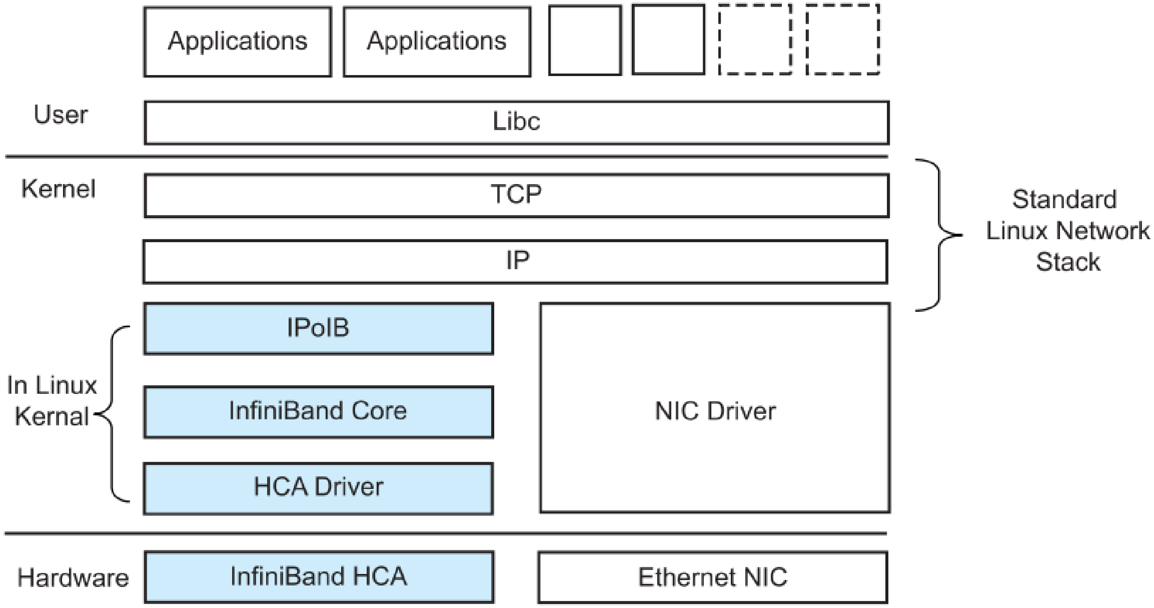


Figure 3. InfiniBand Software Stack

The hardware level is where the physical components exist in the network (i.e. I/O), exchanging electromagnetic waves along copper or fiber waveguides. Here connections are made between a multitude of host devices in varying configurations to provide for network communications. As with Ethernet communications, InfiniBand network traffic enters and exists through these access points before moving farther into the host machine.

Moving into kernel space, the physical components (the HCAs) are controlled by I/O drivers to allow for user space applications to directly control the hardware. An application is executed in user space, through which the device driver maps to an operation [17]. In doing so, a user can take advantage of a wide array of InfiniBand capabilities. At the next level in kernel space, the core kernel modules provide the main services of the IBA. Here important services such as the Verbs Application Programming Interface (API) and Subnet Administrator Client (SAC) reside [16]. It is through these services that InfiniBand distinguishes itself from its Ethernet counterpart. Finally, at the top of kernel space sits the upper layer protocols that allow existing user applications to take advantage of the IBA.

2.3.2.1 InfiniBand Verbs

InfiniBand Verbs are the service through which the user can communicate to the HCA via software. Verbs do not specify the API for the OS, but define the operation for OS vendors to develop a usable API [16]. Verbs describe the parameters necessary for configuring and managing the CA, allocating Quene Pairs (QPs), configuring QP operations, posting work requests to the QP, and getting completion status from the CQ.

When an InfiniBand protocol is used it generates a work request using verbs. A work request may be to load or store memory using Remote Direct Memory Access

(RDMA) operations. Verbs will work with the host OS to describe the IBA operations. Once the work request has been made via the verb, it is subsequently placed in a send or receive queue where it will traverse the network to its destination, after which other layers of the stack will further service the request. A visual representation of this process using verbs is shown in Figure 4 [18].

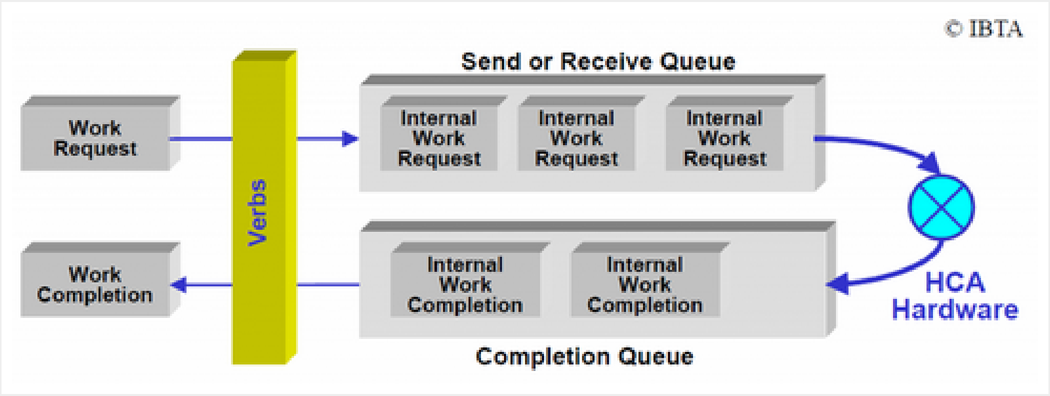


Figure 4. Work Request using Verbs

2.3.2.2 InfiniBand Transport Services

While InfiniBand offers many transport services, this research focuses primarily on its use of RoCE and IP over InfiniBand (IPoIB). In an Ethernet network, packets will traverse the protocol stack, which involves the host OS kernel. The kernel processes packets and determines where to send them, consuming multiple clock cycles of the CPU in the worst case, leading to lowered bandwidth throughput. A visual representation of packet decisions by the kernel is shown in Figure 5 [19].

The IBA avoids this limitation by using RDMA. RDMA is the direct access of memory from one machine to another, without direct CPU involvement. This means that the system's OS kernel is bypassed completely, allowing data transfers to be done

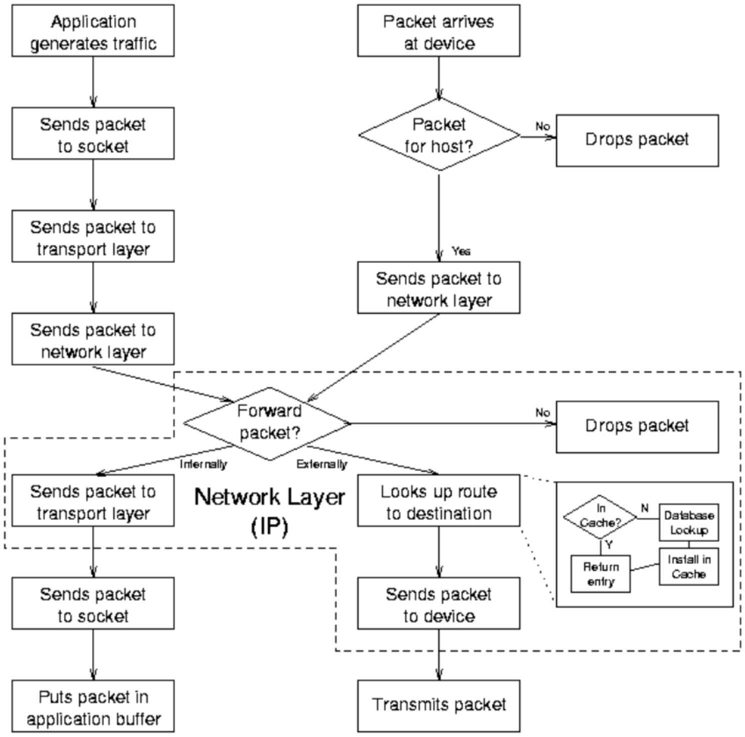


Figure 5. Linux Message Path in the Kernel

by applications directly from user space. User applications can move data directly between virtual memory on different network nodes without the OS intervention [20]. In using RDMA the CPU is used to initiate the communication channel, after which the user application and hardware performing the message passing take control. Throughout the entire process, verbs are used that convey requests to the hardware. This study focuses on RoCE v1, which replaces the physical and data-link layers of the InfiniBand protocol stack with Ethernet [20]. RoCE is capable of the same speeds and low-latency as RDMA and comes in two versions: RoCE v1 allows for communications between two hosts in the same Ethernet broadcast (link-layer protocol) while RoCE v2 allows for packets to be routed outside of a Local Area Network (LAN) (network-layer protocol).

Since RoCE uses Ethernet as its link-layer protocol, this allows for the use of IPoIB [21]. IPoIB is an upper layer protocol that implements a network interface over the IBA. IPoIB encapsulates IP datagrams over an InfiniBand transport service [7]. Once the appropriate kernel modules have been loaded, the service can be enabled using standard Linux tools such as *ifconfig* or *ip*. These will give a standard IP address to the chosen interface. Any application that is configured to use IPoIB will still traverse the Transmission Control Protocol/Internet Protocol (TCP/IP) stack within the kernel [7].

2.3.3 InfiniBand Network Stack

Similar to the Open Systems Interconnection (OSI) network model, the IBA stack is composed of multiple layers: Physical, Link, Network, Transport, and Upper [16]. Each layer operates independently of one another, and also provides a service the next layer above it in the stack. InfiniBand's network layers are shown in Figure 6. Each layer is outlined below:

- **Physical Layer -** The physical layer is what establishes the physical connection between end nodes in an InfiniBand network. It specifies how bits are placed on the wire to form packets and how they are aligned [15]. Specific to InfiniBand, the physical layer defines three link speeds, 1X, 4X, and 12X, ranging from 4 Gbps up to 48 Gbps, fully duplexed [16].

- **Link Layer -** The link layer describes the packet format and protocols for packet operation, flow control and how packets are routed within a subnet [15]. Two types of packets exist here, management and data packets. Management packets are used to configure links and maintenance while Data packets carry information with up to 4K bytes of data per payload [16].

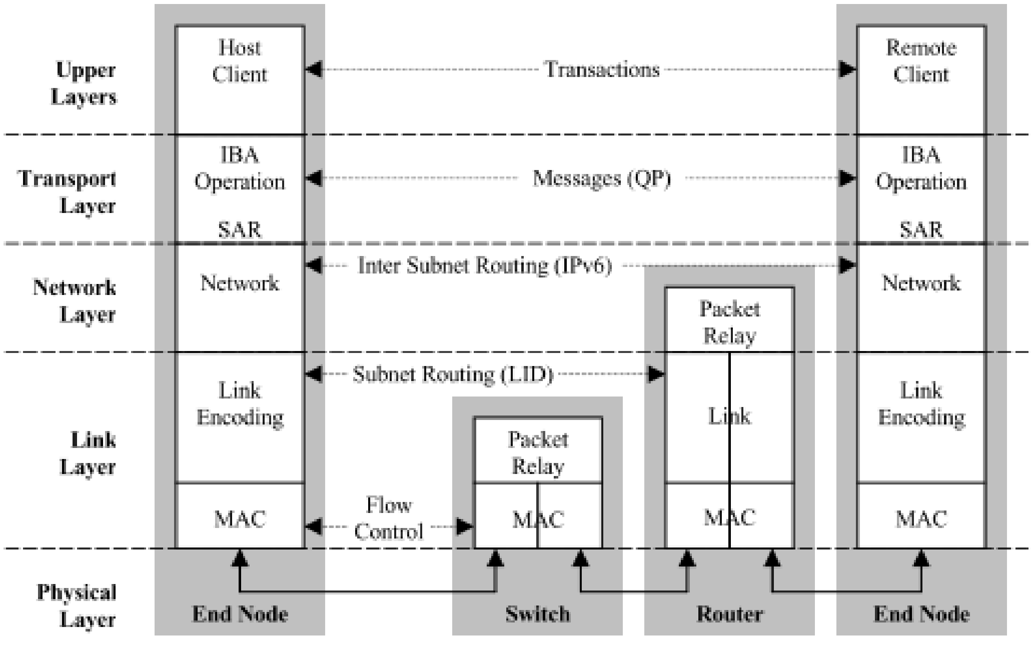


Figure 6. InfiniBand Network Stack Layers

Switching and packet forwarding is handled in the link layer. At system startup, the SM assigns all devices within the network a LID. When packets are sent, the LID is used for addressing, not unlike IP addresses. In combination with a Local Router Header (LRH), these form the addressing portion of packets sent within the subnet.

The IBA link layer provides flow control via a credit based system [15, 16]. In the network, a receiver sends credits to the transmitter on the other end of the link. This value indicates the number of data packets that the receiver can accept. Packets will not be sent unless the receiver signals credits that indicate its receive buffer has space available for new data.

Finally, the IBA link layer provides data integrity via two types of Cyclic Redundancy Check (CRC) per packet: a 16 bit Variant CRC (VCRC) and a 32 bit Invariant CRC (ICRC). The VCRC includes all fields in the packet and is recalculated at each hop, providing link level data integrity between two hops. The ICRC covers only the fields that do not change across hops and provides end-to-end data integrity [16]. An overview of a standard IBA packet is shown in Figure 7 [14].

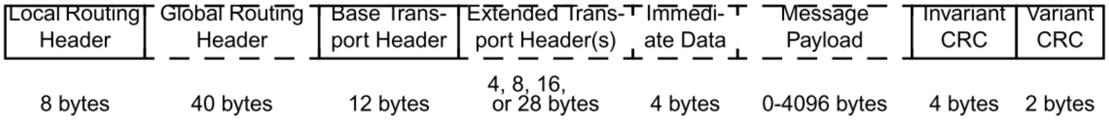


Figure 7. Complete IBA Packet Format

- **Network Layer -** The network layer specifies how to route packets between subnets in an InfiniBand network. Packets get routed by InfiniBand routers, and reach their destinations via the Global Route Header (GRH). The GRH identifies the source and destination ports using a Global ID (GID) in Internet Protocol version 6 (IPv6) address format. As packets travel through subnets, the InfiniBand routers will modify the contents of the GRH and replace the LRH, but the source and destination GIDs do not change and are protected by the ICRC field. Beginning at the source, the InfiniBand router places the GID of the destination in the GRH and the LID in the LRH. Upon arriving at the destination, the final router replaces the LRH using the LID of the destination [15].

