RDMA BASED IP ROUTING PROTOCOLS AND ROUTER ARCHITECTURE

A Thesis by

Nishanth Chinthalapani

B.Tech, Jawaharlal Nehru Technological University, India, 2006

Submitted to Department of Electrical and Computer Engineering
and faculty of the Graduate School of
Wichita State University
in partial fulfillment of
the requirements for the degree of
Master of Science

August 2008

RDMA BASED IP ROUTING PROTOCOLS AND ROUTER ARCHITECTURE

The following faculty has examined the final copy of this thesis for form and content, and recommends that it be accepted in partial fulfillment of the requirements for the degree of Master of Science with a major in Electrical and Computer Engineering.

_____

Ravi Pendse, Committee Chair

_____

M.E Sawan, Committee Member

_____

Krishna K Krishnan, Committee Member

DEDICATION

My parents, sister, and friends for their constant encouragement and support

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Ravi Pendse, for giving me an opportunity to do my Master's thesis under his esteemed and able guidance. I would also like to thank my committee for taking the time to work with me in this endeavor. I am thankful to Mr. Nagaraja Thanthry for his valuable advice, time and help in completing this thesis. Finally, I am forever indebted to my parents and family for supporting and encouraging me throughout my life to achieve higher goals.

ABSTRACT

The Ethernet technology has advanced from the era of fast Ethernet to the era of gigabit Ethernet. The gigabit routers currently available in the market are employing expensive hardware based implementations for improving the throughput [6], which makes the overall cost of the device prohibitively high.

In this thesis the author reviews the existing router architectures and routing protocols and critiques the shortcomings of the existing implementation. This thesis evaluates the drawbacks in the existing infrastructure and proposes an architecture that provides a solution based on the RDMA protocol. The proposed architecture uses the RDMA protocol for transferring the data payload from the ingress interface to the destination interface. In this research the author also presents an analytical mathematical model that can be used for calculating the delay incurred by a packet, memory utilization and CPU utilization for both architectures. The potential benefits by the use of RDMA protocol are also explained in detail in this thesis. The necessity for modifying the update packet structure in the existing implementation of RIP is discussed in detail. Packet payload handling in both architectures is compared and the advantages in the RDMA protocol based implementation are presented.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont.)

TABLE OF CONTENTS (Cont.)

Chapter **Page**

LIST OF FIGURES

LIST OF TABLES

Table Page

LIST OF ACRONYMS

TCP	Transmission Control Protocol
IP	Internet Protocol
RDMA	Remote Direct Memory Access
CPU	Central Processing Unit
I/O	Input and Output
NIC	Network Interface Controller
RNIC	RDMA Network Interface Controller
SQL	Structured Query Language
DDP	Direct Data Placement
ULP	Upper Layer Protocol
LLP	Lower layer Protocol
STag	Steering Tag
OS	Operating System
SCTP	Stream Control Transmission Protocol
IETF	Internet Engineering Task Force
RI	RNIC Interface
CQE	Completion Queue Element
WC	Work Completion
CQ	Completion Queue
SQ	Send Queue
RQ	Receive Queue
TPT	Translation Protection Table
SRQ	Shared Queue
WQ	Work Queue
QoS	Quality of Service

LIST OF ACRONYMS (Cont.)

RIP	Routing Information Protocol
IGRP	Interior Gateway Routing Protocol
OSPF	Open Shortest Path First
SNMP	Simple Network Management Protocol
PC	Personal Computer
TTL	Time to Live
MTU	Maximum Transmission Unit
SLA	Service Level Agreement
ASIC	Application Specific Integrated Circuit
IGP	Interior Gateway Protocol
ACK	Acknowledgement
SYNACK	Synchronized Acknowledgement
PDU	Protocol Data Unit
CMOS	Complimentary Metal-Oxide Semiconductor
DRAM	Dynamic Random Access Memory
UDP	User Datagram Protocol
DoD	Department of Defense

CHAPTER 1

INTRODUCTION

1.1 Introduction

The roots of the internet trace back to ARPANET, which started off as a project of DoD to build a communication network that can survive a nuclear attack. From a robust fully connected mesh network architecture, the internet has undergone numerous changes to become what it is today. The initial routers used in the ARPANET were quite similar in architecture to a multihomed personal computer, where most of the functions were implemented in software by a general purpose CPU. As the size of the network grew the implementation of the functions by the router in software created a bottleneck thus limiting the effective throughput achievable. A lot of research was done in order to improve the throughput and processing power of the router, which resulted in numerous improvements in the architecture of the router. A lot of research is being done in order to minimize the delay experienced by a packet, the CPU utilization and the memory usage. The implementation of critical routing and destination lookup functions in hardware is one of the key developments in the effort to minimize the delay experienced by the packet during transit.

The transfer of data payload among the interfaces within the router requires the processor to copy the payload into the CPU register and then into the buffer of the egress interface. This causes the memory bandwidth bottleneck since the processor power needs to be increased enormously to support the copy operations at a high speed. Apart from the delay the memory bandwidth limitation is a key factor that limits the throughput achievable by the router. The DMA mechanism used in the computers to solve the problem of memory bandwidth can be used in the routers to provide a solution to the same issue. The opening of the main memory of the

hosts to the network in combination with the DMA mechanism forms the nucleus of the RDMA protocol.

The transfer of data from one device to the other can be achieved either by the conventional TCP/IP transfer mechanism or using the RDMA enabled mechanism [1, 15]. The main difference between the data communication over TCP/IP and RDMA enabled procedure is the method in which the packets are copied between the main memories of the devices. TCP/IP method, which is the conventional method, requires the intervention of CPU for copy operations from one device to the other. On the contrary the RDMA enabled method copies the data packets from one device to another without requiring the intervention of CPU. The TCP/IP mechanism can use only the file I/O procedure, but the RDMA enabled mechanism can handle both file I/O and block I/O. Both mechanisms are explained in detail in the following sections.

The volume of traffic flowing through the modern day networks is approaching CPU capacity, if the traditional data transfer mechanisms are to be used the effective bandwidth usage on the network would be limited [4]. Earlier when the CPU power was greater than the network bandwidth, the traditional data transfer methods which required the intervention of the CPU were sufficient to meet the data request demands. Future technologies like the 10GB Ethernet would pose serious problems if the traditional methods continue to be used [4, 5]. RDMA is an attempt to provide a solution to the data transfer issues that will show up in the future.

1.2 TCP/IP communication mechanism

TCP/IP communication mechanism performs the data copy operation with the assistance of the processor. When an application wants to send the information to its counterpart on the remote peer the data is encapsulated appropriately and is retained in the application buffer. An interrupt is sent to the CPU requesting a data transfer to the remote peer. The CPU responds to the interrupt and copies the data from the application buffer to the processor memory. A session is established with the remote peer to initiate the data transfer from the local peer to the remote peer. The connection is established via a three way handshake mechanism [16]. The data is transferred to the remote peer where it is read by the CPU into the CPU memory and is transferred to the application memory from there. This procedure is to be repeated for every incoming packet.

1.2.1 Advantages of the Scheme

The TCP/IP communication mechanism is very simple, in terms of connection establishment and data transfer. The NIC cards available for the TCP/IP procedure are less expensive than the RNIC interfaces required for the RDMA enabled router. The standards for the implementation are well established and tested.

1.2.2 Disadvantages of this scheme

This adds a significant delay and consumes considerable memory and CPU resources. The CPU bottlenecks while using the TCP/IP copy mechanism limit the throughput of a device [1, 15, 4]. The longer the message the greater will be the load on the memory resources of the device. When the network bandwidth gets closer to the CPU processing power the bandwidth cannot be utilized effectively due to the bottleneck presented by the CPU [4].

1.3 RDMA enabled copy mechanism

RDMA allows the reading of main memory of one device by the other, thus avoiding a lot of latency, bandwidth overhead and CPU resources. The RDMA protocol supports zero copy mechanism where data can be written into the system memory without having to copy data from application buffer to system buffer and then into the application buffer via the remote peer's system memory [1, 4]. The memory of the devices is opened for access via the network. Once the connection has been established the data from the memory of one device can be transferred into the memory of another device directly without executing a kernel procedure call.

1.3.1 Advantages of this approach

RDMA also enables the applications to issue read and write requests to the adapters without having to go through the kernel. The requests can be performed in parallel to the other processes even during peak CPU utilization periods.

Taking out the CPU out of the copy mechanism would conserve a lot of CPU and memory resources which can be used for other critical applications, which would otherwise be stuck in the memory copy operations [1, 4].Traditional transactional databases like SQL, Oracle and DB2 have successfully achieved better results when using RDMA in place of traditional TCP/IP mechanism [4].

1.3.2 Disadvantages of this approach

Numerous security concerns are being raised on opening the main memory of a device on the network. Complicated security measures have to be incorporated to the approach to make it secure enough for system memory sharing on the network. The vendors are yet to reach a consensus regarding the design and implementation standards of a RDMA enabled NIC card;

RNIC in short [17, 22]. The API that is required for the RDMA enabled router is deemed to be more complicated than the existing TCP/IP stream interface [19].

1.4 Problem Description

Routers which are considered the most critical devices within a data network, follow the conventional TCP/IP data copy mechanism, which involves the CPU in the packet transfer procedure from the ingress interface to the egress interface. With the Ethernet technology transitioning from fast Ethernet to gigabit Ethernet and efforts being directed to launch the ten gigabit Ethernet technology the conventional router architectures relying on the TCP/IP copy mechanisms will not be able to efficiently utilize the available network bandwidth [4] due to numerous bottlenecks. Considering the developments in Ethernet technology, it is required to make necessary changes to the architecture of the router in order to support the increasing network speeds. In this work the author proposes an RDMA based network architecture which does not involve the CPU's participation in the data transfer mechanism. This approach would remove the bottlenecks that were present due to the involvement of the CPU in the data copy mechanism.

1.5 Organization of Thesis

In this thesis the author proposes modifications to the existing router architecture to extenuate the existing bottlenecks in high bandwidth data networks. The organization of the thesis is as follows. Chapter one gives an introduction to the conventional TCP/IP data communication and the RDMA enabled communication mechanism. Chapter two presents the literature survey, Chapter three presents the proposed modifications to the router architecture and the routing protocols, Chapter four presents a mathematical analysis of both the architectures

along with a detailed analysis of the results, and Chapter five concludes the report with a brief description regarding the future work possible in this area.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

Routers are considered to be the key component of any network; the architecture of the routers is an important feature that can affect the performance of any network. A router design needs to be efficient enough to perform the routing functions efficiently and to maintain accurate route information for the networks in the autonomous system. A router also needs to support network management functions such as generation of SNMP traps and syslog messages [11, 25]

The traditional architecture of a router was similar to that of the PC with numerous interfaces, which performed most of the functions in software on a general purpose processor. The traffic would be received on one or more of the interfaces, the routing functions would be performed and the traffic sent out on the relevant interface. As the internet expanded the maximum throughput achievable was limited by the processor and the main memory.

One of the important factors that contribute to the delay in the forwarding of the packet to the appropriate egress interface is the route table lookup [26]. Research was actively pursued for proposing more efficient route lookup algorithms, but this was not as fruitful as intended due to hardware limitations [12]. The workaround for this issue is the express forwarding mechanism, which is currently implemented in most routers. The route table lookup is performed for the first packet in the flow and the result is cached in the ingress interface that is receiving the stream of packets. This expedites the routing process of the subsequent packets towards the same destination. The cache entries are flushed out after the aging timer expires.

The drawback of this mechanism is that load balancing feature in the event of availability of multiple paths is lost. A more sophisticated implementation of route caching is the forwarding information base table (FIB). In this approach each interface has a copy of the current routing table so that route lookup can be performed locally instead of being performed by the central processing unit (CPU). This is more efficient when compared to the route caching in terms of reducing the delay incurred in lookup, but this does not solve the issue of additional burden on CPU for payload copy mechanism. This is because the CPU has to be involved in copying the payload from the ingress interface to the egress. The solution for this issue can be derived using a modified version of the DMA procedure employed in the modern computers. The subsequent sections explain the basic functions of a router as well as the various kinds of architectures in vogue.

2.2. Basic functions of a router

The basic functions of a router can be broadly classified as routing and packet forwarding [11]. The packet forwarding functions that are performed by the routers are

1. Header Validation: The headers of the packets received by the router are to be validated by the router to ensure that only the packets that meet the specifications are forwarded and the rest are dropped without further processing. The header checksum validation is an example of this function.
2. Packet lifetime control: The time for which a packet needs to be allowed to transit the networks should be controlled by the router else the packet would traverse the network endlessly wasting valuable resources. This is achieved by decrementing the TTL value in the packet header. When the packet traverses a router the value of the TTL is

decremented by one, eventually when the TTL value becomes zero the packet is dropped, thus preventing any possible routing loops in the network.

3. Checksum Recalculation: The checksum of a packet needs to be validated by each and every router through which it traverses through. The need for checksum recalculation arises because the TTL field in the header is decremented at every router it traverses through.
4. Route lookup: The destination IP address is used for searching the routing/forwarding table to determine the output port. The search will help the router determine if the packet is for the router or whether it needs to be sent to any particular output port (unicast) or a set of output ports (multicast).
5. Fragmentation: Many a times the size of the packet can be larger than the MTU of the link. This is when the packet needs to be fragmented based on the requirement, i.e. the MTU of the link.
6. Handling IP options: The router very rarely might need to handle packets that have special processing requirements as indicated by the options field in the IP header.
7. Packet Classification: The traffic engineering and QoS implementation requirements call for classifying the packets based on set parameters. The destination port, destination IP address and the DSCP values in the IP header are used by the router to classify the packet accordingly.

The functions mentioned above belong to the data plane there are a few functions that belong to the control plane and management plane hold equal importance. Routers implement various routing protocols such as OSPF, RIP and IGRP. For maintaining the peer relations between the routers they need to be sending the hello packets, and also the route update packets

which need to be processed and sent by the routers as well as received and processed by the routers. The router needs to be continuously monitored for smooth functionality; the use of SNMP traps is one of the options. If the SNMP traps are configured on the router, then the router needs to send the appropriate messages to the recipient.

2.3 Hardware components

A router can be considered as an interconnection of various interface cards which are controlled by specialized software executed by a central processor. From an architectural perspective the router is a collection of the following functional modules network interfaces, forwarding engine, queue manager, backplane.

