This is to certify that the
dissertation entitled

A Cluster Architecture Supporting Network Emulation

presented by

Pei Zheng

has been accepted towards fulfillment
of the requirements for the

Doctoral degree in Computer Science

Lionel Ni

Major Professor's Signature

6/25/03

Date

A CLUSTER ARCHITECTURE SUPPORTING NETWORK EMULATION

By

Pei Zheng

A DISSERTATION

Submitted to

Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science and Engineering

2003

ABSTRACT

A Cluster Architecture Supporting Network Emulation

By

Pei Zheng

Network research generally requires a simulation or emulation environment to test and evaluate the performance of protocols, algorithms, services, and applications, or to study the complex and highly varying network behaviors in both wireline networks and wireless networks. When the target network is sufficiently large, simulation of the target network will consume a large amount of time and memory, and its result is mostly based on many modeling assumptions. Existing network emulators can only support end-to-end network emulation because they fail to emulate network topology. In addition, network emulators are non-scalable due to the limitation of available physical infrastructure and the one-to-one mapping scheme. In this research, we present a scalable distributed network emulator cluster named EMPOWER, which not only can be used to emulate a large network with moderate cost, but also can generate user-defined network conditions and traffic dynamics at packet level. EMPOWER is highly scalable in that each emulator node could be configured to emulate multiple network nodes according to some predefined topology. By configuring the emulated network nodes on each emulator node across the entire emulator cluster, an emulated large scale network topology can be created using a considerably limited number of emulator nodes, making it possible to

facilitate topology related research. Some significant research issues such as network topology mapping, resource competition in an emulator node, the overhead of emulation, and wireless network emulation, are discussed and addressed. Our emulation experiment with the prototype show that EMPOWER is capable of assisting the study of both wireline and wireless network protocols and applications.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Lionel M. Ni, for his continuous support and help throughout the research project. It was definitely my great pleasure to conduct this interesting and challenging research under his guidance. His broad and profound knowledge and his effective instruction have given me a great help.

I am also very grateful to my academic committee members, Dr. Abdol H. Esfahanian, Dr. Matt W. Mutka, and Dr. Tien-Yien Li for their valuable advice and inspiring comments.

Many of my colleagues have contributions to my dissertation. I would like to thank Manqing Huang, Yong Chen, and Baijian Yang for their insightful comments and suggestions.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

1.1 Introduction and Motivation

The past ten years see an exploding growth of the Internet and related network services and applications. As the network infrastructure shows strong potential to provide even more reliable, adaptive, high-performance services, researcher in network community are focused on a number of research issues in Wide Area Networks (WANs), such as the development and performance evaluation of new protocols to provide fast, reliable and fault-tolerant network communications, the integration of various QoS (Quality of Service) supports in both wireline and wireless networks, and the development of efficient and flexible software for network operations and management. To cope with these issues, a scalable, flexible, and controllable network system is highly demanded for both network researchers and Internet Service Providers (ISPs) in terms of performance evaluation of new protocols, comparison of several related protocols, exploration of specific behaviors, and interactions between multiple protocols or applications.

Five existing approaches are available for this purpose: network simulation, Parallel Discrete Event Simulation (PDES), small-scale testbed, large-scale testbed, and network emulation. Network simulators [1-4] generally provides an event-driven, synthetic conceptual network environment in a single operating system kernel. Most network simulators are able to provide a rich set of protocol modules and extensions and are easy to configure for the simulation of complicated networks. However, those protocol

modules and extensions in network simulators are not real implementations but simulated logical code. Furthermore, simulation time will increase dramatically as the complexity of the simulated network increases. PDES is proposed to improve the simulation speed by executing a single discrete event simulation program on a parallel computer. It is known to be difficult because it is extremely hard to maintain the correct order of computation in a parallel environment [5]. A small-scale testbed is a simple laboratory network environment that may resemble some portion of the target network in terms of network topology or traffic characteristics. Large-scale testbed can be the same of the entire target network, or a similar wide area network. The problem of both a small-scale testbed and a large-scale testbed is that they cannot generate specific network conditions and traffic dynamics that may be critical to test a network system. In addition, a large-scale testbed is costly to establish and most likely unaffordable to researchers. Network emulation is the execution of real network protocol implementation code in a fully controllable and reproducible laboratory network environment, which is usually a Local Area Network (LAN) that is configured to emulate a real network. Unlike network simulation, the protocols and applications as well as the interaction between protocols are "real" in an emulation environment. Network traffic physically traverses the emulation environment, in which underlying protocols and applications are tested and evaluated against user-defined network conditions and traffic dynamics, such as packet latency, link bandwidth, packet drop rate, Bit Error Rate (BER), and link failure. In a sense, network emulation can be regarded real-time network simulation that makes use of underlying computing infrastructure as the running gear to provide a fully controllable testbed.

Existing network emulators can be divided into two categories: standalone emulators and emulator environment. Standalone emulators can be used to emulate a router or an Internet "cloud" with measured end-to-end network parameters and traffic models. As a result, they cannot emulate network topology. An emulator environment consists of several standalone emulators, each of which emulates a real node in the target network. To emulate a wide area network with a large number of network nodes, the same number of emulator nodes is needed according to the one-to-one mapping scheme, which simply re-establishes the target network in a test network with the same number of network nodes and the same network topology. Due to the high cost of such a large scale test network, most emulator environment systems only have less than 10 nodes, thereby in effect fail to support large scale network emulation.

In an effort to support mobile wireless network emulation, some network emulators [6, 7] are specifically designed to support mobile wireless networks. They typically use some analytical models to emulate a wireless link or a channel. Therefore, their emulation capability is largely limited to wireless link emulation rather than emulate MAC layer or above. Wireless network emulators mostly lack means to emulate the mobility of mobile nodes, which is critical to the emulation of a dynamically changing network topology.

In this research, we propose a flexible and scalable distributed network emulator cluster called EMPOWER (EMulation the Performance Of WidE aRea networks). EMPOWER is able to emulate both wireline and wireless networks using a number of commodity computers in a local area network. It is able to faithfully emulate a target network by mapping the target network into a distributed laboratory network

configuration with certain number of computers with certain network topology, and automatically and precisely generates multiple sets of predefined network conditions and traffic dynamics for an emulation experiment. Each emulator node in EMPOWER can be configured to emulate multiple routers and links, making the entire system highly scalable in emulating large wide area networks. With EMPOWER the mobility of a mobile wireless network can be easily emulated in a wireline network. MAC layer and above protocols of mobile wireless networks can also be tested with EMPOWER. Since multiple emulated nodes share the same physical emulator node, the impact of resource competition among emulated nodes in terms of emulation accuracy and scalability must be examined. Our experimental results show that EMPOWER is quite accurate in emulating network performance parameters such as fixed and patterned packet delay, empirical packet delay distribution, fixed and patterned packet loss, and fixed bit error rate. In addition, experiments of network topology emulation and mobile network emulation show that EMPOWER can faithfully emulate network topology.

1.2 Concepts and Terminologies

Prior to a detailed description and analysis of the research in network emulation and simulation, a number of frequently used concepts and terminologies need to be clarified. Please note that our definition and description are based on our understanding of related issues for the ease of our research. Other researcher may use similar and a little different terminologies.

- Network simulation and network simulator

Network simulation is the approach to investigate network research issues by first modeling networks and services using certain abstraction and formulation methods, establish a logical scenario of related components and events, and then perform event driven synthetic computation among all the components in a simulation scenario. Generally simulation makes use of "virtual time" as in contrast to real time for event scheduling. Network simulator is *a standalone program* that provides logical protocol modules and extension as well as scenario generators for network simulation. Network simulation is actually conducted on the application layer of underlying system running the simulation software; simulation results are based on simulated operations of some analytical models of related protocol components.

- Network emulation and network emulator

Network Emulation can be regarded as real-time simulation that uses real computer systems and networks as the platform and protocol modules. Network emulator is a fully controllable and reproducible *network environment* that can be configured to generate specific network conditions and traffic dynamics. Testing modules and extensions are real implementations interacting with the protocol stack of underling system running emulator software. Emulation result are obtained from field measurement in the emulation environment.

- Traffic dynamics

Network traffic dynamics refer to a set of packet-level effects such as packet delay, packet delay jitter, packet drop rate, and bit error rate. One of the major tasks of network emulation is to emulate network traffic dynamics such that

protocols and applications can be tested against specific network effects. A specific emulator software module is needed to generate such network dynamics and apply them to selected traversing packets. This is typically done in the kernel of the emulator system.

- Network conditions

  Network conditions refer to dynamic changes of network links and topology such as link bandwidth change, link failure, and router failure. Network emulator should provide means to generate network conditions to closely mimic a real target network. This function is very important to emulate dynamic network topology.

- Emulation Scenario

  An emulation scenario is a complete specification of emulation configuration for a certain experiment. It should include the configuration of the emulated network, desired network conditions and traffic dynamics to be emulated, measurement methods, and performance metrics.

1.3 Problem Statement

Based on the analysis of existing network emulation research, we present the problem statement of this research as follows.

Using low-cost commodity computer hardware, we intent to design a scalable and flexible network emulator system that supports network topology emulation and wide area network emulation with limited number of computers. The emulator should not only

be able to emulate common network conditions and traffic dynamics, but also be able to incorporate user modules and extensions as plug-ins for specific purposes.

1.4 Our Contribution

Our contribution to the research of network emulation can be summarized as follows.

- We proposed a low-cost, scalable, distributed network emulator cluster architecture. Such an approach is believed to be the first in this area, to the author's best of knowledge. The emulator cluster supports large-scale network topology emulation with a considerably smaller number of machines.
- We investigated the problem of resource competitions and emulation overhead in the design of single node emulator and distributed network cluster. We proposed our techniques to improve network performance of a host-based router, and our methods to determine the emulation capability of both single-node emulator and distributed emulator cluster.
- We show that the proposed emulator cluster can assist us explore a number of research issues such as the impact of non-DS routers in a multi-domain DiffServ (Differentiated Services) network, and performance evaluation of high availability, wide area storage system. Our results from those emulation experiments are obtained by establishing specific network emulation scenarios in the emulator cluster, rather than conducting simplified logical simulations.

Aside from the contribution described above, our research also offers some insight to the simulation and emulation of mobile wireless networks with respect to layer abstraction, mobility emulation, and overhead. We also provide observations of

simulation and emulation of various layers of mobile wireless networks. The research is expected to motivate the research of network emulation.

1.5 Structure of the Content

The rest of the dissertation is organized as follows. A literature review of existing network emulators is outlined in Chapter 2. In Chapter 3, we will present the architecture of EMPOWER and some important research issues, as well as our techniques and solutions to those problems. Some experimental results of the emulator cluster will be presented in Chapter 4. In Chapter 5, we introduce two applications of the emulator cluster. Finally we present our future work plan in the last chapter.

CHAPTER 2 RELATED WORK

2.1 Wireline Network Emulators

Network emulation system, or network emulator, refers to a single host, a device, or a laboratory network environment that is capable of mimicking a real network by applying specific network conditions and traffic dynamics to traversing traffic. As a fully controllable laboratory network, network emulator is able to provide a rich set of modules that can be used to generate network conditions and traffic dynamics, which can be applied to the underlying emulation scenario for any specific purpose. In addition, network emulation may mimic a real network more closely than network simulation since network traffic within network emulation environment and the operation of network protocols and applications are "real" rather than abstracted logical entities or processes of an operating system. The network protocol code to be tested can be directly used without any conversion and modification. In addition, network emulation usually provides the facility for traffic monitoring and tracing, as well as visualization tools to help understand the complex behaviors of the network protocols [8, 9]. Network emulator can be used to assist network research in various ways as follows.

- As a test environment for the performance evaluation of protocols that are executed within the environment.
- As a test environment for the performance evaluation of applications that are executed outside the environment. Traffic of these applications is directed to the environment subjecting to possible network effects.

- As a supporting tool for the development of a new protocol. The implementation of a new protocol can be easily tested and debugged in a reconfigurable network emulator.
- As a supporting tool to evaluate network devices such as routers and switches. These hardware devices can be plugged into the emulation environment, and can be tested against any emulated network effects.

