Implementation of virtual circuits as a switching fabric in virtual modularized network

Siddhartha Bolla

Follow this and additional works at: http://utdr.utoledo.edu/theses-dissertations

Recommended Citation
http://utdr.utoledo.edu/theses-dissertations/796

This Thesis is brought to you for free and open access by The University of Toledo Digital Repository. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of The University of Toledo Digital Repository. For more information, please see the repository's About page.
A Thesis

Entitled

Implementation of Virtual Circuits as a switching fabric in Virtual Modularized Network

By

Siddhartha Bolla

Submitted as partial fulfillment of the requirements for

The Master of Science in Electrical Engineering.

______________________________
Dr. Lawrence Miller, Committee Chair

______________________________
Dr. Ezzatollah Salari, Committee Member

______________________________
Dr. Henry Ledgard, Committee Member

______________________________
Dr. Patricia Komuniecki, Dean

College of Graduate Studies

The University of Toledo

May 2010
Virtual Modularized Network (VIMNet) is a dynamic modularized protocol framework architecture which is based on the principles of object oriented environments which allows applications to utilize protocol modules or choose customized modules. VIMNet is designed to overcome the problems with the current Internet due to its underlying core design which uses the TCP/IP model by providing facilities to enable improved quality of service, reliability and robustness. An important part of this architecture is the connections between the components within the networks to facilitate data transfer. This thesis focuses on the development of virtual circuit based system architecture similar to the one used in ATM.
Acknowledgments

I would like to thank my advisor, Dr. Lawrence Miller, for his supervision and support to accomplish this work. I would like to extend my gratitude to Dr. Ezzatollah Salari and Dr. Henry Ledgard for serving as my thesis committee members.

I would like to thank Jonathan Guernsey, the originator of VIMNet, for providing me with his papers and publications. My special thanks to Prof. Dan Solarek for giving me graduate assistantship for all the semesters I worked for Engineering Technology.

I am very grateful to my family and all my friends for their love and encouragement. Though I am not naming them individually, I would want to show gratitude to all.
Table of Contents

Abstract...................................................................................................................................... iii
Acknowledgements .............................................................................................................. iv
Table of Contents .............................................................................................................. v
Table of Figures................................................................................................................ vii
List of Tables .................................................................................................................. viii

1. Introduction................................................................................................................... 1
   1.1 Objectives .................................................................................................................. 2
   1.2 Thesis organization ................................................................................................. 3

2. The Internet today ........................................................................................................ 4
   2.1 Introduction.............................................................................................................. 4
   2.2 TCP/IP protocol suite............................................................................................... 5
   2.3 The OSI Model ......................................................................................................... 9
   2.4 A Comparison of the OSI model and TCP/IP protocol suite................................. 11
   2.5 Drawbacks of a static protocol framework .............................................................. 13
   2.6 Drawback of the current Internet architecture ...................................................... 14
   2.7 MPLS..................................................................................................................... 17
   2.8 Requirements of the future Internet ....................................................................... 18

3. VIMNet ....................................................................................................................... 24
   3.1 Introduction.............................................................................................................. 24
   3.2 Dynamic protocol framework ............................................................................... 27
      3.2.1 Modules......................................................................................................... 29
List of Figures

2.1 TCP/IP Protocol Stack communication between two hosts ......................... 7
2.2 The OSI Model ............................................................................................. 10
2.3 Comparison between OSI and TCP/IP ......................................................... 12
3.1 VIMNet Proposed Architecture ............................................................... 26
3.2 Inter-modular Communication ................................................................. 32
3.3 End Host System Model ............................................................................ 33
4.1 A basic architecture involving virtual circuits for multicasting ............ 42
4.2 ATM cell header ......................................................................................... 46
5.1 Raw packet ................................................................................................. 48
5.2 Connection setup phase ............................................................................. 49
5.3 Data transfer phase .................................................................................. 50
5.4 Connection tear down phase ................................................................. 52
5.5 Router indicating three left and right ports ........................................... 53
5.6 Internal Routing for Router #1 ............................................................. 55
5.7 Routing function between the source and destination ..................... 56
5.8 A multicast connections scenario ......................................................... 57
List of Tables

3.1 Comparison between TCP/IP and VIMNet ............................................................. 37
3.2 Comparison between MPLS and VIMNet ............................................................... 38
Chapter 1

Introduction

The internet which began as a defense project grew exponentially to the point where it became a critical medium of information. Today’s society has a situation where every individual is connected to the virtual world of information through a fixed end host or a portable device where the desired support for current and future applications is not met with the existing quality of service, security, availability, manageability, extensibility and robustness. With increasing functionalities such as entertainment, streaming live video, voice and many more, the current existing framework lacks in many ways in giving the formerly stated features on which the architecture is formed.

The current architecture of the Internet is based on the TCP/IP protocol suite. With the changing requirements of network applications which are vast and differ drastically from one network application to the other, it is envisioned that the future networks will be dominated by high speed wireless networks similar to WiMAX, multi-gigabit speed Ethernet as well as single and multi-wavelength fiber networks [1]. Unfortunately, the current Internet's static monolithic protocol stack architecture is unable to provide these vast, highly flexible, and adaptable services.
Considering the growth rate of the Internet with an initial size of ARPANET's 4 systems in 1969 to the Internet which had well over 150 million systems in 2000 in the United States alone, shows how much the landscape of the network has changed. This was just the beginning as the world usage growth of the Internet is 380.3% in the last decade (the period of 2000 – 2009) according to Internet World Statistics as of September 30, 2009[2]. With increasing features and functionalities such as cell phones becoming internet ready, online gaming, remotely operating devices like DVR and eBook readers to name a few and many more applications which were science fiction of the past; the desired support for both current and future network applications cannot be provided by the present static protocol architecture and its underlying model; a dynamic protocol framework is needed.

### 1.1 Objectives

Despite various research projects, current techniques are still insufficient, especially for applications that require stringent resource requirements and transmission deadlines. It is foreseen that the next generation of network applications will require true real-time communications, true quality of service (QoS), as well as significantly improved security [1].

In order to achieve these objectives, the future Internet must be designed with the intention of handling the shortcomings of the current Internet. The future internet must be very flexible with respect to adapting changes in the technologies and policies. It should have the ability to integrate new algorithms and protocols, security requirements and
application-dependent processing constraints and needs. Each application connection may contain a different arrangement of protocol modules to provide the required features. It should also be capable of adapting to several networks based on dissimilar protocols [1].

These requirements facilitate the need for a dynamic network that can support various architectures including virtual network architectures where applications can pick and choose or implement customized protocols.

1.2 Thesis organization

This thesis is organized into six chapters. Chapter 2 discusses the current Internet which is based on TCP/IP protocol suite, its working comparing it with the OSI model. Then the drawbacks of the static protocol framework are discussed along with the shortcomings of the current Internet. At the end of the chapter, MPLS, an alternate protocol framework, is discussed along with its limitations. Finally Chapter 2 concludes with a discussion of the requirements of the future internet. Chapter 3 starts with an introduction to the proposed VIMNet architecture which employs dynamic protocol framework including a brief explanation of all components of the architecture. The chapter ends with the comparison of VIMNet with TCP/IP and MPLS architectures. Chapter 4 and Chapter 5 form the central part of this thesis. Chapter 4 describes the role of virtual circuits, its features and functioning where as Chapter 5 discusses about the theory and research done for the implementation of virtual circuits as a switching fabric in the proposed VIMNet architecture. The final chapter summarizes the thesis with conclusions and scope for future work in this area.
2.1 Introduction

The Internet started as ARPANET, a U.S. Department of Defense project in the 1960s, to link several computer networks. Used primarily by academic institutions, scientists and the government for research and communications, the initial functionalities were to connect to each others' computing systems and databases and share data. Gradually the functionalities grew and the ARPANET eventually became the Internet. Initially the Internet was based on the idea that there would be multiple independent networks of rather arbitrary design, beginning with the ARPANET as the pioneering packet switching network, but soon to include packet satellite networks, ground-based packet radio networks and other networks-3. These objectives resulted in the formation of a protocol called Transmission Control Protocol or TCP.