1. Network interfaces: The network interfaces consists of many ports that provide the physical connection to various link types. The port can be considered as the entry and exit point for the physical link and is specific to a physical medium. Besides from housing the ports the network interface provides L2 and L3 processing logic by extracting the relevant headers and forwarding them to the lookup engine. The interface card also provides the encapsulation function to the outgoing packets.
2. Forwarding Engine: When the L3 header is sent to the forwarding engine, the forwarding engine decides as to where the packet needs to be forwarded depending on the values in the header.
3. Queue Manager: The queue manager provides enough buffers to temporarily store the packets when the outgoing link is over subscribed. When the buffers begin to overflow due to congestion in the network the queue manager selectively drops the packets based on the service level agreement (SLA).

4. Backplane: The connectivity among the interfaces is provided by the backplane so that the packet can be transferred to the appropriate egress interface. The legacy backplane architecture is a shared architecture where only one pair of interfaces can communicate at a given instance; however the architecture currently in use is the switched backplane architecture where multiple interfaces can communicate at any given point of time.
5. Route control processor: The route control processor is responsible for populating the routing table and processing updates to be sent to the neighbors. The updates that are sent by the neighbor are processed by the route control processor and the routing updates are used to keep the routing table up to date.

2.4 Types of router architectures

The kinds of architectures can be classified based on the implementation of the packet forwarding function. Based on the above mentioned criteria the architectural approach can be classified as [11]

1) Shared CPU architectures
2) Shared forwarding architecture
3) Shared nothing architecture

2.4.1 Shared CPU architectures

This architecture is similar to the architecture of the conventional computers. A shared backplane connects the multiple line cards and specialized software executed by a general purpose CPU acts as the centralized arbitrating authority. The functional modules such as the traffic manager, forwarding engine are implemented in software.

The route processor function, L2/L3 logic on the ingress and the egress interface is implemented part in software and part in hardware. All the interface cards share the CPU for the above specified functions.

When a packet is received on the ingress interface an interrupt is raised by the interface at the CPU. An interrupt handler copies the packet to the main memory of the CPU, where the CPU performs the route lookup for the egress port and then the packet is subject to queue management and traffic engineering policies. Finally it is written into the buffer of the relevant egress interface. In this architecture it can be observed that the packet traverses the shared backplane twice, once from the ingress interface buffer to the CPU memory and from the CPU memory to the interface buffer of the egress interface.

The main advantages of this architecture are simplicity and implementation. The disadvantages are bottleneck presented by the CPU and the lack of scalability of this architecture, also due to the shared nature of the backplane there would be bottlenecks in terms of the efficient utilization of the backplane bandwidth. It can also observed that most of the CPU cycles are being consumed for by the packet copy procedure; the issue here would be that the other critical functions of the router such as the generation of route updates, processing of the route updates might be starved.

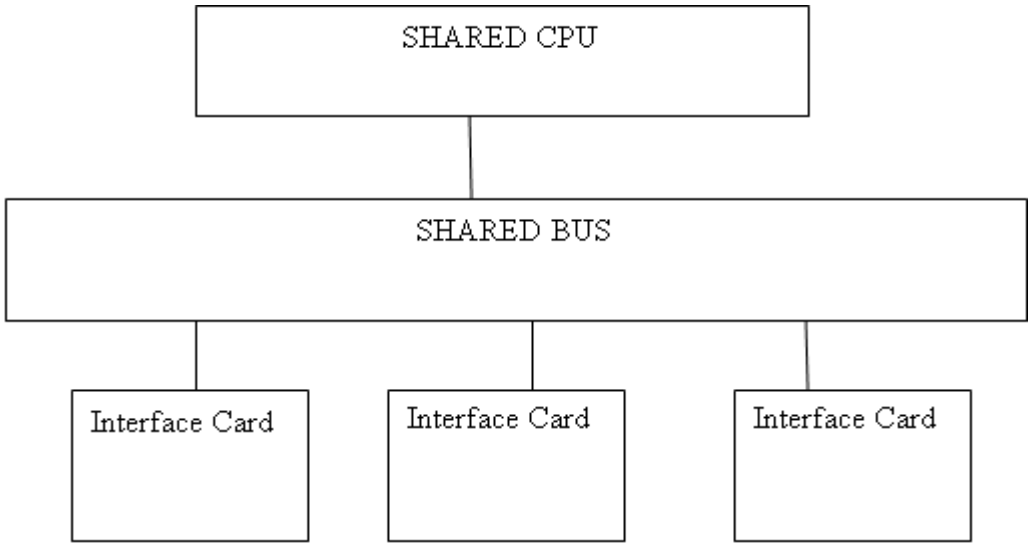

Figure 2.1 Shared CPU architecture

2.4.2 Shared forwarding Engine architecture

In the shared CPU architecture it has been that the major bottleneck is the shared CPU, since the packet has to be copied into the CPU memory and then the route lookup has to be performed [11]. If the shared forwarding engine architecture is used, the forwarding function can be offloaded to the dedicated hardware which is capable of performing the lookup and the forwarding would be much faster than the shared CPU architecture. Each forwarding engine would have a dedicated processor and would be specialized software for performing the route lookup; also the forwarding engine has on board memory which can be used for storing the forwarding table.

In this architecture the line cards are connected to each other via a backplane so that the packets can be transferred to one line card from the other. The shared forwarding engines are connected through a separate backplane called the forwarding backplane; this would help keeping the forwarding traffic separate from the regular traffic.

The shared forwarding engine architecture is used in the architecture of the gigabit routers. The separation of the forwarding and the regular traffic is the key to achieving high throughput, and this is exactly implemented in this architecture. When a packet arrives on the ingress card the IP header is examined to create a packet context, the packet context is sent to the forwarding engine for performing the lookup. Once the lookup has been performed the result is appended to the packet context and is sent back to the ingress line card. The ingress line card will decrement the TTL value and forward the packet to the appropriate egress interface. The packet would be buffered at the egress interface till the queue manager decides to forward the packet out of the interface.

The route processor module is responsible for maintaining the routing table and populating the forwarding engines with the routing tables, which will be referred to as forwarding tables. The main advantage of this architecture is that it can be scaled to architectures supporting higher throughput rates. This is possible because the forwarding and lookup function is offloaded the CPU and is implemented in dedicated hardware rather than software.

The main disadvantage of this architecture is that the key disadvantage of shared backplane limiting the throughput is carried over from the shared CPU architecture [11]. The bandwidth that is available using the shared backplane is very low which might limit the throughput of the router considerably. The workaround for this issue would be to use the switched backplane architecture to improve the bandwidth available for the packet transfers. This would also eliminate the need for a separate backplane for interconnecting the forwarding engines.

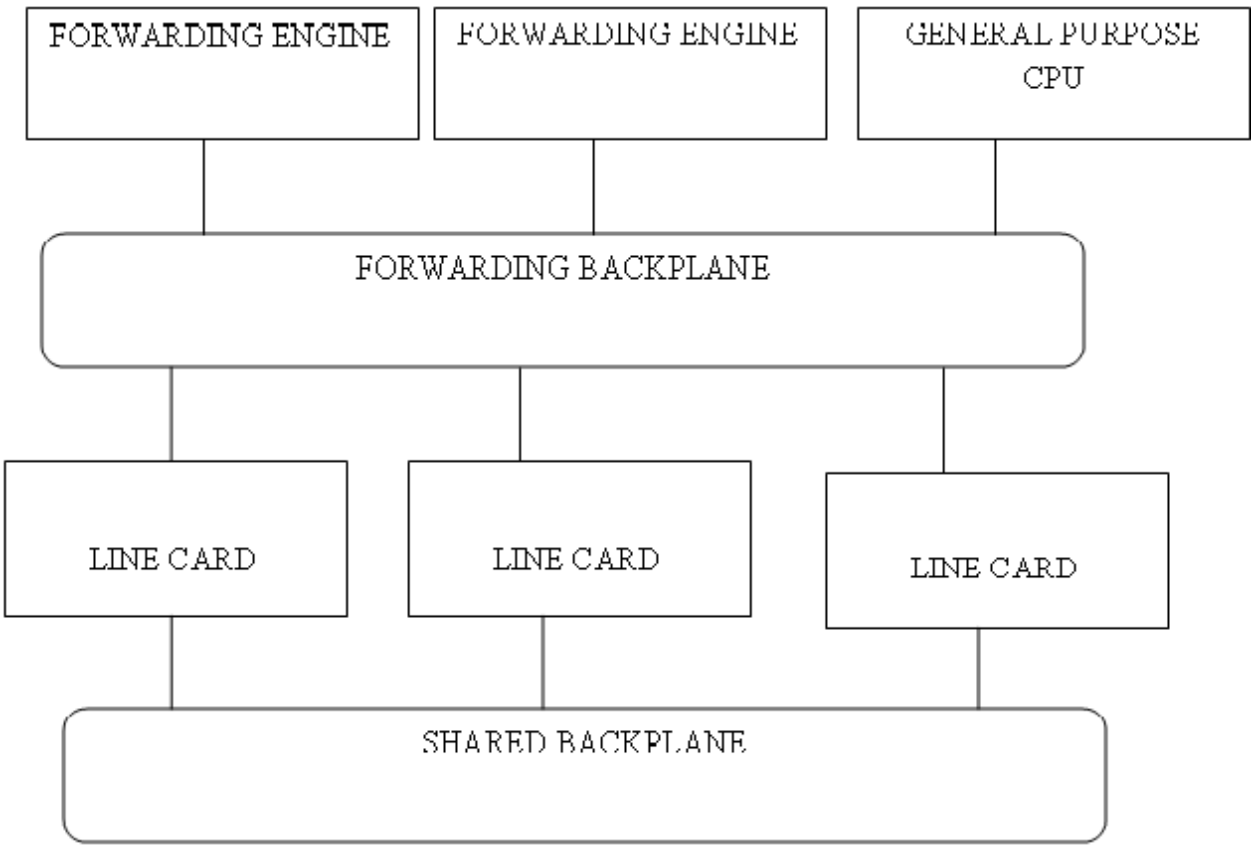

Figure 2.2 Shared forwarding engine

2.4.3 Shared nothing architecture

As mentioned in the earlier section the drawback with the shared forwarding engine architecture is that the shared backplane will limit the bandwidth available hence restricting the throughput. The usage of general purpose processors in the forwarding engine will further restrict the bandwidth due to its dependence on software to perform the tasks intended. If the forwarding engine is placed on the line card then the extra hop through the backplane to the forwarding engine will be eliminated, and using ASIC's to implement it would further reduce the time taken to lookup the egress interface based on the IP address.

The packet forwarding functions are offloaded onto the interface thus reducing the delay incurred by the packet considerably. As most of the functions are implemented in hardware the throughput of the line cards would further increase. The shared nothing architecture does not have anything in common; all the line cards have their own processing hardware embedded on board and interconnected with high speed links.

When a packet arrives at the interface, the packet context with the L2 and L3 logic is created and sent to the forwarding engine. The forwarding engine determines the appropriate egress interface and then the packet is output to the buffer of the egress interface via the switched backplane. The switched backplane is necessary for this architecture as the shared bandwidth would not be able to handle the throughput that is generated by the router.

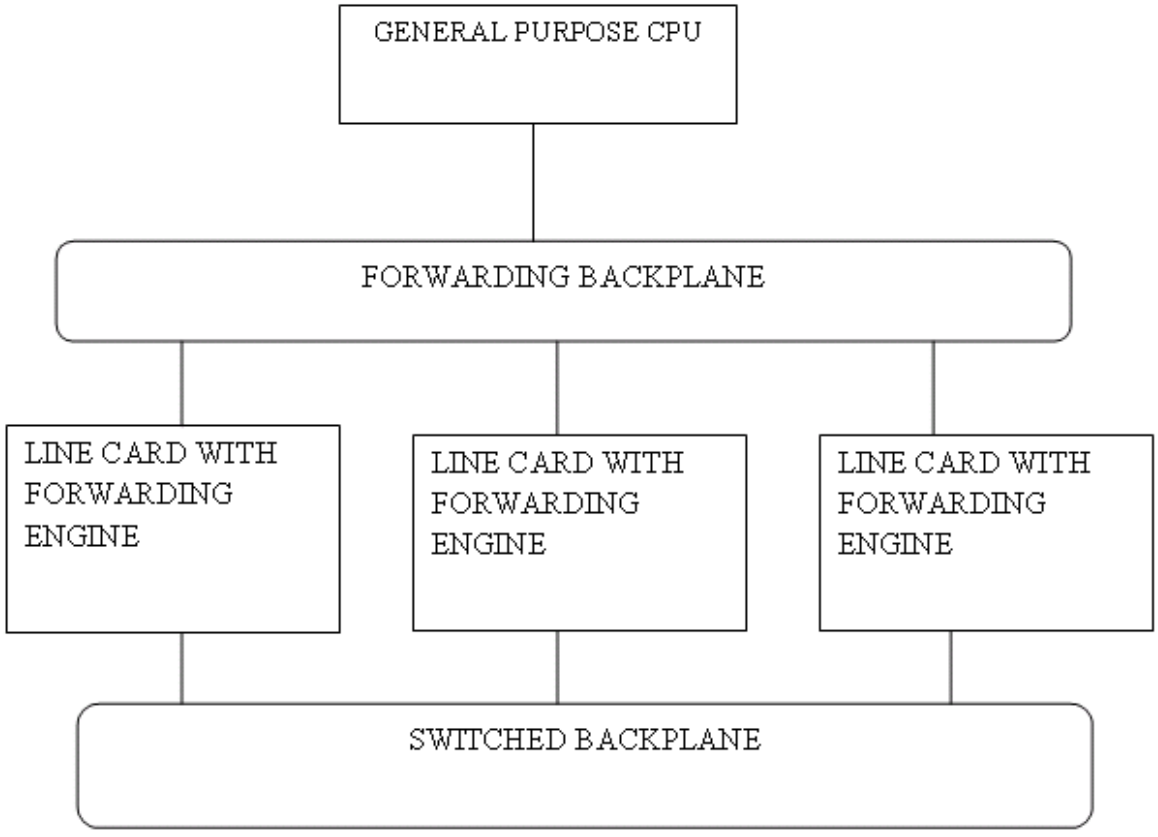

Figure 2.3 Shared nothing architecture

2.5 Introduction to DMA

DMA is a feature present in computers that enable the peripheral devices to read and write data from other peripheral devices without the intervention of the central processing unit. Disk drive controllers, graphic cards and other relatively slow peripheral I/O devices use the mechanism; this allows the CPU to process other requests while the data transfer mechanism is in progress instead of waiting in the idle state for the data transfer mechanism to finish, which would be the case if the programmed I/O mechanism was used. As the peripheral devices are comparatively slower when compared to the system memory, if the CPU has to copy each word from the peripheral devices to the main memory this would render the CPU unavailable for performing other requests. This would place a humungous burden on the CPU, thus affecting the overall system performance. The DMA controller copies the data word by word from the source I/O device to the destination I/O device. The DMA mechanism is initiated upon the execution of the DMA command by the CPU. After the DMA copy mechanism is completed by the controller it generates an interrupt informing the CPU regarding the completion of the copy operation.