- **Transport Layer -** This is where the network and link layer deliver packets to the desired destination, the transport layer ensures the packet arrives at the proper QP and instructs the QP how to process the packet's data. If a packet exceeds the Maximum Transmission Unit (MTU), the transport layer is responsible for segmenting the data into the appropriate sizes. Within a packet exists the Base Transport Header (BTH), that specifies the destination QP and indicates the operation code, Packet Sequence Number (PSN), and partition. The operation code identifies if the packet is the first, last, intermediate, or only packet of a message and specifies if the operation is an RDMA send, write, read or atomic. The PSN is established and incremented each time the QP creates a new packet [15].

2.3.3.1 InfiniBand Network Transaction Flow

When a user wants to send a message through an InfiniBand capable adapter, the user interacts with an IBA CA through QPs, that themselves consist of a SQ and Receive Queue (RQ). From here a work request will be generated that places a Work Queue Element (WQE) in the SQ. Next, the CA detects and accesses the WQE where it interprets the command, validates the WQE's virtual addresses, translates it to physical addresses, and accesses the message data. From here, appropriate headers are added to the packet by the CA, as well as splitting of the message if the MTU is exceeded, before the packet is sent out on the wire. Finally, upon the packet arriving at its destination, any appropriate acknowledgements are sent by the receiving end QPs and if the last packet of the message is received the CA will retire the WQE [15]. A visual representation of a work request being made and overall InfiniBand packet travel can be seen in Figures 4 and 8 [15].

2.4 Relevant Technologies

While the InfiniBand protocol is the primary focus of this research, several technologies are used through this work. In the following sections, the network monitoring and bandwidth tools used, Peripheral Component Interconnect Express (PCIE),

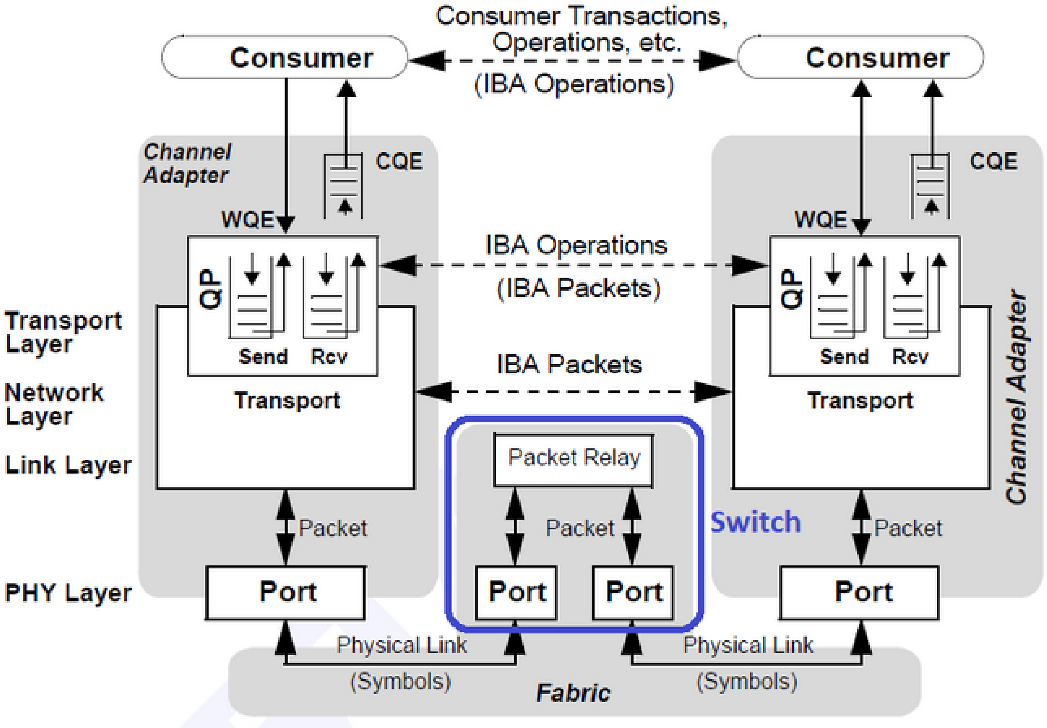


Figure 8. IBA Communication Stack

DPUs and the virtual switching software Open Virtual Switch (OVS) are discussed.

2.4.1 Network Monitoring and Bandwidth Tools

This research makes uses of multiple software tools. For actual monitoring of InfiniBand network traffic, the tools *wireshark/tshark*, *tcpdump*, and *ntopng* are used. All three are open-source and widely used for network traffic monitoring. The network monitoring tools all use a packet capture library, such as *libpcap*, which allows packets to be captured from a live network device or file. In general, these libraries will poll for suitable devices, gain control of them, and then begin to filter and capture incoming network packets. In addition to *libpcap*, *ntopng* also uses the library *PF_Ring*, which is a newer type of network socket, developed by ntop to dramatically improve the speed of packet capture. In newer versions of the library it states that packet capture speeds above 10Gbit and even up to 100Gbit are possible, on multiple network adaptors, with lower packet loss.

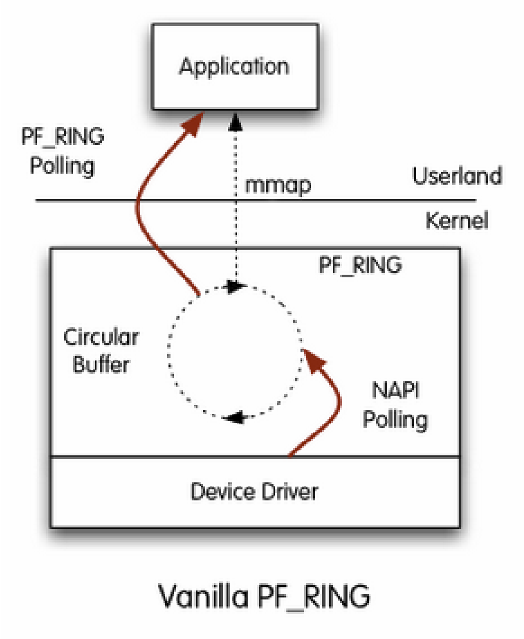


Figure 9. Vanilla PF_RING Packet Capture Library Vverview

As seen in Figure 9 [22], *PF_Ring* polls packets from the NIC by means of the Linux New Application Programming Interface (NAPI), which copies packets from the hardware to a circular buffer. From here incoming packets can be distributed to multiple rings simultaneously drastically improving packet capture speed and lowering packet loss [22].

For traffic generation and bandwidth testing, the tool *iperf* is used extensively in this research. *Iperf* is capable of reporting multiple metrics important to network health, including, but not limited to, bandwidth, latency, jitter, and datagram loss. In the simplest set up, a server is created that opens up a socket to receive traffic and a client is made to send traffic at a specified interval and bandwidth. The following command will configure a client to send TCP/IP traffic at specified bandwidths and report the average bandwidth at one second intervals:

```
$ iperf -c <ip address> -t 120 -i 1
```

The following command will set up a server to accept TCP/IP traffic at specified bandwidths and report the average at one second intervals:

```
$ iperf -s -i 1
```

2.4.2 Peripheral Component Interconnect Express

PCIE is a high-speed serial computer expansion bus standard, capable of high bandwidth. PCIE is a switched lane architecture that operates more like a network than a bus. It has an internal switch that controls several point-to-point serial connections to communicate packets. Each device in the computer that needs data has its own dedicated connection lane via the interconnect, thereby forgoing the need to share bandwidth. Each lane of a PCIE connection contains two pairs of wires: a send and receive, that is capable of sending a specified number of bits per cycle (listed as x1, x2, x4, x8, x16, and x32). In this research, generation 4 PCIE at x16 lane width is used, allowing for a theoretical bandwidth rate of 256 Gbps of data. In practice this will be much lower due to overhead and other system specifications that degrade the bandwidth performance.

2.4.3 Linux Device Drivers

This research makes use of the Linux OS on both host machines and the Nvidia Mellanox Connect-X 5 adapters. Since device drivers are what allow the hardware to be able to communicate with the software, the internals of Linux device drivers are discussed below. Device drivers are distinct black boxes that allow a particular piece of hardware to respond to a well-defined internal programming interface [17]. When a user requests an action, it is carried out by a means of standardized system calls, within kernel space, independent of the specific driver. One feature of Linux is the ability to extend at run-time the set of features offered by the kernel, this includes creating loadable kernel modules. In this research, the majority of device drivers used are implemented as loadable kernel modules, and upon being loaded are represented as device files that the user can directly interact with. All devices used can be looked up and interacted with via the Command Line Interface (CLI) because of the modular nature of the device drivers.

2.4.4 Data Processing Unit

This research makes extensive use of the Nvidia Mellanox BlueField SmartNIC, that hosts the BlueField DPU System on Chip (SoC) on board. A DPU is a newer class of programmable processors that combines three elements to differentiate itself from a Graphics Processing Unit (GPU) and CPU. Namely, a high-performance, programmable, multi-core CPU, a NIC, and a set of programmable acceleration engines capable of offloading applications. While a DPU can be used as a stand-alone embedded processor, it currently finds most use incorporated with programmable SmartNICs. Currently, most DPUs are capable of the following:

- High-speed networking connectivity at speeds of 100 Gbps and higher.
- High-speed packet processing, aided by hardware accelerators.
- Multi-core processors, typically from the ARM family.
- Hardware accelerators used for enhancing cryptography and high-speed storage offloads.
- Allow for the running of an OS separate from the host machine that the BlueField is installed on.

A visual representation of the BlueField DPU used in this research is shown in Figure 10 [23]. As mentioned before, as high network bandwidths become more common place, more processing power is needed in order to handle them. When paired only with a CPU or GPU, that is already handling a multitude of workloads, any extra processing related to network traffic, especially at gigabit speeds, is going to cause some amount of degradation. While not the only reason, this is a major factor in the increasing appearance and use of DPUs and their place in high speed networks is likely to endure.

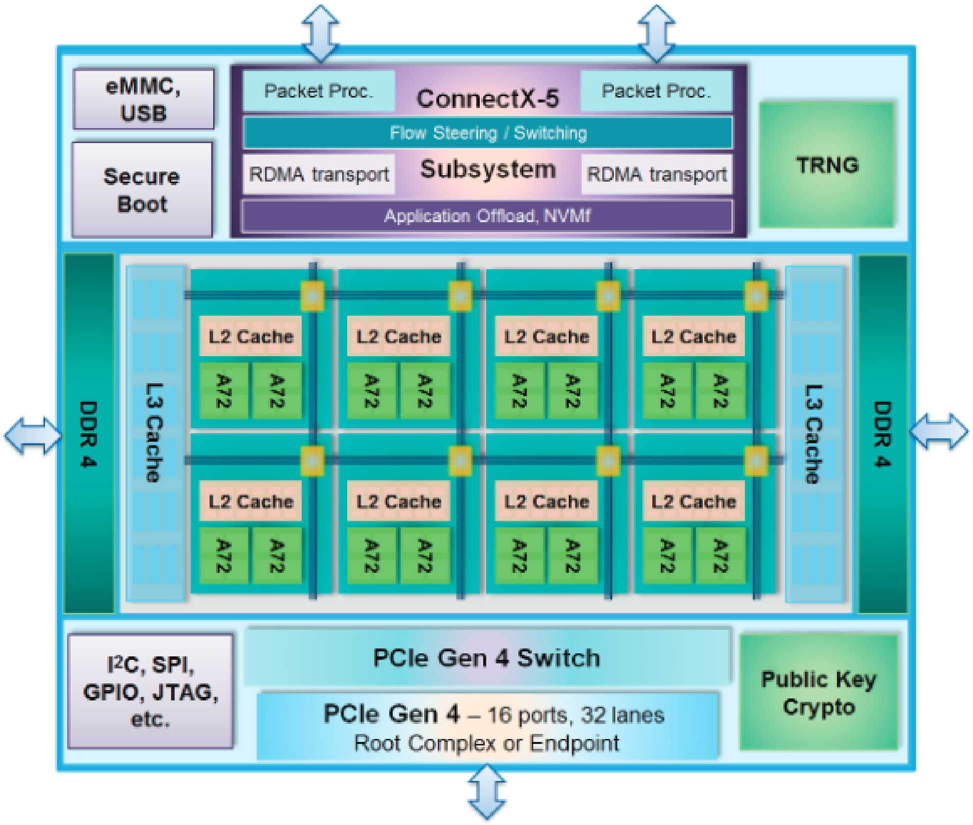


Figure 10. BlueField DPU Architecture

2.4.5 Open Virtual Switch

OVS is a multi-layer software switch, providing security, monitoring, Quality of Service (QOS) and automated control. Its main purpose is to provide a switching stack for hardware virtualization environments [24]. By existing virtually, OVS provides the majority of features a physical switch provides, along with some of its own. OVS consists of eight core elements:

- *ovs-vswitchd* - The daemon that implements the switch.
- *Linux kernel module* - For flow-based switching.
- *ovsdb-server* - A lightweight database server.
- *ovs-dpctl* - A tool for configuring the switch kernel module.
- *ovs-vsctl* - A utility for querying and updating the configuration of ovs-vswitchd.
- *ovs-appctl* - A utility that sends commands to running OVS daemons.
- *ovs-ofctl* - A utility for controlling the OpenFlow features of OVS.
- *ovs-pki* - A utility for creating and managing the public-key infrastructure.

In this experiment, OVS allows network traffic from the host client to be directed through the ARM cores of the Bluefield and then out onto the wire, as shown in Figure 14. This provides an opportunity to directly manipulate and monitor traffic out-of-band, among other capabilities that a software stack allows.

In addition to acting as a virtual switch, OVS allows for the high bandwidth attributed to the IBA via hardware offload. In a traditional network stack, all incoming packets (represented by red lines) are processed via the OS kernel, as shown in Figure 5. Since the CPU must inspect each packet before forwarding to its destination, this results in a much lower bandwidth and is very CPU intensive. As shown in Figure 11, together with OVS hardware offload, Mellanox's Connect-X adapters are able to free up the host workstation CPU and achieve higher bandwidth. In this setup, the first packet reaches the OVS daemon and kernel module within user and kernel space. From here OVS will make the decision to offload all subsequent packets to the Mellanox hardware.