Implementing the Transmission Control Protocol was limited to using virtual circuits initially and worked fine for applications like file transfer, but was largely inadequate for applications like voice packets. This led to a reorganization of the original TCP into two protocols, one being the Internet Protocol (IP) which was used for
addressing and forwarding of individual packets, and the other a separate TCP which dealt with the features such as flow control and recovery of data from lost packets.

The concept of creation of networking standards is to define widely-accepted ways of setting up networks and connecting them together using commonly followed standards. During the inception of TCP/IP, there was another standard called the OSI model which was developed concurrently. The differences between the two architectures are discussed towards the end of this chapter whereas the difference in the structures are shown in the figure 2.1. The rise in popularity of the Internet and its TCP/IP protocols met the OSI suite head on, and in a nutshell, TCP/IP won [2].

2.2 TCP/IP protocol suite

A system that implements protocol behavior consisting of a series of layers is known as a “protocol suite” or a “protocol stack”, hence the name TCP/IP protocol suite. These layers work together to provide reliable and efficient data communication across a network system. From a bird’s eye view, the functionality of the TCP/IP protocol suite is to break up every piece of information and message into pieces called packets, deliver those packets to the proper destinations, and then reassemble the packets into their original form after they've been delivered so the receiving computer can view and use them. TCP accounts for the disassembling and reassembling the packets, whereas IP is responsible for ensuring the packets are sent to the right destination. Protocol suits can be implemented either in hardware or software, or a mixture of both. Typically, only the lower layers are implemented in hardware, with the higher layers being implemented in
software. Being modeled into a layered structure, the layers communicate and utilize the services provided by the each other for a reliable delivery of data. Each layer offers a set of services which are distinguished from the other which leads to distribution of the tasks in the protocol suite. The communication happens between the adjacent layers and not beyond that.

When it comes to communicating between peers over a network or a pair of hosts, the TCP/IP protocols enable communication between them. Each layer has the property that it only uses the functions of the layer below, and only offers the functionality to the layer above. Each layer on one peer has a corresponding layer on the other peer. Each layer is only concerned with communicating to its peer on the other side regarding the task involved but not with immediate next layer in the protocol suite. We must be clear with the communication involved between the peers involving the functionality of the layer and the interaction between the adjacent layers in the protocol suite. This is shown in the figure 2.1 below.

Each protocol formats communicated data and appends information to or removes information from the data. Then the protocol passes the data to a lower layer on the sending host or a higher layer on the receiving host. A brief discussion on the functionalities of each layer is as follows.
Figure 2.1 TCP/IP Protocol Stack communication between two hosts

Application Layer

The application layer defines protocols that applications use to exchange data. It deals with the syntax and semantics of the information exchanged between two systems and enables the user to access the network. Amongst many other responsibilities, some are encryption, compression, mail services, directory services and synchronization.

TCP/IP defines many protocols in this layer including Simple Mail Transfer Protocol (SMTP), File Transfer Protocol (FTP), Hypertext Transfer Protocol (HTTP), Domain Name Service (DNS), Routing Information Protocol (RIP), Telnet, File Transfer Protocol (FTP), Simple Network Management Protocol (SNMP), etc.
Transport Layer

The transport layer takes care of flow control and error control during the process-to-process delivery of the entire message from the source to the destination. Segmentation-reassembly and connection control being the other important functionalities of this layer.

The transport layer uses two protocols, UDP and TCP. UDP, which stands for User Datagram Protocol, does not guarantee packet delivery and applications which use this must provide their own means of verifying delivery. TCP does guarantee delivery of packets to the applications which uses it.

Internet Layer

The internet layer is responsible for the source-to-destination delivery of a packet, possibly across multiple networks. It ensures that each packet gets from its point of origin to its final destination. Other functionalities being logical addressing and routing.

The Internet Layer supports Internetworking protocol (IP). IP, in turn, uses four supporting protocols: ARP, RARP, ICMP and IGMP.

Network Layer

The network interface layer coordinates the functions required to carry a bit stream over a physical medium and transforms this raw transmission facility into a reliable link. Packet routing, synchronizing bits, physical addressing, error control and access control being its important functionalities [12].

It uses low level protocols such as ICMP, IP, and IGMP. In addition to the routing protocols such as RIP, OSPF, and EGP.
2.3 The OSI model

OSI model stands for Open Systems Interconnection model. The OSI model was created by International Standards Organization in order to standardize the protocols used in different layers. It is primarily used today as a teaching tool. It conceptually divides a network into seven layers as shown in the Figure 2.2 instead of five layers, as in TCP/IP. The lower layers deal with electrical signals, chunks of binary data, and routing of these data across networks. Higher layers cover network requests and responses, representation of data, and network protocols as seen from a user's point of view.

The OSI model was originally conceived as a standard architecture for building network systems and indeed, many popular network technologies today reflect the layered design of OSI. The important principles that were applied to arrive at the seven layers are summarized below.

1. The number of layers should be large enough that distinct functions need not be thrown together in the same layer out of necessity and small enough that the architecture does not become unwieldy.

2. Each layer should perform a well-defined function.

3. The layer boundaries should be chosen to minimize the information flow.
There are two additional layers in the OSI model compared to the TCP/IP as shown in the figure above. They are as follows.

**Session Layer:**

The Session layer allows users on different machines to establish sessions for communication between them which may involve synchronization, dialog control, token management and synchronize data flow direction. In TCP/IP its characteristics are provided by transport layer.

**Presentation Layer:**

The Presentation Layer handles data format information for networked communications. This involves dealing with the syntax and semantics of the information transmitted. This
is done by converting data into a generic format that could be understood by both sides.
This function is provided by Application Layer in TCP/IP.

2.4 A Comparison of the OSI model and TCP/IP Protocol suite

It is apparent from Figure 2.3 that the OSI and TCP/IP reference models have much in common. The adoption of TCP/IP does not conflict with the OSI standards because the two protocol stacks were developed concurrently. In some ways, TCP/IP contributed to OSI, and vice-versa [2].

Both are based on the concept of a stack of independent protocols. Also, the functionality of the layers is roughly similar. For example, in both models the layers up through and including the transport layer are there to provide an end-to-end network-independent transport service to processes wishing to communicate. Again in both the models, the layers above the transport are application-oriented users of the transport layer.

Despite these fundamental similarities, the two models also have many differences. Several important differences exist, which arise from the basic requirements of TCP/IP which are: a common set of applications, dynamic routing, and connectionless protocols at the networking level, universal connectivity and packet-switching.

The main differences between the OSI architecture and that of TCP/IP relate to the layers above the transport layer and those at the network layer. OSI has both, the session layer and the presentation layer, whereas TCP/IP combines both into an
application layer. The requirement for a connectionless protocol also required TCP/IP to combine OSI’s physical layer and data link layer into a network level.

There are three concepts are central to the OSI model: services, interfaces and protocols. The TCP/IP model did not originally clearly distinguish between these ideas [2]. As a result, the protocols in the OSI model are better hidden than in the TCP/IP model and can be replaced relatively easily as the technology changes

![Diagram of OSI and TCP/IP layers]

**Figure 2.3 : Comparison between OSI and TCP/IP**
The OSI reference model was devised before the corresponding protocols were invented. With TCP/IP, the reverse was true i.e. the protocols came first, and the model was really just a description of the existing protocols.