2.6 Introduction to RDMA

RDMA is structured over the DDP layer, and it enables a device on the network to access the system memory of a remote peer. The applications are provided with memory read and write services. The applications can implement the kernel bypass procedure to bypass the kernel and communicate with the interface for fetching the data from the remote peer. The DDP offers two buffer models namely the tagged buffer model and the untagged buffer model [1]. The tagged buffer model need not be processed by the ULP layer before being written into the data sink

buffer, the untagged buffer model requires that the message be handed over to the remote peer ULP before being written into the data sink buffers [1].

2.7 Architectural goals of RDMA

The high level architectural goals for RDMA are as follows [1]

1. Provide a mechanism for the local peer to transfer data to a previously advertised buffer at the remote peer without requiring a copy operation. This is referred to as RDMA write data transfer operation.
2. Provide a mechanism for the local peer to retrieve data from a previously advertised buffer at the remote peer without requiring a data copy operation. This is referred to as RDMA read data transfer operation.
3. To provide a mechanism for the local peer to send data directly to an unadvertised buffer at the remote peer. This is referred to as the send operation.
4. To enable the local ULP to use a specific mechanism to signal the completion of the data transfer operations initiated previously.

5. The operations on a single RDMA stream will be delivered to the remote ULP in the order they were submitted to the local ULP.

6. To Provide RDMA capabilities to individual streams when multiple streams are supported by the LLP.

2.8 Overview of the services provided by the protocol

The RDMA protocol supports seven operations, but for the read operation each operation generates exactly one RDMA message. The RDMA operations currently available are (i) Send,

(ii) Send with invalidate, (iii) Send with solicited event, (iv) Send with solicited event and invalidate, (v) RDMA write, (vi) RDMA read, (vii) Terminate.

1. Send: The send operation uses a send message to transfer data from the data source to an unadvertised buffer in the data sink. This uses the DDP untagged buffer model to transfer the data to the data sink buffer.
2. Send with invalidate: The send with invalidate function is similar to the send message but the only difference is that the buffer which is identified by the value in the STag field will be invalidated and it cannot be accessed by any other device till it is revalidated by the ULP of the remote peer.
3. Send with solicited event: Send with solicited event is similar to the send operation, the data from the source is sent to the unadvertised buffer of the data sink. But upon completion the remote peer will send a solicited event message, provided the recipient is configured to generate one. The send with solicited event is used by the local host when the message being sent is an important one. Any message sent using the send with invalidate option will have to be handled by the ULP at the remote peer in an expedited manner.
4. Send with solicited event and invalidate: The send with solicited event is the send process where the local peer writes the data to the unadvertised buffer of the remote peer, the buffer identified by the value in the STag field will be invalidated; it cannot be accessed till the ULP of the remote peer revalidates the buffer. Upon the successful delivery of the message to the recipient and if the recipient is configured accordingly, a solicited event is generated.

5. RDMA write: The RDMA write mechanism transfers the message to the explicitly advertised buffer. The RDMA write mechanism uses the tagged DDP buffer model to transfer the ULP message to the data sink tagged buffer. The ULP at the remote peer advertises the STag, buffer length, tag offset using a ULP specific mechanism. The data sink buffer into which the ULP message was delivered still remains valid even after the message has been written into it. The buffer needs to be invalidated by the ULP of the local peer or the ULP of the remote peer. The buffer is invalidated using a send with invalidate or send with invalidate and solicited event. The buffer cannot be used again till the ULP of remote peer re enables it.
6. RDMA access read: The RDMA read operation uses the DDP tagged buffer model to transfer the message from the remote peer which is the data source to the local peer which in this case would be the data sink. The local peer which is the data sink initializes enables the buffer and advertises the STag, tag offset and the buffer length to the remote peer which is the data sink. The advertisement message semantics are ULP specific, i.e. it varies on how the implementation of the ULP will include the advertisement semantics in the kernel of the OS. The read request message uses the DDP untagged buffer model to advertise the STag, tag offset and the buffer length to the read request queue of the remote peer. Similar to the RDMA write operation the data source buffer would remain valid till the ULP of the local peer invalidates it or the remote peer invalidates it using a send with invalidate message. The data sink buffer can only be invalidated by the ULP of the data sink. In this research the author will be considering the send and RDMA write operations as the router would write the data payload to the next hop buffer and not reading from the buffer of the next hop router.

2.9 Layering of RDMA

The RDMA layer lies below the ULP and above the transport protocol layer, currently RDMA is supported only over TCP and not over UDP [1], which is another transport protocol. The MPA is used for layering; the complete functionality will be explained in the fore coming sections. In brief the MPA layer aids the DDP layer in the identification of the DDP header in the encapsulated payload.

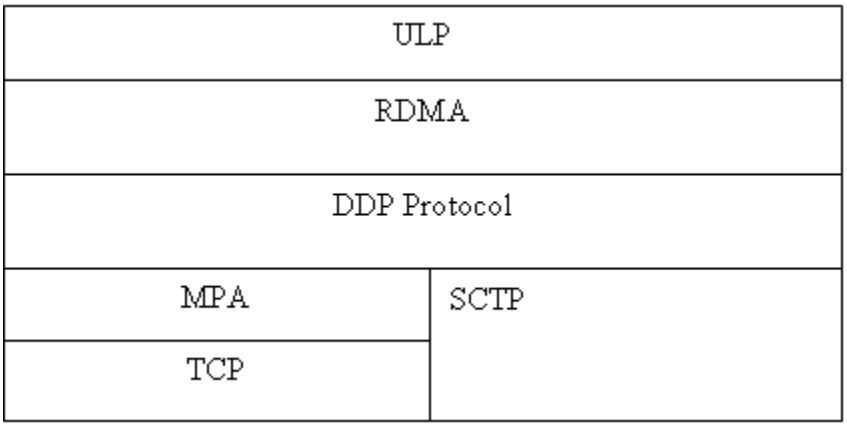

Figure 2.4 RDMAP layering

A combination of RDMA, MPA and DDP is referred to as iWARP suite which is placed over TCP or SCTP. The iWARP suite of protocols can be placed over any transport protocol available but currently the only transport protocol supported by iWARP is TCP and SCTP, currently work is in progress in order to extend support for UDP. The header layering of the iWARP protocols over TCP is shown as below.

A key point to remember here is that the DDP layer uses the STag and the tag offset values provided by the RDMA layer for placing the data in the data sink buffers [1, 24].

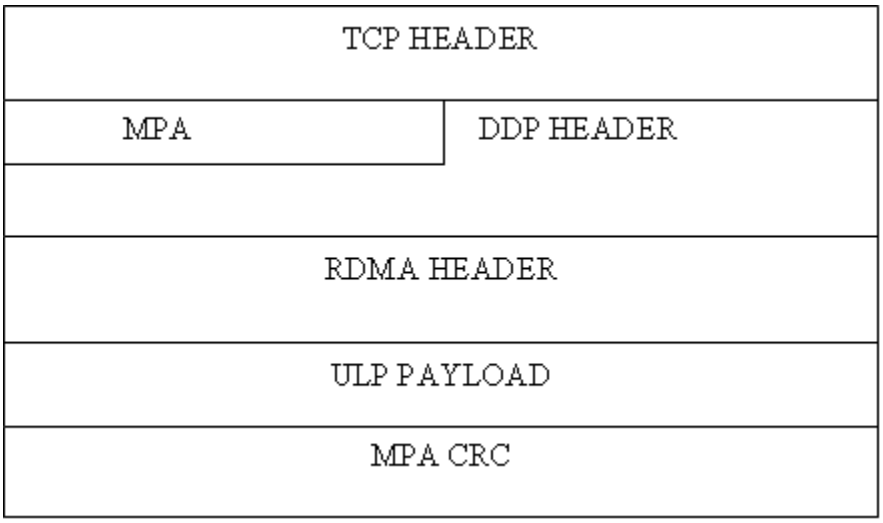

Figure 2.5 RDMA header alignments

2.10 Interactions of ULP and RDMA

The interaction mechanism between the ULP layer and the RDMA layer is very important because this determines the flow of data between the layers and this also defines the semantics that are to be followed while the data is being handed off between layers. This work considers the ULP layer and the RDMA layer interactions for the send operations and the RDMA write operation, since these are the only operations that have been extensively used.

2.10.1 ULP and RDMA interactions for send operations

For the send operation types the interaction between the ULP and the RDMA layers can be divided into the interactions at data source and the interactions at the data sink. They would be discussed in separate sub sections.

2.10.1.1 Interactions at the data source

At the data source the following information is passed to the RDMA layer by the ULP layer

1. ULP message length
2. ULP message

3. The type of send operation that would be used for the process, i.e. plain send operation or a variant of the send operation.
4. The invalidate STag if the send operation is a send with invalidate or a send with solicited event and invalidate.
5. A negotiated indication that would mark the completion of the send operation. This is an optional feature that is dependent on whether or not the data source and sink are configured so.

2.10.1.2 Interactions at data sink

The RDMA layer passes the following to the ULP, after the send operation is completed successfully.

1. ULP message length
2. ULP message
3. An event if the data sink is configured to generate one
4. An invalidate STag if the operation was send with invalidated or send with solicited event and invalidate.
5. If the operation had encountered an error and terminated prematurely, the data sink RDMA layer will pass the error message to the ULP of the data sink which will be sent to the RDMA later of the data source, which is passed on to the ULP of the data source.

2.10.2 Interaction of ULP with RDMA for a RDMA write operation

The interaction at both the data source and the data sink would be quite similar to the interactions for the send operation, but for the buffer tag information which is exchanged considering that the RDMA write is a tagged buffer operation. For the write operation the DDP

header is present instead of the RDMA header, but the DDP header contains a segment for the RDMA control variables [1].

2.10.2.1 Interactions at data source

The ULP passes the following information to the RDMA layer at the data source.

1. ULP message length
2. ULP message
3. Data sink STag
4. Data sink tagged offset
5. A pre-negotiated indication that would inform the data source regarding the successful completion of the operation. This is an optional feature that is dependent on whether or not the data source and sink are configured so.

2.10.2.2 Interactions at data sink

At the data sink the RDMA message is not handed over to the ULP but is directly placed in the ULP buffer, since the RDMA write operation uses a DDP tagged buffer operation. If the RDMA write operation terminates in error then the RDMA layer at the data sink passes on the terminate message to the ULP at the data sink. The ULP of the data sink passes the terminate message to the RDMA layer of the data source at the first chance available.

2.11 Header formats for various operation types

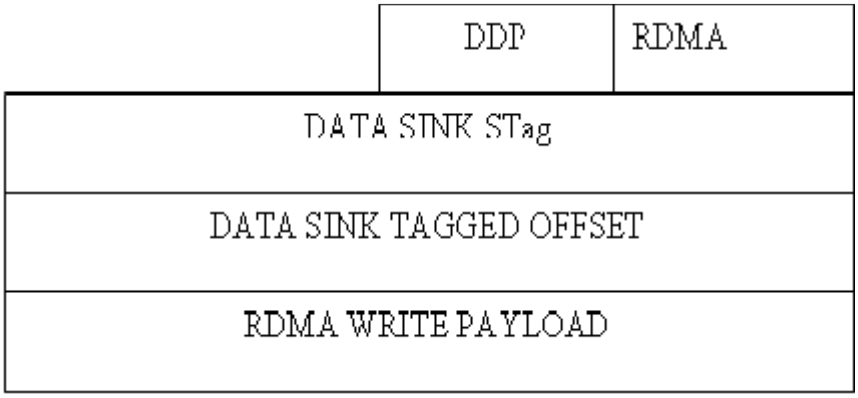

Figure 2.6 DDP header format for RDMA write

The RDMA control field and the DDP control field will be explained in the forthcoming paragraphs. The control fields have information regarding the opcode which dictates the operation that is to be performed with the payload.

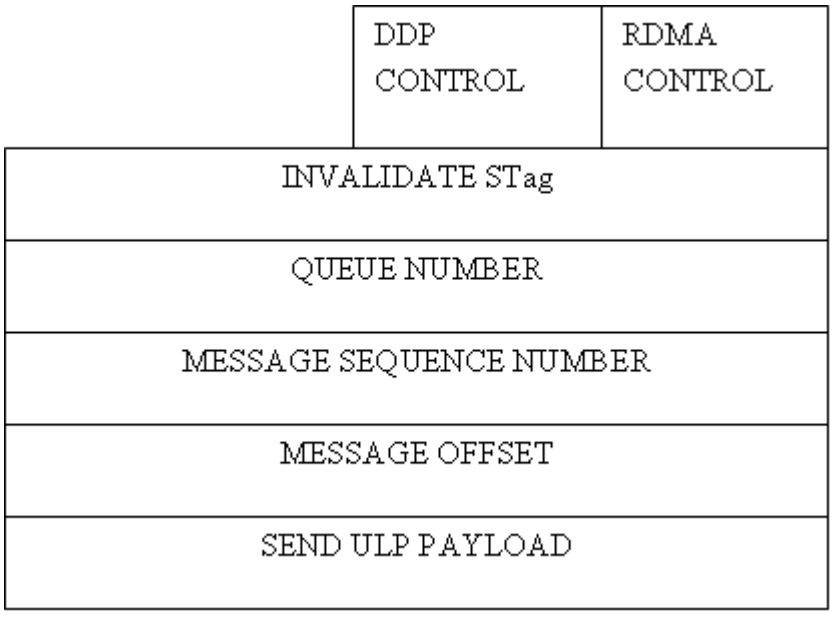

Figure 2.7 DDP header format for send operation

The invalidate field in the header is valid only if the send operation type is a send with invalidate or a send with solicited event and invalidate. The queue number is used for conveying the queue into which the message needs to be written into.

1. Queue 0 is used by RDMA at the data sink to handle the send operations.
2. Queue 1 is used for the RDMA read message.
3. Queue 2 is used for the RDMA terminate message.