There are several methods to classify network emulation systems. Based on the type of the emulated network, network emulation systems can be divided into two categories, wireline network emulation system such as NIST NET [9], ONE [10], Delayline [11], Dummynet [12], WAN emulator for TCP Vegas evaluation [13], ENTRAPID [14], IP-TNE [15], Shunra Cloud [16], and wireless network emulation systems such as JEMU [6], Seawind [7], trace-based mobile network emulation system from Carnegie Mellon University [17, 18], emulation facility provided by ns simulator [1, 19] and an extension from Kubinszky [20]. Most network emulation systems are computers or a local area network consisting of several computers, while some commercial emulation systems are hardware devices such as PacketStorm [21] and PacketSphere [22]. The major difference among existing network emulation systems is the abstraction level of the emulation scheme. Some network emulation systems such as NIST NET and ONE abstract the emulated wide area network to a single network "cloud" with a set of parameters or distributions such as packet delay, delay jitter, link bandwidth, packet loss, etc, while others try to emulate each router and link in an emulated network in accordance with some analytical network models. In the latter case, the emulated network topology is

mapped to the emulation system in a one-to-one fashion. We select some examples of emulation systems to describe different approaches to the underlying research problem.

2.1.1 NIST NET

NIST NET [9] aims to emulate an IP network on a single Linux router. Two operation modes are supported by NISET NET: trace-driven mode and interactive mode. In trace-driven mode, real traffic traces in the target network are used to control the emulator. In interactive mode, one must determine a set of network performance parameters of the target network prior to an emulation experiment, then the emulator, the Linux router, is configured with these parameters to generate corresponding network conditions and traffic dynamics. With the emulator as the gateway of test hosts, end-to-end network protocols and applications executing on test hosts can be tested and evaluated against those predefined network conditions and traffic dynamics in the emulator, which is fully controlled by a kernel module. Similar emulation systems include ONE [10], Shurna Cloud [16], ENDE [8], each of which approximates an emulated network to a single router. The advantage of this approach is that it masks the details for the emulated network and the emulator is easy to configure as only one single machine is involved in emulation configuration. The limitation of these emulation systems is that only end-to-end network traffic effect can be emulated. For example, they are unable to evaluate any routing protocols. Further, it is usually difficult to determine performance parameters of a target network that will be used for emulation configuration, as measurement experiment or analytical models are required for this purpose.

2.1.2 WAN Emulator for TCP Vegas Evaluation

Unlike the single router approach described above, this WAN emulator consists of a dozen workstations in a local area network. Each workstation can be configured to emulate a network node, and each network interface on a workstation can be configured to emulate a link. As a result, a network topology can be mapped to the emulator by a simple one-to-one mapping scheme. Internally, a pseudo-device is created on each workstation to apply specific propagation delay, transmission delay, queueing delay to each traversing packet. In contrast to NIST NET where the entire target network is abstracted to a set of parameters, this approach aims to emulate a target network in link level, i.e., each link and network node in the target network could have a corresponding *emulated* identity in the emulator. Protocols and applications that cannot be tested with NIST NET may be tested with this emulator. However, the problem of this emulator, as well as some similar ones such as Dummynet [12], and Emulab [23], is scalability. According to the one-to-one mapping scheme, one must have the same number of workstations available to emulate a target network, which is costly and unaffordable for most network researchers. Moreover, as a number of workstations are used for emulation, these emulators are not flexible in that they cannot be easily configured to adapt to different emulation scenarios.

2.1.3 Emulab

The Utah Emulab, also called the Utah Network Testbed, is a unique type of experimental environment: a universally available "Internet Emulator," which provides a balance between control and realism [23]. Several hundred machines, combined with

secure, user-friendly web-based tools, and driven by ns-compatible scripts, allow users to remotely reserve, configure and control machines and links down to the hardware level. Even the operating system disk contents may be securely and fully replaced with custom images. A major limitation of Emulab is that each machine can only emulate one network node. There must be another machine in the middle to to shape the traffic between two network nodes. Moreover, this is a very expensive emulation environment and it is hard to scale. Surprisingly, the only information about Emulab is on the web [23]. We could not find any published papers on this project.

2.1.4 The Emulation Facility in VINT/ns

As an extensible and powerful discrete event-driven network simulator, VINT/ns has been widely used in network research community. The emulation facility [19] in VINT/ns is developed to combine ns simulation and network emulation such that the rich resources available in ns such as protocol modules, algorithms, and visualization tools, can be used for the purpose of network emulation. However, this approach inherits the fundamental limitation of network simulation: real protocol implementations and applications cannot be evaluated without conversions, and simulation assumptions might strongly affect the simulation results.

2.2 Wireless Network Emulators

The network emulation systems described above are mostly considered to be general-purpose network emulators. Their capabilities are largely limited by the nature of wireline network emulation. As wireless network research becomes increasingly active recently, the need for wireless network emulators is evident. Wireless network emulators

have to cope with the difference between wireline networks and wireless networks from the emulation's point of view. More specifically, wireless emulators must offer means to emulate the characteristics of wireless channels and highly dynamic mobile topology.

Existing mobile wireless network emulators can be divided into three categories: central-control wireless emulators, simulation-combined wireless emulators, and trace-based mobile network emulation. We discuss each of them along with some examples.

2.2.1 Central-Control Wireless Emulators

Emulators in this category abstract the entire wireless mobile network to a model with a set of parameters. With these parameters, the emulator is able to reproduce some network effects by applying network conditions and traffic dynamics to each packet passing by. A mobile wireless network emulation experiment is conducted by connecting mobile hosts to the central control emulator. This category may also include some general purpose network emulators in addition to one specifically designed for wireless mobile network emulation. Some general purpose network emulators falling into this category are ONE [10], Dymmynet [12], and NIST NET [9]. Typical network parameters supported in these emulators include packet delay distribution, packet drop pattern, bandwidth, and queueing discipline. Although these emulators perform well for wireline network emulations, they are likely to fail meeting one of the specific requirements of mobile wireless network emulations, i.e., the capability to emulate dynamic network conditions. Wireless-specific network emulators, such as Seawind [7], allow one to define multiple set of parameters that are constantly varying during an emulation experiment. Additionally, specific channel models are provided to closely mimic a unidirectional wireless link. Another example of central-control mobile wireless

network emulator is JEmu [6, 24]. In JEmu, mobile hosts connected to the emulator host are not allowed to move. The emulator program, a standalone application, accepts radio layer messages from those mobile hosts and determines whether or not to forward the messages to the destination based on the location information of the mobile hosts. If there is a collision on the mobile host once the message is forwarded, the message is then dropped and returned to the radio layer. Central-control emulators are easy to configure, and their parameter sets are usually sufficient for the performance evaluation of end-to-end network protocols. However, this approach fails to provide a test environment for some other topology-related protocols such as multi-hop ad hoc routing protocols, which is now highly active in mobile network research community.

2.2.2 Simulation Combined Wireless Emulators

Mobile wireless network emulators in this category aim to combine network emulation with existing full-fledged network simulators such that the rich resource available in network simulators can be used in an emulation experiment. The most known example in the category is VINT/ns [19, 20]. As an extensible and powerful event-driven network simulator, VINT/ns has been widely used in network research community. A set of wireless and mobile extension has been developed by Monarch project at CMU [20, 25]. The emulation facility in VINT/ns is able to capture and direct traffic into the simulator. Within the simulator, protocol modules, algorithms, and visualization tools can be incorporated into the emulation in an automatic fashion. In addition, arbitrary mobility can be generated with the help of the simulator. The advantage of this approach is that it offers a large amount of simulation resources in the central simulator, comparing to the central-control approach that only has a limited number of network parameters available

for configuration. However, similar to central-control approach, the simulationcombined approach also lacks of the support for the evaluation of real topology-related protocols. Furthermore, since the emulation work is mostly conducted in the simulator, this approach inherits the intrinsic drawbacks of network simulation, as discussed in the previous section.

2.2.3 Trace-based Mobile Network Emulation

This approach, more specifically trace modulation, consists of three distinct phases: data collection phase, trace distillation phase, and modulation phase [18]. In the first phase, one needs to collect traces from a target mobile wireless network using some traffic monitoring tools. In the second phase, a empirical wireless network model with the collected traces will be formed. In the last phase, the target wireless network effects could be reproduced in a wireline network. This approach is particularly significant because the wireless network effects that it reproduces originate from real network traces, while other approaches can only approximate the network effects according to some analytical models. However, similar to the central-control approach and simulation-combined approach, the trace-based approach can only provide a reproducible environment for the performance evaluation of end-to-end protocols and applications. Additionally, since each mobile host in the target network should be traced for an emulation experiment, this approach cannot emulate a large-scale wireless network because of the high cost in collecting traces. Even for the emulation of a small wireless LAN with several laptop computers and handheld devices, it is still impractical to repeat the same trace collection many times while still maintaining sufficient accuracy. In addition, traced-based emulation of a target wireless network is limited to traced network

conditions and traffic dynamics, which may not capture specific situations that required by some emulation scenario.

In addition to the three categories, some general purpose network emulators can also be used to emulate a simple wireless network topology, such as [13, 26]. The problem of these emulators is that the number of nodes that they can emulate is limited by the one-to-one mapping scheme, in which each mobile host in the target network is mapped to an emulator node in the emulation experiment.

2.3 Problems of Network Emulation

Although network emulation has many advantages, it has been criticized with the following major drawbacks [27]:

- Emulation systems are expensive to build. Typically a network emulation system either involves kernel modification, or need a large number of machines for wide area network emulation.
- Emulation systems are difficult to reconfigure and share. Most emulation systems lack of a simple configuration interface. Configuration of an emulation experiment requires a number of error-prone steps in existing emulation systems.
- Emulation systems have limited flexibility. By flexibility we mean that emulation systems should be able to incorporate new modules without modifying its architecture.

More specifically, the emulators described above share the same disadvantages: these approaches do not provide means to emulate a network topology. Instead, only one

single router can be emulated using a single-node approach, and the emulated network has to be aggregately simplified as a *cloud*, which provides controllable network conditions and traffic dynamics such as bandwidth, packet delay, and packet loss rate. Thus this kind of approach may only be used to test end-to-end protocols and applications, making it impossible to facilitate some network research that is concerned with network topology.

In addition, existing emulation systems are usually non-scalable in two aspects: First, the maximum number of routers that an emulation system is able to emulate is less than or equal to the number of emulator hosts in the system. In particular, some network emulators simply abstract a wide area network to a single gateway with empirical network characteristics [8, 10, 11]. Others take a simple one-to-one mapping approach to establish an emulated network that consists of exactly the same number of machines as the target network. Therefore, the cost of large-scale network emulation will be prohibitively high for most researchers. Second, the maximum emulated bandwidth that an emulation system can achieve is restricted to the bandwidth limit of the physical emulation environment. If the physical emulation environment is a fast Ethernet, it can only emulate a bandwidth up to 100Mbps.

As for existing wireless network emulators, while all the existing approaches described above address the problem of wireless network emulation with different methodologies, they share the following drawbacks:

- Unable to emulate a wireless network topology, or unable to emulate a complex network topology without a high cost, which is highly needed for current wireless network research.

- Unable to emulate the mobility of each individual mobile node. Test experiment involving wireless mobility is extremely hard to conduct due to cost of repetitive mobility generation and the high varying nature of wireless network.
- Unable to provide an interactive interface for users to easily reconfigure the emulator. Some emulators need to change system calls. Some are not extensible because they are specially designed for certain protocols or services.

Based on the literature review presented above, we plan to address the underlying research issue using another approach: a new network emulation framework called EMPOWER. We choose to emulate a target network in link level rather than network level such that network topology can be emulated in EMPOWER. Unlike any other network emulators to the author's knowledge, EMPOWER can emulate multiple routers and links with one single emulator node according to certain network topology. Thus a network topology of a target network can be mapped into an emulation configuration with an emulated network topology consisting of a small number of emulated nodes, making it possible to evaluate both end-to-end protocols and other topology-related protocols. EMPOWER supports the emulation of a variety of network parameters in an automatic manner, which allows one to generate multiple sets of network conditions and traffic dynamics that can be applied to traversing network traffic in the emulation environment. We will present the details of EMPOWER in the next chapter.

CHAPTER 3 EMPOWER ARCHITECTURE

3.1 Design Objectives

Network emulator refers to a workstation or a group of connected workstations that can mimic a target network by imposing predefined network effects to traversing traffic. Network emulators are highly needed in research community when the target network is not accessible, or the target network cannot be used as a test environment. Even if the target network is accessible, one still needs an emulator as a fully controlled reproducible test environment to facilitate protocol development and performance evaluation. Based on the need and the applications of network emulation, the design objectives of a network emulator can be summarized as follows.

First, a network emulator must be an open and extensible network environment such that any protocol modules can be loaded into the emulation system without major modifications to the module and the emulator architecture. In other words, a network emulator is a platform that should adapt to various protocol modules, rather than specifically designed for certain protocols.

Second, a network emulator must be scalable in terms of emulation capacity and accuracy. One cannot afford a costly emulation environment with a large number of workstations. Thus an emulator should provide means to reduce the number of workstations required for a specific emulation configuration. Moreover, the emulation accuracy should not be affected when emulating large networks. Otherwise the overhead of the emulator cannot be ignored and its impact must be carefully considered.