Another difference is in the area of connectionless versus connection-oriented communication. The OSI model supports both connectionless and connection-oriented communication in the network layer, but only connection-oriented communication in the transport layer. The TCP/IP model has only connectionless mode in the network layer but supports both modes in the transport layer.

Despite the problems with the OSI model, it has proven to be exceptionally useful for discussing computer networks. The reverse is true of TCP/IP: the model is practically nonexistent, but the protocols are widely used.

2.5 Drawbacks of a static protocol framework

To provide structure to the design of network protocols, network designers organize protocols, and the network hardware and software that implement the protocols in layers. Each protocol belongs to one of the layers. One of the main drawbacks of layering is one layer may duplicate lower layer functionality. For example, many protocol stacks provide error recovery on both a link basis and end to end basis. The second potential drawback is functionality at one layer may need information that is present only in another layer. As we have seen, in TCP/IP, only adjacent layers communicate with each other owing to its standard monolithic layered protocol stack structure. In this regard, the principle of layering is the culprit, and many researchers have been trying to find an alternative. Though the layering principle reduces the complexity of the protocol stack design, it
restricts scalability, functionality, flexibility and the ability to make changes. Hence, it results in a linear system where a layer can communicate and utilize the services of the layer only above or below it.

2.6 Drawbacks of the current Internet architecture

For a better understanding of the shortcomings of today’s internet, we need to understand the motivation and reasoning which lead to the development of it. The top level goal for the DARPA Internet Architecture was to develop an effective technique for multiplexed utilization of existing interconnected networks. The following list summarizes a more detailed set of goals which were established for the design of current Internet architecture [3].

1. Internet communication must continue despite loss of networks or gateways.

2. The Internet must support multiple types of communications service.

3. The Internet architecture must accommodate a variety of networks.

4. The Internet architecture must permit distributed management of its resources.

5. The Internet architecture must be cost effective.

6. The Internet architecture must permit host attachment with a low level of effort.

7. The resources used in the internet architecture must be accountable.

This set of goals might seem to be nothing more than a checklist of all the desirable network features. But the important thing here is to understand that these goals are in order of importance. If we change the order, it results in entirely different network
architecture. It is very clear from these set of goals, that security was the last concern of the above architecture. This resulted to the security problems in the current internet. But, if we develop a new internet today, it is obvious that security would need to be a primary concern.

Networks are risky for at least three major reasons. First and most obvious, more points now exist from which an attack can be launched. A second reason is that the physical perimeter of the computer system has been extended. Messages received may be of uncertain origin; messages sent are often exposed to other systems on the net. The third reason is more subtle, and deals with an essential distinction between an ordinary dial-up modem and a network. Modems, in general, offer one service, typically the ability to log in; there are vulnerabilities in the login service, but it is a single service, and a comparatively simple one. Networked computers, on the other hand, offer many services: login, file transfer, disk access, and remote execution, etc. These services are more complex and difficult to protect.

The existing Internet architecture also stands in the way of new technologies. Networks of intelligent sensors that collectively monitor and interpret things like factory conditions, the weather, or video images could change computing as much as cheap PCs did 20 years ago. But they have entirely different communication requirements. Future networks aren't going to be PCs docking to mainframes. It's going to be about some car contacting the car next to it. All of this is happening in an embedded context. Everything is machine to machine rather than people to people. With today's architecture, making such a vision reality would require more and more patches.
The current internet has shortcomings not only in security, but also in a large number of other different areas including Quality of Service (QoS), robustness, availability, reliability, management, extensibility and adaptability. The National Research Council (NRC) claims, based on results of a 2001 workshop, that the internet is “ossifying” to the point of being unable to change or integrate new technology and new services from which it would significantly benefit.

New high-bandwidth applications have arisen, and many more are expected to arise in the near future, imposing high "Quality of Service" demands on ISPs (Internet Service Providers). Therefore, current and future strong demands for high baud rates per user, as Internet usage increases, require network technology to adapt quickly to the new needs. There are physical factors which restrict or will restrict future required capacity of the network. Those restrictions are based primarily on the bounded capability of future IP routers, to forward "quickly enough" incoming packets to the proper destinations, due to several physical limitations, like finite (not zero) memory access time (needed for searching in the routing table the proper destination port) or switch time (needed to connect input and output ports) of the router. According to many research results on optic cables used for physical transfer of information (i.e. on the physical layer level), a bandwidth in Tbps (Terabytes per second) range is reachable. However, available, commercial, state-of-the art IP routers, can work only at Gbps (Gigabytes per second) rate. So, the most serious, limiting factor on the expansion of Internet is basically the routing capability of IP routers, not bandwidth.

The future Internet needs to support all the different kinds of internets and synchronize them too. Technologies attempting to make this available on the current
internet are very unsuccessful. Hence, the new internet should natively support them all. To make the matters worse, in the future, it is expected that the services required by an application will dynamically change during the course of a connection, and so the network needs to be highly agile and flexible. It is well known that significant quantities of fiber optic cables have already been laid in the ground by numerous telecommunications companies, which are sitting idle [4]. The Internet in spite of its widespread usage, is failing to make the optimum use of these cables.

2.7 MPLS

MPLS stands for Multi Protocol Label Switching, an alternate technology which was created to enable service providers to offer additional services for their customers, scale their current offerings, and exercise more control over their growing networks by using its traffic engineering capabilities [5]. MPLS has a major impact on the Internet in recent years and is seen as a natural fit for organizations deploying wide area applications. It has provided additional security by utilizing Label Switched Paths (LSPs) and allowing the provisioning of Virtual Private Networks (VPNs) [6].

Still, MPLS solutions do not fully address many of the security issues plaguing the Internet today, for example, the access points can be still be hacked. Point to multipoint communication is not possible with MPLS. Network configuration has to be changed which is very costly as special MPLS routers have to be employed. Layer 3 IP/VPNs are proprietary, i.e., one provider, one network, this too adds to the cost. It works only with IP; other standards cannot be used over the network without conversion to Layer 3 [1]. MPLS can be expensive as larger carriers charge for prioritizing traffic in
their annual cost. Apart from all these, there is no provision of adapting new technologies and policies in MPLS.

### 2.8 Requirements of the future internet

The design of any future Internet infrastructure is affected by three unalterable factors. First, the Internet will always be a network of networks. Second, the bandwidth limitation of wired connections will be bounded by the theoretical limitations of fiber optic cables. Finally, wireless communication will be bounded by the restrictions of Maxwell's equations [6].

International Telecommunication Union Telecommunication Standardization Sector (ITU-T) Study Group 13 defined a next generation network in Recommendation Y.2001 as “A packet-based network able to provide telecommunication services and able to make use of multiple broadband, QoS-enabled transport technologies, and in which service-related functions are independent from underlying transport-related technologies. It enables unfettered access for users to networks and to competing service providers and/or services of their choice. It supports generalized mobility which will allow consistent and ubiquitous provision of services to users.”

Recommendation Y.2001 further defines the NGN by the following fundamental characteristics [7]:

- Separation of control functions among bearer capabilities, call/session, and application/service
- Decoupling of service provision from transport, and provision of open interfaces
• Support for a wide range of services, applications, and mechanisms based on service building blocks (including real-time/streaming/non-real-time and multimedia services)

• Broadband capabilities with end-to-end quality of service (QoS)

• Interworking with legacy networks via open interfaces

• Generalized mobility

• Unrestricted access by users to different service providers

• A variety of identification schemes

• Unified service characteristics for the same service as perceived by the user

• Converged services between fixed/mobile

• Independence of service-related functions from underlying transport technologies

• Support of multiple last mile technologies

• Compliance with all regulatory requirements, for example, concerning emergency communications, security, privacy, and lawful interception

Apart from the above stated requirements, the National Science Foundation has specified some features to make the design of the next generation network foolproof. The new architecture discussed in this thesis comprises of all these conditions and characteristics. Following are the assumptions made [2]:

1. **High capacity fiber optic links are going to connect all fixed, or “wired”, network connections.**
The telecommunication industry is laying fiber optic cables and as the trend continues, even the “last mile” will be bridged very soon.