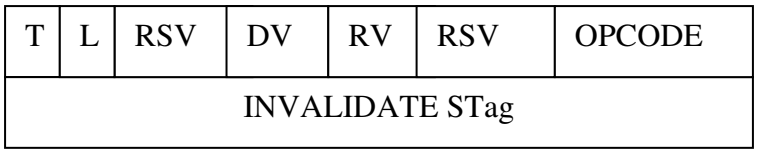

Figure 2.8 Control fields

Figure 2.8 depicts the control fields of DDP and RDMA respectively, which are 8 bits in width. The 'T' field value if 1 represents a tagged DDP buffer model used for RDMA read and RDMA write operations, for an untagged DDP buffer model message it is 0. The value of the 'L' field represents whether the length of the message is communicated between the DDP and the RDMA fields. Currently as per IETF specifications all the operations require that the length of the message has to be communicated between the DDP and the RDMA protocols hence the value of this field is always 1. The DV filed is used to denote the version of the DDP protocol and the RV field is used to denote the version of RDMA protocol. The opcode represents the type of operation that needs to be performed. The opcode for RDMA write and send operations are 0000b and 0011b respectively.

2.12 Data transfer using RDMA write

The RDMA write message references a tagged buffer; hence the data source RDMA requests the DDP layer to mark the message as tagged by substituting the value of 'T' with 1. The RDMA layer does not define as to how the buffers are used at the data sink; the ULP at the data sink should define how the buffers are utilized for the RDMA write operation. The ULP may choose to advertise a non-contiguous set of buffers and the task of writing the messages to the appropriate buffer needs to be handled by the DDP and the RDMA layer of the data sink.

The size of each RDMA message negotiated with the data sink depends on the size of the TCP segment parameter negotiated by the data source and the data sink [24]. The handling of buffer advertisement is supervised by the ULP of the data sink. The RDMA layer should submit messages to the DDP layer in the order they were received from the ULP [2]. The local peer needs to use multiple RDMA write messages to write into multiple tagged buffers [1].

2.13 RDMA stream management

The RDMA stream management consists of two stages, the RDMA stream initialization and RDMA stream termination.

2.13.1 RDMA stream initialization

The RDMA stream initialization takes place after the LLP stream has been created, the LLP stream refers to the transport protocol stream such as the TCP stream. After the LLP stream is initialized successfully it is transitioned into an RDMA enabled stream. Once a stream is initialized to RDMA enabled mode the stream should be transporting only RDMA messages till it is terminated. The number of RDMA writes waiting in the queue should be negotiated during the session transition from LLP to RDMA stream. The ULP should be determining the

acceptable number of messages that can be outstanding at a given point of time. The local ULP should negotiate with the remote ULP regarding the number of outstanding messages.

2.13.2 RDMA stream teardown

The RDMA stream can be terminated by ULP graceful termination, RDMA abortive termination and LLP abortive termination. If the ULP terminates the stream gracefully this is referred to as a ULP graceful termination, the ULP of either the data source or the data sink can initialize this teardown. After the ULP graceful termination either the source or the sink can initiate a LLP graceful termination. The RDMA abortive termination enables the RDMA protocol to issue a terminate message describing the reason for the termination.

The LLP abortive termination is due to an LLP error, the stream is torn down instantaneously without sending a terminate message to the RDMA layer. This kind of teardown is highly undesirable since the layers above LLP would have no knowledge regarding the reason for teardown and all the operations that have competed successfully will have to be assumed as completed in error even though it is not the case.

RDMA abortive termination is initiated when the stream encounters a RDMA error or an LLP error. It may not always be possible for the RDMA protocol of the side which encounters the error to the able to send a terminate message to the other end. If the error is encountered by the LLP and the LLP stream is torn down before the message is conveyed to the RDMA layer. Before the RDMA is able to send the terminate message to the opposite entity or the stream is terminated before the LLP of the end which has encountered the error can convey it to the RDMA layer of the opposite end.

2.14 Error management in RDMA

There are no inbuilt RDMA or DDP error management functions as of now. Under ideal condition when things are working fine then the message delivery to the remote peer is guaranteed by the LLP but when an error surfaces at the RDMA or the LLP layer then the stream should be terminated abortively.

The main causes for these errors are poor programming at the ULP layer, when an error is encountered at the RDMA or the LLP layer it is less stressful to terminate the stream rather than to try and maintain it. This can be said because it is always complex to troubleshoot the cause for the error while keeping the stream active.

The LLP layer has to tear down the applicable stream when it encounters an error; if the LLP cannot tear down an individual stream then it has to mark the stream as an erroneous stream and should not allow further flow of information on the stream. When all the LLP streams are marked as erroneous the LLP has to tear down the stream. When a stream is abortively terminated the RDMA messages that were previously sent to the remote peer should be deemed as completed with an error, however RDMA read is an exception to this rule.

2.15 Introduction to RNIC verbs

The RNIC verbs are a representation of the behavior of the RDMA hardware, software and firmware to the host that has the RDMA services available on it. The RNIC verbs specify the services that can be provided to the host by the RDMA protocol [2, 12]. The complete behavioral description of a system can be provided by using the RDMA interface (RI) and RDMA verbs.

The RNIC interface provides the entire semantics for the services made available by an RNIC that is compliant with the RNIC verb. A verb is an operation that can be performed by an interface [2].

2.16 Overview of RNIC model

The Application that uses the services of the RNIC verbs is referred to as the verb consumer. The communication between the peer nodes is through verbs, the following functions are managed via the verb. The connections, memory and queue access is managed by the RNIC verbs. The submission and retrieval of work requests are also handled by the RNIC verbs. The RNIC verbs perform the tasks requested by the consumer.

The RNIC interface (RI) acts as a middleware between the consumer and the RNIC verb. The RI may consist of hardware, software and firmware. The RNIC interface submits the work requests and retrieves the completed work requests on behalf of the consumer. The work requests submitted on behalf of the consumer are converted to work queue elements. Similarly the completion queue elements (CQE) are converted into work completion (WC).

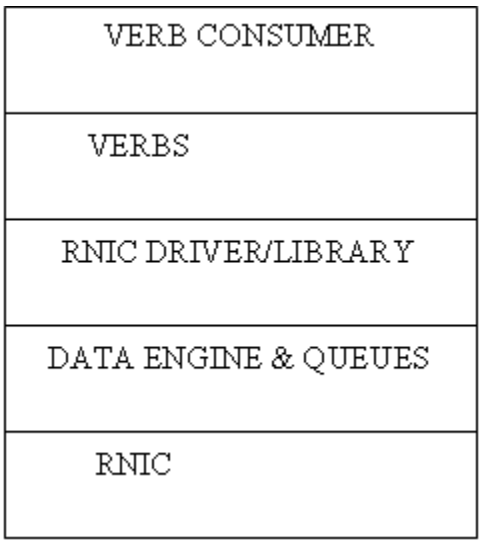

Figure 2.9 RNIC model

The data engine consists of the queues and the translation tables. The data engine can be pictorially represented as in figure 2.8. SQ refers to send queue, RQ refers to receive queue, SRQ refers to shared queue, and CQ refers to completion queue. The queue pair context refers to the context that exists for a SQ and RQ pair. The translation table contains the information regarding the offset to physical buffer translation.

The send queue is used for the outgoing messages, i.e. the outbound operations are intended to use the send queue and the receive queue is used for the incoming tasks. The send queue and the receive queue together are referred to as a queue pair. If an application sends and receives information simultaneously then it requires a dedicated queue pair, for the uninterrupted processing of work requests in either direction.

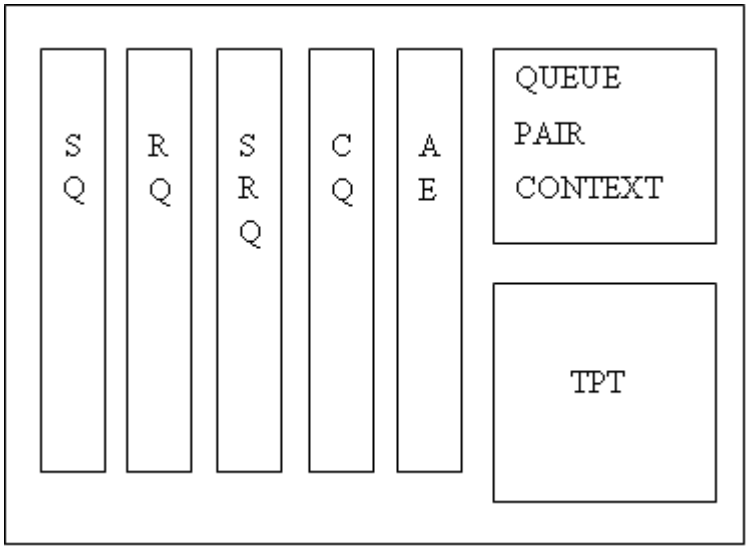

Figure2.10 Representation of data engine

2.17 Work submission model

The consumer uses the send queue (SQ) for all outbound send/write messages. For the inbound send/write messages the consumer uses the receive queue (RQ) or the shared queue (SRQ). A send or a write operation in the outbound direction along with the required information such as the STag, tag offset, buffer length is referred to as a work request.

The types of work requests are send, send with invalidate, send with solicited event and invalidate, RDMA write, RDMA read. The RNIC interface (RI) converts the work requests (WR) to work queue elements (WQE), the WQE is submitted to the work queue (WQ). After the WQE is submitted to the WQ the control is returned back to the consumer, the RI cannot modify the WR once the control is returned to the consumer.

2.18 Completion model

Consumer sets the completion event handler; this completion event handler is responsible for handling the post completion events on behalf of the consumer. Considering the WR in the WQE is completed and the completion of the task is signaled to the RI, a CQE is generated.

For a receive queue (RQ) the procedure is same, when the operation is completed the WQE translates into a CQE. This is when a shared queue is not associated with the RQ. But when a shared queue (SRQ) is associated with the RQ then the RNIC would directly convert the WQE and put it in the completion queue (CQ). Upon the successful completion of the request the event handler is notified regarding the completion which in turn notifies the verb consumer.

For the traditional send queue (SQ) models the consumer should post the number of expected WR that need to be handled per connection. Hence the consumer at the data sink can go for either posting the number of WR's equal to the bandwidth of the connection or enable the

data flow control so that the data source would back off till the WR's in the queue are attended to. Both of these approaches are not efficient, the first option might allocate more resources than needed and the second option might induce unwanted delay in the processing of the incoming WR's. The workaround for this issue is implementing the shared queue model [11]. The shared queue model is inspired by the architecture of few storage systems, where a shared pool of buffers is shared amongst a set of transport connections.

The shared queue concept is used in the following manner; the SRQ is associated with a QP on the RNIC. When the WR's are received on the SRQ associated with a QP the WQE's are transferred from the SRQ to the RQ associated with the queue pair.

2.19 Conventional Routing mechanism

The routing procedure in a network is fundamentally dependant on the IP address [25, 11]. The routers are primarily concerned about the logical connectivity between the networks. The routers transport the packet from one network to the other, the transfer of the packet between individual hosts is not handled by the routers. The destination IP address consists of network portion and the host portion. The network mask helps the router determine the network portion and the host portion of the IP address.

When a packet arrives on the ingress interface the packet might need to be buffered depending on the existing network conditions, i.e. the amount of traffic that is flowing through the network. Then the process of destination lookup begins when a packet context is created for the packet by the CPU, the packet context is constructed from the information present in the packet header such as the destination IP address, source IP address, type of packet [10, 11]. The packet context is used to perform a packet lookup; the packet lookup can either be done by the CPU or by the forwarding engine on the interface. If the router has a specific forwarding engine

to perform the route lookup the packet context is forwarded to the forwarding engine, else the packet context is used by the CPU to perform the route lookup and find the closest match to the destination address in picture. Later the packet context can also be used for traffic engineering and QoS implementations [10].

After the network portion of the destination is determined the router searches its routing table for the longest match for the network portion [25]. After the longest match is found the packet is written into the buffer of the appropriate egress interface. The routing tables are maintained in the memory of the router. The routing tables can be learnt dynamically by the router or they can be manually configured on the router. When a router is first powered up it does not know anything except its directly connected interfaces.

There are two kinds of routing protocols the distance vector protocols and the link state distance vector protocols [25, 11]. The distance vector protocols use the hop count as their metric they use it as a benchmark to determine the preference that can be given to a particular route over another route to the same destination. The distance vector algorithms advertise the information in their routing table to its neighbors only where as the link state protocols advertise their directly connected network information throughout the network. RIP and IGRP are examples of distance vector algorithms where as OSPF is an example for link state vector algorithms.

The router may store the routing information in the cache so that the packets to follow in the stream can be fast switched to the appropriate egress interface [11]. The first packet of a stream is analyzed and a lookup is performed and then the packets to follow are sent out the exact same egress interface that the first packet was sent out on. The information regarding the

egress interface to be taken to reach a particular network is stored in the cache on the ingress interface on which the stream reaches the router [25].

CHAPTER 3

PROPOSED CHANGES TO ROUTER ARCHITECTURE AND IP ROUTING PROTOCOLS

3.1 Functional overview of router architecture

A router is a layer 3 device which looks up the information in the header of an IP packet and forwards the packet to the appropriate egress interface based on the information present in the header. From an architectural view point the router can be considered to be an interconnection of interface cards. From a functional perspective a router is composed of interfaces, line cards, forwarding engine, queue manager and backplane. In this research the author proposes the architecture for a RDMA enabled router.

3.2 Line card architecture

The major component of the router that is to be emphasized on is the NIC card because it is the functional element of the router which implements the L2 and L3 logic. The conventional NIC card will not match the specifications for the implementation of RDMA over TCP. The modifications proposed in the previous chapter would require an interface card which can process RDMA, DDP headers along with the conventional TCP and IP headers.

The major advantage of using RDMA over TCP/IP is that it can handle block level and file level I/O simultaneously. This makes it suitable for almost all the applications available in the present day scenario. In the conventional NIC cards the TCP/IP stack is implemented in software by the kernel of the device operating system. This places additional burden on the CPU of the router since the connection establishment procedure would be handled by the operating system kernel, which requires additional CPU cycles.

The functions of the TCP stack such as connection establishment and data stream segmentation can be implemented in hardware. This is required since the TCP connection establishment could create bottlenecks as the number of connection establishments are more in the RDMA enabled router than the conventional router.

The major advantage of using RDMA over TCP/IP is that it can handle block level and file level I/O. This makes it suitable for almost all the applications available in the present day scenario. The interface cards should be capable of handling DDP segments and also should be capable of comprehending the information regarding the MPA marker and the DDP header, which is used to determine the end of the header and the destination buffer information respectively.