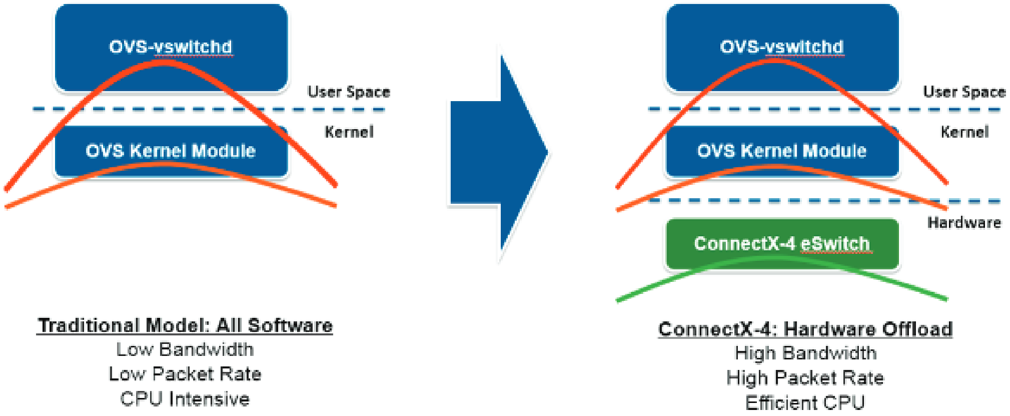


Figure 11. OVS offload Frees up CPU to Provide Performance Benefits

2.5 Related Work in IBA Security

Early research into the security of the IBA revealed several vulnerabilities without known mitigation solutions. While significant effort was spent in improving bandwidth and latency performance, less effort was spent in securing the network, arguably due to its use in HPC settings. Listed below are some of the security research efforts and their relevant findings.

2.5.1 A Framework for Cyber Vulnerability Assessments of InfiniBand Networks:

This research conducted a cyber vulnerability assessment on an IBA network to learn the possible vulnerabilities that may exist in the IBA [3]. The research states that since the IBA was designed as a data center technology, logically separated from the Internet, standard defensive mechanisms such as packet encryption were not implemented. As with many new technologies, the assessment concluded that, while security holes do exist in the IBA, most bad actors have not taken significant interest in InfiniBand. The author goes on to warn that this may change as the IBA is more widely adopted, and this may come to fruition as InfiniBand now comprises over 70% of supercomputers [2]. However, IBA remains resistant to cyber attacks.

2.5.2 Implications and Limitations of Securing An InfiniBand Network:

This research showed that network packets crafted using InfiniBand verbs are unable to be monitored by standard networking monitoring tools [7]. Differing from an Ethernet network, InfiniBand traffic uses the set of semantics called verbs to describe operations that take place between an HCA and consumer (receiver of network traffic) [15, 20]. In combination with other aspects of the IBA, these allow traffic crafted with verbs to bypass the OS kernel to achieve the IBA's high bandwidth and low latency through RDMA operations. An unfortunate aspect of this is that modern traffic analyzers cannot sniff these network packets because they never traverse the TCP/IP stack in the kernel. Ultimately the research concluded that the key to securing an InfiniBand network may reside in hardware offload.

2.5.3 Security Enhancement in InfiniBand Architecture:

This research highlighted the IBAs promising features for clusters and Storage Area Networks (SANs) [5]. It further pointed out the IBA specifications lacked sufficient security features, leaving it vulnerable to possible exploitation. The authors stated the most serious vulnerability in the IBA was the lack of authentication of network traffic since InfiniBand authenticates packets solely by checking the existence of plaintext keys in a packet. A new authentication mechanism was proposed that treated the ICRC field as an authentication tag that is compatible with the current IBA specification. Upon implementation, the authors found the new tag enhanced the IBAs authentication capabilities without hampering its performance and incurred only marginal performance overhead.

2.6 Summary

This chapter began by providing a brief overview of the NIST cybersecurity framework and the core functions used in this research. Following this, the chapter described the basic concepts and protocols of the IBA, including the hardware and software architecture. Additionally, relevant technologies used in this research were discussed, including the network monitoring and bandwidth testing tools, device interconnects and drivers, as well as the processing units and switching software used. Finally, recent literature concerning InfiniBand security was discussed and the conclusions made in each were presented.

III. InfiniBand Case Studies

3.1 Objective

The research objective of this thesis is to characterize the ability of common network traffic analyzers to monitor an InfiniBand network, with emphasis on Ethernet and the RoCE protocol. Past research concerning InfiniBand occurred within the HPC domain and focused primarily on its architecture. Security aspects, such as network monitoring, remain largely unexplored. Thus, to obtain this knowledge, three case studies are performed, exploring mechanisms for monitoring an InfiniBand network. In particular, this work examines the difficulties in using common network traffic analyzers on an InfiniBand network to showcase the limitations of monitoring the InfiniBand protocol. Additionally, the thresholds laid out previously for a network monitoring solution for InfiniBand are examined, in regards to the requirements of *Identify*, *Detect*, and *Negative Impacts*.

3.1.1 Testbed Setup

While the IBA allows for both InfiniBand and Ethernet (using RDMA and RoCE, respectively), Ethernet was chosen as the link layer protocol for this study. Previous research into the limitations of security applications on the InfiniBand protocol already determined that InfiniBand itself cannot be monitored without security penalties [7]. As they are currently the largest producer of InfiniBand hardware in the market, Nvidia Mellanox equipment is used for these experiments. However, the experiments outlined in this work should be applicable to other InfiniBand capable hardware as well. The testbed setup and hardware used in this study is outlined in the next sections:

3.1.1.1 Host Workstation:

The experimental setup consist of two host workstations, powered by an Intel Xeon Silver 4114 processor (2.2 Ghz, 10 Cores, 20 Logical Processors) with 128GB of RAM. Each workstation is running Ubuntu 18.04 LTS 64-bit, kernel version 5.0.4-56-generic.

3.1.1.2 Network Configuration:

The network configuration used in this study is shown in Figure 12. The two host workstations each have a Connect-X 5 adapter installed. Each adapter will be connected "back-to-back" via a 100 Gbps Active Optical Cable (AOC). For this research, no switch is required, and the link layer protocol used is Ethernet, allowing for RoCE operations to be used.

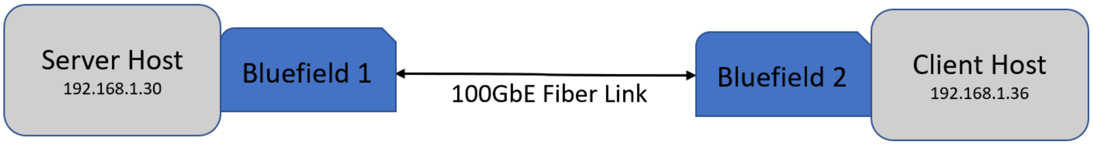


Figure 12. Network Diagram of RoCE 100Gbps with Connect-X 5 Adapter

3.1.1.3 BlueField DPU Programmable SmartNIC:

Within each workstation, an Nvidia Mellanox BlueField SmartNIC is installed. The SmartNIC contains 16 ARMv8 A72 cores and 16GB of RAM. Each card uses a Connect-X 5 adapter as the HCA which acts as the physical interface for the network. Each card is running Ubuntu 18.04 LTS 64-bit [25]. Additionally, the BlueField has two modes of operation as shown in Figure 14. In the default *Separated Host* mode, both the host workstation and SmartNIC ARM OS act as separated entities, communicating with each other or with the network via the Connect-X 5 module of the SmartNIC. In *SmartNIC* mode, the host workstation communicates with the network via the SmartNIC ARM cores. When the host workstation sends data into the network the packets are sent via the PCIE interface, into the NIC where it is received by the virtual interface *pf0hpf* (this is the host-workstation-facing interface of OVS). The packets are processed by the OVS virtual switch and then sent out on the network-facing interface *p0*. Finally, the packets are then sent to the Connect-X 5 HCA which sends the packets out onto the network. For this research *SmartNIC* mode is used [26].

At initial startup, the BlueField is defaulted to *Separated Host* mode and needs to be changed to *SmartNIC* mode. To do this the following commands are issued to the Mellanox OpenFabrics Enterprise Distribution (OFED) drivers.

```
$ mst start
$ mlxconfig -d /dev/mst/mt41685_pciconf0 s INTERNAL_CPU_MODEL=1
```

First, *mst start* loads the appropriate kernel modules and saves PCI configuration headers in the appropriate directories. Next, *mlxconfig* is invoked which allows the user to change device configurations without changing the firmware on board. Once invoked, this command tells the CPU mode (indicated by 1 here) to be changed for the device (the BlueField) designated by *mt41685*.

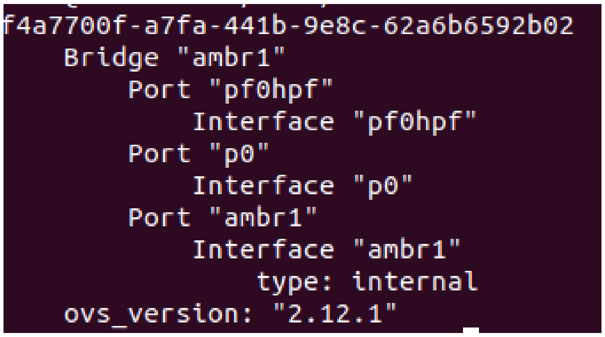
Before being able to send network traffic from host workstation to host workstation, OVS needs to be invoked to bridge the virtual interfaces *p0* and *pf0hpf*. At its default, network traffic can be sent to the host-facing network interface *pf0hpf*, but it is not connected to anything, so no traffic will ever make it out onto the wire. To fix this the following commands are issued to OVS on the BlueField:

```
$ ovs-vsctl add-br ambr1

$ ovs-vsctl add-port ambr1 p0

$ ovs-vsctl add-port ambr1 pf0hpf
```

In these commands, *ovs-vsctl* is invoked to created a bridge called *ambr1*. Once created, the ports *p0* and *pf0hpf* are then added to the bridge as shown below in Figure 13. Once completed, network traffic now flows from host workstation, through the ARM cores on the BlueField and then finally out onto the wire to its destination.



```
f4a7700f-a7fa-441b-9e8c-62a6b6592b02
    Bridge "ambr1"
        Port "pf0hpf"
            Interface "pf0hpf"
        Port "p0"
            Interface "p0"
        Port "ambr1"
            Interface "ambr1"
                type: internal
    ovs_version: "2.12.1"
```

Figure 13. OVS Bridge on BlueField

3.1.2 Data Collection and Metrics

For this research, several metrics were collected and analyzed across all case studies to analyze the efficiency of the monitoring tools, as shown below:

- Bandwidth: Each case study will report the average bandwidth in each *iperf* trial to determine how much TCP/IP traffic is being sent from client to server and show that the monitoring tools are not imposing negative effects on the network.

- Packets Received: Each case study will report the number of packets received by the monitoring tools.

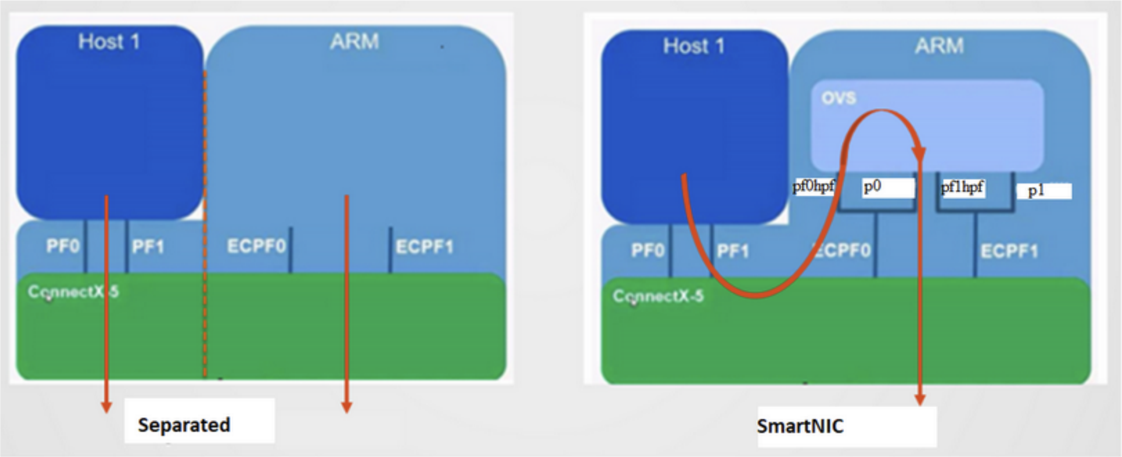


Figure 14. BlueField Modes of Operation

- Packets Dropped: Each case study will report the number of packets dropped by the monitoring tools.
- Data Consistency: Each case study will report the standard deviation, mean, and coefficient of variation for all runs performed. In doing so, an analysis of how consistent each run was can be made.

Additionally, a baseline was set using the Linux utility *ip* on the server host machine. The tool *ip* can report many statistics, but for this study its ability to show received and dropped packets will be used. While the *ip* tool will drop some packets, overall it captures 99% of all TCP/IP traffic coming into, and out of the client host machine. With that in mind, the number of packets dropped as a result of the monitoring tools not being capable of handling the influx of traffic can be collected.

3.1.3 Monitoring Tool Requirements Evaluation

As stated earlier in Section 2.2.1, three requirements were established to meet the threshold for a possible monitoring solution for an InfiniBand network. In each case study several monitoring tools are used in varying configurations to test their effectiveness at monitoring InfiniBand traffic. How each monitoring tool used in this research will be evaluated on meeting all the listed requirements is listed below:

- **Identify** - To meet this requirement, a threshold of 10% of all network traffic captured during the trial duration must be met. While each monitoring tool captures network traffic in slightly different ways, metrics common to all include received packets, dropped packets, and bandwidth. Each case study runs a specific number of trials at varying bandwidths for each monitoring tool. To calculate captured network traffic, the following steps are applied according to each case study:

  1. Perform specified number of trials at each bandwidth level, for each monitoring tool.
  2. Record the number of received and dropped packets reported by each monitoring tool.
  3. Account for the baseline packets dropped at each bandwidth level, for the monitoring tool being analyzed.
  4. Calculate the mean number of received and dropped packets, at each bandwidth level, for each monitoring tool.
  5. Divide the mean dropped packets by the mean received packets for each bandwidth level.
  6. Check if 10% threshold is met.