2. **Wireless communications will power the mobile computing devices to the future Internet.**

Existing wireless standards, such as 802.11a/b/g, already support the connection of wireless devices to the Internet across short distances. The phrase, “bridging the last mile” is bound to become a reality with the recent approval of 802.16a protocol called WiMax that supports broadband wireless Internet access to users as far as 30 miles away from a transmission tower.

3. **New router functionality will exist in the form of general purpose processors embedded on network router interface cards.**

With the recent announcement of the introduction of the new line of routers from Cisco, new kinds of applications where significant portions of the application are embedded into the router interface cards is not a distant possibility. This technology will allow routers to execute application specific and network protocol stack specific algorithms at many different layers in the protocol stacks.

4. **Routers will be capable of dynamically loading new protocol modules at the granularity of a single connection.**

Since the framework is dynamic, the routers are given the capability of dynamically loading new protocol modules. When a new connection is being established, the router loads the protocols required for a packet to traverse from the source and end host following the requirements of the host.

5. **The single “flat” Internet consisting of interconnected networks will be substituted by a collection of hierarchically structured and interconnected Internets.**
Though the current internet is made of many different kinds of internets, it doesn’t fully support all of them. The internet was originally made to be an open, “always on” networks and hence the problems with security, QoS, etc. The new internet will be more structured and interconnected along with the option of scalability to different kinds of internets.

6. **Virtual Circuit Switching is used for the future Internet.**
   Taking a cue from ATM, the prerequisite is that all packets of a specific connection will follow the same path to the destination. This will help the network in a lot of ways. Apart from helping in the reduction of traffic due to the creation of the route at the connection establishment time, the connection is released once the data reaches its destination, thus, releasing the routes for further use. It also reduces the overhead required to maintain reliability, breaking up of data into packets and finding all the possible routes to reach the destination.

7. **The future Internet will be highly redundant with a significant amount of replication of resources and underlying infrastructure.**
   As already discussed, there are many fiber optic cables sitting idle. These can be utilized to send multiple parallel link segments which in turn can be utilized in a variety of ways including the support of fault tolerance, security, network management, and embedded primary network services such as route tracing and sophisticated name services. Since it is difficult to setup distinct physical links for wireless networks, distinct connections can be used where data can be broken up into chunks and also transmitted to the destination using spread spectrum techniques.

8. **Core services providing network services is one of the requirements of the network infrastructure of the future Internet.**
The core services include the ability to access and update forwarding tables, low level unique network identification addressing and name translation, support redundant private paths, packet route tracing/tracking, network measurement functions, and the ability to dynamically upload network protocol modules, stack layers, and virtual sub-layers, on a connection by connection basis. The core services are further discussed in the following chapter.

9. **For wireless communications, the packet sizes will need to range from relatively small to relatively large.**
   TCP/IP has a restriction on the maximum network MTU, packet, and message sizes to be not larger than 64K. Though, for performance reasons in wireless communications, it will still be necessary to support the current relatively small MTU and packet sizes utilized in the current Internet, for fiber optic technology however, we assume the packet sizes of at least 1 Mbit (or possibly larger) with message sizes being much larger still. Using current fiber optic technology, an OC-768 link is able to support 39,813 1-Mbit packets per second.

To achieve the NRC’s vision, the future internet should handle many of the current internet’s shortcomings. The new internet should be highly adaptable to rapid changes in technologies, policies and principles. The ability to immediately incorporate new features, new algorithms and protocols that will be required to provide strict resource allocation requirements, new improved security features, and processing requirements that change on a per connection basis.

All these diverse requirements necessitate the need for a change from the current internet. The new internet should be comprised of a conglomerate of overlapping virtual
network architectures provided by a dynamic protocol framework. In order to realize this new network and protocol framework, a new infrastructure foundation that is suitable for supporting a dynamic protocol framework which is flexible, resilient to network changes, dynamically configurable, secure, connection-oriented and redundant is required.
Chapter 3

VIMNet

3.1 Introduction

In order to achieve the goals described in the previous chapter, the future internet must be designed with the intention of handling the shortcomings of the current Internet. It will need to be highly adaptable to handle rapid changes in technologies and policies. It will have to possess the ability to immediately integrate the new algorithms and protocols required to provide stringent resource allocation requirements, increased security requirements, and unique processing requirements via application customized protocols. Each application connection may contain a different arrangement of protocol modules to provide the required features.

The future Internet will also need to be designed with the ability to integrate multiple kinds of networks based upon radically different protocols. It will need to enable different authorities, countries, and regions of the world to form their own radically distinct virtual networks and still be able to interconnect these networks without the use of hard-coded gateways, which are inherently adverse to changes. These diverse requirements necessitate the need for a new dynamic Internet architecture, one that can
support multiple virtual network architectures based upon application customized protocols.

This chapter describes such an architecture called VIMNet which comprises and takes into consideration all the features that an ideal internet should be composed of. Most of the information in this chapter is related to current research in SimNet laboratory under Dr. Lawrence Miller. Some of the contents in this chapter reflect on Dr. Lawrence Miller’s work on VIMNet.

VIMNet is a connection oriented network. Our intention is to move away from connectionless packet switching systems, such as TCP/IP, because they inherently provide an unreliable service. To make these connections reliable, we need to augment several heavy weight protocols. Sometimes this causes severe loss of network transmission bandwidth as more packets are required for retransmissions due to transmission errors, network failures and congestion in the random paths the packet may traverse.

VIMNet uses fiber optic cables to connect network devices in a point to point architecture. Again we are assuming that, by the time VIMNet is commercialized, fiber optic cables will be laid everywhere. There are two different types of paths between any two network devices, such as hosts, routers, etc. These are called private paths and public paths. For each primary data path group, there will be at least one private connection which will be used for a variety of uses including security, router management, and connection setup and connection management. These private paths are very secure and can be accessed only by authorized people. Public paths are used for general data
transmission. All these paths are redundant in number. The proposed architecture for VIMNet is shown in the Figure 3.1 [1]. The FPGA acts as a network interface card; it collects the data from both private and public paths. Then it divides this data into two different chunks, a private chunk and a public chunk. The private chunk contains header related information. When a connection establishment packet is sent from source to destination, it contains all the information regarding the modules to be loaded for that particular connection at each location. The FPGA hands over that connection establishment packet to the core services.

Figure 3.1 VIMNet Proposed Architecture
The Module Management System, which is a part of the core services, generates an XML mapping sheet based on the information from the connection establishment packet. The Module Management System sends this XML sheet to the module chain kernel.

The XML sheet mentioned above contains information about the required module names, the DLLs name which contain required modules, the DLL locations, the Class names, Mapping information, etc. The Module Management System initiates a new process for that particular module chain and sends this XML mapping sheet to the chain kernel. Module Management System manages all the processes, which are generated. The chain kernel parses this XML sheet, and then load’s required DLL’s dynamically based on the mapping instructions. Thus, a module chain will be formed for that particular connection as shown in the Figure. 3.1 [1]. This process is explained more clearly in the following sections.