3.2.1 Functional Specifications of the line card

TCP uses 8 bit wide fields for data communication but the size of the payload is undetermined, on the contrary MPA uses fixed size PDU's. TCP segments are to be framed so that they comply with the MPA specifications, the MPA frames the PDU's in such that each TCP segment will have one FPDU. The step wise procedure for the transmission of data is as follows

1) The RDMA layer converts the RDMA write requests to DDP messages.
2) The DDP layer fragments the DDP segments handed over by the RDMA layer based on the buffer length advertised by the remote peer.
3) The DDP segments are handed over to the MPA layer, where they are appended with header and CRC, as well as inserted with markers. From this point these segments are referred to as FPDU's.
4) The MPA delivers the FPDU's to the TCP layer where they are fragmented so that each TCP segment contains exactly one FPDU. The segments are appended with the required

TCP header information and are handed over to the IP layer. To make the task of fragmentation simple the ULP defines the maximum size of the RDMA write message equal to the maximum TCP fragment size.

5) At the IP layer the required outing information is appended to the TCP segment from where on it is referred to as an IP packet.
6) The IP packet leaves the interface and is routed over to the destination as specified in the header.

When the packet is received at the data sink the IP header is stripped off as it is no longer required.

1) The TCP segments are reassembled based on the sequence number values present in the respective fields. It is important to remember that the MPA markers still exist with the FPDU.
2) The MPA markers still remain attached to the FPDU since they are required by the DDP to determine the end point of the DDP header.
3) The data sink buffer information in the DDP header can be used to write the data into the buffers directly.
4) The key point in here is that the DDP segments are not handed over to the RDMA layer to be written into the data sink buffers. They are directly written into the data sink buffers, per the information provided in the DDP header.

3.2.2 Hardware functionalities

From the hardware perspective the architecture of the RNIC should be similar to that of the Line card, i.e. most of the critical control operations should be implemented in hardware and should be kept local to the interface card. The interface should be capable of performing L2 and

L3 packet processing, similar to the conventional NIC cards. Along with all of this it should also be capable of processing the DDP header, this is very critical in terms of data placement as all the information is present in the DDP header.

Forwarding Engine functionality can be implemented on the line card in hardware since the incoming packets contain the egress interface information, hence the forwarding engine need not perform any lookup regarding the egress interface based on the destination. The forwarding engine also has to maintain a table of the session identifiers in order to forward the packet received to the appropriate destination. The reasons that justify the moving of the forwarding engine onto the interface are as follows,

1) To minimize the number of times the packet traverses the backplane.
2) The DDP header contains all the required information regarding the egress interface. During the session establishment the STag, tag offset and the buffer length are advertised by the destination interface. The information has to pass through the ingress interface via which the session establishment procedure was initiated by a neighboring router; the information is also temporarily stored on the ingress interface. This information is cached only while the session is in progress, after the session is terminated by either the data source or data sink the information is flushed out. It should have the capability to cache the information of all the parallel sessions in progress. The identifier field in the TCP header can be used to differentiate multiple sessions.

The ingress interface should populate the forwarding table based on the buffer advertisements being broadcasted by the egress interface to the egress interface of the upstream router. There are two methods by which the table can be populated

a) By inspecting all the packets during the initial phases of the session establishment the device can determine the buffers that are being advertised by the data sink. This method of populating the forwarding is not efficient because the interface has to inspect all the packets that are being exchanged by the source and the destination during the session establishment.

b) Another method would be to have the data sink forward a copy of the advertisements to the forwarding engine of the ingress interface, thus populating the forwarding table. The session control should populate the forwarding table of the interface accordingly when a session has been successfully initialized. After the session being terminated the entries in the cache would be flushed out, this responsibility is carried out by the session control procedure. The session control procedure can be implemented in software or in hardware based on the requirements of a platform.

The session control mechanism would oversee the session establishment procedure between various interfaces, the information regarding the buffers that are being advertised by the data sink to the data source.

The buffer advertisements by the data sink should be propagated via the session control mechanism only, this would allow the session control procedure to record the buffer advertisements and then populate the forwarding engine of the intermediate interface.

Traffic Manager is responsible for implementing the service level agreements. The quality of service implementations can be employed on the ingress as well the egress interfaces. The inclusion of the traffic manager/queue on the interface is much required to expedite the handling of the packets. The OS would be passing the required instructions to the traffic manager

based on the configuration input by the user. The packets would be selectively dropped by the manager based on the congestion levels on the backplane. Hence it becomes extremely important to have a backplane designed that would provide decent bandwidth to keep up with the line rate of the interfaces. The traffic manager would be responsible for implementing the QoS policies on the data streams that are passing through the router. As mentioned earlier for minimizing the overall cost of the system the author would be opting for shared buffer rather than dedicated buffers. The packets would be buffered and then would be inspected by the traffic manager.

Shared buffer: The advances in the CMOS technology have increased the capacity of the on chip memory and have also reduced its cost to a great extent. However it is always advisable to use on chip memory efficiently to keep the overall cost of the system in an affordable range. Memory allocation from the shared buffer will be supervised by a kernel process that would be running in the background, when any of the interfaces have to use a portion of the memory as a buffer they require sending a memory allocation request to the process. The process would check the shared memory block for available segments and would allot the segments to the requesting interfaces, however in this method it cannot be guaranteed that the interface would be allotted the amount of memory requested. It all depends on the amount of free memory available.

As of now prioritization mechanism is not available in most of the RTOS kernels. The most common circumstance under which the interfaces request for buffers was when the backplane would get congested. But this is not completely true since in the modern day devices it is absolutely critical to implement QoS and ensure that the SLA's are always reached. For implementing QoS and SLA the respective fields in the IP header need to be examined by the traffic/queue manager if this is to be achieved in real time as the packets are being sent to the O/P interface to be transmitted this would require high amounts of on chip memory, instead the

packets can be buffered in the shared buffers and suitable decisions can be made based on the information in the header.

Backplane can be considered as the backbone for any network device, since it provides the interconnection between the entry and exit points for the traffic. With the growing demand for bandwidth and speed it becomes all the more critical to design the backplane to provide wire speed transfer. The most common backplane implementation schemes are the shared backplane, switched backplane.

TCP offload engine is the implementation of TCP stack in hardware. For the proposed architecture this is required since the TCP sessions that have to be established are more when compared to the conventional router. Hence if the TCP stack is implemented in software this would place a lot of burden on the CPU, to avoid this TCP stack is implemented in hardware on the interface.

3.3 Router backplane architecture

The shared backplane architecture can lead to congestion during periods of high burst of traffic since the cumulative bandwidth required by all the interfaces cannot be provided by the backplane. The second reason is that the designing of high speed shared bus is very expensive and very difficult from an engineering perspective. The electrical loading and the density of the electrical connecters required makes it a gigantic task from the engineering perspective. With the technology available as of today the maximum bandwidth available would be 20 GB/s. For the high end routers with line rate of 2.4 Gbps, 40Gbps of bandwidth is required at a minimum.

The switched back plane architectures are capable of providing high bandwidths as the connections from the line cards to the central switch regulates the connection that is capable of transferring data from the ingress line card to the egress line card. The connections are made by

closing the circuit at the required places; this effectively translates to creating a point to point link from the ingress to the egress card. The CMOS technology has allowed the creation of 1Gbps chip to chip serial links. In the next few years the link capacity would go up to 4Gbps. An additional advantage regarding the chip to chip serial link is that issues such as electromagnetic interference would be very minimal. Crossbar switched architecture can support multiple connections simultaneously, which makes it non-blocking internally. The concept of parallelism is also possible in the switched crossbar architecture which is not possible in the shared bus architecture. If there are multiple crossbar switches in the backplane then they are interconnected to a centralized scheduler that provisions the connections to one line card at any given point of time.

Due to the bandwidth requirements for the high performance RDMA routers the author prefers the switched backplane architecture rather than the conventional shared backplane architecture. The switched backplane architecture provides the platform enough bandwidth to ensure that the throughput is not jeopardized by potential congestions in the backplane.

A schematic representation of the proposed architecture of a router with RDMA capabilities is depicted in figure 3.1.

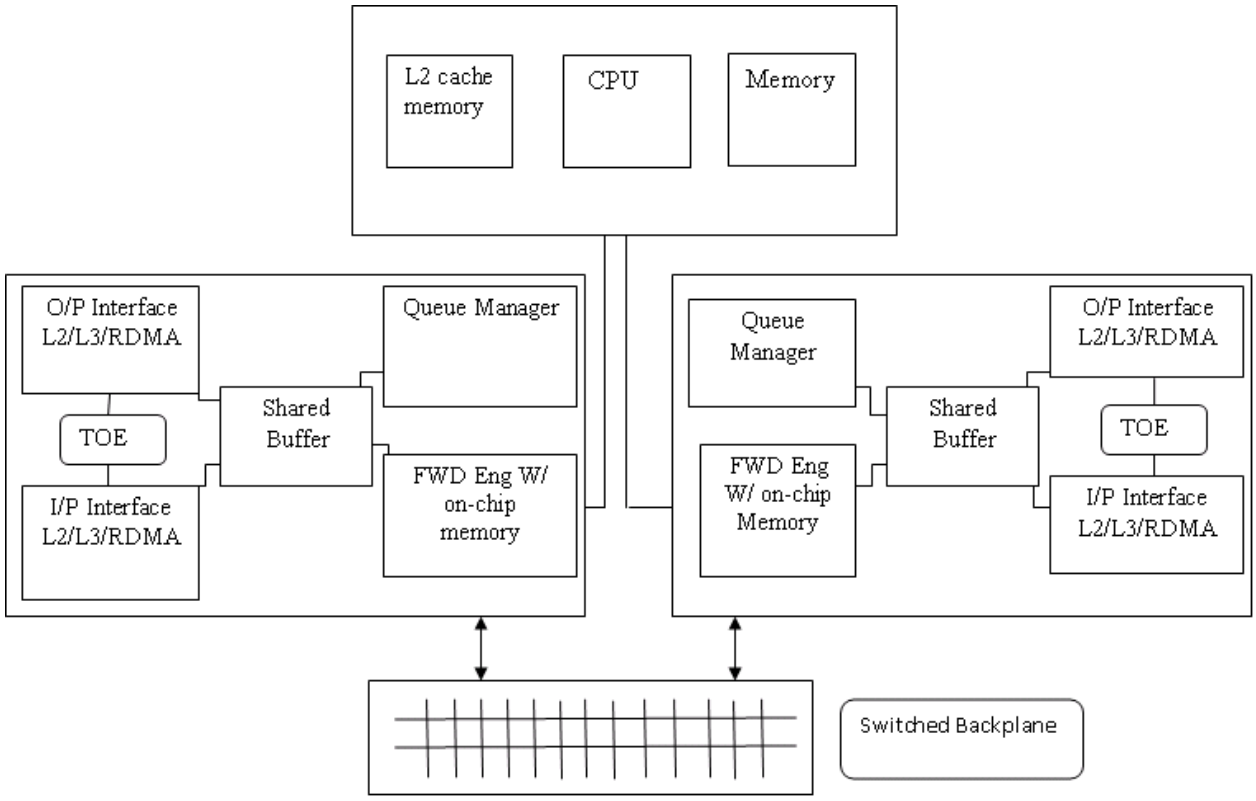

Figure 3.1 Proposed architecture for an RDMA enabled router

3.4 RDMA stream management between interfaces

For the RDMA stream to be established it is to be ensured that the TCP stream has successfully initialized prior to it. The TCP stream under usual conditions would be initialized between a client and a server. But here it is required that the TCP session be established between two RNIC interfaces. The task of aiding the TCP session establishment can be handed over to

1. The operating system of the router.
2. The TCP offload engine sitting on the interface.

The choice of implementation is left to the system architect. The decision is governed by design and economic specifications. The interfaces can be deemed ready to establish an RDMA

session as soon as the SYN ACK is received from the other end. The TCP session automatically transits to RDMA enabled mode after the connection is completely established.

The ULP of both devices negotiate on the number of outstanding outbound RDMA messages, this negotiation can be done separately from the TCP windowing specifications or the value negotiated by the TCP layer. If the RDMA negotiates the outstanding messages in the queue different from the TCP windowing segment then there are pros and cons for this method. The delay in the packet transit time would be incremented as the negotiation procedure would take certain amount of time. The second disadvantage of this method is that if the negotiated value of the RDMA outstanding messages is more than that of the TCP windowing segment then it might cause LLP errors in the transfer process and the stream might be terminated using the non-graceful termination procedure.

All the above mentioned short comings can be reduced by configuring the RDMA outstanding message size equal to TCP windowing size. But if appropriate measures are taken at the application layer to prevent occurrence of errors then the system designer can go for either of the options.

During the session establishment the values of tag offset and the starting value of the STag should be provided to the local peer by the remote peer. To summarize all the parameters that are to be available at the local peer before the session can be determined as completely establishes are

1) TCP windowing size
2) Outstanding RDMA write messages acceptable by either peers.
3) Tag offset and the starting value of the STag.

Stream Teardown: The RDMA stream can be terminated in three ways ULP graceful termination, RDMAP abortive termination, LLP abortive termination.

1. The ULP graceful termination is initiated by the ULP upon successful completion of all the RDMA write requests. When the outbound queue at the local peer is empty this indicates that all the RDMA write requests have been completed successfully. This is when the data source initiates the session termination sequence.
2. LLP abortive termination is caused when an error creeps up with the TCP session between the local and remote peers. This causes the RDMA session to be torn down without any messages being sent to either peer. This method of teardown is highly undesirable.
3. RDMAP abortive termination is invoked when either an RDMAP error surfaces or an LLP error is reported by the LLP to the RDMAP layer. This error may or may not result in a loss of data. At times the data loss may occur when an error is detected by the remote peer but the remote peer does not get a chance to send a terminate message to the local peer. When an error surfaces at the LLP or the RDMA layer of the remote peer, the remote peer tries sending a terminate message at the earliest opportunity. But it might not be always possible for the remote peer to send the error message before it terminates the session. Hence the messages that were completed prior to the termination have to be retransmitted as they are assumed to be completed in error. Currently RDMA does not have any inbuilt error recovery procedures.

3.5 Writing the message to the destination buffer

The data source transfers the data packets into the data sink using the RDMA write operation. The ULP at the data sink has to enable the data sink tagged buffer and advertise the buffer length, tagged offset, and STag. The information is propagated to the data source using a ULP specific mechanism, meaning that the firmware on the RNIC card should handle the negotiation. This uses the tagged buffer mechanism; hence the intervention of the ULP is not required. The data packets can be directly written into the data sink buffers. The data sink buffers remain valid until the ULP of either the local or the remote peer invalidates it using a send with invalidate option.