- **Detect** - To meet this requirement, a threshold of 80% of all network traffic captured during the trial duration must be met. As stated previously, each case study will run a set number of trials at varying bandwidths for each monitoring tool. To calculate captured network traffic, the following steps were applied according to each case study:

    1. Perform specified number of trials at each bandwidth level, for each monitoring tool.
    2. Record the number of received and dropped packets reported by each monitoring tool.
    3. Account for the baseline packets dropped at each bandwidth level, for the monitoring tool being analyzed.
    4. Calculate the mean number of received and dropped packets, at each bandwidth level, for each monitoring tool.
    5. Divide the mean dropped packets by the mean received packets for each bandwidth level.
    6. Check if 80% threshold is met.

- **Negative effects to Network** - To meet this requirement, an average bandwidth level within 5% of the bandwidth used for each trial must be met. In order to calculate average bandwidth, the following steps are applied according to each case study:

    - *Bandwidth Loss* - Repeat these steps for each bandwidth being tested for the required number of runs.

        1. For each bandwidth level, perform trial using *iperf* without monitoring network traffic, for the monitoring tool being used.

2. Record average bandwidth as reported by *iperf*.
3. For each bandwidth level, perform trial using *iperf* with the monitoring tool being used.
4. Record average bandwidth as reported by *iperf*.
5. Compare the average bandwidth from each trial to determine if they are within 5% of each other.

3.2 Case Study 1: Host-based Monitoring, Hardware Offload Enabled

The first case study tests if current network monitoring tools can monitor InfiniBand applications. For this study, *tcpdump* and *wireshark* are used. *tcpdump* uses the CLI while *wireshark* uses a Graphical User Interface (GUI). A visual of the network configuration for this case study is shown in Figure 15.

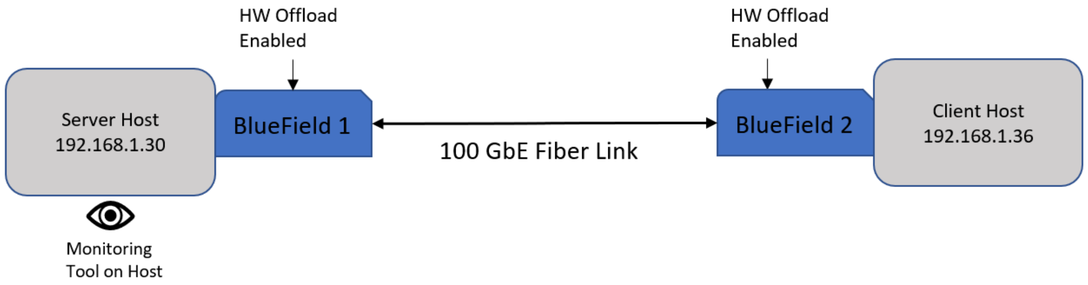


Figure 15. Network Configuration for Case Study 1

Under typical conditions, an InfiniBand application using RDMA would bypass the OS kernel and therefore be hidden from network monitors. Nvidia Mellanox adapters could employ vendor tools such as the "Offloaded Traffic Sniffer" to capture the TCP/IP packets on the desired interface [7]. However, because Ethernet is used as the link layer protocol for this research, thus using the RoCE protocol for DMA operations, it can be shown that network packets can be captured without these custom tools on Nvidia Mellanox and other vendor specific hardware. In general, capturing network traffic that used RDMA or RoCE is the same. Both use DMA operations, the main difference lies in using either InfiniBand or Ethernet as the link layer protocol for network communications. The main benefit of RoCE is that no specific tool is needed to monitor the incoming traffic. Any common network monitoring tool can see TCP/IP traffic on the NIC interface with RoCE, a benefit arising from using Ethernet as the underlying link layer protocol.

This case study uses the InfiniBand/Ethernet 100GbE with BlueField Connect-X 5 Adapter network configuration. A total of 50 trials are performed, five for each of the two monitoring tools, at each of the five bandwidth levels. In each trial, *iperf* will be used to send TCP/IP packets and report the average bandwidth. The trial duration is 120 seconds, reporting the average bandwidth at one second intervals throughout the trial. Hardware offload is enabled for this case study and the bandwidth level is varied from 1, 3, 5, 10 and 25 Gbps. The goal is to determine whether the monitoring tools can capture the TCP/IP traffic at bandwidths at and above 1 Gbps, as bandwidths above this tend to result in a large volume of lost and dropped packets.

3.2.1 Test Steps:

1. Configure host machines to enable IPoIB, allowing network traffic to be sent via Layer 3.

2. Start *iperf* Receiver on the server host at desired bandwidth limit.

3. Initiate network monitoring tool on server host and specify appropriate interface to capture packets.

4. Run *iperf* Sender on the client host to send TCP/IP packets to receiver.

5. After 120 seconds of capture, terminate network monitoring tool.

6. Record 120 bandwidth samples captured in trial run.

7. Repeat steps 2-5 for remaining bandwidth trials.

Upon test completion, the baseline number of received and dropped packets, as stated in Section 3.1.2, is compared to the number of received and dropped packets from the case study. The number of dropped packets reported by *ip* is subtracted from the number of dropped packets reported by the monitoring tool.

Next, each bandwidth levels trial is measured for the variation of the data collected. To do this a statistics test of the Standard Deviation (SD), mean number of dropped packets, and the Coefficient of Variation (CV), is reported. For this and subsequent case studies, if the CV is less than 10% then it will be assumed that the data collected was consistent across all bandwidth level trials.

3.3 Case Study 2: BlueField Monitoring, Hardware Offload Disabled

This case study examines the monitoring capability throughput of current network monitoring tools when tasked with monitoring InfiniBand applications from the BlueField itself. In this scenario a degree of anonymity is gained as network traffic is captured from the NIC hardware as opposed to the host workstation. The tools for Case Study 2 are set-up similar to those in Case Study 1. However, because the BlueField only has a CLI, the study relies on *tshark*, which is the command line equivalent of *wireshark*. Another tool, *ntopng* was later added to test the capability of a flow-based traffic analyzer, a capability arising from the BlueField having OVS in its Linux installation. A visual of the network configuration for this case study is shown in Figure 16.

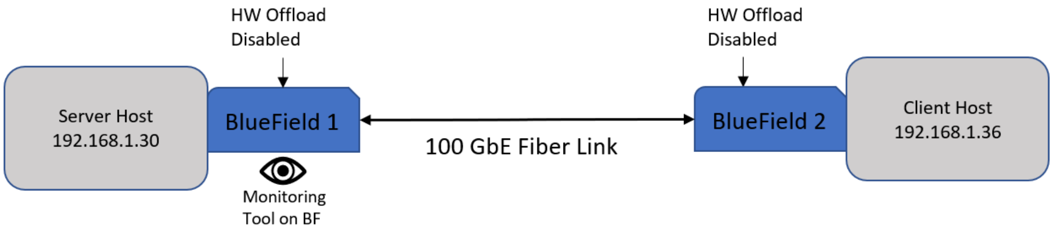


Figure 16. Network Configuration for Case Study 2

The main purpose of *ntopng* is to act as a collector for NetFlow records that will be configured on, and sent from OVS on the BlueField. NetFlow is a network protocol, originally developed by Cisco for collecting IP traffic information and monitoring network flow. For this Case Study, OVS on the BlueField is configured to send NetFlow records to a collector (*ntopng*) which is located on the server host machine. From here the collector analyzes the flow and updates network statistics for observation and analysis in a web browser on the client host machine. The command to enable NetFlow records on OVS to be sent to a collector is shown below:

```
$ ovs-vsctl -- set Bridge ambr1 netflow=@nf -- --id=@nf \
create NetFlow targets=\"192.168.1.30:5566\" \
active-timeout=30
```

In this command, *ovs-vsctl* is invoked first, and the bridge *ambr1* is assigned a NetFlow that is identified as *nf*. This bridge connects the virtual interfaces *pf0hpf* and *p0*, as seen in Figure 14. The command specifies that the NetFlow record is sent to the collector at the IP address 192.168.1.30 via port 5566. Finally, an active timeout of 30 seconds is set, splitting the flow data into several packets, with one flow of data arriving every 30 seconds. Additionally, OVS is configured to disable hardware offload, in order to verify that the monitoring tools are capable of capturing packets at lower bandwidth speeds. To disable hardware offload in OVS the following command is issued:

```
$ ovs-vsctl set Open_vSwitch . other_config:hw-offload=false
```

For this experiment, the InfiniBand/Ethernet 100GbE with BlueField Connect-X 5 Adapter network configuration is used. As in Case Study 1, *iperf* is used to send TCP/IP traffic and report average bandwidth. 45 trials are run, five for each monitoring tool and at each bandwidth. The trial duration is 120 seconds and bandwidth is varied to 1, 3, and 5 Gbps. The reason this case study has hardware offload disabled (and consequently why it is capped at 5 Gbps as opposed to 25 Gbps) is because TCP/IP traffic cannot be captured by the BlueField with it enabled. When a packet capture is initiated on the BlueField and then later viewed for analysis, the only captured traffic is RoCE operations. This is because only the first packet reaches the OVS daemon, all subsequent packets are instructed to be offloaded to the Connect-X 5 NIC as seen in Figure 11. At the packets' destination (server host workstation), TCP/IP traffic is able to be seen in a parallel packet capture on the server host. Due to this, hardware offloading must be disabled to see any traffic on the BlueField for this case study.

3.3.1 Test Steps:

1. Configure both server and client host machines to enable IPoIB, which allows network traffic to be sent via Layer 3.

2. Start *iperf* Receiver on the server at desired bandwidth limit.

3. Initiate network monitoring tool on BlueField and specify appropriate interface to capture packets on.

4. Run *iperf* Sender on the client to send TCP/IP packets to Receiver.

5. After 120 seconds of capture, terminate network monitoring tool.

6. Record 120 bandwidth samples captured in trial run.

7. Repeat steps 2-5 for remaining bandwidth trials.

As in Case Study 1, the number of dropped packets reported by the *ip* utility is subtracted from the number of dropped packets reported by the network monitoring tools. This is done to correct for packets not dropped by the monitoring tools themselves. Additionally the statistics test done in Case Study 1 is also performed here to observe if the data collected was consistent among all tests.

3.4 Case Study 3: BlueField Monitoring, Hardware Offload Enabled

This Case Study examines monitoring capability throughput of InfiniBand applications from the BlueField itself at maximum bandwidth. The tools and set-up are similar to Case Study 2, however OVS is now configured to allow for hardware offload. A visual of the network configuration for this case study is shown in Figure 17. As shown in Figure 11, hardware offload allows for network traffic to be handled by the HCA instead of having to traverse the host OS kernel, freeing the CPU of intensive operations and improving bandwidth. To enable hardware offload in OVS the following command is issued:

```
$ ovs-vsctl set Open_vSwitch . other_config:hw-offload=true
```

With hardware offload enabled, the full bandwidth of the Connect-X 5 adapter and InfiniBand applications can be achieved. The previously mentioned network tools

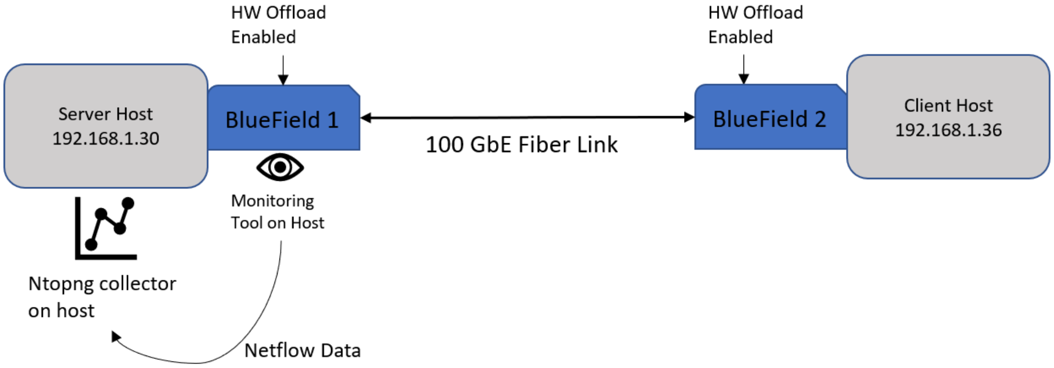


Figure 17. Network Configuration for Case Study 3

can be evaluated for how effectively they monitor InfiniBand applications. For this case study, the InfiniBand/Ethernet 100GbE with BlueField Connect-X 5 Adapter network configuration is used. Once again, *iperf* is used to send TCP/IP traffic and report average bandwidth. A major difference from the previous case studies is the number of trials and tools. Only *ntopng* is used in this case study and the bandwidth is varied to 1, 3, 5, 10, and 25 Gbps for a total of 25 trials, five for each bandwidth speed. The reason for only using *ntopng* is due to the previous case studies revealing that *wireshark* and *tcpdump* were incapable of handling high bandwidths. However, this was not the case for *ntopng* therefore it is the only tool used in this case study.

3.4.1 Test Steps:

1. Configure both server and client host machines to enable IPoIB that allows network traffic to be sent via Layer 3.

2. Start *iperf* Receiver on the server at desired bandwidth limit.

3. Initiate *ntopng* on BlueField and specify appropriate interface to capture pack-

ets on.

4. Run *iperf* Sender on the client to send TCP/IP packets to Receiver.

5. After 120 seconds of capture, terminate network monitoring tool*ntopng* trial run.

6. Repeat steps 2-5 for remaining bandwidth trials.

3.5 Summary

This chapter described the experimental set-up and test process of this thesis. It presented how each host workstation was configured as well as how to correctly configure the InfiniBand network. Additionally, the BlueField was described and the methods to correctly set it up was explained. Next the three case studies that were conducted were presented. Each case study's goal was explained and steps to set up the experiments and collect data were described.