3.2 Dynamic Protocol framework

The standard monolithic layered protocol stack only allows communication between adjacent layers. This is due to the principle of layering that has been the foundation of protocol stack design and is currently being challenged. The principle of layering reduces the complexity of the system and the design process, but it also restricts the overall functionality and extensibility. The result of this principle is that any layer can only utilize the services of layers below it, and in addition, it must utilize them through the lower adjacent layer, functioning as a proxy. In the past, simple data transfer communication was sufficient to fulfill the needs to network applications. However, since
then there have been increases in the functionality needed from the infrastructure. These needs have included increased security, flexible communication formats, QoS support, real-time support and built-in protection from malicious communication attacks. This has brought about new concepts such as virtualization of network architectures, peer-2-peer networks and overlay networks.

Many attempts to shoehorn these concepts into the current Internet have been tried, but even on those that have succeeded, the workload burden has been upon the end-host systems. All of this has brought about the need for a new protocol model [13]. This model should allow for complex interactions within the protocol environment and significantly increased flexibility that can be easily expanded upon to provide support for and implementation of more complex systems and techniques. Computer's already have a system that is flexible and easily expanded upon that provides a desirable initial model for network communication that is their operating systems. Operating systems allow processes and the kernel to intercommunicate using methods such as inter-process communication (IPC) and message parsing [8]. This communication can be process-to-kernel or process-to-process, as long as the involved parties have implemented the proper required API calls. This allows for flexible, complex intercommunication within the operating system and it is a desired starting point for the implementation of our dynamic protocol environment. In essence, our dynamic protocol framework is a simplified and modified miniature operating system dedicated to network communication. The framework is composed, on the most basic level, of modules that can be connected together at run-time to provide any desired virtual network architecture. The core services support this by providing the basic virtual network architecture modules and the
infrastructure services needed to use them. The rest of the framework includes interfaces, collection objects, and management objects that manage and utilize the modules to provide communication connections for applications [14].

To manage the modules that is module storage, module loading and interconnection, the framework includes a module repository system, module loader and connection manager. The framework utilizes the core services to access the network infrastructure and its respective facilities. The functionality of the core services can be leveraged and expanded by the extra modules that are utilized during an application's connection. This allows for the implementation of advanced techniques and virtual network architectures, including overlay networks, virtual private networks and peer-2-peer networks. Given the open framework design, almost any algorithm or system can be designed and used.

3.2.1 Modules

The smallest defined object is the module. Protocols are broken up into modules, that all implement a common API, so they can be utilized by the protocol framework. As the network itself is inspired from an operating system, just like the processes in an OS, the modules also communicate with other modules located in any of the different protocol chains that handle the communication responsibilities associated with connection. No module has a particular role except for the core services which provide the dynamic protocol framework. Core services can be one single module or a collection of modules put together to perform the tasks.
The only restriction that any module has is that it should implement the module API interface specification so that it is compatible with the core services and the dynamic protocol framework and can be in sync with the rest of the modules. All the other modules can be custom built to implement anything from a simple mathematical calculation to something as complex as a peer-2-peer mesh network architecture. The framework uses these modules and a set of mapping instructions from the applications to generate dynamic protocol chains to implement the desired network system. To maximize the flexibility of the network system, the framework will allow the application to customize and/or design modules to be used in the dynamic protocol chain. Module chains can also contain all the preexisting protocol stack architectures, like TCP/IP, OSI, ATM etc. This enables to reuse the existing protocol stacks in the proposed VIMNet.

### 3.2.2 Module Chaining and Communication

The operating system approach used by the dynamic protocol framework spurs the usage of two inter-protocol module communication (IPMC) techniques in order to facilitate communication between modules. These two techniques are:

1. Direct Chaining.
2. Indirect Chaining.

As shown in Figure 3.2 [1], in the Indirect chaining technique, the communication is disjoint and haphazard. This leads to confusion, an increase in complexity, and thus eventually leading to deadlocks and race conditions which in turn will lead to a diminished QoS, real-time communication, security, etc.
To surmount this problem, VIMNet makes it necessary for every module to conform to the API in the core service. By conforming to the API, the modules that are loaded on the core services are connected to each other by a “stack-like” direct chain. This protocol chain can be visually thought of as a monolithic stack composed of a series of layers. But since there is no standard number of layers in VIMNet, these chains are dynamically formed as requested by the end host application.

Coming back to direct and indirect chains, with the help of the API, these can be implemented individually or in unison. The framework allows this without much complexity and with efficiency. In all connections, one master chain exists that provides the connection to the application. To provide additional functionality, there can be any number of indirect chains linked to the master chain. This is illustrated in Figure 3.2 [1]. For the initial test bed, we are using Direct Only chains. Indirect chains will be implemented in the second phase of the project.

3.2.3 Software Model and Connection Manager

Since the dynamic protocol framework works like an operating system, VIMNet uses the same software model for both the router and the end host systems of the network. The system model of the end host system is illustrated in Figure 3.3 [1].
Figure 3.2 Inter-modular Communication

The connection manager maintains all the virtual circuit connections. It has a system that handles all of loaded and linked protocol chains and controls the flow of chains along with their application connections. The connection manager makes use of a module repository, a prerequisite validation object, a module loader object and a protocol chain linker object to generate the required chains.

The protocol chains are created when the end host decides to establish a connection and so creates its own protocol chain first and then creates the connection
establishment packet. Once the packet reaches the destination and the connection is established, the connection manager takes care of all the requirements of the interconnection of the master chain connection to the application. The packet contains the end hosts’ connection data, a list of all the modules required to create the protocol chain, a connection mapping for the modules and other required data.

As the packet passes through the network infrastructure building a virtual circuit, the requested router modules are loaded and connected. When the packet reaches its destination(s), the requested protocol chains are built and then an acknowledgement is sent, so that actual data transfer can proceed.

Figure 3.3 End Host System Model
3.3 Virtual Circuits

In our approach, the network infrastructure is based upon a virtual circuit/virtual path based system similar to that of ATM and MPLS networks. The virtual circuits provide the connections between components within the networks and between the end-hosts [9]. The core services provide the data structures and functions to create, manage and dispose of virtual circuits. Virtual circuit management includes forwarding of data through the virtual circuit, the control of data structures containing the virtual circuit data and the monitoring for activity to provide a timeout procedure to dispose of inactive virtual circuits that are not using their allocated network resources.

3.4 Connection Establishment

Once the core services provide the data structures and functions, the virtual circuit is built using them, i.e. a connection is established. A virtual circuit is just a one way path from the sender to the receiver. Therefore, for two ways communication, a return path is also necessary, i.e. paths for connection request packet and the connection acknowledgment packet. During the course of a connection establishment, the services interact with the virtual circuit, routing, constraint, resource allocation, and network infrastructure services. We discuss connection establishment process in detail in the following sections.

3.5 Redundant Paths

The base of the new design is a series of hierarchically layered mesh networks composed of multitudes of virtual circuit routers and high speed single and multi-wavelength optical
fiber trunk connections. These trunk connections are further divided into 2 types – private paths and data paths, such that every path has at least one data path and one private path. But since there is a lot of underground optical fiber cables already laid, it is assumed that there would be a dozen or more data paths grouped with several private paths which could be used in parallel or redundantly in practicality.

One of the architectural requirements for the future Internet’s network infrastructure is the provisioning of redundant data path (RDPs) links. These redundant links will be organized into collections of redundant data paths (RDPs). Each of these RDPs will be split into two.

1. Primary Data Paths (PDPs)/Reserved Path Links (RPLs)
2. Redundant Private Paths (RPPs)/ Application Path Links(APLs)

The user data flows across PDPs and the network management and administration data flowing across RPPs. In the design, it is assumed that every physical link in a path consists of multiple parallel link segments. This design is intended to scale all the way down to the end hosts.