Third, an emulator must be able to be flexibly configured and operated for various network topologies and emulation parameters. The physical layout of the emulator environment should stay intact while emulating different target networks. Instead, software modules in the emulator should facilitate the configuration of an emulation experiment.

3.2 EMPOWER Architecture

3.2.1 Overview

As a distributed network emulator cluster, EMPOWER (EMulation the Performance Of WidE aRea networks) essentially consists of a number of workstations in a local area network. The workstations can be divided into two categories: emulator node and test node. Each emulator node has been equipped with multiple 4-port 100Base-TX Network Interface Cards (NICs). Each test node typically has one NIC with one Ethernet port. All the ports connect to fast Ethernet switches. Figure 3.1 shows an example of the physical layout of EMPOWER. Whatever the target network topology is, the physical layout of EMPOWER will not change. The workstations in EMPOWER are low-cost commodity computers running Linux. Therefore the number of emulator nodes can be increased without high cost.

In order to generate specific network effects that can be applied to traversing network traffic, EMPOWER provides multiple *Virtual Devices* (VDs) as software modules loaded into each emulator node. To emulate network topology, EMPOWER employs an efficient topology mapping scheme called *Virtual-Router mapping*. TCP and UDP Traffic generators are executed on test nodes. Note that in network emulators such as

EMPOWER, the kernel of the underlying emulator node will physically handle each traversing packet. Tested modules are incorporated into corresponding layers of the underlying emulator system such that they are being evaluated against user defined network effects. With appropriate configurations, any available protocol modules can be loaded into the kernel's network stack without any modification. Furthermore, specific packet level operations could be hooked into a VD.

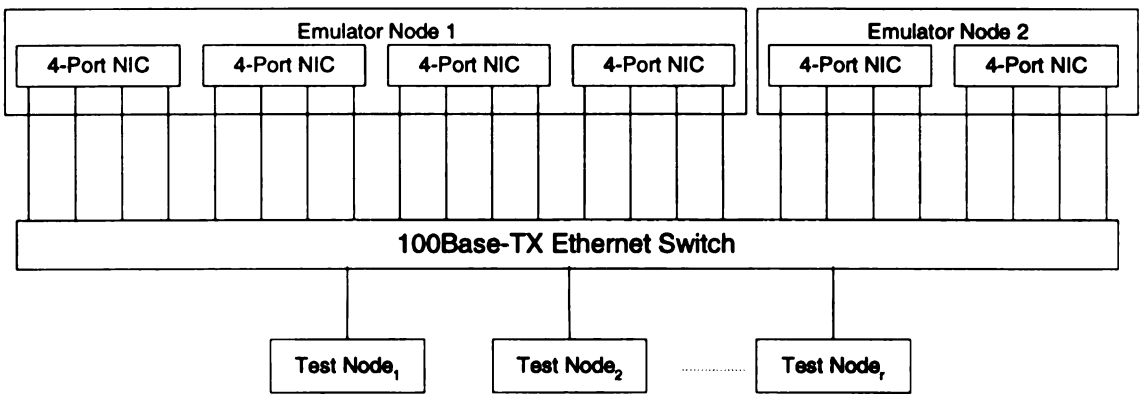

Figure 3.1 An example of the physical layout of EMPOWER

3.2.2 The Virtual Device Module

The basic component of EMPOWER is a VD that intercepts traversing packets and applies predefined network conditions and traffic dynamics. A VD must be attached to a physical network port in an emulator node. They share the same IP address. In the kernel of an emulator node, the routing table has been modified to divert an egress packet flow from the kernel's IP layer to the VD, where specific packet level operations are defined. After that the packet flow will be redirected to the underlying network port and transmitted to the media.

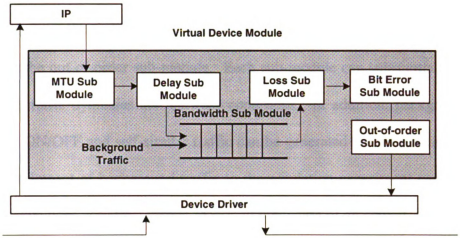

Figure 3.2 Virtual Device (VD) in EMPOWER

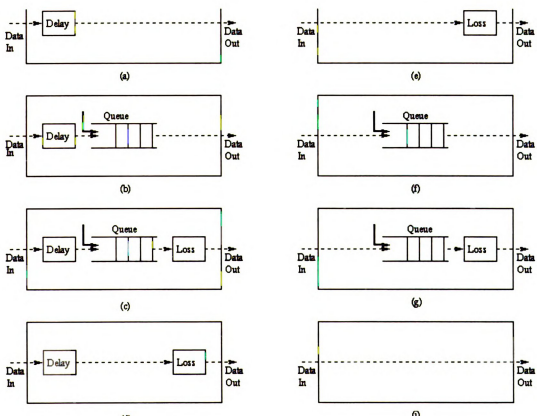

Figure 3.3 Single node emulation mode

As shown in Figure 3.2, six sub-modules exist in a VD module: the MTU sub-module, the delay sub-module, the bandwidth sub-module, the loss sub-module, the bit error sub-module, and the out-of-order sub-module. Each sub-module can substantially incur one aspect of IP traffic dynamics and network conditions. In addition, background traffic [28] such as ON/OFF and self-similar traffic can be generated within the emulator node such that the impact of background traffic to the emulation experiment can be studied. The VD module does not affect packet-receiving procedure of an emulator node. Packets entering the underlying network port traverse the network layers of the operating system in its original way. A VD module also supports trace-driven emulation, in which a trace file of real network traffic is used as the input to the emulator instead of traffic generated by some applications. In the EMPOWER prototype, the VD module is implemented as a kernel loadable module in Linux.

With various combinations of emulated network effects, one could have a number of emulation mode applied to traversing traffic. A list of single node emulation modes in conjunction with the combination of link bandwidth emulation, packet delay emulation, and packet loss emulation, is shown in Figure 3.3 [29]. The single node emulation mode are (a) Delay Mode, (b) Delay-Bandwidth Mode, (c) Delay-Bandwidth-Loss Mode, (d) Delay-Loss Mode, (e) Loss Mode, (f) Bandwidth Mode, (g) Bandwidth-Loss Mode, and (h) Mode7 [28]. A GUI interface is also provided to easily configure the network conditions and traffic dynamics and to apply background traffic for each VD module.

3.2.3 Traffic Statistics

For a network emulator to be a powerful tool for network emulation, it must provide useful statistics about the experiment to the user. Though it is difficult to come up with a

general set of useful statistical indicators we have chosen a set of statistics we feel will be useful to a large community of users. The statistics reported by the emulator depend upon the mode being used. Table 3.1 gives a list of all statistics generated by the emulator along with the modes during which they are reported. The emulator reports both aggregate statistics and statistics computed over the last *n* seconds, where *n* is user configurable.

Table 3.1 Traffic statistics available from the emulator

Statistics Reported	**Description**	**Relevant Modes**
Total packets	Total number of packets received by the emulator.	0-7
Packets Dropped	Total number of packets dropped by the emulator	0-6
Loss Percent	Percent of packets dropped by the emulator	0-6
Mean Delay	Average delay incurred by packets passing through the emulator.	0-3,5,6
Max Delay	The maximum delay a packet incurred in passing through the emulator.	0-3,5,6
Min Delay	The minimum delay a packets incurred in passing through the emulator.	0-3,5,6
Avg Data Size	The average size of all the user data packets passing through the emulator.	0-7
Avg Queue Len	The average length of the queue in the Queue Module as seen by the data packets.	1,2,5,6
Load Percent	The ratio of the average queue length to the maximum queue length reported as percent.	1,2,5,6

3.2.4 Emulation Layout

In EMPOWER, a router in the target network will be mapped to a virtual router in an emulator node. The router and its mapped virtual router have the same number of interfaces and share the same network configuration such as IP address and routing table. Thus a real network topology with certain number of routers is mapped to a virtual topology with the same number of virtual routers in one or more emulator nodes. Each virtual router is independent of each other in terms of packet forwarding mechanism and routing table maintenance. Even if a packet is sent out from one virtual router to another in the same emulator node, it has to traverse a physical network port and an Ethernet switch, and then be received by another network port. This feature ensures that the packet handling process of EMPOWER closely mimic the same operation in the target network. Otherwise the packet will be directly forwarded between two network ports without going through user-defined packet handling operations and network effects. Figure 3.4 shows an example of emulation configuration with two emulator nodes generating four virtual routers; each virtual router has four virtual device modules attached to network ports.

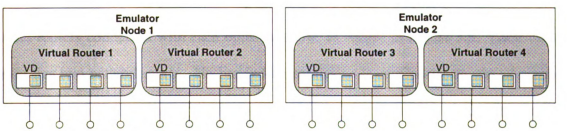

Figure 3.4 An example of EMPOWER emulation configuration

In out prototype system, there are two types of machines as the emulator nodes. One (*em1*) has an Intel Pentium 4 1.7GHz CPU with 128 MB PC800 RDRAM memory, 32bit 33MHz PCI bus and 400MHz system bus. The other (*em2*) has an AMD K6 300MHz processor with 128MB PC100 SDRAM memory, 33MHz PCI bus and 100MHz system bus. DLink DFE-570TX 4-port Network cards are used as the NICs. Each has four Digital "tulip" 21143 chips and one 21152 PCI-PCI bridge on board.

3.3 The Impact of Resource Competition

3.3.1 Resource Competition and Network Throughput

The major effect of system resource competition in an emulator node is the network throughput degradation in heavy traffic load. Specifically, when there is only one virtual router with two network ports in an emulator node, the packet forwarding throughput of the virtual router is quite close to a standalone host-based router. However, when more network ports are added to the virtual router or more virtual routers are generated in the emulator node, the throughput of packet forwarding decreases sharply. To investigate the impact of resource competition among multiple virtual routers within an emulator node in terms of network throughput, we set up a "virtual router chain" with certain number of virtual routers connected with each other in a row, as shown in Figure 3.5. Each virtual router has two network ports. No bandwidth emulation is configured on virtual routers. The IP address assignment and the routing table configuration for each virtual router ensure that network traffic between test nodes is correctly routed. UDP traffic with various packet arrival rates is fed into the router chain. We use 1470-byte UDP packets to minimize the overhead of IP and Ethernet frame headers to the end-to-end network

throughput. Note that when there are 4 virtual routers in the chain, the effective traffic load to the emulator node is 400 Mbits/s since at any time there are 4 network ports receiving packets either from the test node or from other network ports on the same emulator node. Similarly, the actual network throughput can be roughly calculated as 4 times of the end-to-end throughput between two test nodes. For simplicity we only show end-to-end throughput in the following figures. In this experiment the emulator node (named *eml*) is a workstation with a Pentium 4 1.7GHz processor. The result is shown in Figure 3.6. When there are 4 virtual routers (8 network ports) forwarding packets in *eml*, the maximum end-to-end throughput can hardly reach 60 Mbits/s.