1) Upon the receipt of the data packet the destination IP address is looked up by the RNIC card against the forwarding table to determine the egress interface. After the egress interface is determined an RDMA session is established with the egress interface. The IP packet received is suitably encapsulated using the RDMA header. The resulting packet with the respective fields populated would be as follows

<table>
<tr><td rowspan="2">MPA (8 bytes)</td><td>T
(1)</td><td>L
(1)</td><td>000b
DV(2)</td><td>RV</td><td>Resv</td><td>010b
Opcode</td></tr>
<tr><td colspan="6">Data sink STag</td></tr>
<tr><td colspan="7">Data sink Tagged Offset</td></tr>
<tr><td colspan="7">RDMA write ULP Payload (IP packet)</td></tr>
</table>

Figure 3.2 RDMA write header format

2) The placement of the data packets in the buffer of the data sink is different from that of the send with invalidate. There the packets are handled by ULP since packet transfer is done

using the untagged buffer model where as in here tagged buffer model is used, hence the data packets can be written to the data sink buffers directly.

3) The lookup is performed by the forwarding engine, whose forwarding table is pre-populated with the information regarding the destination buffers into which the data payload has to be delivered.

4) After the lookup is performed the pay load is delivered into the data sink buffer. After each data sink buffer is populated a message is generated to indicate the successful completion of the task to the data source. This is followed by the tearing down of the connection.

3.6 Overview of routing functionality

IP routing protocols are used to propagate information regarding the destinations that can be reached by a router in a major network. The routers propagate the information using the routing update packets they send out either periodically or change in a network. The routers process the update packets and populate the routing tables with the information in the routing update packets. The frequency at which the update packets are sent depends on whether the routing protocol configured is a link state or a distance vector protocol. A distance vector protocol send out update periodically where as the link state vector protocol sends out updates only when there are changes in the states of links.

The existing implementation of IGP routing protocols is not sufficient to reap the full benefits of the RDMA based routing infrastructure. The reason is explained as follows. For the proposed routing architecture to be more effective the router needs to implement a mechanism where in the data source has information regarding the exit interface of the next hop device so that it can queue the packets at the egress interface directly. The current update packets

that are only provide the next hop information in order to reach the destination, where as in the proposed scheme the routers are required to have information regarding the egress interface of the next hop. For solving the issue the existing structure of the update packet where there is no provisioning for the information regarding the egress interface of the next hop router requires to be modified.

To present an example as to what needs to be modified with regards to the packet structures of the update packets we will be considering the update packet structure of RIP. Routing Information Protocol (RIP) is one of the simple and widely used routing protocols that are deployed in numerous small networks. It is an open source routing protocol unlike other routing protocols like IGRP and EIGRP which are Cisco proprietary. When a router interface running RIP is initialized, the router sends out update packets with complete routing table of the router generating the packet. The packets are sent out every 5 seconds, this acts as a keep alive between neighboring routers and this prevents the routes from being deleted from the routing tables of the neighboring routers. The update packet of RIP is as shown in figure 3.2 on the next page

Command	Version	Unused

Address Family Identifier	Unused
IP Address	
Unused	
Unused	
Metric	

Figure 3.3 Existing update packet structure of RIP

The update packet has fields for the IP address of the destination and the metric; it has no fields for specifying the egress interface to its neighbor. And one more requirement that has to be kept in mind is that the next hop field should be capable of being modified by the receiving router similar to the metric field. This would enable the receiving router to queue the packets directly to the egress interface of the neighboring router. The proposed packet format is depicted in figure 3.4.

Command	Version	Unused
Address Family Identifier		Unused
IP Address		
Unused		
Egress Interface		
Metric		

Figure 3.4 Proposed update packet structures

The router that would be sending the update packet would be modifying the field as required and it would broadcast the update packet as required. The incrementing of the metric field can be implemented in a different manner, i.e. the sending router can increment the field or the receiving router can increment the hop count. But the Egress interface field should be populated by the router originating the route information.

When the update message is received and processed by the next hop it would know the next hop for reaching the destination advertised in the routing table along with the egress interface to which the packet has to be delivered to. When the data source has the packet payload ready to be sent out of the interface it would establish a session with the egress interface of the next hop.

CHAPTER 4

MATHEMATICAL MODELLING

In this chapter the author presents a comparative study of a conventional router and an RDMA enabled router. The parameters considered for the study are the transit delay experienced by a packet, load on the CPU in and memory utilization. Transit delay refers to the total delay experienced by the packet from the data source to the data sink buffers. The load on the CPU refers to the instances where the intervention of the CPU is required.

4.1 Total delay experienced by a packet during its transit

The total delay experienced by a packet from the origin to the destination is the sum of connection time (Δ_{TC}), packet formation at source (Δ_{DS}), processing at intermediate hops (Δ_{IH}), processing at the final hop (Δ_{FH}). A detailed analysis of the above mentioned factors is performed in the following subsections.

4.1.1 Connection establishment

TCP Connection establishment time is the difference in time stamps of the SYN packet and the first data packet. The packet processing delay at the receiver is also taken into consideration. Δ_C is the time at which the first SYN packet was seen on the wire and Δ_{IP} is the time at which the first data packet was seen on the link, the time for TCP connection establishment (Δ_{TC}) is the difference of Δ_C and Δ_{IP}.

$$\Delta_{TC} = \Delta_{IP} - \Delta_C \quad (1)$$

$$\Delta_C = \Delta_{PDS} \quad (2)$$

Hence (1) can be written as

$$\Delta_{TC} = \Delta_{IP} - \Delta_{PDS} \quad (3)$$

For a RDMA enabled router there would be an additional delay incurred for transitioning the TCP connection to an RDMA enabled connection, which can be represented as (Δ_R).

The TCP connection establishment time for a conventional router (Δ_{TCC}) would be

$$\Delta_{TCC} = \Delta_{IP} - \Delta_{PDS} \quad (4)$$

The TCP connection establishment delay for an RDMA enabled router (ΔTCR) would be

$$\Delta_{TCR} = \Delta_{IP} - (\Delta_{PDS} + \Delta_R) \quad (5)$$

The delay incurred in transitioning the stream to an RDMA enabled stream is very minimal [1].

4.1.2 Packet processing delay

As mentioned in the previous chapters the update packet of RIP is being cited as an example. The proposed RIP update packet would include an additional field for the egress interface for the next hop router. The inclusion of the egress interface for the next hop would enable the data source router to identify the exact interface with which it has to establish the session.

After a data packet is received from a host with the source and destination addresses specified in the IP header, the router has to encapsulate the packet with another IP header with the address of the egress interface as the source address and the address of the egress of the next hop as the destination.

The delay that is incurred in looking up the next hop destination address can be represented as (Δ_{IPH}). The delay that is incurred for obtaining the relevant parameters for the TCP and RDMA headers is included in the Delay for connection establishment.

When a packet is received at the ingress interface of the next hop, the ingress interface has to process the IP header to find out the destination address of the packet. The delay incurred in processing of the IP packet (Δ_{IPP}) is present in the conventional router as well as the RDMA enabled router.

For an RDMA enabled router in addition to the IP header the RDMA header also needs to be processed. The delay incurred in processing the RDMA header can be put as (Δ_{RH}). The RDMA header contains the information regarding the data sink buffers into which the packet is to be written into.

In a conventional router the egress interface needs to be looked up based on the destination address in the IP header. The lookup time (Δ_{DL}) depends on the following factors number of routes present in the routing table (E_N), dedicated hardware for performing the lookup (E_H) and the presence of the route in the look up cache (E_C).

It can be put as the lookup delay (Δ_{DL}) is inversely proportional to the product of E_H, E_C and directly proportional to the number of routed in the routing table (E_N).

$$\Delta_{DL} \alpha E_N \quad (6)$$

As the number of routes in the routing table increase the time required to lookup the appropriate egress interface increases.

$$\Delta_{DL} \alpha 1/ E_H * E_C \quad (7)$$

When the conditions being represented by the factors are present they are to be substituted with 1, which is the maximum value of those factors. If the inverse is true then they are to be substituted by 0, which is the minimum value of the factors. If any one of the factors equals to zero then this would result in the increase in the egress interface lookup delay (Δ_{DL}).

Traffic engineering is being implemented in a big way in numerous networks, which calls for applying the configured policies and dropping the packets that are not in lieu with the policies. For checking the packets against the rules they need to be inspected, which calls for buffering of the packet in the memory of the traffic manager and delving into the IP header fields. The delay incurred due to traffic engineering can be represented as (Δ_{TE}).

On the similar lines many network administrators implement QoS on their devices for configuring the appropriate priority based on the type of traffic, the multimedia traffic which is intolerant to delay jitters is given the highest priority. Hence the amount of delay due to QoS procedure (Δ_{QoS}) that can be experienced by a packet is directly proportional to the probability quotient of the packet being a multimedia packet (P_{MM}).

$$\Delta_{QoS} \alpha 1/P_{MM} \quad (8)$$

The maximum value of P_{MM} is 1, which is applicable if the packet is a multimedia packet. If the packet is a non-multimedia packet then the value of P_{MM} would be 0, which is the minimum possible. If P_{MM} is zero then as per the equation the delay experienced by the packet would shoot up.

In this analysis the delay due to the packet copy mechanism is not considered, it is assumed that the copy procedure delay is zero. To sum up the overall delay experienced by a packet at 'N' conventional intermediate hops (Δ_{PPI}) can be written as

$$\Delta_{PPIC} = N * (\Delta_{IPP} + \Delta_{DL} + \Delta_{TE} + \Delta_{QoS}) \quad (9)$$

The overall packet processing delay experienced by the packet at 'N' RDMA enabled intermediate hops is

$$\Delta_{PPIR} = N * (\Delta_{IPP} + \Delta_{TE} + \Delta_{QoS} + \Delta_{RH}) \quad (10)$$

The total delay that is experienced by a packet from its source to the destination with conventional routers as the intermediate hops would be

$$\Delta_{ODC} = \Delta_{PPIC} + \Delta_{TCC} \quad (11)$$

The total delay experienced by a packet from its source to the destination with RDMA enabled routers as its intermediate hops would be

$$\Delta_{ODR} = \Delta_{PPIR} + \Delta_{TCR} \quad (12)$$

4.2 Analysis of memory utilization

A router, similar to a PC has a primary and a secondary memory. For a PC the random access memory is the primary memory and the hard drive is considered as the secondary memory. The primary memory is used for temporary storage of instructions or program data; whereas the secondary memory is used for the storage of application software and user data. If the router is analyzed on the similar lines, the DRAM can be considered as the primary memory of the router and the flash card can be considered as the secondary memory.

The DRAM of the router is used for supporting the I/O process of in the router; it is also used for storing the program instructions and various parameters that are being used by the processes of the router. The memory that is allocated for the interface buffers is actually allocated from the I/O buffer. There are special processes that are executed by the CPU for the management of various memory regions.

In this section the author would be analyzing the memory utilization patterns for both architectures. The author would also be analyzing the memory usage during the packet processing and transfer to the egress interface.

4.2.1 Memory utilization for a conventional router

When a packet first arrives at the ingress interface of the router there might be numerous packets that might have arrived before it, so the packet might have to be stored in the buffer until all the packets before it have been processed and sent out of the queue. Hence the memory that is to be allocated for the ingress interface depends on the amount of traffic that is being received on the ingress interface. It is assumed that the TCP streams terminate on a router, the amount of buffer that is to be allocated at a certain point of time depends on the number of TCP connections that are terminating on that particular interface during that instance.

If there are 'N' connections that are terminating on the ingress interface at a given instance of time and if (M_{SB}) is the minimum amount of buffer that is required per stream to sustain a reasonable load, the total amount of buffer required for an interface (M_{TB}) can be written as

$$M_{TB} = M_{SB1} + M_{SB2} + \ldots.. + M_{SBn}$$

$$M_{TB} = \sum_{i=1}^{N} M_{SBi} \quad (13)$$

If a router has ‘P’ interfaces then the total amount of buffer consumed by all the ingress interfaces (M_{IB}) would be the summation of the amount of buffer required per interface (M_{TBi})

$$M_{IB} = M_{TB1} + M_{TB2} + \ldots.. M_{TBi}$$

$$M_{IB} = \sum_{i=1}^{P} M_{TBi} \quad (14)$$

The analysis performed on the ingress interface is equally applicable to the buffers required for the egress interfaces. The justification for the above statement is that the router may not be able to send the packets out of their respective egress interface because of the sheer number of packets that have to be output per second. The packet needs to be buffered and then sent out of the egress interface.

If there are ‘N’ connections that are originating out of the egress interface at a given instance of time and if (M_{SoB}) is the minimum amount of buffer that is required per originating stream to sustain a reasonable load, the total amount of buffer required for an interface (M_{ToB}) can be written as

$$M_{ToB} = M_{SoB1} + M_{SoB2} + \ldots.. + M_{SoBn}$$

$$M_{ToB} = \sum_{i=1}^{N} M_{SoBi} \quad (15)$$

If a router has ‘P’ interfaces then the total amount of buffer consumed by all the egress interfaces (M_{EB}) would be the summation of the amount of buffer required per interface (M_{ToBi})

$$M_{EB} = M_{ToB1} + M_{ToB2} + \ldots.. M_{ToBi}$$

$$M_{EB} = \sum_{i=1}^{P} M_{ToBi} \quad (16)$$

All the packets that have arrived are to be forwarded to the CPU or the forwarding engine for performing the egress interface lookup based on the destination address. If the router's CPU is to perform the route lookup then the packet needs to be copied into the CPU memory and the lookup procedure needs to be performed on the packet context. The ingress interface makes a copy of the packet context and sends it to the processor memory for it to perform the lookup function. Assuming that there is appropriate caching mechanism on the interface in order to cache the lookup results so that they can be used for future forwarding, at least one packet from an individual stream has to be looked up for successful forwarding of all the packets of a stream.

The size of the IP header is 20 bytes, assuming that there are 'N' active streams terminating at an ingress interface at any given point of time. At peak load conditions the number of connections terminating on an ingress interface can be anywhere from few hundreds to a thousand.

A Cisco 3620 router has 16KB of data cache, even if assumed that the whole 16KB is available for the packet contexts that need to be processed it can accommodate only 820 streams, which is not adequate for peak load conditions. Practically the cache has to be shared by the packet contexts and the data variables called in by other active processes.

After the lookup is performed the packet context [11] needs to be appended accordingly, and sent back to the ingress interface. The packet context is used by the interface to send the packet over to the appropriate I/O buffer of the egress interface. The egress interface needs to have a packet context for encapsulating the packet accordingly to forward to the next hop.