However, where the core system may have tens to hundreds of parallel link segments, the edge networks and end hosts may only have a few. Ideally, each end host should have several primary data connections coupled with at least one private data connection. It is possible to implement these two connections using a multiplexed single path connection that requires a module on the local router to de-multiplex and redirect packets to their correct path.
RPPs will provide a secure medium through which security and system information can be exchanged regarding user data transmission. They can be utilized in a variety of ways, including the support of fault tolerance, security, network management, and embedded primary network services such as route tracing and sophisticated name services. The security information can be used to securely and uniquely identify the source hosts/networks for user data entering the network, to securely track data traversing the network, to securely exchange encryption keys, and to transmit message identifiers to identify the types, size, and content of user messages traversing the network. The intent is to provide mechanisms which enhance the security of data traversing the network, and which makes the network infrastructure itself more impervious to threats. In addition to providing secure paths for the exchange of network security information, RPPs can be utilized for many other control purposes as well. They can be used for exchanging many types of control and management information, including channel establishment messages, routing protocol update packets, congestion notification packets, congestion management messages, informational messages containing network performance statistics and network self monitoring and measurement data. In VIMNet, RPPs are utilized to establish the connection also.

Along with the protocol modules chains suggested for the inter-modular communication, the use of APLs and RPLs greatly improves security in the network. Multi-Protocol Label Switching (MPLS) used Label Switching Paths (LSPs) and allowed the usage of Virtual Private Networks (VPNs) to improve the security problems in TCP/IP networks along with addressing majority of problems like QoS, availability, reliability and traffic engineering. Even then, MPLS doesn’t fully solve the security
issues of TCP/IP. However, using RPLs, the network can securely track and manage the network by allowing network devices to exchange information without causing any bandwidth problems.

RPLs provide the facility to construct private and secured data packets that encode information like encryption keys, CRC and parity codes and securely transmitted trace route data, etc. Also RPLs are used to exchange information regarding modules, their construction, chain formation, updates, newer versions or even the modules themselves.

3.6 Comparison between TCP/IP and VIMNet

The differences between TCP/IP and VIMNet are as shown in Table 3.1:

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>TCP/IP</th>
<th>VIMNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flexibility of the framework</td>
<td>Ossified</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Fixed number of layers with predetermined functions.</td>
<td>Highly Flexible</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Layers dynamically loaded at runtime as per the need of the application.</td>
</tr>
<tr>
<td>Security</td>
<td>Not Secure Enough</td>
<td>Very Secure</td>
</tr>
<tr>
<td></td>
<td>• Packets can be sniffed.</td>
<td>• Data is divided into two chunks and sent through private and public paths.</td>
</tr>
<tr>
<td>Reliability</td>
<td>Packet Switching</td>
<td>Virtual Circuit Switching</td>
</tr>
<tr>
<td></td>
<td>• Packets can be lost.</td>
<td>• The probability of losing a packet is very less.</td>
</tr>
</tbody>
</table>
### Table 3.1: Comparison between TCP/IP and VIMNet

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>TCP/IP</th>
<th>VIMNet</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Congestion Problem</strong></td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>• Routers not capable of handling traffic.</td>
<td>• Very efficient routers.</td>
</tr>
<tr>
<td></td>
<td>• Packets of same connection may follow different paths.</td>
<td>• Path fixed for packets of various connections.</td>
</tr>
<tr>
<td></td>
<td>• Congestion dealt with during routing.</td>
<td>• Congestion taken care of at the time of connection establishment.</td>
</tr>
<tr>
<td><strong>Network Bandwidth Utilization</strong></td>
<td>Not Satisfactory</td>
<td>Excellent</td>
</tr>
<tr>
<td></td>
<td>• Allows very small packet size; bandwidth wasted.</td>
<td>• Allows large packets; entire bandwidth used.</td>
</tr>
<tr>
<td><strong>Packet Size</strong></td>
<td>Very Small</td>
<td>Large</td>
</tr>
<tr>
<td></td>
<td>• Less information.</td>
<td>• More information.</td>
</tr>
<tr>
<td></td>
<td>• High error rate.</td>
<td>• Very low error rate.</td>
</tr>
<tr>
<td></td>
<td>• More processing time.</td>
<td>• Reduced processing time.</td>
</tr>
<tr>
<td><strong>Extensibility</strong></td>
<td>Poor</td>
<td>Excellent</td>
</tr>
<tr>
<td></td>
<td>• Unable to support new technologies like IPv6 and multicasting.</td>
<td>• Modularized, hence capable of supporting every kind of technology.</td>
</tr>
</tbody>
</table>

3.7 Comparison between MPLS and VIMNet

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>MPLS</th>
<th>VIMNet</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Flexibility of the framework</strong></td>
<td>Not Flexible Enough</td>
<td>Highly Flexible</td>
</tr>
<tr>
<td>4. Can be used with different type of networks</td>
<td>6. Layers dynamically loaded at runtime as per the need of the application.</td>
<td></td>
</tr>
<tr>
<td>5. Fixed number of layers with predetermined functions.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Security</strong></td>
<td>Not Secure Enough</td>
<td>Very Secure</td>
</tr>
<tr>
<td>7. Packets can be sniffed.</td>
<td>• Data is divided into two chunks and sent through private and public paths.</td>
<td></td>
</tr>
<tr>
<td>Characteristics</td>
<td>MPLS</td>
<td>VIMNet</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Reliability</strong></td>
<td>Virtual Circuit Switching</td>
<td>Virtual Circuit Switching</td>
</tr>
<tr>
<td></td>
<td>• The probability of losing a packet is very less.</td>
<td>• The probability of losing a packet is very less.</td>
</tr>
<tr>
<td><strong>Congestion Problem</strong></td>
<td>Low</td>
<td>Lower</td>
</tr>
<tr>
<td></td>
<td>• Routers not capable of handling traffic.</td>
<td>• Very efficient routers.</td>
</tr>
<tr>
<td><strong>Network Bandwidth Utilization</strong></td>
<td>Not Satisfactory</td>
<td>Excellent</td>
</tr>
<tr>
<td></td>
<td>• Allows very small packet size; bandwidth wasted.</td>
<td>• Allows large packets; entire bandwidth used.</td>
</tr>
<tr>
<td><strong>Packet Size</strong></td>
<td>Very Small</td>
<td>Large</td>
</tr>
<tr>
<td></td>
<td>• Less information.</td>
<td>• More information.</td>
</tr>
<tr>
<td></td>
<td>• High error rate.</td>
<td>• Very low error rate.</td>
</tr>
<tr>
<td></td>
<td>• More processing time.</td>
<td>• Reduced processing time.</td>
</tr>
<tr>
<td><strong>Extensibility</strong></td>
<td>Poor</td>
<td>Excellent</td>
</tr>
<tr>
<td></td>
<td>• Unable to support new technologies like IPv6 and multicasting.</td>
<td>• Modularized, hence capable of supporting every kind of technology.</td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td>Expensive</td>
<td>Economical</td>
</tr>
<tr>
<td></td>
<td>• Special MPLS routers need to be configured and employed.</td>
<td>• As ViMNet is software-based, its functionality can be added to current routers.</td>
</tr>
</tbody>
</table>

Table 3.2: Comparison between MPLS and VIMNet.
Chapter 4

Virtual Circuits

4.1 Introduction

Virtual circuits are a connection oriented communication between two nodes which uses packet switching technology. All the packets pertaining to a data flow follow the same path from source to destination. This guarantees the delivery of the message and reliability. The destination sends back an acknowledgment after receiving each packet. The source on receiving the acknowledgement understands that the message has reached the destination without any errors. In case of any errors detected at the destination, the destination simply discards the packet as if it did not receive any packet. Hence there won’t be any acknowledgement sent back to the source. The source upon not getting any acknowledgement sends the packet again assuming that the packet sent was lost and did not reach the destination. This features accounts for the reliability of the data flow.