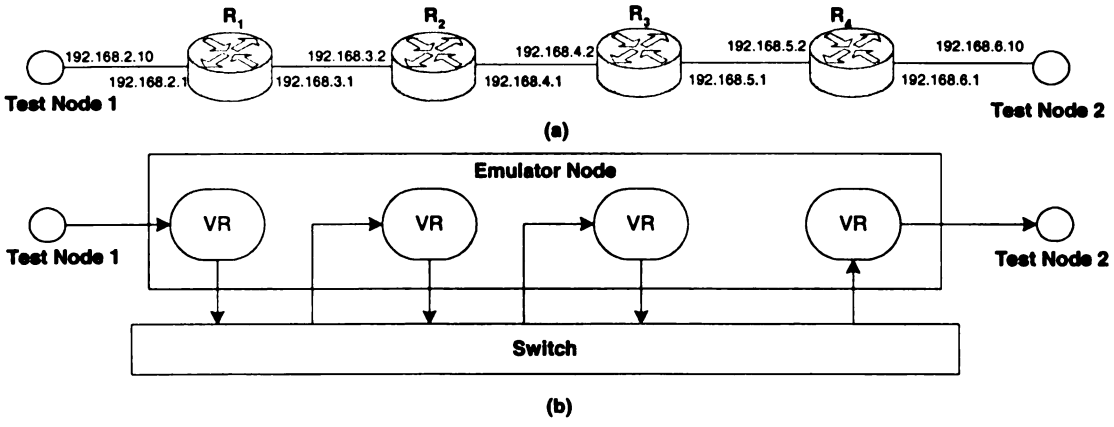

Figure 3.5 A virtual router chain

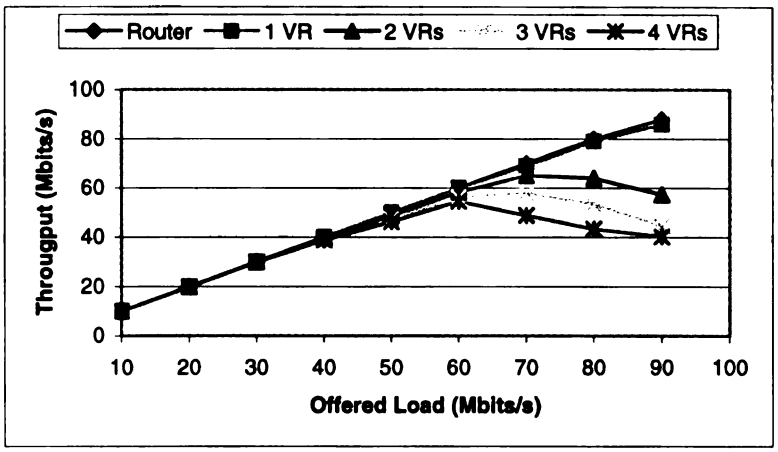

Figure 3.6 Network throughput of the virtual router chain on *em1*

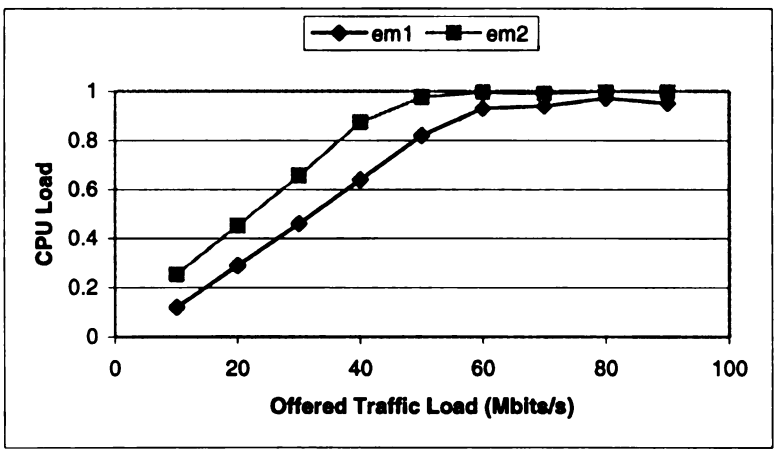

Figure 3.7 CPU load with various traffic loads

Given a host-based router with proper Ethernet connections, there are a number of factors that may affect the maximum network throughput of the router. Hardware factors may include CPU speed, PCI bus bandwidth, system bus bandwidth, buffer size of the network interface card, whereas software factors are the interrupt-driven architecture of the operating system and device drivers.

In Figure 3.7, we can clearly see the CPU's impact on the network throughput. *em1* is the emulator node with a Pentium 4 1.7GHz CPU, whereas *em2* has an AMD K2 300Hz CPU. Both emulator nodes use 32-bit 33MHz PCI bus, which provides a maximum

throughput of 1.056 Gbits/s. This is sufficiently large for 10 100Base-TX Ethernet ports in the worst case. However, if an emulator node uses a Gigabits Ethernet interface for a specific emulation, the PCI bus may become the bottleneck. In this case, a 66MHz PCI bus is required to provide a doubled throughput. Similar to PCI bus, the impact of system bus should be examined as well. *em1*'s bus clock rate is 100MHz. With QDR (Quad Data Rate) the system bus is boosted to 400MHz, giving a maximum throughput of 12.8 Gbits/s. The memory bus on *em1* provides even higher throughput. Thus they are not the bottleneck of the network throughput. For *em2*, the throughput of the system bus is 3.2 Gbits/s, which is theoretically large enough for more than 30 100Base-TX Ethernet ports.

The network interface cards we used on both emulator nodes are full duplex 100Base-TX 4-port network cards with an 8K buffer on each port. Experiment on individual port shows that it can achieve a maximum throughput of 92 Mbits/s for one-direction UDP traffic. Taking account of the overhead of IP header and Ethernet frame header, and the overhead of operating system, it is quite reasonable.

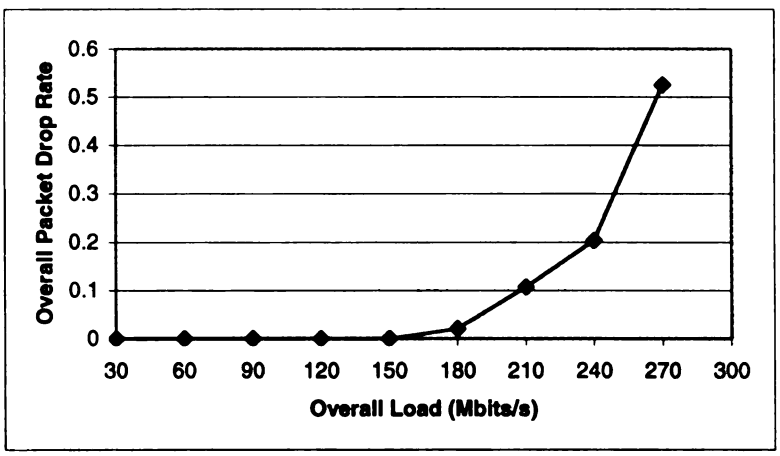

Figure 3.8 Packet drop rate with various traffic loads on *em2*

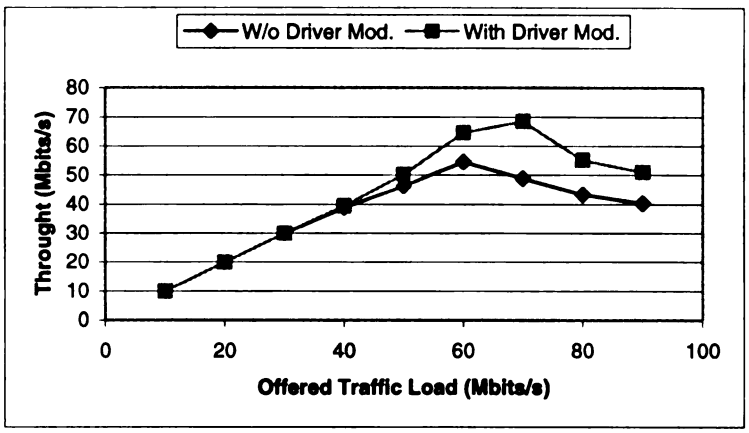

Figure 3.9 Network throughput of the virtual router chain with driver modification

For the software factors, specifically the interrupt-driven scheme of the operating system, we could find a way to improve the maximum network throughput of a host-based router. Generally, when a packet is received on a network interface card, an interrupt will be raised by the hardware device. The interrupt handler in the device driver will copy the packet from device buffer to kernel memory via DMA or PIO. The packet is enqueued and a soft interrupt is then raised to notify the CPU for further processing on different network layers. If the packet's destination is a process on the local system, the packet will be copied to user memory. Otherwise, after the next hop of the packet is determined, it will be sent to the transmission queue of another network device. Then the device driver will be called and the packet will be transmitted to the network. When there are a large number of packets coming from multiple network devices, the upper level processing of queued packets may not be scheduled. As a result, the queue will be full and no further packets from network devices can be delivered to the queue. Thus all incoming packets will be dropped. Figure 3.8 shows the packet-drop experiment result when three UDP flows are fed into *em2*.

We choose to modify the device driver of our network interface card such that the basic I/O architecture of the operating system stays intact. Our interrupt mitigation scheme is to find the appropriate number of events that an interrupt may process in the network device driver while maintaining a low average delay of the packets no more than 10% of the original value. Our experiment on both *em1* and *em2* show that 40 is such a number that can meet our requirement. The network throughput of the virtual router chain on *em1* with 4 virtual routers is shown Figure 3.9. Note that the improvement can only be observed when the offered traffic load is quite high. The maximum network throughput has been increase by 23% on *em1*.

Our experiment shows that bandwidth emulation accuracy of a single virtual router in an emulator node is less than 4% [30], given that the emulated bandwidth is less than the maximum throughput of the emulator node. To investigate the impact of resource competition and overhead of packet handling of EMPOWER on the bandwidth emulation accuracy, we setup another virtual router chain on *em1*. In this case each link of a virtual router has been configured to emulate a certain bandwidth, which is less than the maximum end-to-end throughput of the emulator node we obtained from previous experiment. The number of virtual routers in the chain varies from 1 to 6. The emulated bandwidth ranges from 10 Mbps to 70 Mbps, depending on the number of virtual routers in the chain. Our experiment results show that when multiple virtual routers exist in an emulate node and more than one links have been configured to emulate specific bandwidths, the emulation accuracy drops (up to 8%) slightly comparing to the single router case. In particular, when the emulated bandwidth is less than 50 Mbps, the difference between the emulated bandwidth and the effective bandwidth can be almost

ignored (less than 2%). We observe that the competition for the CPU among various virtual device modules substantially affects the bandwidth emulation accuracy. But the impact is not as obvious as that on the packet forwarding of virtual routers.

3.3.2 Resource Competition and Packet Delay Emulation

As a network emulator cluster, EMPOWER should be able to incur user defined arbitrary packet delay for traversing packets. Today's Internet has large range of packet delays from less than 1 millisecond inside an enterprise network to more than 100 milliseconds from coast to coast in North America. As a result, the emulated packet delay in EMPOWER must be scalable to cover this range. There are two problems for the emulation of packet delay. One is the emulation of small delay value, in which the emulator node must provide sufficient time accuracy to generate this delay for a packet. The other is for both small and large delay emulation. The emulation of these delays should not strongly affect the emulation of other network parameters on any virtual routers within an emulator node.

The time accuracy of Linux is 1 jiffy (10 milliseconds) on i386 platform, which is not acceptable for EMPOWER. We can either increase the system interrupt frequency to obtain a finer time resolution or constantly count the real time clock in the timer implementation. The latter approach is rejected because by now there is no way to access the real time clock from a kernel module. Since a time accuracy of 1 millisecond is required for specific packet delay emulation, the system interrupt frequency must be at least 1000. Our experiment shows that the modification to the interrupt frequency causes a maximum overhead of 3%, which is acceptable for most applications.

Resource competition in an emulator node may also affect packet delay emulation in EMPOWER. Previous result show that with a single virtual router in an emulator node, EMPOWER is quite accurate in emulating packet delay [30]. When there are multiple virtual routers in an emulator node and each virtual router may impose specific packet delays to some packets, the resource competition, specifically the CPU competition, will affect the packet delay emulation.

3.4 Virtual Router Mapping

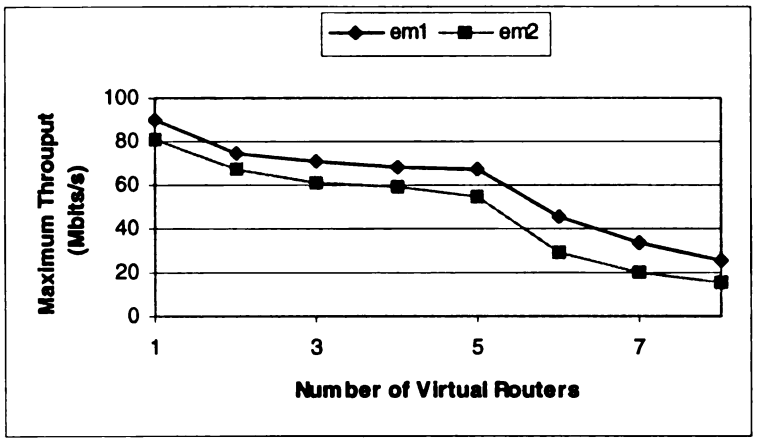

Figure 3.10 Maximum end-to-end throughput with various number of virtual routers

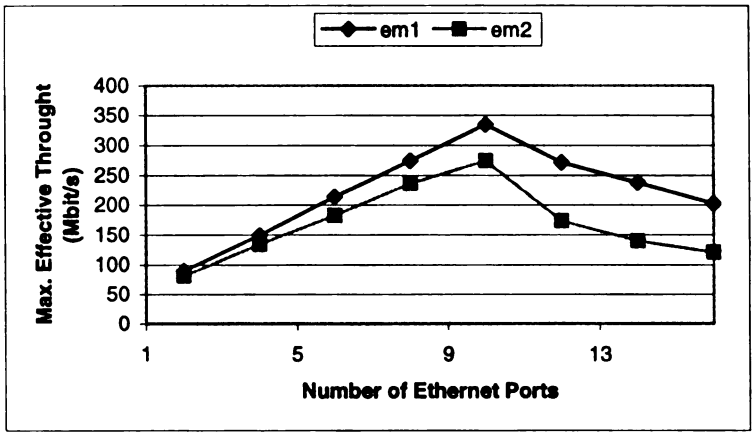

Figure 3.11 Maximum effective throughput with various number of Ethernet ports

The virtual-router mapping scheme is proposed to efficiently and accurately map a target network to an emulation configuration in EMPOWER. Given a target network with certain number of routers in a specific topology, assume each router can be emulated by a virtual router in terms of link bandwidth and packet delay accuracy, the target network will be partitioned into several blocks such that each block can be emulated by an emulator node and the number of blocks is minimal.