IV. InfiniBand Case Studies Results

4.1 Overview

This chapter highlights the results of the three case studies performed. First, the results of the baseline packet loss test, performed before each case study, is discussed. This shows the percentage of network packets lost due to the hardware and OS alone, without any network monitoring tools used. Following this, each case study's results are presented and discussed. A short recap of each case studies goal is presented, followed by the subsequent results. Each case study's percentage of packet loss is shown and discussed. Additionally, the network monitoring tools used for each case study are evaluated based on the requirements of *Identify*, *Detect*, and *Negative Impacts*. Finally, the results of the last case study are used to present the software and hardware set-up suggested by this research for a monitoring solution for an InfiniBand network.

4.2 Results

4.2.1 Packet Loss Baseline

Before starting the case studies, a baseline of packets dropped by the OS kernel was done using the Linux tool *ip*, running on the server host machine. Regardless of the network monitoring tool used, some packets will be dropped by either the server host OS kernel, or the hardware interface receiving packets. While many factors can contribute to a packet being dropped or lost, this commonly happens due to hardware issues (such as faulty cables or hardware not capable of routing effectively), software issues, and insufficient bandwidth to name a few. Due to this reality, at the beginning of each case study, a bandwidth test was conducted using *iperf* at each bandwidth level used in the case study. Once completed the number of packets dropped were then

Table 3. Baseline Packet Loss Reported by *ip*

Study	Bandwidth	Packets Received	Packets Dropped	Percent Dropped
Case Study 1	1 Gbps	10414492	0	0.0%
	3 Gbps	31242709	153	0.1%
	5 Gbps	52070648	193	0.0%
	10 Gbps	104137417	5212	0.5%
	25 Gbps	225989529	10987	0.5%
Case Study 2	1 Gbps	10414521	0	0.0%
	3 Gbps	31242631	148	0.1%
	5 Gbps	52070535	199	0.0%
Case Study 3	1 Gbps	10414537	0	0.0%
	3 Gbps	31242595	177	0.1%
	5 Gbps	52070499	201	0.0%
	10 Gbps	104136650	5170	0.5%
	25 Gbps	260115117	11086	0.4%

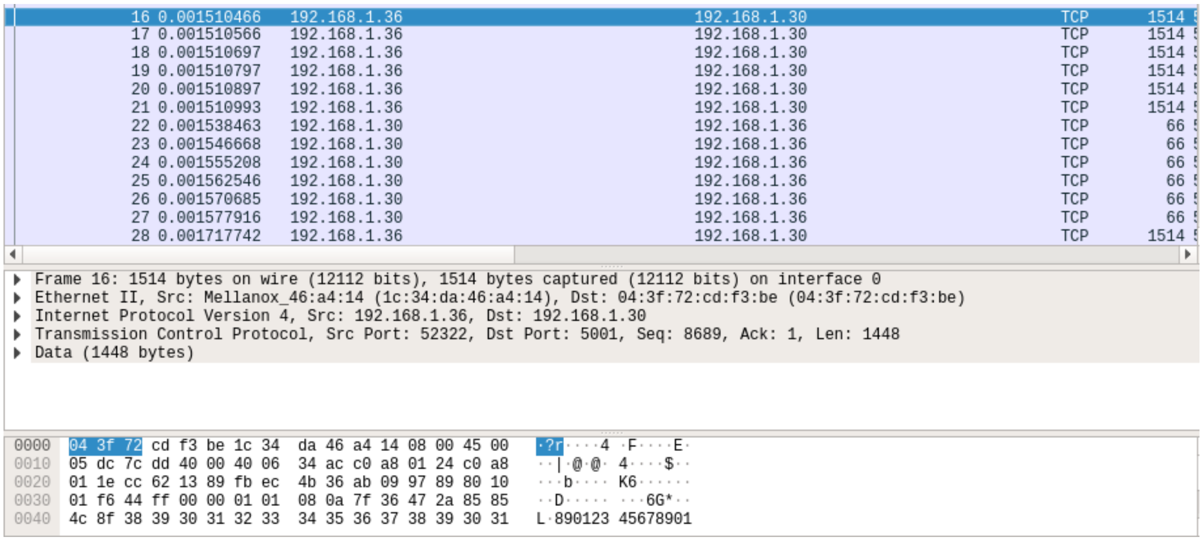
subtracted from the total number of dropped packets reported by each monitoring tool to normalize comparisons.

The data collected is shown in Table 3. It can be seen that at each bandwidth level tested, less than 1% of all packets during the test were dropped, well within acceptable standards. With this data applied to the results of the network monitoring tools in each case study, it is possible to make a better claim on the effects of using the monitoring tools, most importantly whether they are imposing any negative effects on the network. While it is expected that the tools will drop some packets, due to any number of factors, it is not desirable for the monitoring tools themselves to be degrading the network. Again, using these baseline numbers provides a better overall picture of the effect of the tools on the InfiniBand network.

4.2.2 Case Study 1: Results

This study was designed to explore the capabilities of common network monitoring tools on an InfiniBand network. The first test used the network analyzers *wireshark* and *tcpdump* to attempt to capture packets on the server host machine (192.168.1.30).

In all cases, the tools were able to capture TCP/IP traffic on the host workstations, as shown in Figure 18.



```
16 0.001510466   192.168.1.36                       192.168.1.30              TCP       1514
17 0.001510566   192.168.1.36                       192.168.1.30              TCP       1514
18 0.001510697   192.168.1.36                       192.168.1.30              TCP       1514
19 0.001510797   192.168.1.36                       192.168.1.30              TCP       1514
20 0.001510897   192.168.1.36                       192.168.1.30              TCP       1514
21 0.001510993   192.168.1.36                       192.168.1.30              TCP       1514
22 0.001538463   192.168.1.30                       192.168.1.36              TCP         66
23 0.001546668   192.168.1.30                       192.168.1.36              TCP         66
24 0.001555208   192.168.1.30                       192.168.1.36              TCP         66
25 0.001562546   192.168.1.30                       192.168.1.36              TCP         66
26 0.001570685   192.168.1.30                       192.168.1.36              TCP         66
27 0.001577916   192.168.1.30                       192.168.1.36              TCP         66
28 0.001717742   192.168.1.36                       192.168.1.30              TCP       1514

▸ Frame 16: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
▸ Ethernet II, Src: Mellanox_46:a4:14 (1c:34:da:46:a4:14), Dst: 04:3f:72:cd:f3:be (04:3f:72:cd:f3:be)
▸ Internet Protocol Version 4, Src: 192.168.1.36, Dst: 192.168.1.30
▸ Transmission Control Protocol, Src Port: 52322, Dst Port: 5001, Seq: 8689, Ack: 1, Len: 1448
▸ Data (1448 bytes)

0000  04 3f 72 cd f3 be 1c 34  da 46 a4 14 08 00 45 00   ·?r····4 ·F····E·
0010  05 dc 7c dd 40 00 40 06  34 ac c0 a8 01 24 c0 a8   ··|·@·@· 4····$··
0020  01 1e cc 62 13 89 fb ec  4b 36 ab 09 97 89 80 10   ···b···· K6······
0030  01 f6 44 ff 00 00 01 01  08 0a 7f 36 47 2a 85 85   ··D····· ···6G*··
0040  4c 8f 38 39 30 31 32 33  34 35 36 37 38 39 30 31   L·890123 45678901
```

Figure 18. Wireshark Analysis of Captured InfiniBand Packets

However, further analysis shows that when increasing the bandwidth, the ability to capture all traffic begins to diminish. What starts as trivial at 1 Gbps becomes ineffective at max bandwidth. As seen in Figure 19 both monitoring tools begin to drop off in effectiveness quickly once bandwidth is increased to 3 Gbps. Notably, *wireshark* begins to drop off dramatically at 5 Gbps and is incapable of keeping up at higher bandwidths. *Tcpdump* performs better than wireshark above 3 Gbps, however it still drops a large percentage of packets. A monitoring tool drops packets when the buffer space allocated to it by the OS fills up [27].

Following the overall performance of the network monitoring tools, a look at the statistics of all dropped packets resulting from each bandwidth test gives an estimate of collection accuracy. As stated in Section 3.1.3, a total of 50 trials were performed, with five conducted at each bandwidth level, for each monitoring tool. The mean of dropped packets at each bandwidth was calculated, followed by the SD of the dropped

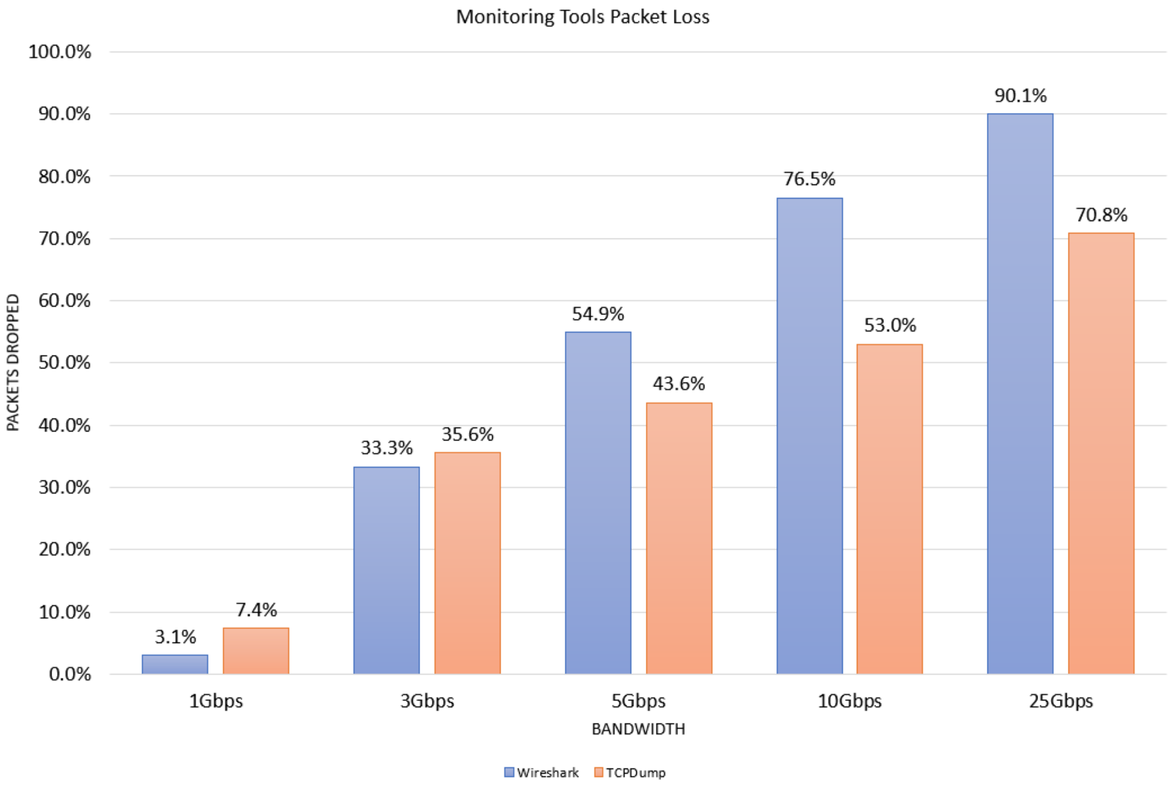


Figure 19. Comparison of Dropped Packets at Varying Bandwidths on the Host Server

packets. Then, by dividing the SD by the mean number of dropped packets, the CV was calculated. As shown in Table 4, the CV column shows that all tests were under a CV of 5%. The CV is a measure of the SD relative to the mean, allowing a simple, unit-less measure of the spread of the collected data. For this experiment, this implies that data collected was generally consistent across all trial runs and, barring possible outliers, additional runs will most likely produce similar results.

Next, an analysis of how well each tool met the requirements for a monitoring solution was performed. Recall that the three requirements involved are *Identify*, *Detect*, and *Negative Impacts to Network*. As shown in Table 5, both *wireshark* and *tcpdump* are only able to meet all three requirements at a bandwidth of 1 Gbps. Ideally all requirements would be met for each bandwidth speed. The ability to

Table 4. Case Study 1 Packet Loss: Data Variance

Monitoring Tool	Bandwidth	Packets Dropped - Mean	of Packets Dropped - Std. Dev.	CV
Wireshark	1 Gbps	3.1%	493.25	2.4%
	3 Gbps	33.3%	1079.03	0.2%
	5 Gbps	54.9%	5450.47	0.2%
	10 Gbps	76.5%	2488.52	0.1%
	25 Gbps	90.0%	46663.91	0.0%
TCPDump	1 Gbps	7.4%	372.61	0.8%
	3 Gbps	35.5%	5240.27	0.7%
	5 Gbps	43.6%	9643.42	0.7%
	10 Gbps	53.0%	169169.98	4.7%
	25 Gbps	70.8%	229920.93	2.9%

monitor InfiniBand traffic at a maximum speed of 1 Gbps is not helpful when typical InfiniBand speeds are in the tens and hundreds of gigabits per second.