The circuit set up is done in such a way that after a successful data transfer, the connection is terminated. This accounts for efficient utilization of resources. This set up is analogous to a phone call. When a caller wants to communicate with another person,
there is a connection setup and as soon as the conversation is finished the connection is terminated.

4.2 Virtual Circuit numbers

The circuits along the path are identified by a number know as Virtual Circuit numbers (or VC numbers) which is unique to the circuits along the path. All these numbers are stored in a table as entries which are called as VC number transition table [10]. In a full duplex communication these numbers will be different for each switch to indicate the flow of data in different directions utilizing the same virtual circuit.

Once a VC is established between source and destination, packets can be sent with the appropriate VC numbers. Because a Virtual Circuit has a different Virtual Circuit number (VC number) on each link, an intermediate packet switch must replace the VC number of each traversing packet with a new one in the translation table as soon as the packet leaves and enters a new virtual circuit.
Figure 4.1: A basic architecture involving virtual circuits for multicasting[1].

4.3 Phases in Virtual Circuits:

The three phases involved in the operation of Virtual Circuits are

1. Connection set up.

2. Data Transfer.

3. Connection termination.
During the setup phase, the source contacts the network layer (in the TCP/IP model), specifies the receiver address, and waits for the network to setup the Virtual Circuit. The network layer determines the path between sender and receiver, i.e., the series of links and switches through which all packets of the VC will travel. In the VIMNet model, the core services upon the request for connection set up identify the receiver and its details like the physical and network address. Then it designs a route path and sets up the virtual circuit. Once the packet leaves one circuit and enters another core services sends a request for updating the tables with the VC numbers for the route. This will be dealt more in detail in chapter 5.

Once the connection has been setup the data flow takes place and this can be fully duplex. Once the data transfer phase is completed, the source or the destination informs the network layer (in the TCP/IP model) about the status of the data transfer and request for connection termination. The network layer then sends a circuit termination message which deletes all the entries of the transition tables at the switches which stored the VC numbers for the circuits formed during the connection setup. In the VIMNet model, the Core Services send the circuit termination packet to all the components involved and the entries in the transition tables will be deleted. This phase is also called as tear down phase.

The VC numbers can be same for the links along the route. This helps in reducing the length of the VC field in the header. Also this repeated usage helps the management function of the network be simple as it doesn’t has to generate different numbers everytime and keep a record of it. Once the link is used for a common VC number the
routers just have to exchange information on the VC number for that particular connection. When the connection is torn down all the information in the tables is erased.

While using the virtual circuits the, switches in the network must maintain the state information for the ongoing connections. When a new connection is established, the entry for this particular connection across the switches must be updated to virtual circuit translation table [10]. Now, let us briefly discuss about the usage of virtual circuits in X.25 and ATM.

4.4 Virtual circuits in X.25:

X.25 is one of the oldest packet-switched services. It implements two types of virtual circuits. One being virtual calls and permanent virtual circuits. The virtual calls are established when the need for a circuit is only desired, which is again torn down when the call is complete (i.e. when the data transfer is complete). This is done using X.121 addresses which are a three digit Data Country Code or DCC concatenated with a network digit making it a four digit Data Network Identification code or DNIC. The permanent virtual circuits are permanently established and does not need the usage of above addressing system for a call setup. The PVC are associated with numbers called as logical channel identifiers. These numbers identify a specific logical channel between the subscriber and the network. These significance of these numbers is only local corresponding to the link between the subscriber and the network. The range of possible logical channels is split into four groups which are [10]

1. Channels assigned to the permanent virtual circuits.

2. Channels assigned to incoming virtual calls
3. Channels assigned to full duplex (two way) virtual calls

4. Channels assigned to outgoing virtual calls.

Frame relay also employs a similar system to X.25 but does not implement error control and flow control. It can be called as a second generation X.25. The virtual calls are replaced by the term switched virtual circuits and the permanent virtual circuit is used here too [11].

4.5 Virtual circuits in ATM:

In ATM terminology, the packets are referred to as cells and the Virtual Circuits are called as Virtual Channels. The transmission involves, breaking down the datagram into cells and transmitting them. At the receiving end the cells are reconstructed. There is no retransmission involved and in case of an error in a cell, it is rectified at the receiving end. If it is not able to correct the error, it simply drops the cell and doesn’t send a request for retransmission. Here when the datagram is to be sent to a destination, the router indexes its routing table and determines the destination IP address. The router indexes a routing table called ARP table with the IP address of the destination and determines the ATM routing path for it. This involves creating a Virtual channel. Now each channel is identified with an identifier called as Virtual Channel Identifier or VCI. Once the destination VCI is identified it is mapped to the ARP table. Once this process is done, the datagram is segmented into cells and transmitted through the channel. Most of the times in architecture these channels are permanent. They are called Permanent Virtual Channels or PVC. For these channels the datagram sending side maintains a table that maps the ATM address to the VCIs [10]. If the VCs are not permanent, the VCs follow the 3 steps
of connection establishment, data transfer and connection teardown. All this is done dynamically. The usage if a Permanent Virtual Channel negates the use of the above mentioned three stage process. A typical ATM header contains a VCI, payload type PT, cell loss priority bit CLP and header error checksum or HEC byte. Note that data is not resent in ATM as any loss or error occurs is fixed at the destination end. The ATM cell header is shown in the following figure 4.2 [10].

<table>
<thead>
<tr>
<th>VCI</th>
<th>CLP</th>
<th>PT</th>
<th>HEC</th>
</tr>
</thead>
</table>

Figure 4.2: ATM cell header

Now that we discussed about the concepts of virtual circuits and its use in the technologies like ATM, let’s move onto the next chapter where the implementation of these concepts in the Virtual Modularized network is dealt.
Chapter 5

Implementation of Virtual Circuits in VIMNet

The last chapter introduced the concept of Virtual Circuits and gave a brief overview. We will now discuss the implementation of the Virtual Circuits in the Virtual Modularized network. As mentioned in the section 4.3, the implementation involves 3 stages which are Connection setup, Data transfer Connection termination.

5.1 Connection Setup:

In the connection setup phase, the source which wants to send data to a destination sends a connection request to the core services. The core services which provide the connection establishment services that contain connection request packet structures and functionality and the connection establishment process starts within the dynamic protocol framework. Upon receiving the request, the core services identify the destination and its address. Then it maps a virtual circuit connection between the source and the destination involving various links covering the routers. A transition table will be formed by the core services at each node through which the data hops i.e. at each router. After the protocol framework has generated its customized protocol setup, it then generates a connection establishment packet. This packet contains the destination’s connection data given by the
core services, information about the protocol environment to be used and any necessary constraint data. This is shown in the figure 5.1 [1].

<table>
<thead>
<tr>
<th>Virtual Circuit Identifier</th>
<th>Data Segment</th>
<th>CRC and Parity</th>
<th>Trace route trailer</th>
</tr>
</thead>
</table>

**Figure 5.1: Raw packet**

Now this information is sent back to the source which now ends a connection establishment packet through the path of the virtual circuit. When this packet reaches the destination(s), all of the information is processed and then the destination sends a denial or an acknowledgement back to the source indicating that the connection is set up. Now we should note that the acknowledge May/may not use the same path depending allocation of resources. Even when the acknowledgement packet traverses the same path as the connection packet in the reverse direction, the virtual circuit numbers along the links in the reverse direction will be different. Different numbers are used to indicate the direction of the data flow along the virtual circuit. After receiving the acknowledgement, the source starts the process of data transfer thus completing the three way handshake. The process happening during the connection setup can be seen in the follow chart below.
5.2 Data transfer:

The source starts transmitting the packets along the route upon receiving the acknowledgement from the destination. The virtual circuit translation table will be updated at each node with virtual circuit numbers on the router along the links as the data traverses through them. For every packet received at the destination, the destination sends
back an acknowledgement to the source about the received packet. This ensures the reliability of the data transfer involved. The figure 5.3 shows a flow chart of the data transfer phase.