An emulator node with multiple Ethernet ports is a host-based router. The emulation capability of an emulator node is simply determined by its maximal network throughput, which is bounded to a variety of factors such as the CPU speed, system interrupt frequency, and PCI bus speed, as described in the previous section. This maximum network throughput varies with the number of network ports involved in forwarding packets. In order to find the relationship between these two parameters, we setup a "virtual router chain" with both two types of emulator nodes, in which each virtual router has two network ports. The network throughput of the router chain with various number of active network ports are measured. As shown in Figure 3.10, the more network ports are assigned to virtual routers, the smaller the maximum end-to-end throughput will be. Clearly the relationship between the number of assigned network ports and the maximum throughput is not liner. We show in Figure 3.11 the maximum effective throughput with different number of Ethernet ports in an emulator node. The maximum effective throughput is calculated by multiplying the maximum end-to-end throughput and the number of virtual routers in the virtual router chain. We can see that when there are more than five virtual routers with ten network ports in an emulator node, the maximum

network throughput drops fast due to resource competition in the emulator node. In effect, data in Figure 3.11 serves as a physical constraint to network mapping for an emulation configuration as an emulator node can only be configured to emulate a portion of the target network with limited aggregate network throughput. Otherwise, the emulator cannot guarantee the validity of emulation result.

3.5 Network Mapping and System Scalability

3.5.1 Maximum Network Provision

The scalability of the whole system in terms of emulation capacity is largely determined by the virtual-router mapping scheme and the network throughput of an emulator node with certain number of network ports. The virtual-router mapping scheme is proposed to map a target network to an emulation configuration in EMPOWER. Specifically, given a target network with certain number of routers or nodes in a specific topology, we need to develop an efficient algorithm that can partition the target network into a number of blocks such that each block can be accurately emulated by an emulator node and the number of emulator nodes is minimal.

We define Maximum Network Provision (MNP) of an emulator node with certain number of active Ethernet ports as the maximum packet-forwarding throughput over all the active ports. The MNP can be regarded as the measurement of an emulator node's capability of forwarding packets using a specific number of ports. We have conducted an experiment to measuring the MNPs of two types of emulator nodes, *em1* and *em2*. In the experiment, a number of emulated nodes are configured to form a node chain. Each node has been assigned two Ethernet ports. Test nodes are UDP traffic generators. We setup

four scenarios with different number of nodes in the node chain in order to see the impact of number of Ethernet port on the MNP. The result is shown in Figure 3.12. A configuration with a certain number of active ports will show an MNP ranging from 91 Mbits/s to 335 Mbits/s. The maximum MNP reaches its peak when the active port number is 10.

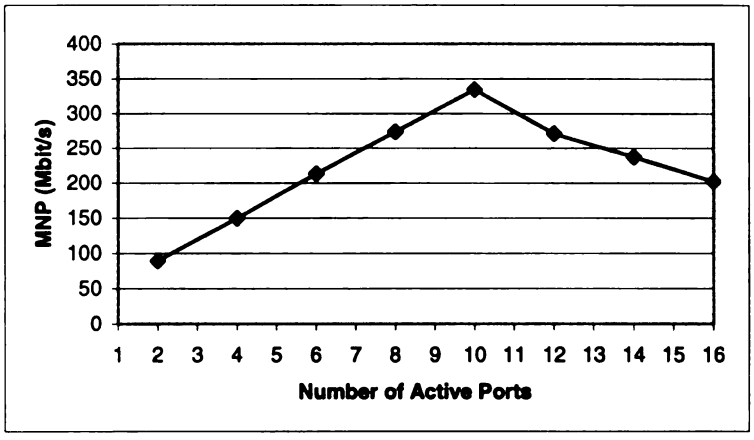

Figure 3.12 MNPs with different number of active ports

When an emulator node has been configured to emulate a certain set of nodes with various bandwidths, it should be able to ideally provide an effective throughput that can handle all the link bandwidths in the node set. For example, as shown in Figure 3.13, an emulator node is configured to emulate three nodes. One node has three interfaces (links); the others have two interfaces (links). To achieve accurate bandwidth emulation, the emulator node, with seven active ports, should be able to provide an emulated network throughput for all the emulated links.

3.5.2 Aggregate Network Capacity

We define Aggregate Network Capacity (ANC) of a network or a portion of a network with certain number of nodes as the sum of the link bandwidths among all the nodes in

the network, plus bandwidths of links between nodes in the network and outside nodes including traffic sources and sinks. Thus ANC theoretically represents the upper bound of the sum of flows a network may achieve. As for the example in Figure 3.13, its ANC is 170Mbps.

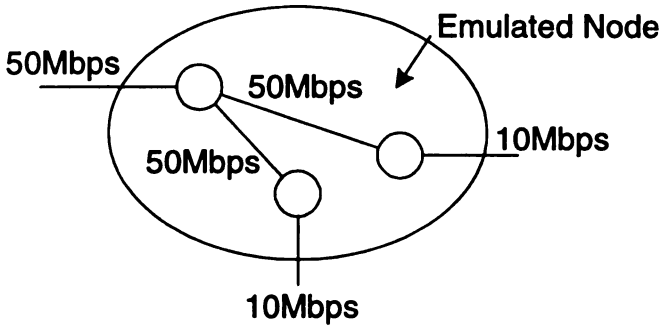

Figure 3.13 An example of single emulator node emulation

If the ANC of a certain number of emulated nodes is too large to be emulated by an emulator node, one or more emulated nodes have to be removed from the node set to reduce the ANC. Consequently, the problem of improving the scalability of the emulator cluster has been reduced to two separated problems: One is to improve the maximum network throughput of a emulator node, which has been discussed in the previous section; the other is to develop an algorithm that can efficiently partition a target network into a number of node sets such that each set can be emulated by a emulator node and the number of emulator nodes is minimal.

3.5.3 The Mapping Algorithm

We propose the following heuristic best-fit Bin-Packing algorithm to address this problem. Traditional Bin-Packing problem, which has been proved to be NP-hard [31], is to place a number of products with various weights into some equal-capacity bins such that the number of bins used is minimal. The constraint of traditional bin-packing problem is the bin capacity. In our case, nodes in the target network are the products; emulator nodes with a limited number of ports and MNP are the bins. Note that the bin size varies due to the fact that MNP will change with number of ports. Importantly, there are two constraints, the port number and the ANC. Both of them can be regarded as weights of products. In particular, not only the ANC of an assigned set cannot exceed its corresponding MNP, but also the number of ports in the set cannot exceed the port number that determines the MNP. The basic idea is the to put as many as nodes that have large single-node ANCs into a set that can be emulated by an emulator node with an MNP larger than the set's ANC. When a set is done, remove all its nodes from the target network, and repeat the steps from the beginning.

Given a target network with n number of nodes $v(i), i \in [1, n]$. Link bandwidths of node $v(i)$ can be denoted by $B(i) = \{b_{i,1}, b_{i,2}, \cdots, b_{i,l(i)}\}$, where $l(i)$ is the number of links on node $v(i)$. Let $MNP(i), i \in [1, p]$ denotes the MNP with i number of active ports. p is the total number of ports available on a physical node. We assume for all nodes, $l(i) < p$. Let $P(j), j = [1, k]$ denotes the resulting partitions of the target network. We further assume that for any single node $v(i)$, $ANC(v(i)) \leq MNP(l(i))$. Let T denotes a temporary node set. Initially T is NULL. The ANC algorithm is described as follows:

Step 1:Find a node $v(i)$ with the maximum single-node ANC among all the remaining nodes. Put $v(i)$ in T. Update $ANC(T)$.

Step 2: Find one of T's adjacent nodes $v(i')$ with the maximum single-node ANC among all the remaining nodes. If there are two such nodes, choose the one with the smaller number of ports. Let d be the total number of ports in $T + v(i')$. If

$$ANC(T) + ANC(v(i') < MNP(d)$$

Then put $v(i')$ in T and repeat Step 2. Otherwise, discard $v(i')$ from the selection (but $v(i')$ is still in current network). Find another T's adjacent node with the second maximum ANC. Repeat the above operation, and so on.

Step 3: If such a node cannot be found in T's adjacent nodes, put T into $P(j)$, increase j, and remove T from current network. Reset T to NULL.

Step 4: If no nodes exist in current network, stop. Otherwise, go to Step 1.

Using the ANC algorithm, the emulator cluster is able to generate a set of partitions that can be assigned to a number of emulator nodes to generate the corresponding virtual topology. As a result, a target network is accurately mapped to a proper configuration of such emulator nodes in the emulator cluster. To illustrate the algorithm, an example of the application is shown Table 3.2 and Figure 3.14. Note that with only three emulator nodes we are able to emulate eight nodes with respect to network throughput in a real network environment. In addition, we have applied the algorithm to a number of sample networks. Figure 3.15 shows the results. The dash line denotes the MNP of *em1*, while the solid line denotes the MNP of *em2*. Symbols such as "*" for *em1* and "x" for *em2* represent the node set result, i.e., a pair of ANC and port number for either *em1* or *em2*.

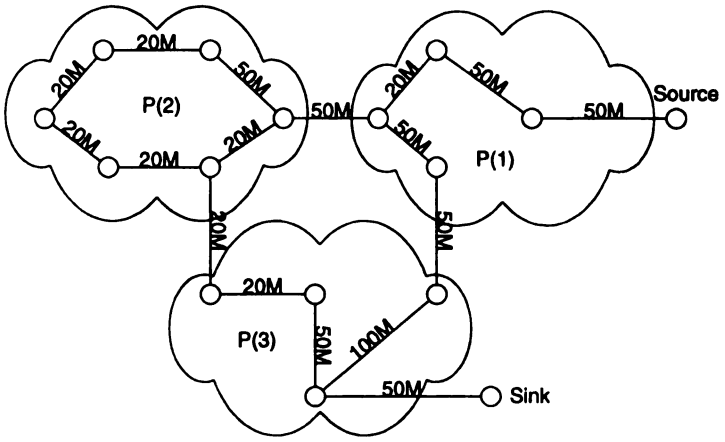

Figure 3.14 An example of network mapping using the algorithm

Table 3.2 An example of the network mapping algorithm

Node Set	Port Number	ANC (Mbits/s)	MNP (Mbits/s)
P(1)	9	270	304
P(2)	12	220	271
P(3)	9	290	304

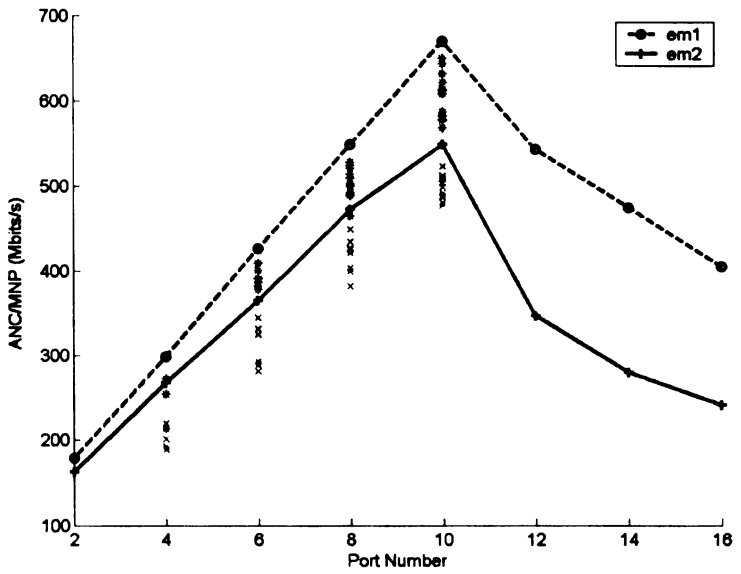

Figure 3.15 Example of node sets using two types of emulator nodes

3.6 Emulation Accuracy

Another design issue of EMPOWER is the improvement of the emulation accuracy of network effect such as packet delay, link bandwidth and packet drop rate. Time accuracy is the key to the emulation accuracy of EMPOWER. We choose to increase the system interrupt frequency to obtain a finer time resolution, while still maintaining a low overhead of less than 3%, as described in the previous section.