Table 5. Case Study 1: Monitoring Tool Requirements Performance

Monitoring Tool	Bandwidth	Identify	Detect	Negative Impacts
Wireshark	1 Gbps	✓	✓	✓
	3 Gbps	✓		✓
	5 Gbps	✓		✓
	10 Gbps	✓		✓
	25 Gbps	✓		✓
TCPDump	1 Gbps	✓	✓	✓
	3 Gbps	✓		✓
	5 Gbps	✓		✓
	10 Gbps	✓		✓
	25 Gbps	✓		

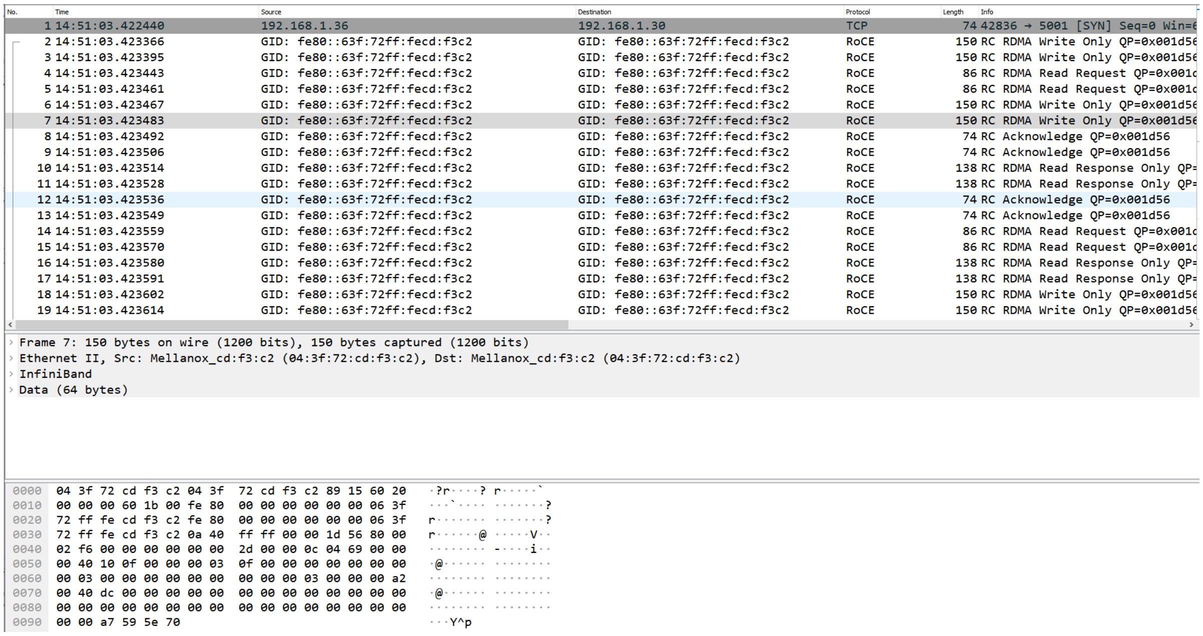
At the conclusion of Case Study 1, the overall packet loss of each monitoring tool and the variation of those lost packets have two implications. First, network traffic created using Ethernet and RoCE operations can be captured using common network monitoring tools. As shown in Figures 18 and 19 as well as Table 4, both monitoring tools were capable of receiving the network traffic. Second, unfortunately, at speeds above 1 Gbps, the monitoring tools are ineffective at capturing all network traffic. Both *wireshark* and *tcpdump* begin to drop well over 30% of all incoming packets beginning at 3 Gbps and this only worsens with increased bandwidth. This implies that while traffic capture is possible, very high speeds are still an issue for common network monitoring tools. Additionally, as shown in Table 5, the monitoring tools do not meet the requirements outlined earlier for a monitoring solution for an InfiniBand network, implying the need for another solution for successful traffic monitoring.

4.2.3 Case Study 2: Results

Case Study 2's purpose was to determine if the same monitoring tools used to capture InfiniBand traffic on the host workstation could function on the BlueField NIC. With hardware offload enabled, TCP/IP traffic was not able to be captured on the BlueField itself. As shown in Figure 20, the first packet that was captured is the initial TCP/IP packet, while every subsequent packet is offloaded to the NIC. Following this, only RDMA operations can be seen. While this is interesting and provides insight into how the BlueField is operating, the capability to see packet data is unfortunately lost. Further study required disabling the hardware offload feature.

Once hardware offload was disabled, the test was performed according to the procedure described in 3.3. As seen in Figure 21, *tshark* and *tcpdump* are incapable of effectively capturing all traffic, with *tshark* dropping a minimum of 65.7% of packets and *tcpdump* effectively dropping all. Most interesting is *ntopng's* results showing very minimal packet loss. In order to confirm that the count of packet loss was accurate, the number of packets reported by the monitoring tools was compared with the Linux tool *ethtool* which reports numerous statistics on traffic received on a specified interface. This confirmed that the reported metrics from the monitoring tools were accurate. Additionally, *ntopng* allows for the live capture of packets in specific time intervals, which also confirmed minimal packet loss by examining the pcap files.

Next, as in Case Study 1, a comparison of the dropped packet statistics at each



```
No.  Time              Source                        Destination                   Protocol  Length Info
   1 14:51:03.422440   192.168.1.36                  192.168.1.30                  TCP           74 42836 → 5001 [SYN] Seq=0 Win=6
   2 14:51:03.423366   GID: fe80::63f:72ff:fecd:f3c2 GID: fe80::63f:72ff:fecd:f3c2 RoCE         150 RC RDMA Write Only QP=0x001d56
   3 14:51:03.423395   GID: fe80::63f:72ff:fecd:f3c2 GID: fe80::63f:72ff:fecd:f3c2 RoCE         150 RC RDMA Write Only QP=0x001d56
   4 14:51:03.423443   GID: fe80::63f:72ff:fecd:f3c2 GID: fe80::63f:72ff:fecd:f3c2 RoCE          86 RC RDMA Read Request QP=0x001d
   5 14:51:03.423461   GID: fe80::63f:72ff:fecd:f3c2 GID: fe80::63f:72ff:fecd:f3c2 RoCE          86 RC RDMA Read Request QP=0x001d
   6 14:51:03.423467   GID: fe80::63f:72ff:fecd:f3c2 GID: fe80::63f:72ff:fecd:f3c2 RoCE         150 RC RDMA Write Only QP=0x001d56
   7 14:51:03.423483   GID: fe80::63f:72ff:fecd:f3c2 GID: fe80::63f:72ff:fecd:f3c2 RoCE         150 RC RDMA Write Only QP=0x001d56
   8 14:51:03.423492   GID: fe80::63f:72ff:fecd:f3c2 GID: fe80::63f:72ff:fecd:f3c2 RoCE          74 RC Acknowledge QP=0x001d56
   9 14:51:03.423506   GID: fe80::63f:72ff:fecd:f3c2 GID: fe80::63f:72ff:fecd:f3c2 RoCE          74 RC Acknowledge QP=0x001d56
  10 14:51:03.423514   GID: fe80::63f:72ff:fecd:f3c2 GID: fe80::63f:72ff:fecd:f3c2 RoCE         138 RC RDMA Read Response Only QP=
  11 14:51:03.423528   GID: fe80::63f:72ff:fecd:f3c2 GID: fe80::63f:72ff:fecd:f3c2 RoCE         138 RC RDMA Read Response Only QP=
  12 14:51:03.423536   GID: fe80::63f:72ff:fecd:f3c2 GID: fe80::63f:72ff:fecd:f3c2 RoCE          74 RC Acknowledge QP=0x001d56
  13 14:51:03.423549   GID: fe80::63f:72ff:fecd:f3c2 GID: fe80::63f:72ff:fecd:f3c2 RoCE          74 RC Acknowledge QP=0x001d56
  14 14:51:03.423559   GID: fe80::63f:72ff:fecd:f3c2 GID: fe80::63f:72ff:fecd:f3c2 RoCE          86 RC RDMA Read Request QP=0x001d
  15 14:51:03.423570   GID: fe80::63f:72ff:fecd:f3c2 GID: fe80::63f:72ff:fecd:f3c2 RoCE          86 RC RDMA Read Request QP=0x001d
  16 14:51:03.423580   GID: fe80::63f:72ff:fecd:f3c2 GID: fe80::63f:72ff:fecd:f3c2 RoCE         138 RC RDMA Read Response Only QP=
  17 14:51:03.423591   GID: fe80::63f:72ff:fecd:f3c2 GID: fe80::63f:72ff:fecd:f3c2 RoCE         138 RC RDMA Read Response Only QP=
  18 14:51:03.423602   GID: fe80::63f:72ff:fecd:f3c2 GID: fe80::63f:72ff:fecd:f3c2 RoCE         150 RC RDMA Write Only QP=0x001d56
  19 14:51:03.423614   GID: fe80::63f:72ff:fecd:f3c2 GID: fe80::63f:72ff:fecd:f3c2 RoCE         150 RC RDMA Write Only QP=0x001d56

> Frame 7: 150 bytes on wire (1200 bits), 150 bytes captured (1200 bits)
> Ethernet II, Src: Mellanox_cd:f3:c2 (04:3f:72:cd:f3:c2), Dst: Mellanox_cd:f3:c2 (04:3f:72:cd:f3:c2)
> InfiniBand
> Data (64 bytes)

0000  04 3f 72 cd f3 c2 04 3f  72 cd f3 c2 89 15 60 20   ·?r····? r·····`
0010  00 00 00 60 1b 00 fe 80  00 00 00 00 00 00 06 3f   ···`···· ·······?
0020  72 ff fe cd f3 c2 fe 80  00 00 00 00 00 00 06 3f   r······· ·······?
0030  72 ff fe cd f3 c2 0a 40  ff ff 00 00 1d 56 80 00   r······@ ·····V··
0040  02 f6 00 00 00 00 00 00  2d 00 00 0c 04 69 00 00   ········ -····i··
0050  00 40 10 0f 00 00 00 03  0f 00 00 00 00 00 00 00   ·@······ ········
0060  00 03 00 00 00 00 00 00  00 00 00 03 00 00 00 a2   ········ ········
0070  00 40 dc 00 00 00 00 00  00 00 00 00 00 00 00 00   ·@······ ········
0080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0090  00 00 a7 59 5e 70                                  ···Y^p
```

Figure 20. Packet Capture of RoCE Traffic on the BlueField with Hardware Offload Enabled

bandwidth level was completed. As stated in 3.3, a total of 45 trials were performed (not 50 trials as in Case Study 1, due to two bandwidth levels not being tested because of limitations from hardware offload), with five conducted at each bandwidth level for each monitoring tool. As previously, the mean number of dropped packets at each bandwidth level was calculated, along with the SD and CV. As shown in Table 6, the CV column shows that all tests were under a CV of 6%. As stated in 3.2 this implies that it is safe to conclude that all data collected was consistent across all trial runs and, barring any outliers, additional trials will most likely produce similar results.

Next, as in Case Study 1, an analysis of how each tool met the requirements for a monitoring solution was accomplished. As shown in Table 7, *tshark* and *tcpdump* fail to meet all three requirements for all bandwidth levels. However, *ntopng* met the requirements outlined at each bandwidth level, suggesting that *ntopng* could be capable of monitoring an Infiniband network at high bandwidth, which is expanded

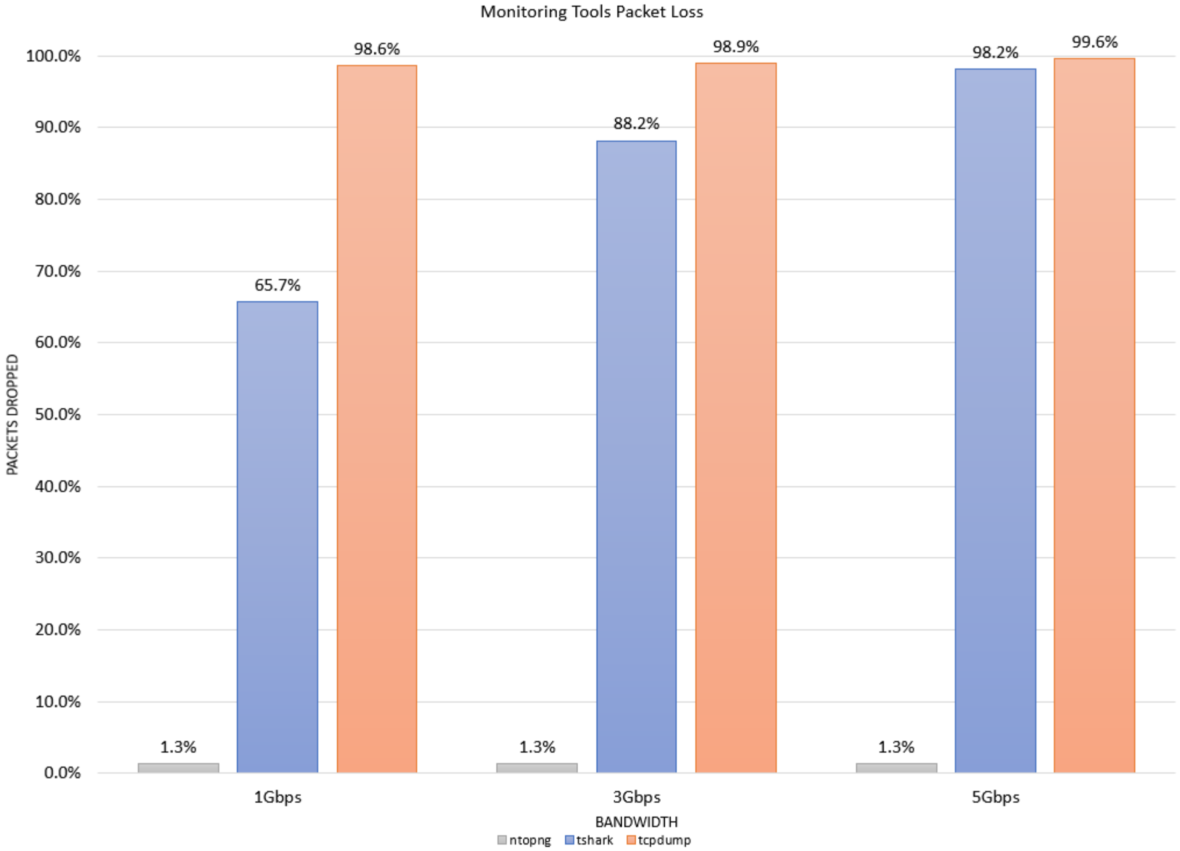


Figure 21. Comparison of Dropped Packets at Varying Bandwidths on the BlueField

Table 6. Case Study 2 Packet Loss: Data Variance

Monitoring Tool	Bandwidth	Packets Dropped - Mean	Packets Dropped - Std. Dev.	CV
Tshark	1 Gbps	65.7%	2043.18	0.4%
	3 Gbps	88.2%	19554.85	0.9%
	5 Gbps	98.2%	90484.81	2.8%
TCPDump	1 Gbps	98.6%	10687.54	1.5%
	3 Gbps	98.9%	31744.95	1.3%
	5 Gbps	98.2%	16916.16	0.5%
Ntopng	1 Gbps	1.3%	492.94	5.1%
	3 Gbps	1.3%	1231.38	4.1%
	5 Gbps	1.4%	1050.55	2.1%

upon in Case Study 3's results.