Looking at the number of memory copy operations for each packet, the first copy operation would be from the application buffer which in this case would be the I/O buffer of the

ingress interface to the CPU buffer for the lookup, after the lookup is performed the packet is to be transferred to the application buffer, which is the I/O buffer of the egress interface. If there are 'N' active streams terminating on an ingress interface at a given instance of time the CPU memory per interface (M_{CPUI}) required would be the summation of the product of the average packet size in a stream (P_s) and the number of packets in the stream (S_P). This can be put mathematically as

$$M_{CPUI} = ((P_{S1} * S_{P1}) + (P_{S2} * S_{P1}) + \ldots\ldots + (P_{sn} * S_{P1}))$$

$$M_{CPUI} = \sum_{i=1}^{N} (P_{Si} * S_{Pi}) \quad (17)$$

If a router has 'P' interfaces then the total amount of CPU memory consumed (M_{TCPU}) would be the summation of the amount of buffer required per interface (M_{CPUI})

$$M_{TCPU} = M_{CPU1} + M_{CPU2} + \ldots.. M_{CPUi}$$

$$M_{TCPU} = \sum_{i=1}^{P} M_{CPUi} \quad (18)$$

To summarize the total memory utilization (M_{TC}) by a conventional router for the destination lookup, buffering of the packets and the memory utilization in the CPU can be written as

$$M_{TC} = M_{TCPU} + M_{EB} + M_{IB} \quad (19)$$

4.2.2 Memory utilization in a RDMA enabled router

In the proposed architecture for the RDMA enabled router the lookup for the destination buffers is not performed by the CPU but by the forwarding engine present on the interface. As mentioned earlier the forwarding table is populated with the details of the advertised buffers for each of the sessions by the CPU. The STag, tag offset and the buffer length information is included in the packet context formed by the ingress interface. The packet context is sent to the forwarding engine present on the interface which writes the packet directly into the destination buffers.

The look up time required for an RDMA session is very minimal since the forwarding table already has the details of the interface to which the buffers being advertised belong to. The on chip memory of the forwarding engine has the buffers that were advertised and the interface that has advertised them. Buffering the packet stream is not required in this context as all the details regarding the destination buffer can be obtained from the RDMA header which can be mapped to a specific egress interface.

When the packet is transferred to an egress interface they may need to be buffered since the egress interface needs to establish sessions with the egress interface of the next hop. The total amount of memory required for buffering the packets on the egress interface (M_{EBR}) can be written as

$$M_{EB} = M_{ToB1} + M_{ToB2} + \ldots. M_{ToBi}$$

$$M_{EB} = \sum_{i=1}^{P} M_{ToBi} \quad (20)$$

This is the total amount of memory required by the router for buffering the egress packets. The need for buffering the packet context in the CPU is eliminated by placing the forwarding engine on the interface itself.

$$M_{TR} = M_{IB} + M_{EB} \quad (21)$$

The overall memory that is required for transferring a stream of packets from the ingress interface to the egress interface is very nominal in the RDMA enabled router when compared to the conventional router. This is advantageous in terms that the memory requirement during the peak load period can be easily met since plenty is available to be assigned to the egress buffers of the interfaces, and another advantage with this approach is that the CPU can be freed from the task of performing the copy operations and can be used for other functions.

4.3 Analysis of CPU utilization

In this section the author will be analyzing the CPU utilization for both architectures. Analysis regarding the CPU utilization when the packet arrives at the interface in a step by step manner is performed.

4.3.1 CPU utilization for the conventional router

This section will consider the scenarios of the previous sections. The author presents a detailed analysis of the CPU utilization for functions such as

a) Performing the route lookup for an egress interface based on the destination.
b) The copying of the packet context into the CPU memory.
c) Copying the packet from the application memory, i.e. the ingress buffer to the CPU memory.
d) Writing the packet from the CPU memory to the buffer of egress interface.

Assuming a packet size of 'N' bytes, the word size of the CPU is at least 32 bits we represent the word size as 'X' and the processor frequency is 'Y' megahertz. The author also assumes that the forwarding cache of the router being considered is empty or there was a cache miss.

Packet arrives on the ingress interface a packet context needs to be created a packet context is a description of the packet in brief. The packet context should at least have the destination IP address of the packet in order to lookup the egress interface to which the packet needs to be transferred.

A packet context is created by the CPU from the IP header of the packet. The 32 bit destination IP address is to be read into the CPU memory for the creation of packet context. Each copy cycle consumes a CPU cycle. Hence the CPU cycles required for the creation of a packet context (TPC) is one.

$$T_{PC} = 1 \tag{22}$$

After the creation of the packet context the CPU needs to search the routing table for the appropriate egress interface for the packet. The closest match for the destination IP address needs to be searched for deciding upon the next hop and thus the egress interface. The routing table needs to be searched for by the router's CPU for determining the exact match. The routing table is present in the DRAM, which is to be loaded into the CPU memory for performing the search. The routing table size varies depending on the kind of router in consideration, i.e. access router has fewer routes when compared to the routing table of a core router. For performing the search the routing table is loaded in blocks into the memory of the CPU and a search is performed to

determine the closest match for the IP address in picture. This process is repeated till an appropriate match has been found.

The number of CPU cycles required for searching the routing table depends on the number of routes in the routing table (E_N).

$$T_L \; \alpha \; E_N \tag{23}$$

Another factor that affects the CPU utilization is the probability of finding the closest match within the first block of the routing table loaded into the CPU. The probability factor for finding the route in the first block of the routing table loaded can be represented as (P_{FB}). The CPU utilization is directly proportional to the proportionality factor.

$$T_L \; \alpha \; P_{FB} \tag{24}$$

After the lookup is performed successfully, the packet needs to be transferred to the buffer of the appropriate egress interface. The packet copy operation needs to be performed by the CPU. The first step is to copy the packet from the application memory to the CPU memory. As mentioned earlier if the packet size is 'N' bytes and if the processor is an 'X' bit, 'Y' mega hertz processor.

The packet size in bits would be N * 8, the processor can load 'X' bits per copy cycle into the CPU memory. If a single read consumes one entire copy cycle, the number of cycles required for copying the entire packet would be (N * 8) / X. Since the packet needs to be copied from the CPU memory to the egress buffer. This would require another (N * 8) / X. Hence the total number of CPU cycles required for transferring a packet from the ingress buffer to the egress buffer (T_C) equals 2 * (N * 8) / X

$$T_C = 2 * (N * 8) / X \quad (25)$$

As an example if a packet size of 1500 bytes is considered, it is the most common packet size [10], and a 32 bit and 100 MHz processor. The packet size in bits translates to 1500 * 8 = 12000 bits. The number of CPU cycles required = 12000 / 32 = 375 cycles. 100MHz translates to $100 * 10^6$ cycles per second. The similar logic is applicable for writing the packet to the application memory of the egress interface from the CPU memory, thus the total number of cycles required for copying a packet from CPU memory to the buffer of the egress interface. This implies that the processor can copy 200,000,000 packets of size 1500 bytes.

The above figure appears humungous, but the CPU needs to perform numerous other tasks that are critical for the proper functioning of the system, which place high burden on the CPU thus limiting the overall packet throughput of the router.

To summarize the total number of the CPU cycles required for a packet to be transferred from the ingress interface to the egress interface (T_T) would be

$$T_{TC} = T_{PC} + T_L + T_C \quad (26)$$

4.3.2 CPU utilization in a RDMA enabled router

The author would be analyzing the CPU utilization for the RDMA enabled router for the scenario that was described in the previous section. The assumptions made in the previous section would be valid for the present section too.

Considering the similar scenario as described in the previous section, when a packet arrives the packet context needs to be created for the packet that has arrived on the ingress interface. The packet context in the RDMA router is not the same as in the conventional router,

where the packet context comprises of the destination IP address. The packet context would consist of the STag and tag offset which are 32 bits each. Hence the CPU needs to copy the required information twice into the memory of the CPU in order to create the packet context.

$$T_{PCR} = 2 \tag{28}$$

Here the author concentrates on the CPU cycles required for the copy mechanism; they will not be concentrating too much on the CPU cycles that are required for the processing of the instructions.

After the packet context is created, it is forwarded to the forwarding engine on the interface of the router. The forwarding engine with the on chip memory has the buffer to interface map, which is populated by the CPU while the session negotiation is taking place. The number of CPU cycles required for populating the forwarding table can be represented as (T_{FP}).

The value of T_{FP} depends on the number of buffers being advertised by the egress interface, which depends on the number of packets in the stream (S_P) that are to be written into the buffer. If the routes that are to be populated to the forwarding table are more than the register capacity of the CPU then it would require more number of CPU cycles to populate the forwarding table on the onboard forwarding engine.

$$T_{FP} \alpha\, S_P \tag{29}$$

In the RDMA enabled router the packet write mechanism from the ingress interface to the egress interface is not a CPU consuming process because the RDMA buffer information in the RDMA header is adequate enough to place the packet from the ingress interface to the egress interface.

To summarize the total CPU utilization for the RDMA enabled router (T_T) is the sum of CPU utilization for the populating of the forwarding table (T_{FP}) and the CPU utilization for creation of packet context (T_{PC}).

(30)

4.4 Results

This section presents a practical evaluation of the equations discussed in the mathematical model. The results are presented in a tabular format in figures 4.1 and 4.2; table 4.1 assumes a packet size of 64 bytes. 'N' represents the number of streams terminating on the interface; 'P' represents the number of interfaces on the router, 'S_P' is the size of the packet and 'X' is the size of the CPU register.

N	2	Δ_{TCC} (ms)	5	M_{SB} (bytes)	6400	T_{PC}	1
P	2	Δ_{TCR} (ms)	6	M_{TB} (bytes)	12800	T_L	3
P_{iS}	100	Δ_{IPP} (ms)	2	M_{IB} (bytes)	25600	T_C	32
S_P	64	Δ_{DL} (ms)	3	M_{SoB} (bytes)	6400	T_{TC}	36
X	32	Δ_{TE} (ms)	2	M_{ToB} (bytes)	12800	T_{PCR}	2
		Δ_{QoS} (ms)	1	M_{EB} (bytes)	25600	T_{FP}	3
		Δ_{RH} (ms)	2	M_{CPUI} (bytes)	16800	T_{TR}	5
		Δ_{PPIC} (ms)	16	M_{TCPU} (bytes)	33600		
		Δ_{PPIR} (ms)	20	M_{TC} (bytes)	84800		

		Δ_{ODC} (ms)	25	M_{TR} (bytes)	51200		
		Δ_{ODR} (ms)	20				

Table 4.1 Results with a packet size of 64 bytes and a 32 bit processor

N	2	Δ_{TCC} (ms)	5	M_{SB} (bytes)	12800	T_{PC}	1
P	2	Δ_{TCR} (ms)	6	M_{TB} (bytes)	25600	T_L	3
PiS	100	Δ_{IPP} (ms)	2	M_{IB} (bytes)	51200	T_C	64
PS	128	Δ_{DL} (ms)	3	M_{SoB} (bytes)	12800	T_{TC}	68
X	32	Δ_{TE} (ms)	2	M_{ToB} (bytes)	25600	T_{PCR}	2
		Δ_{QoS} (ms)	1	M_{EB} (bytes)	51200	T_{FP}	3
		Δ_{RH} (ms)	2	M_{CPUI} (bytes)	29600	T_{TR}	5
		Δ_{PPIC} (ms)	16	M_{TCPU} (bytes)	59200		
		Δ_{PPIR} (ms)	20	M_{TC} (bytes)	161600		
		Δ_{ODC} (ms)	25	M_{TR} (bytes)	102400		
		Δ_{ODR} (ms)	20				

Table 4.2 Results with packet size of 128 bytes and 32 bit processor

N	2	Δ_{TCC} (ms)	5	M_{SB} (bytes)	6400	T_{PC}	1
P	2	Δ_{TCR} (ms)	6	M_{TB} (bytes)	12800	T_L	3
P_{iS}	100	Δ_{IPP} (ms)	2	M_{IB} (bytes)	25600	T_C	16
P_S	64	Δ_{DL} (ms)	3	M_{SoB} (bytes)	6400	T_{TC}	20
X	64	Δ_{TE} (ms)	2	M_{ToB} (bytes)	12800	T_{PCR}	2
		Δ_{QoS} (ms)	1	M_{EB} (bytes)	25600	T_{FP}	3
		Δ_{RH} (ms)	2	M_{CPUI} (bytes)	16800	T_{TR}	5
		Δ_{PPIC} (ms)	16	M_{TCPU} (bytes)	33600		
		Δ_{PPIR} (ms)	20	M_{TC} (bytes)	84800		
		Δ_{ODC} (ms)	25	M_{TR} (bytes)	51200		
		Δ_{ODR} (ms)	20				

Table 4.3 Results with a packet size of 64 bytes and 64 bit processor

The graph in Figure 4.1 depicts the memory utilization for the conventional router and the RDMA enabled router against various packet sizes. It is evident that the memory utilization for the conventional router is far greater than the memory utilization of the RDMA enabled router, as mentioned in one of the previous sections this is due to the memory that is used by the router for data transfer mechanism.