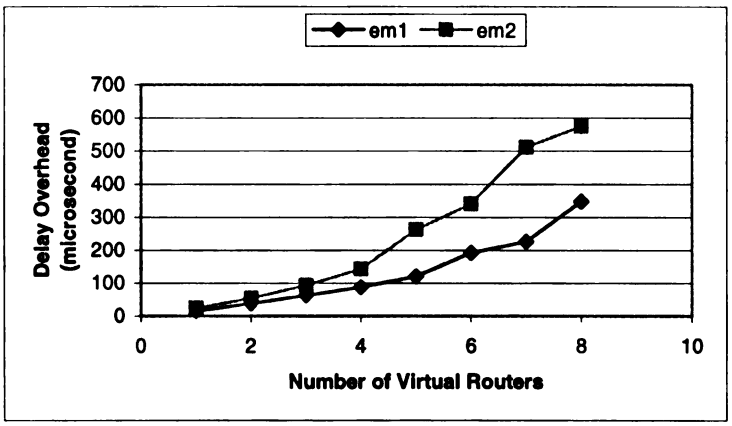

Figure 3.16 Virtual router delay overhead

To find the overhead of multiple virtual routers in an emulator node, we measure the average end-to-end delay of a UDP flow on *em1* that will traverse a virtual router chain with various numbers of virtual routers, including no virtual routers in the traffic path. There is no packet delay emulation on each virtual router. We assume the packet forwarding delay of the fast Ethernet switch is constant regardless of number of port used. Note that even two adjacent virtual routers are generated by the same emulator; packets from one virtual router must be first physically transmitted to the fast Ethernet switch before arriving at the other virtual router. The result in Figure 3.16 shows that when the number of virtual routers increases from 1 to 4, the average packet delay

incurred by each virtual router also increases from 15 microseconds to 22 microseconds. If the time accuracy of the emulator node is close to 100 microseconds, these overheads must be considered and compensated in the virtual device queueing procedure in order to achieve accurate packet delay emulation. If the time accuracy of the emulator is sufficiently larger than the delay overhead of virtual router such as 976 microseconds with a system interrupt frequency of 1024 per second, we may simply ignore this overhead.

3.7 Traffic Models

EMPOWER supports background traffic generation that enables performance evaluation of protocols and applications with respect to in-kernel cross traffic. The background traffic generator needs various statistical models that determine the characteristics of Internet traffic[28]. However, due to the complex nature of Internet traffic, it is rather difficult to create an efficient and effective workload model. Paxson [32] showed that the Poisson model fails for emulating wide area networks. EMPOWER provides several traffic models to generate background traffic. In the packet train model traffic originating from a source is modeled as trains of packets with the inter-train interval comparatively larger than the inter-car interval within a train. The analytical modeling of such sources is done by ON/OFF sources. The aggregate Internet traffic is considered to be a superposition of such ON/OFF sources. Taking this model into consideration a facility is provided in the emulator to set up multiple ON/OFF sources. The outputs of these sources are multiplexed together to generate the background traffic. In addition, we incorporate self-similar model proposed in [33] into our emulator. It is

based on the principle that superposition of many ON/OFF sources with strictly alternating ON and OFF periods and whose ON periods or OFF periods exhibit the Noah effect will produce aggregate traffic that exhibits long range dependence. In EMPOWER, we use Pareto distribution to determine the lengths of the ON and OFF periods. As in the packet train model during ON periods fixed size packets are generated at a constant rate. In addition to the analytical traffic models discussed above, EMPOWER also provides an empirical method to approximate Internet traffic.

3.8 Wireless Network Emulation

To facilitate the emulation of mobile wireless networks, EMPOWER has been designed to support the emulation of wireless networks in a wireline network. Similar to the wireline network emulation, a mobile node in a target network is mapped to a *Virtual Mobile Node* (VMN) in an emulator node. A VMN physically consists of two network ports, one for ingress traffic and the other for egress traffic. Since a virtual device only affects egress traffic, one single virtual device is attached to a network port for egress traffic to impose predefined network effects, as shown in Figure 7. Particularly, each VMN has to maintain a list of mobility events, which describes the movement of the corresponding mobile node in the target network. Each entry in the mobility event list consists of a time value indicating the time the event fires, and the updated list of neighbors of the mobile node. As a result, at any given moment, the emulated mobile node is able to find out who are its neighbors by simply consulting its neighbor list. When a packet is directed to a VMN from IP layer of the emulator node, the corresponding virtual device will lookup the neighbor table to check if the next hop of the

packet is in the neighbor table. If the next hop is in the neighbor table, the packet will be forwarded to the next hop (another VMN). Otherwise, the packet will be silently dropped because in reality the radio signal of current node cannot reach that node. Thus the movement of a node has been substantially mapped to a dynamic neighbor list. By scheduling changes to the neighbor list of nodes across the entire network, EMPOWER is able to emulate the mobility of the network and its consequence in terms of node accessibility and routing dynamics. A similar approach of mobility emulation is described in [6], in which a receiver-side filter program executed on a central control workstation is used to determine if an ingress packet should be accepted by a mobile node connecting to the control workstation. Our approach differs from it in that, instead of using a central standalone program that may become the bottleneck of the emulation system, EMPOWER employs a distributed architecture in which each VMN maintains a packet filter independently.

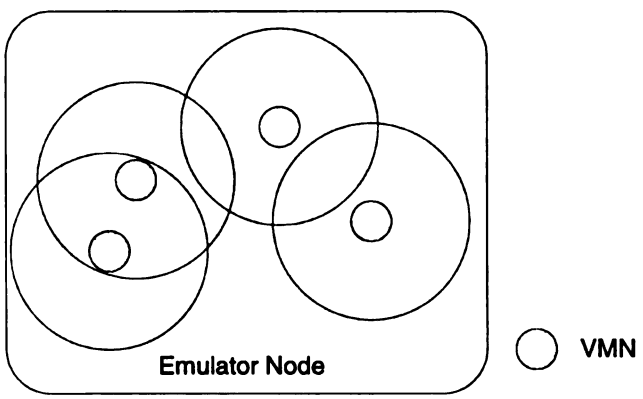

Figure 3.17 Logic view of wireless emulation in EMPOWER

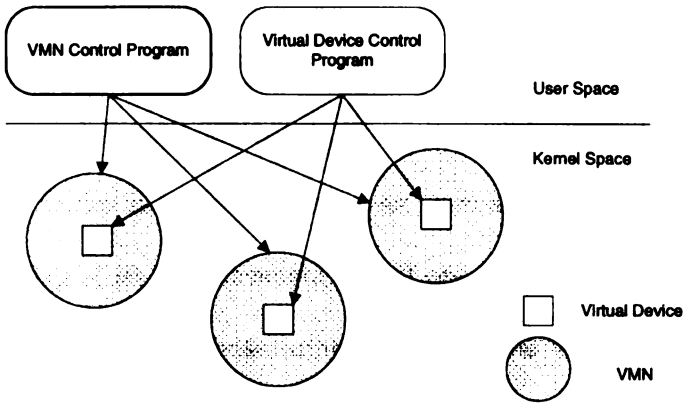

Figure 3.18 Software modules in an emulator node

VMNs and VDs run in the kernel space, while two control programs, VMN control program and virtual device control program, are executed in user space such that one can conveniently modify the configurations of VMNs and VDs. Specifically, the VMN control program is responsible for generating all the VMNs based on the target network information in a configuration file. It also maintains the correct schedule of the mobility events. The VD control program configures the property set of each virtual device and system-wide settings according to the emulation scenario. These may include IP address, network mask, ARP entry, routing tables, routing rules, queueing disciplines. Both the VD control program and VMN are able to be scheduled according to a predefined emulation scenario for network dynamics emulation.

A critical design issue of mobile wireless network emulation in EMPOWER is to make it flexible in emulating different layers of mobile wireless networks. More specifically, EMPOWER should not only provide means to emulate network behaviors beyond network layer (including network layer), but also facilitate the emulation of data link layer. In particular, most existing approaches use simulation to evaluate MAC layer

protocols such as [34] and [35]. We intend to emulate MAC layer protocols by using IP layer operations in EMPOWER. To achieve this, a special emulated MAC layer (eMAC) has to be added to each VMN. eMAC resides between the virtual device and network layer. Whenever a packet is about to be sent out from a VMN, eMAC of the VMN will check the channel flag of its current neighbors. If one of its neighbors has a channel flag raised, that neighbor is active in transmitting now. Thus the VMN has to backoff for next try. Once occupying the channel successfully, it will raise the channel flag to indicate the occupation. The channel flag will be reset when the transmission is over. In addition, similar to packet transmission between VMNs, RTC/CTS can be sent from one VMN to another for a complete channel operation. eMAC is also the place where a backoff algorithm can be incorporated into EMPOWER.

When it comes to the emulation of wireless networks, one critical issue is the choice of layers for abstraction. Because of the inherent difference of physical layer between wireline network and wireless network, if EMPOWER is used to emulate a wireless network in a wireline network, we have to focus on the layer we are interested with and use abstraction for others. For example, when emulating the performance of a MAC layer protocol for wireless LAN, a good choice of emulation is to use a physical layer model to simulate radio layer, and implement the MAC layer protocol in eMAC.

Open research issues in wireless network emulation include global time coordination among multiple physical emulator nodes, physical layer emulation, hybrid emulation, and emulation of wireless networks other than wireless LAN. In order to achieve high time accuracy in wireless emulation, a global time coordination mechanism in the system is needed to maintain proper scheduling of node movements and topology dynamics. NTP

cannot be used for this purpose because it focuses on time accuracy over a quite long period of time at the expense of short-term clock drift. We might periodically send UDP time update packets from one emulator node to others. The overhead of the time update in terms of packet delay emulation and its impact on a skewed clock are yet to be examined. Currently EMPOWER does not support the emulation of radio layer. We plan to incorporate some physical layer models into EMPOWER.

3.9 Other Research Issues

Two major drawbacks of network simulation are the simulation speed and simulation scalability. Network simulators will consume a large amount of time when simulating a large-scale network, making it unacceptable to emulate large networks. An emulator is actually a real-time simulator that does not have the simulation time problem. However, the scalability of an emulator remains a critical research issue. More specifically, we need to compare EMPOWER with network simulators such as ns [1] from a variety of perspectives including single-node simulation scalability, multiple-node simulation scalability, and the upper bound of large-scale network simulation. Some measurements and quantitative analysis with respect to the scalability of simulators and emulators are needed.

CHAPTER 4 VALIDATION AND RESULTS

4.1 Network Effect Validation

In emulating a predefined network conditions and traffic dynamics, EMPOWER imposes a set of network parameters on each packet passing by. These parameters are packet latency, link bandwidth, packet drop rate, bit error rate, MTU change, and out-of-order delivery. The validation of these parameters is important for further experiments. The parameter of packet latency can be fixed value, user defined patterns, or an empirical delay distribution. The parameter of link bandwidth can be a fixed value to indicate the upper bound of the emulated bandwidth. The parameter of packet drop rate can be a fixed value or user defined drop sequence.

For the packet latency experiment, we only generate one single virtual router in an emulator node because our intent is to validate the basic component of EMPOWER. The emulator node is the gateway between two test nodes. The packet latency parameter of the active virtual router is set to a wide range from 1 millisecond to 1 second (In Figure 4.1, We only show the packet latency range from 1 millisecond to 60 milliseconds, where the difference between the expected value and measured value is larger than the others). On one of the two test nodes, we send 100 ICMP ECHO REQUEST packets to the other test node, one for each second. To improve the time accuracy of the system, we increased the system interrupt frequency from 100 to 1024. The new time accuracy of the system becomes 976 microseconds, more than 10 times smaller than the default one.

As shown in Figure 4.1, the measured packet latencies are quite close to the expected values.

To validate the link bandwidth emulation, we generate a four-virtual-router chain in an emulator node. The link bandwidth parameter of one virtual router is set to a range from 100Kbps to 60Mbps to avoid the overhead of resource competition. We send a total volume of 10Mb UDP packets from one of the test nodes to the other, and calculate the throughput with the measured transmission time. As show in Figure 4.2, the measured bandwidth is very accurate when the expected bandwidth values are less than 50Mbps. Beyond that value, there is noticeable difference since the system is too busy to handle all the network interrupts on all active Ethernet ports, as discussed in the previous section. To achieve a higher bandwidth in the virtual router chain, we may partition the virtual router chain into two blocks, and move one block to another emulator node such that the effective network throughput of the block is larger than that of the entire network, as shown in Figure 3.10.