The packet loss of each monitoring tool observed in Case Study 2 has several

Table 7. Case Study 2: Monitoring Tool Requirements Performance

Monitoring Tool	Bandwidth	Identify	Detect	Negative Impacts
Tshark	1 Gbps	✓		✓
	3 Gbps	✓		✓
	5 Gbps			✓
TCPDump	1 Gbps			✓
	3 Gbps			✓
	5 Gbps			✓
Ntopng	1 Gbps	✓	✓	✓
	3 Gbps	✓	✓	✓
	5 Gbps	✓	✓	✓

implications. First, when using standard network monitoring tools, the hardware offloading capabilities of the BlueField are not available. In this configuration, the ability to see individual TCP/IP packet data is lost. Second, network traffic created using Ethernet and RoCE operations can be captured using common network monitoring tools on the BlueField when hardware offload is disabled, albeit only at lower bandwidths. As shown in Figure 21 and Table 6, both monitoring tools were capable of receiving network traffic. Finally, capturing traffic effectively is not possible via *tshark* and *tcpdump*. Looking at Figure 21 *tcpdump* and *tshark* drop over 99% of network packets at maximum bandwidth. This points to a need for another method for monitoring network traffic.

Fortunately, *ntopng*s results are promising. At each bandwidth level, only a small percentage of packets are dropped; slightly over 1%. (Dropped here implies they were not captured and processed by the network interface). The consistency of each test and the efficiency of the network monitoring tool suggest that *ntopng* and its flow-based traffic monitoring capability may be a promising alternative option for monitoring InfiniBand traffic. The possibility of combining the offloading capabilities of the BlueField and the efficiency of *ntopng* at higher bandwidth was the focus of Case Study 3.

4.2.4 Case Study 3: Results

Case Study 3's purpose was to determine if *ntopng*, after showing promising results monitoring InfiniBand traffic on the BlueField in Case Study 2, could monitor IBA traffic on the BlueField at higher bandwidths. As in the prior case studies, all other monitoring tools explored dropped significant percentages of packets as bandwidth increased. This is likely due to either the packet capture library being slow, copying packets between user and kernel space, or buffer space within the kernel OS overflowing quickly. Case Study 2 showed that *ntopng* captured InfiniBand traffic very effectively up to 5 Gbps. As stated in 3.3, *ntopng* is a flow-based monitoring tool. In this experiment *ntopng* executes on the server host workstation, while the flow data is configured on and sent from the BlueField via OVS. Thus, it seemed possible that hardware offload could be taken advantage of (to obtain high bandwidth speeds) and still monitor from the BlueField itself without losing the ability to see packet data.

Upon performing the trials, not only were TCP/IP packets visible, and captured at all bandwidths, but *ntopng* dropped very few packets during testing. As shown in Figure 22 from 1 Gbps up to 25 Gbps, *ntopng* only dropped between 1.3% to 2.6% of all network packets. To confirm that packet loss was accurate, the number of packets reported by *ntopng* was compared with the Linux tool *ethtool* running on the server host machine, as in Case Study 1. An inspection confirmed that *ntopng* was seeing all generated traffic and that its reported metrics were accurate.

Next, as in Case Study 1 and 2, a comparison of the statistics of the dropped packets at each bandwidth level was completed. As stated in 3.4, 25 tests were run, with five conducted at each bandwidth. As previously, the mean of each bandwidth level test's dropped packets were calculated, with the SD and CV of the dropped packets calculated as well. As shown in Table 8, the CV column shows that all trials experienced a CV of no more than 5%. As stated in 3.2 this implies that it is safe

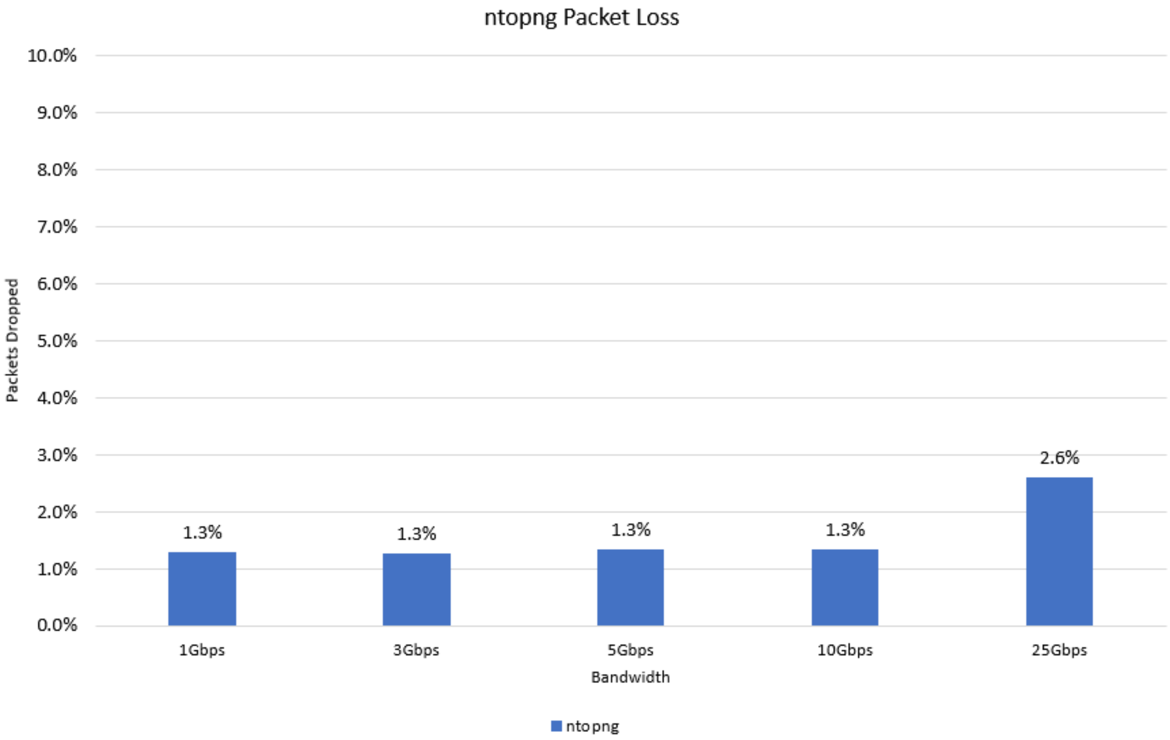


Figure 22. Dropped Packets at Varying Bandwidths on the Host Server Using Ntopng

to conclude that data collected was consistent across all trials and, barring outliers, additional trials would likely produce similar results.

Table 8. Case Study 3 Packet Loss: Data Variance

Monitoring Tool	Bandwidth	Packets Dropped - Mean	Packets Dropped - Std. Dev.	CV
Ntopng	1 Gbps	1.3%	466.00	5.0%
	3 Gbps	1.3%	1040.50	3.7%
	5 Gbps	1.4%	347.24	0.7%
	10 Gbps	1.4%	317.27	0.3%
	25 Gbps	2.6%	1765.02	0.4%

Next, as in the previous case studies, an analysis of how each tool met the requirements for a monitoring solution was performed. As shown in Table 9, *ntopng* met every requirement at each bandwidth level. In all cases, over 97% of network traffic was captured, and average bandwidth was within 1% of expectation. This suggests that *ntopng* is potentially capable of effectively monitoring an InfiniBand network.

Table 9. Case Study 3: Monitoring Tool Requirements Performance

Monitoring Tool	Bandwidth	Identify	Detect	Negative Impacts
Ntopng	1 Gbps	✓	✓	✓
	3 Gbps	✓	✓	✓
	5 Gbps	✓	✓	✓
	10 Gbps	✓	✓	✓
	25 Gbps	✓	✓	✓

In summary, at the conclusion of Case Study 3, the overall packet loss of *ntopng* at the varying bandwidth levels has several implications. First, network traffic created using Ethernet and RoCE operations can be captured using *ntopng* on the BlueField. Hardware offload could be enabled on the BlueField, allowing for high network speeds, and packet data was viewable in the packet capture files. Second, based on the results shown in Figure 22 and Table 8, it can be concluded that *ntopng* and its implementation of a packet capture library in conjunction with the Mellanox hardware has drastically reduced packet loss. Compared to *wireshark* and *tcpdump* from Case Study 1, packet loss has decreased at max bandwidth by over 95% in both cases.

There are several possibilities that could explain why this could be happening. The first, and probably simplest explanation is that *ntopng* uses the packet capture library *PF_Ring* and, as stated in 2.4.1, this newer library allows for much faster packet capture speeds. This is mostly possible due to the circular buffer which allows incoming packets to be distributed to multiple rings simultaneously. At very high bandwidths, packets are coming in extremely fast, at well over hundreds of millions of packets per second. Being able to distribute these to multiple buffers at the same time could lead to lower overall packet loss for the network. Second, it is possible that since the information being sent is coming from the BlueField, via OVS and sent to the *ntopng* collector on the host server workstation, that not involving the host server workstation CPU as intensively is freeing processor power to process incoming packets.

In typical network traffic, the CPU will process incoming packets and determine where they should be sent. Normally an intensive process, when multiplied by hundreds of millions of packets the hit to bandwidth performance becomes more profound at higher speeds. Any offload from the host server workstation should lead to some performance improvement on network bandwidth.

4.3 Monitoring Software and Hardware Set-up

The set-up for this possible solution is fairly simple. Each host workstation in the network that will not strictly monitor InfiniBand traffic would need a NIC that has a DPU capable of the high speeds and offloading capabilities of the BlueField. As stated in 2.4.4, DPUs are typically deployed in high speed network environments where bandwidth, packet processing and encryption are desired. These features meet the *Identify* and *Detect* functions as well and only further the need for a DPU in this set-up. Additionally, they will have a some form of OS installed that supports a virtual switching software similar to OVS. This will allow for flows, like NetFlow or Sflow, to be configured to be sent off the NIC. Recall that most flow-based protocols enable the ability to not only monitor network traffic on a separate host, but provide for a much higher level of network visibility to aide in network analysis. A simple, visual representation of this set-up is shown in Figure 23. Finally, a host workstation would need to act as the collector of the flow data. As was shown in Case Study 3, *ntopng* took on this role (though another tool with the same capabilities could be used). This open-source monitoring tool was used to interpret the incoming flow and InfiniBand traffic and then present it in a way that a user could analyze.

Based on the capabilities of the monitoring tools and the results found in Case Study 3, one should be able to monitor InfiniBand traffic at line rate without loss of fidelity in the network. While this does not cover every potential avenue, it is an

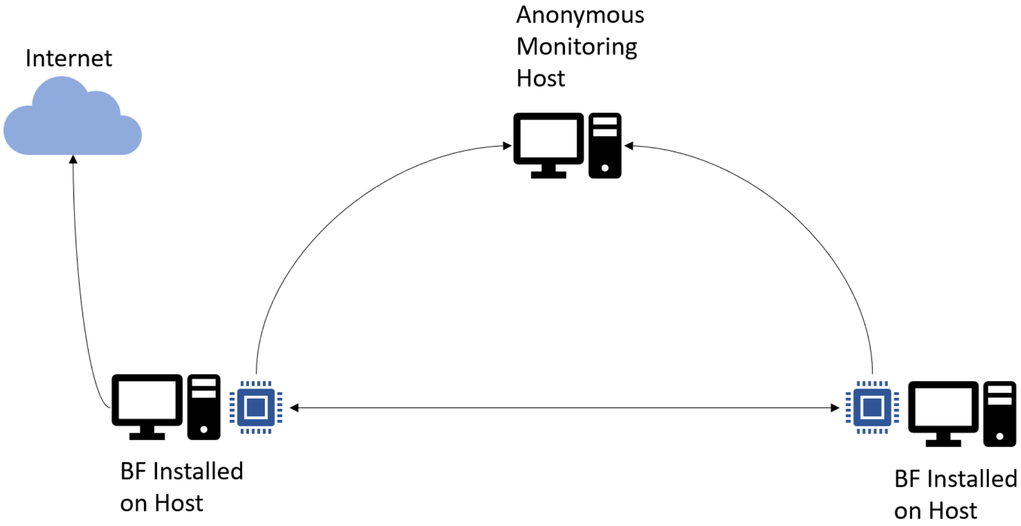


Figure 23. Example InfiniBand Network Monitoring Set-up

important start for a comprehensive security set-up for an InfiniBand network.

4.4 Summary

This chapter analyzed the results of the three case studies regarding the capabilities of network monitoring tools on an InfiniBand network. Case Study 1 demonstrated that common network monitoring tools can capture InfiniBand traffic using Ethernet and RoCE operations, however at very high bandwidths their effectiveness is diminished. Case Study 2 showed that attempting to capture InfiniBand traffic directly from the BlueField NIC yielded similar results for the same monitoring tools, capturing the traffic but dropping almost all packets. Notably, a newer monitoring tool, *ntopng* had excellent performance and presented itself as a possible solution to monitoring InfiniBand traffic. Case Study 3 took and tested its network monitoring capabilities at very high bandwidths and yielded very low packet loss. In conjunction with a high speed, InfiniBand-capable adaptor, this tool provides a possible network monitoring solution for an InfiniBand network. The three case studies concluded that typical network monitoring solutions do not translate well to an InfiniBand network, however by using Ethernet as the link layer protocol together with a monitoring tool like *ntopng* it is possible to effectively monitor InfiniBand traffic.

V. Conclusion

5.1 Overview

This chapter summarizes the work performed for this research concerning the capability of monitoring an InfiniBand network. It reiterates the motivation behind monitoring an InfiniBand network and how it is possible to retain the benefits of InfiniBand without compromising on performance. It discusses the benefits of the monitoring solution proposed while also highlighting any drawbacks and challenges that existed during experimentation. The chapter closes by discussing the significance of the research performed as well as future work that needs to be done to ensure the overall security of an InfiniBand network.

5.2 Summary

This research focused on the ability to passively monitor an InfiniBand network and the challenges associated with this task. It described the features of the IBA, including the key hardware components, software and network architecture unique to InfiniBand. The NIST cybersecurity framework was discussed and three key functions were identified as requirements for a network monitoring solution for an InfiniBand network. Related work involving the IBA and its current level of security was presented to lay a foundation for why this research's goals were an important basis for an overall security solution for the IBA.