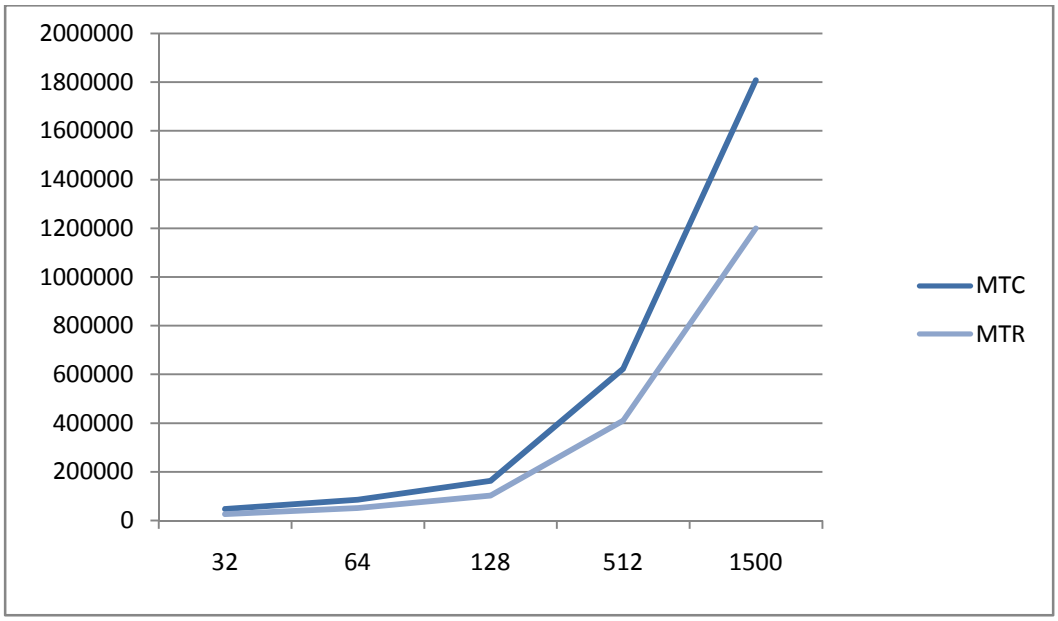

Figure 4.1 Memory utilization vs. packet size

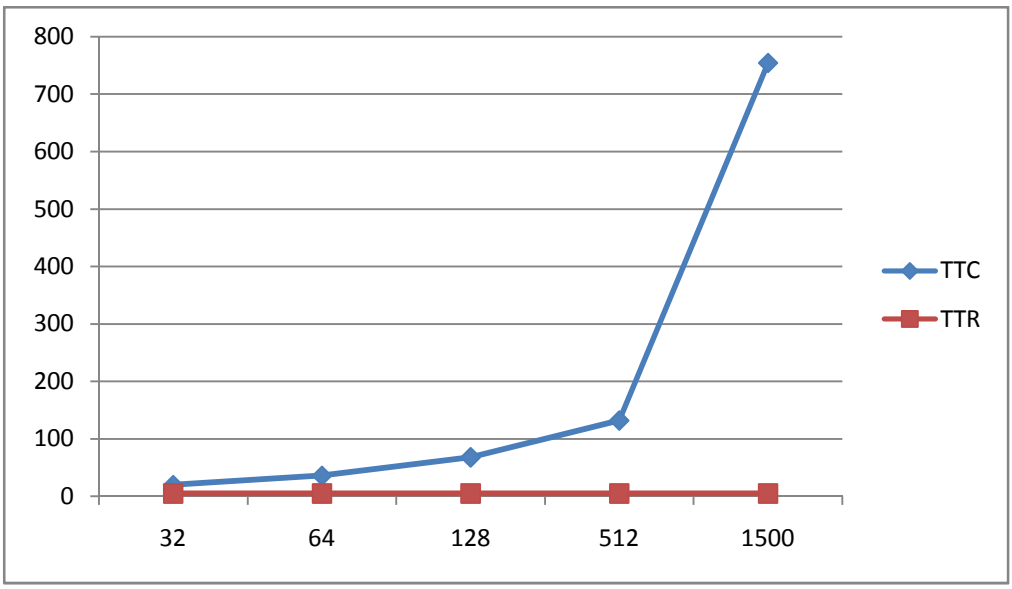

Figure 4.2 CPU utilization vs. packet size

Figure 4.2 presents a comparative analysis of the number of CPU cycles that are required by both routers for increasing packet sizes. The CPU utilization increases as the size of the packet increases because the greater the packet size in comparison to the register size of the processor the more number of cycles are required to transfer the packet from the ingress buffer to

the egress interface. The CPU cycles consumed by the RDMA enabled router are constant since the payload does not involve intervention of the CPU. The same holds good for Figure 4.3.

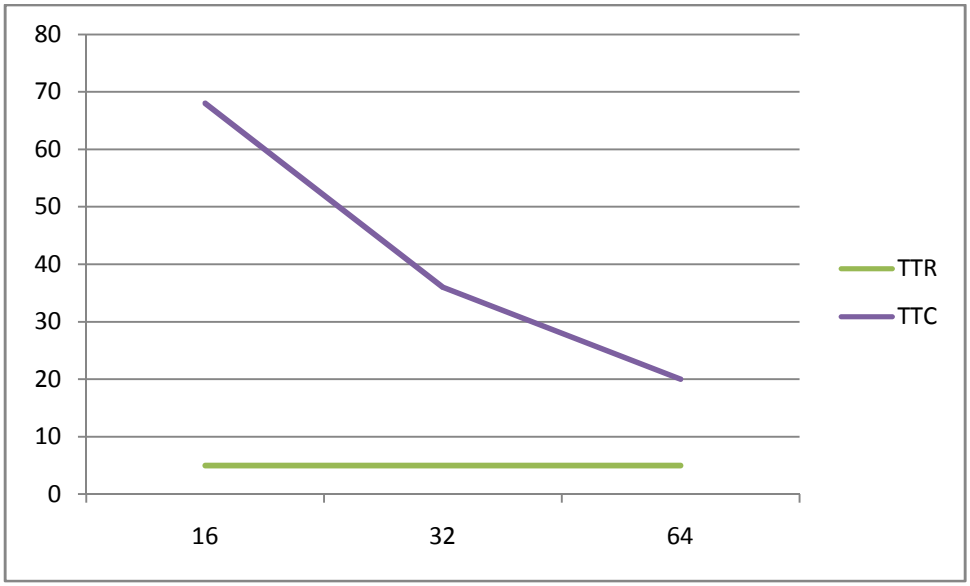

Figure 4.3 CPU utilization vs. Register size

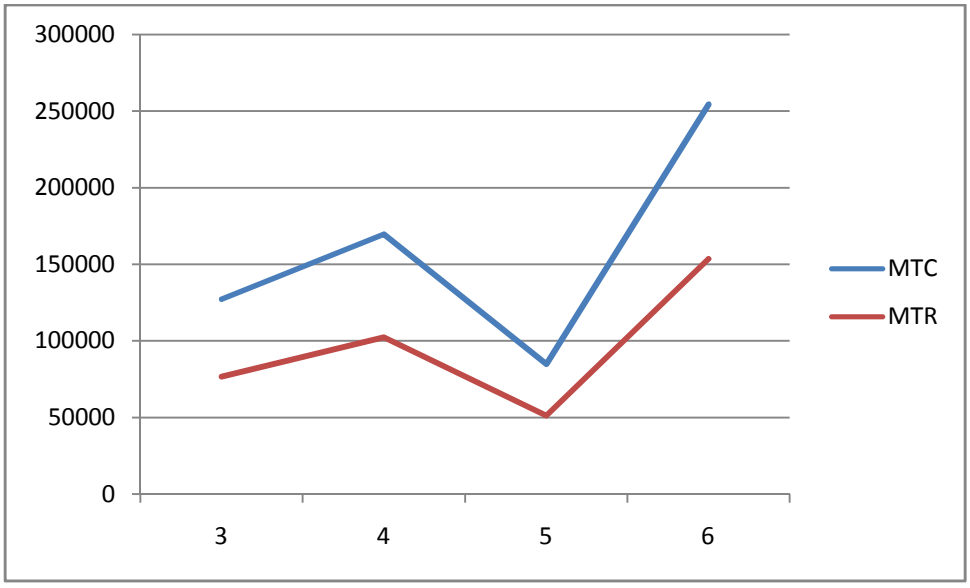

Figure 4.4 Memory utilization vs. number of interfaces

As depicted in Figure 4.4 it can be observed that the memory utilization of a conventional router increases as the number of interfaces increase due to CPU copy mechanism that is required for transferring the payload from the ingress interface to the egress interface.

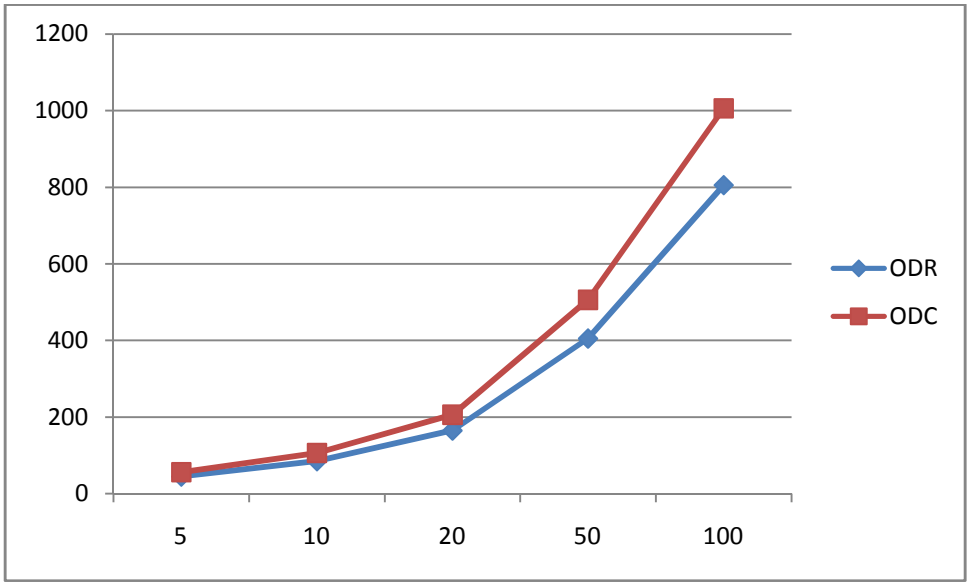

Figure 4.5 Delay incurred vs. number of active streams

The delay incurred by a packet while transiting an RDMA enabled router is less than the delay incurred when it transits a conventional router, this is pictorially depicted in Figure 4.5

CHAPTER 5
CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

An analysis of the results obtained revels that the memory and CPU utilization of the router with RDMA capabilities is less than that of the conventional router which allows the allocation of these resources to critical processes, thus ensuring the availability of the CPU and memory when requested by critical processes. With the RDMA process the CPU cycles required when compared to a conventional router is four times less. The lesser the CPU cycles required the lesser the burden on the CPU. Even the memory required by the RDMA enabled router is one and a half times the memory utilized by the conventional router. Reduced utilization of memory translates to reduced load on the buffer.

In this thesis, the author proposes the modifications to the conventional IP routing protocols, taking RIP as an example. The author has also proposed architecture for the RDMA based router that provides better throughput when compared to the conventional router, which has issues with processor and backplane bottle necks. The workarounds for the bottlenecks have been suggested and implemented in the proposed architecture. In the last chapter the author has mathematically analyzed the CPU utilization, delay experienced by the packet, memory utilization for both architectures.

The delay experienced by the packet is more in the conventional routers due to the lack of specialized hardware for performing the egress interface lookup based on the destination IP address, the lookup delay constitutes a major portion of the overall delay experienced by a packet. This delay is reduced in the RDMA enabled architecture as the destination buffer

information is present in the RDMA header that enables the forwarding engine to transfer the payload directly to the data sink buffer.

The CPU utilization is more in the conventional router since the packets have to be transferred from the ingress buffer to the egress interface by CPU copy mechanism, this would consume CPU cycles based on the size of the packet. The use of RDMA protocol eliminates the necessity for CPU mechanisms.

The memory requirement in the RDMA enabled router is less than that of the conventional router considering the fact that the need for an ingress buffer is almost eliminated as the lookup time is negligible and the forwarding table is populated with all the required details well ahead of time. The egress interface buffer might be required under conditions of peak loading.

5.2 Future work

Currently the RDMA protocol is supported only over connection oriented protocol such as TCP, but there are numerous applications that use the services of UDP for minimal transport overhead. This warrants the extension of RDMA support for connectionless protocols. The buffer advertisement is a ULP specific mechanism; if this function is offloaded from the ULP this would further conserve CPU resources, which is always a welcome improvement. Future work aiming at proposing an efficient QoS implementation method can also be pursued.

An efficient buffer advertisement and handling mechanism can add to the increase in throughput of a router. A real time implementation of a buffer controller in hardware is also desired.

REFERENCES

LIST OF REFERENCES

[1]. RFC on RDMA, RFC 5040, http://www.ietf.org/rfc/rfc5040.txt

[2]. RDDP verb draft, http://tools.ietf.org/html/draft-hilland-rddp-verbs-00

[3]. MPA draft, http://www.ietf.org/proceedings/06jul/slides/tcpm-4.pdf

[4]. The case for RDMA, an internet draft, http://www.cs.duke.edu/ari/publications/draft-csapuntz-caserdma-00.txt

[5]. Study of internet router architectures, by Kiran Mukesh Misra and Freddy Kharoliwalla, Michigan State University, http://citeseerx.ist.psu.edu/viewdoc/summary;jsessionid=AEAFCC1A8C6842C7566DDCFCFE2E7457?doi=10.1.1.91.3402

[6]. Fast switched backplane for a Gigabit router, by Nick McKeown, Stanford University

[7]. Modeling a router, by Ronald Rousseau and Iris Vandecasteele, Industrial Sciences and Technology, Belgium

[8]. Impact of BGP on CPU utilization, Sharad Agarwal, Chen-nee Chuah, Supratik Bhattacharya and Christophe Diot, University of California.

[9] Performance Analysis of an RSVP-Capable Router, Tzi-cker Chiueh, Anindya Neogi and Paul Stripe, State University of New York at Stony Brook and Reuters Information Technology Inc.

[10]. Network Routing, algorithms, protocols and structure By Deepankar Mehdi and Karthigeyan Ramaswamy, Morgan Koffman Publications.

[11]. Advanced router architectures, By Axel K. Kloth, Taylor and Francis Group, CRC press

[12]. RDMA and DDP overview, Renato Resio, RDMA Consortium.

[13]. An overview of RDMA over IP, Alynn Romanov, Cisco Systems, Stephen Bailey, Sandburst Corporation.

[14]. Memory Mapping and DMA, http://lwn.net/images/pdf/LDD3/ch15.pdf

[15]. Introduction to RDMA, http://en.wikipedia.org/wiki/Remote_Direct_Memory_Access

[16] Transmission Control Protocol, http://en.wikipedia.org/wiki/TCP_handshake#Connection_establishment

[17]. A critique of RDMA, http://www.hpcwire.com/features/17886984.html

[18]. Overview of RDMA, http://www.networkworld.com/details/5221.html

[19]. Tutorial of the RDMA model, http://www.hpcwire.com/features/17887604.html

LIST OF REFERENCES (Cont.)

[20]. Using RDMA to improve processing performance, http://www.embedded-computing.com/articles/id/?2079

[21]. RDMA: Tiny interface, big gains- archi – TECH, http://www.itbusinessedge.com/blogs/atc/?p=397

[22]. Is RDMA that bad, http://insidehpc.com/2006/08/20/is-rdma-really-that-bad/

[23] RDMA over TCP Breaking the Gigabit Ethernet bottleneck, http://www.networkworld.com/columnists/2004/080204tolly.html

[24]. RDMA protocol: improving network performance, technology brief, http://h20000.www2.hp.com/bc/docs/support/SupportManual/c00589475/c00589475.pdf

[25]. Advanced IP routing in Cisco networks, By Terry Slattery and William Burton, New York Mc Graw Hill Professional.

[26]. Express forwarding in Cisco routers, http://www.cisco.com/en/US/products/sw/iosswrel/ps1828/products_tech_note09186a00801e1e46.shtml