For the packet drop rate experiment, we use a single virtual router to generate specific packet drop rates. We send 500 ICMP ECHO REQUEST packets from one test node to the other, measuring the effective packet drop rate. As shown in Figure 4.3, the difference between the expected packet drop rate and the measured packet drop rate is very small in a rate less than 5% when the drop rate is less than 50%, after that there is up to 10% difference between them due to the error of our random number generator in the virtual device module. We plan to replace our integer-based random number generator with a software module that uses the random number generator provided by the Linux kernel.

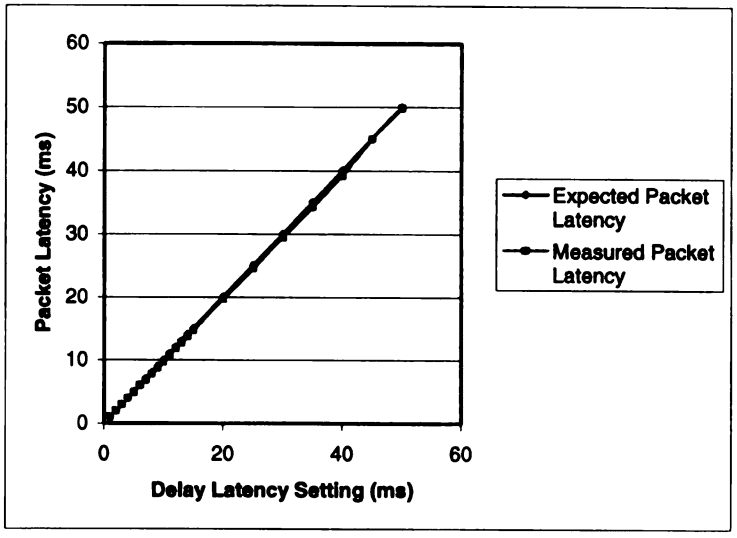

Figure 4.1 Packet delay validation

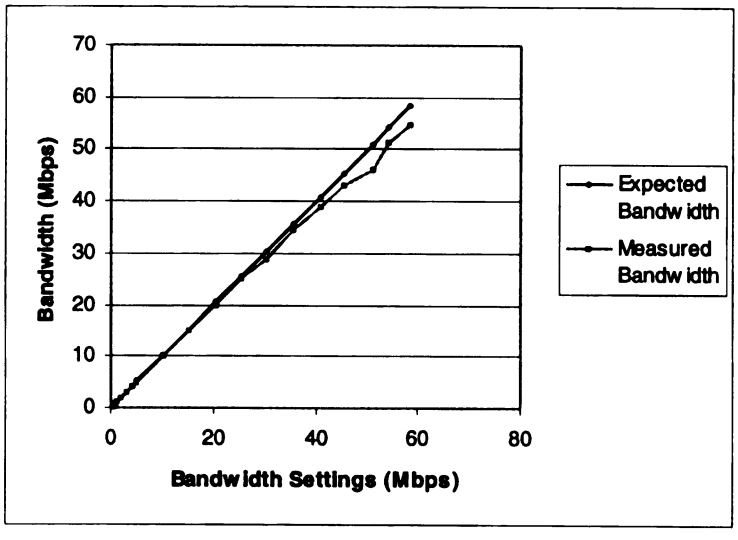

Figure 4.2 Link bandwidth emulation validation

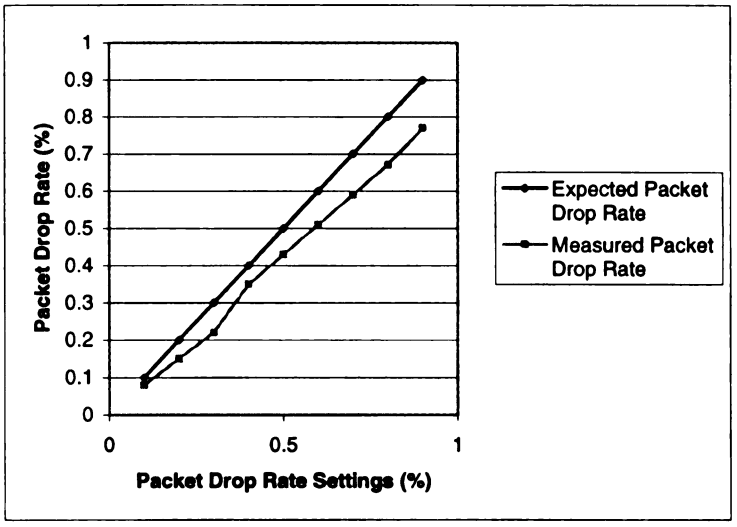

Figure 4.3 Packet drop rate validation

Based on the network parameter validation experiments, we conclude that EMPOWER is quite accurate in emulating these network parameters. The virtual device in EMPOWER also supports some other network parameters such as MTU, out-of-order packet delivery, bit error rate, etc. The validation experiments show that EMPOWER can be used as a central-control network emulator as well, although it provides more functionality than a common central-control network emulator.

4.2 Multiple Nodes Emulation

To illustrate how EMPOWER can be used to emulate a network topology, we present a sample network emulation experiment. The sample target network is shown in Figure 4.4, in which three routers exist. $Router_1$ and $Router_3$ have three physical ports. $Router_2$ has two physical ports. Each physical port has been marked with the last two octets of its IP address. A packet flow is generated between $PP_{12.1}$ on $Router_1$ and $PP_{16.1}$ on $Router_2$. To improve the complexity of the sample emulation, asynchronous routing is used in this

case. More specifically, network traffic entering the sample network at physical port $PP_{12.1}$ will be forwarded $PP_{14.2}$ through $PP_{14.1}$ on $Router_1$; network traffic entering the sample network at $PP_{16.1}$ will be forwarded to $PP_{15.1}$ through $PP_{15.2}$ on $Router_3$.

The sample target network is mapped to an emulation configuration as shown in Figure 4.5. Since there are a total of eight physical ports in the sample target network, an emulator node with eight network ports is sufficient to emulate the sample topology. Three virtual routers, VR_1, VR_2, and VR_3 have been generated. Both VR_1 and VR_2 have three network ports; VR_3 has two network ports. All the network ports connect to a fast Ethernet switch. Three virtual links are generated to emulate the physical links in the sample target network.

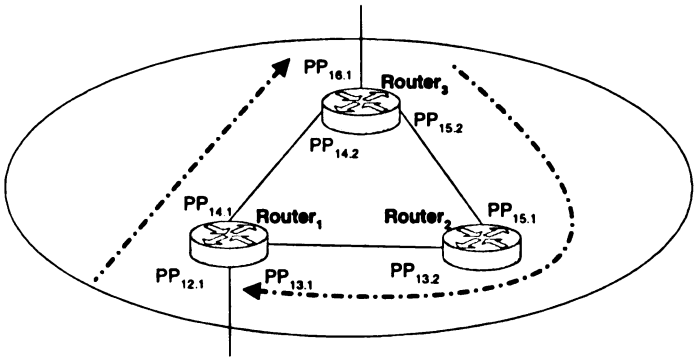

Figure 4.4 A sample target network

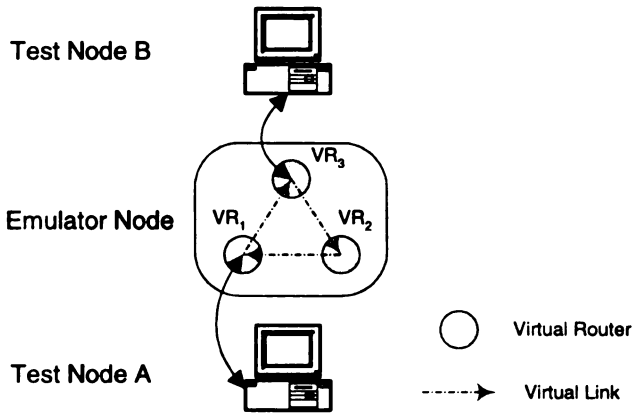

Figure 4.5 The emulation configuration of the sample target network

To generate network traffic and measure the end-to-end performance, we setup two test nodes, Test Node A and Test Node B. The gateway of Test Node A is the network port for $PP_{12.1}$. The gateway of Test Node B is the network port for $PP_{16.1}$. We use ttcp [36] as the traffic generator on both test nodes. In order to show the flexibility of EMPOWER, we change the network topology shortly after the data transmission begins by removing the unidirectional route of VR_3-VR_2-VR_1 from VR_3's routing table. Thus Test Node A cannot receive the acknowledgement from Test Node B. After several seconds, a new unidirectional route, VR_3-VR_1 for the traffic from Test Node B to Test Node A is established by adding the route to VR_3's routing table. A new unidirectional virtual link (from VR_3 to VR_1) is generated as well. As a result, data transmission between two test nodes resumes. The time sequence graphs of the TCP connection between two test nodes on each virtual router are shown in Figure 4.6, Figure 4.7, and Figure 4.8, respectively. Note that in this scenario, VR_2 forwards packets from Test Node B to Test Node A for about 3.8 seconds. After that there is no route for packets

from Test Node B to Test Node A, until a new route is added to VR_3 at about 10.5 seconds.

In addition to the experiment of topology mapping, we made another experiment on the emulation of network parameters. The virtual links in the sample emulation are configured to generate specific values of propagation latency and bandwidth, as shown in Table 4.1. We send 60 ICMP packets from Test Node A to Test Node B using "ping" program, measuring the round trip time of each packet. Similar to the previous experiment, the unidirectional route of VR_3-VR_2-VR_1 is removed at some time during the experiment, and a new unidirectional route of VR_3-VR_1 is added after some period of time. Figure 4.9 shows the round trip time of each ICMP packet during the experiment. There are three stages in this graph. Stage 1 shows an average round trip time of 7 milliseconds for ICMP packets. No response is measured in Stage 2, indicating a broken route between the sender and receiver. Then in Stage 3 those ICMP packets show an average round trip time of 5 milliseconds, after a new route has been installed on VR_3. Therefore, the route change in the sample network has been faithful emulated in our emulation configuration.

Our sample experiments show that the prototype of EMPOWER is capable of mapping a network topology to an emulation configuration. Performance analysis, measurement and traffic monitoring can be done on each virtual router to provide more insight of the underlying protocols and applications.

As an open emulation environment, EMPOWER is capable of incorporating any existing network protocol modules that are available in Linux. For example, to examine the impact of non-DS domains in a multi-domain DiffServ network, DiffServ module

[37] has been loaded into each emulator node of EMPOWER. Detailed of this application will be presented in the next Chapter. We have conducted several emulation experiments with EMPOWER to explore the underlying research issue [38].