This thesis consisted of three case studies, each revealing the capabilities of common network monitoring tools as applied to the representative InfiniBand network. Case Study 1 explored the ability to monitor InfiniBand traffic using Ethernet as the link layer (and the RoCE protocol) in conjunction with hardware offload, on the server host workstation. Results obtained in Case Study 1 highlighted that common network monitoring tools are capable of monitoring InfiniBand traffic that uses Ethernet as the link layer. However, these tools were shown to be incapable of effectively capturing all network traffic at speeds above 1 Gbps. All monitoring tools tested dropped a significant number of TCP/IP packets. This case study proved that common monitoring tools are only marginally effective at capturing InfiniBand traffic.

Case Study 2 tested the ability to monitor the same type of InfiniBand traffic in Case Study 1, but now on the BlueField SmartNIC and analyzed the efficiency of the monitoring tools. Case Study 2 demonstrated that the same network monitoring tools used in Case Study 1, but directly implemented and executed on the BlueField SmartNIC, are capable of capturing InfiniBand traffic. However, they drop a significant number of packets. Unfortunately, the advantages of hardware offload are not available, as evidenced by the results of this case study; the only traffic captured by these tools came from RDMA operations performed by the hardware. Interestingly enough, one of the new tools, *ntopng*, a flow-based monitoring tool, was not only capable of capturing the InfiniBand traffic, but was also able to capture the majority of TCP/IP packets with minimal packet loss. This case study showed the promise for a different type of monitoring tool and showed the need to further test it at higher bandwidths.

Case Study 3 observed the capabilities of a flow-based monitoring tool at high bandwidths in conjunction with hardware offload. The same tool was configured to accept flow data from the BlueField SmartNIC, to the *ntopng* collector on the server host workstation at bandwidths up to 25 Gbps. Not only could the tool capture traffic at these high speeds, but it was also able to drop a minimal number of TCP/IP packets, just as in Case Study 2. Case Study 3 determined that a potential monitoring solution for an InfiniBand network could reside in a flow-based tool like *ntopng*, using Ethernet as the link layer protcol, and RoCE operations. Finally, a brief but simple introduction to a monitoring solution for a InfiniBand network was introduced. Using the *Identify* and *Detect* functions from the NIST framework, the necessary requirements for this solution were described and the hardware needed was introduced.

5.3 Discussion

Below we discuss the contributions of this research, as well as limitations that were encountered during experimentation, and issues concerning security:

5.3.1 Benefits

The benefit of a viable network monitoring solution for a high speed network like InfiniBand is being able to apply low-cost, but reliable and proven security tools. These already exist in most standard network installations, but high speed networks bring with them more challenges. With InfiniBand already present in the top ten supercomputers in use today, a usable security solution can be the catalyst to mainstream adoption of InfiniBand, bringing the benefits of fast data transfer with it.

5.3.2 Limitations, Challenges & Security

While this set-up shows promise, it has limitations. Ideally, traffic in the network would be sent to a mirrored port, away from the main network, to be discreetly monitored. However, the current implementation of OVS imposes a significant performance degradation when applying port mirrors. In testing, bandwidth never exceeded 5Gbps with this implementation. Additionally, research into offloading IP Security (IPsec) to the InfiniBand hardware may prove fruitful. This capability would greatly increase the security of InfiniBand traffic, but it is not present in the current generation of the BlueField processor. It should be noted that currently released information from Mellanox states that the next iteration of the BlueField will have this capability [28].

5.4 Future Work

As the IBA becomes more prevalent in industry, and the evolving threat of cybersecurity incidents continues unabated, there are other areas that should be researched and explored. Listed below are areas of interest that could expand on this research:

- **100 Gbit Line Rate Encryption:** The hardware used throughout this thesis was Nvidia Mellanox's BlueField SmartNIC, the first iteration of their DPU family of InfiniBand capable hardware. While this SmartNIC had many capabilities, it was limited in others, specifically in offloading IPsec encryption. Prior research showed that IPsec operations impose a large performance hit on InfiniBand, and concluded that the BlueField's offloading capabilities could solve this [7]. While partially true, the BlueField was missing the hardware accelerators to make this possible. During this research Nvidia Mellanox released the BlueField 2 SmartNIC which claims to be able to perform IPsec encryption by offloading to the NIC hardware, at 100 Gbit speeds [28]. Research into testing the offloading capabilities of the BlueField 2, supported by the information discovered in monitoring InfiniBand traffic in this thesis would be useful.
- **Security SDK:** Nvidia Mellanox developed a Security SDK that acts as a Deep-Packet inspection tool. Used in conjunction with a monitoring solution that can see InfiniBand traffic, a potential Intrustion Detection System (IDS) could be created that can further process packets in an InfiniBand network. Future research could deploy the Security SDK and evaluate its effectiveness, together with already-proven security practices for the IBA. As stated in prior

research, the Security SDK should attempt to secure all types of InfiniBand traffic [7].

- **Root of Trust:** The BlueField SmartNIC was used heavily in this research, and newer versions of this hardware are expected to be used in the future. Since these devices sit in the middle of the network and are key pieces in network communication, hosting their own OS on board, it makes sense to research the security of those boot processes. Root of Trust can be explored on the BlueField SmartNIC to determine if the hardware architecture chosen for the SoC has any vulnerabilities that would need to be mitigated to ensure network security for any environment that InfiniBand might be used in.

5.5 Conclusion

The three case studies in this paper explored the capabilities of modern network monitoring tools within an InfiniBand network and looked to find a viable option. Case Study 1 demonstrated that common monitoring tools can capture InfiniBand traffic at lower bandwidths, due to inefficiencies in the packet capture libraries or low buffer space in the OS kernel. Case Study 2 showed that putting these same tools on the InfiniBand NIC incurred the same penalties as in Case Study 1, however another tool, *ntopng* that uses flows and a more efficient packet capture library was capable of very low packet loss at much higher bandwidths. Case Study 3 then showed that increasing this speed to max bandwidth incurred minor packet loss, posing as a possible and viable option for monitoring an InfiniBand network. This study indicates that options exist to monitor InfiniBand networks and that further research should be done to find the optimal set-up for low-cost, efficient and reliable monitoring solutions.

Bibliography

1. J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach.* Morgan Kaufmann, 2011.

2. J. Dongarra, H. Meuer, M. Meuer, H. Simon, and E. Strohmaier, "Top 500 The List," November 2020. [Online]. Available: https://www.top500.org/lists/top500/2020/11/ [Accessed: 2020-12-08]

3. D. Schmitt, S. Graham, P. Sweeney, and R. Mills, "A Cyber Vulnerability Assessment of InfiniBand Networking," in *Critical Infrastructure Protection XIII.* Springer, Cham, 2019, pp. 179–205.

4. M. Lee and E. J. Kim, "A Comprehensive Framework for Enhancing Security in InfiniBand Architecture," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, pp. 1393–1406, September 2007.

5. M. Lee, E. J. Kim, and M. Yousif, "Security enhancement in InfiniBand architecture," in *International Symposium on Parallel and Distributed Processing (IPDPS).* Denver, CO: IEEE, April 4-8, 2005, pp. 1530–2075.

6. K. P. Subedi, D. Dasgupta, and B. Chen, "Security analysis on InfiniBand protocol implementations," in *IEEE Symposium Series on Computational Intelligence (SSCI).* Athens, Greece: IEEE, December 6-9, 2016.

7. L. Mireles, "Implications and Limitations of Securing an InfiniBand Network," Master's thesis, Air Force Institute of Technology, 2020.

8. F. Schneider, "Performance evaluation of packet capturing systems for high-speed networks," Ph.D. dissertation, Technical University of Munich, 2005.

9. S. Alias, S. Manickam, and M. Kadhum, "A Study on Packet Capture Mechanisms in Real Time Network Traffic," in *2013 International Conference on Advanced Computer Science Applications and Technologies.* Kuching, Malaysia: IEEE, December 23-24, 2013, pp. 456–460.

10. M. Dashtbozorgi and M. Azgomi, "A High-Performance Software Solution for Packet Capture and Transmission," in *2009 2nd IEEE International Conference on Computer Science and Information Technology.* Beijing, China: IEEE, August 8-11, 2009, pp. 407–411.

11. Mellanox-Technologies, "Introduction to InfiniBand," Tech. Rep., 2003. [Online]. Available: https://www.mellanox.com/pdf/whitepapers/IB\

textunderscoreIntro\textunderscoreWP\textunderscore190.pdf [Accessed: 2020-01-26]

12. "Framework for Improving Critical Infrastructure Cybersecurity," *Proceedings of theAnnual ISA Analysis Division Symposium*, pp. 9–25, 2018. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04162018.pdf [Accessed: 2020-12-15]

13. NIST, "Guide to Intrusion Detection and Prevention Systems (IDPS)," Tech. Rep., 2012. [Online]. Available: http://www.reference.com/go/http://csrc.ncsl.nist.gov/publications/nistpubs/800-94/SP800-94.pdf [Accessed: 2020-12-31]

14. R. Buyya, T. Cortes, and H. Jin, *An Introduction to the InfiniBand Architecture*, 2002, pp. 616–632.

15. IBTA, "InfiniBand TM Architecture Specification Volume 1 Release 1.4," Tech. Rep., 2020. [Online]. Available: https://www.infinibandta.org/ibta-specifications-download/ [Accessed: 2020-04-30]

16. Melanox-Technologies, "InfiniBand Software and Protocols Enable Seamless Off-the-shelf Applications Deployment," Tech. Rep., December 2007. [Online]. Available: https://www.mellanox.com/pdf/whitepapers/WP\textunderscore2007\textunderscoreIB\textunderscoreSoftware\textunderscoreand\textunderscoreProtocols.pdf [Accessed: 2020-02-02]

17. J. Corbet, A. Rubini, and G. Kroah-Hartman, *Linux Device Drivers.* Newton, MA: O'Reilly Media, 2005.

18. J. Langston, "InfiniBand Survey," pp. 1–8, 2008. [Online]. Available: http://blogs.cae.tntech.edu/jwlangston21/files/2008/08/infiniband.pdf [Accessed: 2020-01-03]

19. G. Herrin, "Linux IP Networking," 2000. [Online]. Available: https://www.cs.unh.edu/cnrg/people/gherrin/linux-net.html#tth_chAp2 [Accessed: 2020-09-22]

20. P. MacArthur and R. Russell, "A Performance Study to Guide RDMA Programming Decisions," in *2012 IEEE 14th International Conference on High Performance Computing and Communications.* Liverpool, UK: IEEE, June 25-27, 2012, pp. 778–785.

21. Mellanox-Technologies, "RDMA Aware Networks Programming User Manual," Tech. Rep., 2015. [Online]. Available: https://www.mellanox.

com/related-docs/prod\textunderscoresoftware/RDMA\textunderscoreAware\textunderscoreProgramming\textunderscoreuser\textunderscoremanual.pdf [Accessed: 2020-09-22]

22. ntop, “Vannilla PF_Ring,” 2020. [Online]. Available: https://www.ntop.org/guides/pf\textunderscorering/vanilla.html [Accessed: 2020-12-02]

23. N. Mellanox, “NVIDIA Mellanox BlueField Data Processing Unit (DPU),” 2020. [Online]. Available: https://www.mellanox.com/files/doc-2020/pb-bluefield-dpu.pdf [Accessed: 2020-12-17]

24. Open-Virtual-Switch, “What is Open vSwitch,” 2020. [Online]. Available: https://docs.openvswitch.org/en/latest/intro/what-is-ovs/ [Accessed: 2020-11-28]

25. Mellanox-Technologies, “NVIDIA Mellanox BlueField SmartNIC for InfiniBand Ethernet,” 2020. [Online]. Available: https://www.mellanox.com/files/doc-2020/pb-bluefield-vpi-smart-nic.pdf [Accessed: 2020-11-28]

26. Mellanox, “BlueField SmartNIC Modes,” 2019. [Online]. Available: https://community.mellanox.com/s/article/BlueField-SmartNIC-Modes [Accessed: 2020-11-28]

27. Tcpdump, “Man Page of TCPDUMP,” 2020. [Online]. Available: https://www.tcpdump.org/manpages/tcpdump.1.html [Accessed: 2020-12-02]

28. N. Mellanox, “NVIDIA BlueField-2 DPU Data Center Infrastructure On A Chip,” 2020. [Online]. Available: https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/documents/datasheet-nvidia-bluefield-2-dpu.pdf [Accessed: 2020-11-01]

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704–0188*

1. REPORT DATE *(DD–MM–YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From — To)*
21–03–2021	Master's Thesis	Jun 2019 — Mar 2021

4. TITLE AND SUBTITLE Infiniband Network Monitoring: Challenges and Possibilities	**5a. CONTRACT NUMBER** **5b. GRANT NUMBER** **5c. PROGRAM ELEMENT NUMBER**
6. AUTHOR(S) Hintze, Kyle D., Capt, USAF	**5d. PROJECT NUMBER** 18G230 **5e. TASK NUMBER** **5f. WORK UNIT NUMBER**
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT-ENG-MS-21-M-048
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory 2241 Avionics Circle WPAFB OH 45433-7765 Attn: Steven Stokes COMM 937-528-8035 Email: steven.stokes@us.af.mil	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/RYWA **11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Within the realm of High Performance Computing, the InfiniBand Architecture is among the leading interconnects used today. Capable of providing high bandwidth and low latency, InfiniBand is finding applications outside the High Performance Computing domain. One of these is critical infrastructure, encompassing almost all essential sectors as the workforce becomes more connected. InfiniBand is not immune to security risks, as prior research has shown that common traffic analyzing tools cannot effectively monitor InfiniBand traffic transmitted between hosts, due to the kernel bypass nature of the IBA in conjunction with Remote Direct Memory Access operations. If Remote Direct Memory Access over Converged Ethernet is used instead, it is possible to restore traffic visibility in novel ways. This research shows that this approach, together with an InfiniBand capable adapter, allows common traffic analyzing tools to be used to monitor network traffic without unnecessarily sacrificing the bandwidth and performance of InfiniBand.

15. SUBJECT TERMS

InfiniBand Architecture, Network Monitoring, Cybersecurity

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Dr. Scott Graham, AFIT/ENG
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*
U	U	U	UU	83	(937) 255-6565 x4581; scott.graham@afit.edu