Figure 5.3: Data Transfer phase.
5.3 Connection Teardown:

After the completion of the data transfer, the destination sends another acknowledgement that it received the complete data segments. Upon receiving this acknowledgement the source sends a message to the core services about the completion of the data transfer. The core services then generates a connection tear down packet. The connection tear down packet basically erases all the information in the virtual circuit transition tables. The core services send this packet along the route of the virtual circuit. At each router, the virtual circuit translation table is reset i.e. all the entries in the table are deleted. This would tear town all the links associated with each router and hence the virtual circuit is terminated. The process can be seen in the flow chart below.
After the completion of the data transfer, the destination sends the source an acknowledgement.

The source sends a request to the core services for connection tear down.

The core services send a connection termination packet along the virtual circuit.

The entries in the VC translation tables are deleted.

Figure 5.4: Connection tear down phase

Now the above scenario reflects a unicast situation. A multicasting situation would reflect multiple unicast scenarios. This can be seen in figure 5.8.
5.4 The router port numbering:

The ports on the router are classified as left side and right side for convenience. Figure 5.5 shows this depiction.

![Router indicating three left and right ports](image)

**Figure 5.5: Router indicating three left and right ports**

Let us assume there are 9 routers in the network excluding the source and destination. Each router has the same configuration as the other and has six ports altogether, three on each of its sides. Let us also assume that the data enters each router through the left side and exits to the destination from the right side. The three ports on the left side indicate the incoming ports and the three on the right side indicate the outgoing ports. The numbering for each port is a 3 digit number in our implementation. If the 3 digit number is of the form ABC, A indicates the router number, B indicates the side of the router and C indicates the port number. Let’s say in the network of 9 routers, they be
named 1 through 9 in the increasing order. The first router will be named router 1. Hence A is 1 for router 1, 2 for router 2, 3 for router 3 and so on, thus the range of A being 1-9. B would be 1 if it is a left side port and 2 if it is a right side port, thus the range of B being 1-2. Since we assumed that each side of the router has 3 ports, C ranges from 1-3 with the first being 1, the second being a 2 and the third being a 3.

In the above example, the network is limited to 9 routers and the ports are two sided with three on each side. If we want to extend the number of routers or the number of ports, we just have to increase the range of A, B and C. For example if the number of routers is 99, then the numbering on each port would be a four digit number in format ABC with A being a two digit number. This can be seen in the figure below.

5.5 Internal routing for a Router:
The data entering through the ports in the router can choose their own path for their journey towards the journey depending on the factors like available ports for transmission, congestion or traffic. Now this is chosen route is monitored and updated in the virtual circuit translation table. For indicating this route, we simply concatenate the addresses of the entry port and exit port of the router. For example consider the following figure 5.5.
Here in the above figure, the data enters the router through port#2 and exits the router through port#1. The address for the data entering the port is 112 and the address for the data exiting the port is 121. Hence this route can be identified by assigning it as 112121. Looking at the this six digit number in the translation table, the route can be said as the data is entering the router#1 from the left side through the second port and exiting the router through first port on the right side.

Let us consider one more example where the data from source A is transferred to the destination B through a single router in an architecture which consists of 9 routers. The source and the destination have different identifiers assigned on the network. Let us
assume that identifier for the source A be a two digit number and the identifier for B be a four digit number for convenience to distinguish them from the routers. The routing which takes place is shown in screenshot below.

In the screenshot below (figure 5.7), there are two paths one for transmission of data from the source A to the destination B and the other a return acknowledgement from the destination B to the source A. We have to note here that the return path need not be the same path as transmitted from the source.

![Routing diagram]

Figure 5.7: Routing function between the source and destination

We have the flexibility of using a different channel as once the transaction is completed the circuits are torn down anyway. This is an example of a unicast scenario. For a multicast scenario, the following screenshot (figure 5.8) is an example.
5.6 Role of Core Services:

The core services play an important role in all the three phases involved in implementation of virtual circuits in the Virtual Modularized network. Upon receiving a connection request, the core services assign the identifiers for the source and destination. Once the initial job of assigning the identifiers is done, the core services update the translation tables in the routers. When the connection packet is sent and the connection is
setup, the data transfer begins. When the data transfer begins, it follows the route mapped by the core services along the links of the virtual circuit. Now during the connection tear down phase, once the acknowledgment is received back the executable file which is associated with the virtual circuit translation tables is terminated by the core services.

In addition to the basic virtual circuit services, the core services must extend these services to provide the ability to generate sub-virtual circuits. These sub-virtual circuits will then be able to be combined by the utilization of special modules to implement virtual network architectures. These modules will load onto the routers that are the interconnection point of sub-virtual circuits during connection establishment. The modules are delegated control over the routing of data packets for the virtual network architecture. They can then forward or replicate and forward data packets as necessary to provide the desired virtual network implementation. This can be used for direct implementation of multicast, broadcast or any cast connections. These virtual network architecture modules will be provided directly by the core services. Beyond these default modules, dynamic protocol framework modules can implement other virtual network architectures [1]. An example of multicasting virtual network architecture can be seen in the above Figure 5.7.
Chapter 6

Conclusions and Future Work

The initial chapters indicated the need for an overhaul of the current architecture of the internet for achieving better quality of service and reliability. Chapter 2 discussed the TCP/IP and OSI reference models and made a comparison. The drawbacks of the existing internet are discussed and the areas which are needed to be addressed are highlighted. It also explained the drawbacks of the static protocol framework. Chapter 3 explained the need for a dynamic protocol framework and introduced the concept of Virtual Modularized network. Chapter 4 explained the concept of Virtual circuits and discussed its various stages of operation. Chapter 5 of this thesis clearly explained the implementation of the Virtual circuits in the Virtual Modularized network.

This thesis gave an overview of the most important portions of the VIMNet architecture. It justified why the new architecture is better than the old one by comparing the corresponding features. It discussed the components of VIMNet like Core Services, Dynamic Protocol Framework and Network Infrastructure, describing their characteristics and working. A comparison with existing protocol suites of TCP/IP and OSI model along with the alternate architecture of MPLS was made. This augmented the need for a better quality of service and robustness in the architecture. Virtual circuits, a
key part of the routing architecture and its implementation in VIMNet involving different stages of routing are discussed. The numbering for the routers’ ports have been proposed and shown the ease of their use in implanting the routing function.

Future work for this design would be incorporating a suitable routing algorithm into the proposed virtual circuit implementation according to the specifications of the dynamic protocol framework. This would implement a best possible routing scenario in all the data transfers involved in the architecture. The design and implementation of the Virtual circuits concepts proposed in this thesis are designed for an architecture which can implement the numbering scheme for the routers. The initial examples were shown with 9 routers and 99 routers. Extending the number of routers and ports would be increasing the number if digits and their significance to the port numbering involved. The implementation of this design was not done in a real-time system as the Core services functionalities could not be utilized because of the unavailability of the hardware setup involving the FPGA.

Once the Initial Core Services work in a real network with multiple routers and hosts, the implementation can be integrated with the Core services running the functionalities for assigning the identifiers for the different nodes involved. Hence the connection request process, the translation tables and the connection can be extended for bigger architectures. In an attempt to keep the implementation as general as possible, the programming language used here is native.
References