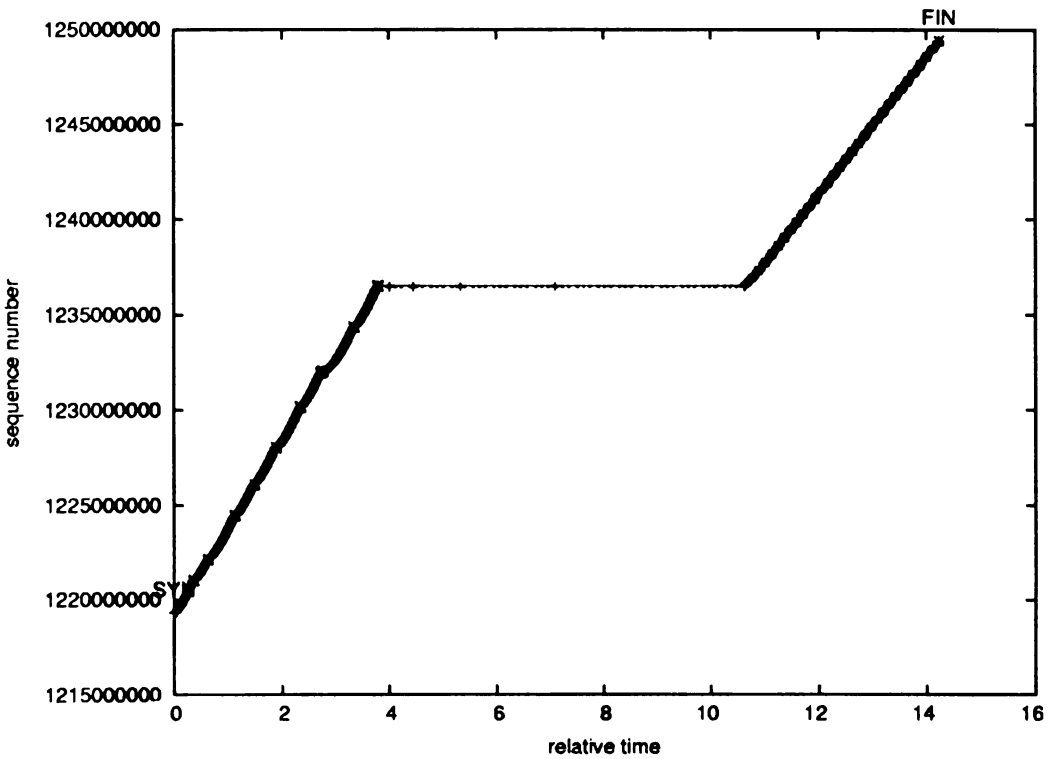

Figure 4.6 Time sequence graph on VR_1

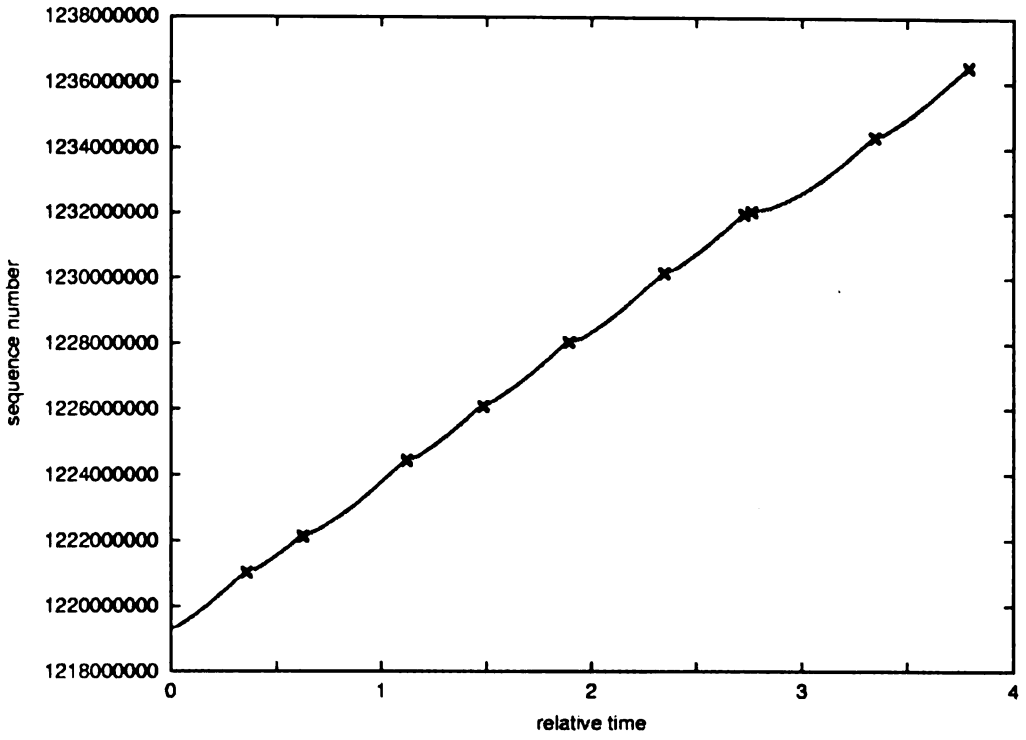

Figure 4.7 Time sequence graph on VR_2

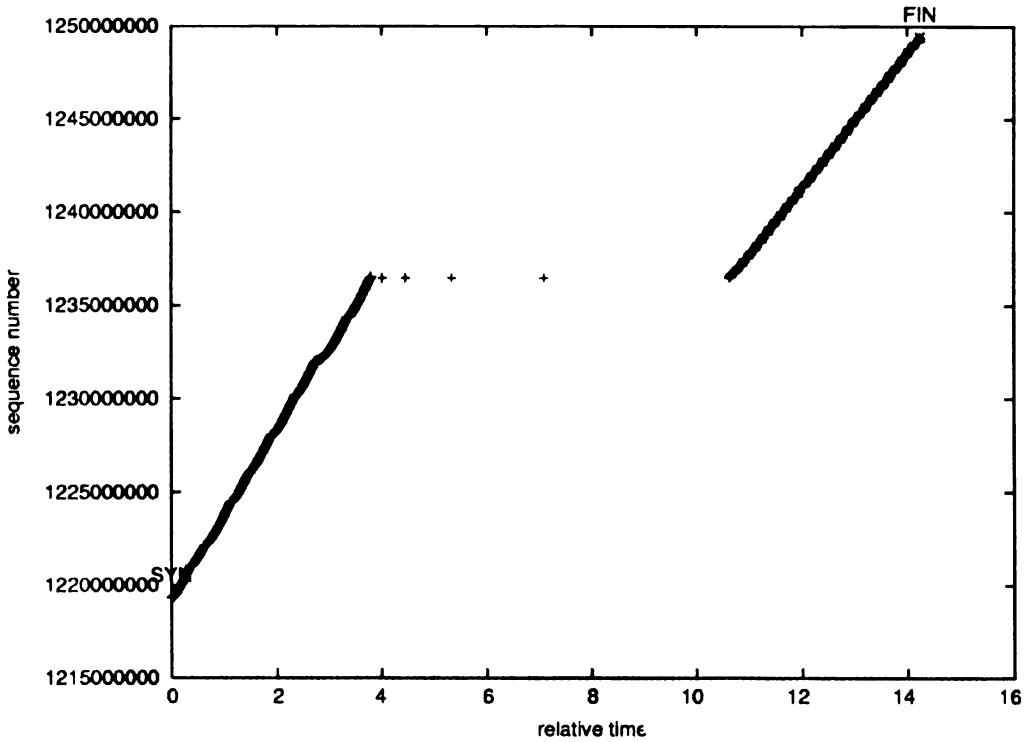

Figure 4.8 Time sequence graph on VR_3

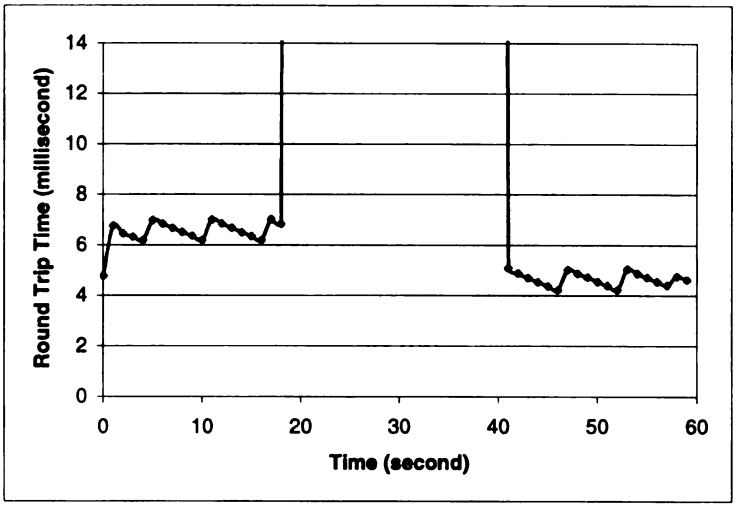

Figure 4.9 Round trip time of ICMP packets

Table 4.1 Network parameter settings

Virtual Link	Propagation Latency (millisecond)	Bandwidth (Mbps)
VR_1->VR_3	3	50
VR_3->VR_2	2	100 (default)
VR_2->VR_1	2	100 (default)
VR_3->VR_1	2	25

4.3 Wireless Network Emulation

To demonstrate EMPOWER's support for mobile wireless network emulation, we now introduce a sample emulation of a multi-hop mobile ad hoc network. The sample network consists of four mobile nodes, A, B, C and D, as shown in Figure 4.10. Node A

and Node C are two handheld devices with wireless LAN enabled. They do not move in this scenario, while Node B and Node D are two laptop computers that move in a predefined manner. The mobility information of this scenario is shown in Table 4.2. The origin of the 2-dimension coordinates is at the position of A. At time t=0 second, B has an initial coordinates of (20, 0) and begins to move upwards, C has an initial coordinates of (40, 0), D has an initial coordinates of (20, -18). At time t=15 seconds, B has moved to (20, 25) and then stays there. D begins to move upwards. At time t=18 seconds, D moves to (20, -15) and keeps moving upwards. At time t=33 seconds, D moves to (20,0). The distance between A and C is 40 feet, B and D are in the middle perpendicular to A and C. Each mobile node has a transmission range of 25 feet. The moving speed of B and D is 1 foot per second. Note that the transmission range and moving speed can be any values. We choose these values to simplify the illustration of mobility emulation.

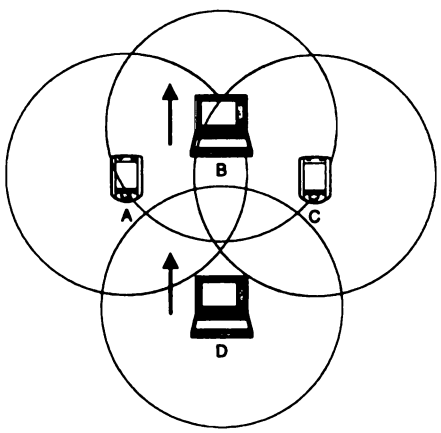

Figure 4.10 A sample mobile wireless network

Table 4.2 Mobility information of the sample scenario

Node	Time (second)			
	0	15	18	33

A	(0,0)			
B	(20,0)	(20,25)	(20,25)	(20,25)
C	(40,0)			
D	(20, -18)	(20,-18)	(20,-15)	(20,0)

In this scenario, A and C cannot communicate with each other directly because of the distance between each other. Initially B works as a gateway for A and C to forward packets to and from these two mobile nodes. When B is moving upwards and out of the transmission range of A and C (t=15s), A and C lose the connection between each other. At the same time D is moving upwards. After three seconds (t=18s), D is within the range of A and C, thus it works as a gateway for A and C to forward packets between them.

The network parameters of this sample network are shown in Table 4.3. We choose these parameters in order to demonstrate comprehensive emulation of the sample network. We can also obtain these parameters by measuring the performance of a real target network.

Table 4.3 Network parameters used in the sample emulation

Link	Latency (ms)	Bandwidth (Mbps)
A->B	3	10
B->A	2	10
B->C	2	10
C->B	<1	10
A->D	3	10

D->A	2	5
C->D	<1	10
D->C	4	5

The target network with its determined network parameters is mapped to an emulation configuration in EMPOWER. More specifically, the configuration file is read by the VMN control program, then four VMNs, VMN_A, VMN_B, VMN_C and VMN_D, are generated and properly configured according to the information in the configuration file. Since only four VMNs with eight virtual devices have to be generated, one emulator node with eight Ethernet ports is sufficient. In particular, the VMN control program constructs a mobility event list for each VMN, which will be used to schedule the events for each VMN at a specified time. In addition, the routing table of each VMN has to be modified due to the changes in the neighbor table. For example, at the time when D is about to enter the transmission range of A and C, the routing tables of VMNs for A and C must be modified to reflect this change. We plan to implement some proposed multi-hop ad hoc routing protocols in EMPOWER such that the installation and removal of a route for a VMN can be done automatically.

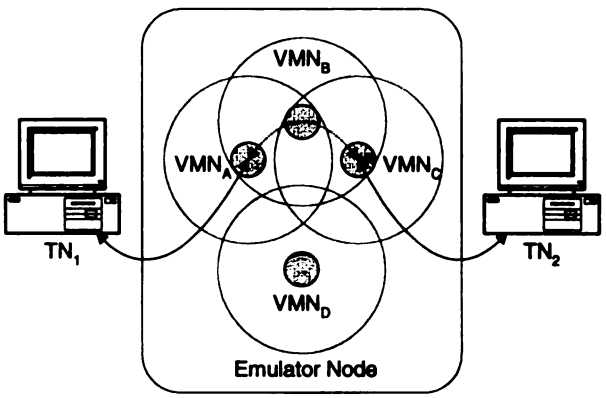

Figure 4.11 The emulation configuration of the sample network

We conduct the sample network emulation with two test nodes and an emulator node in EMPOWER, as shown if Figure 4.11. VMN_A is the gateway of one test node TN_1, while VMN_C is the gateway of the other test node TN_2. VMN_B or VMN_D will be the gateway of VMN_A and VMN_C, depending on their positions at that time. The routing tables of VMN_B and VMN_D ensure that packets from TN_1 to TN_2 will be forwarded to VMN_C, and packets from TN_2 to TN_1 will be forwarded to VMN_A. First, to verify the correct mobility event schedule of each VMN, a number of ICMP ECHO REQUEST are sent from TN_1 to TN_2. The round trip time of each packet is shown in Figure 4.12. In the first 15 seconds while B works as the gateway between A and C, the round trip times are on an average of 7 milliseconds. After that, there is no connection between A and C for 3 seconds. When D moves within the range of A and C and becomes the gateway, the round trip times are on an average of 9 milliseconds. Note that the time accuracy in EMPOWER is almost 1 millisecond.

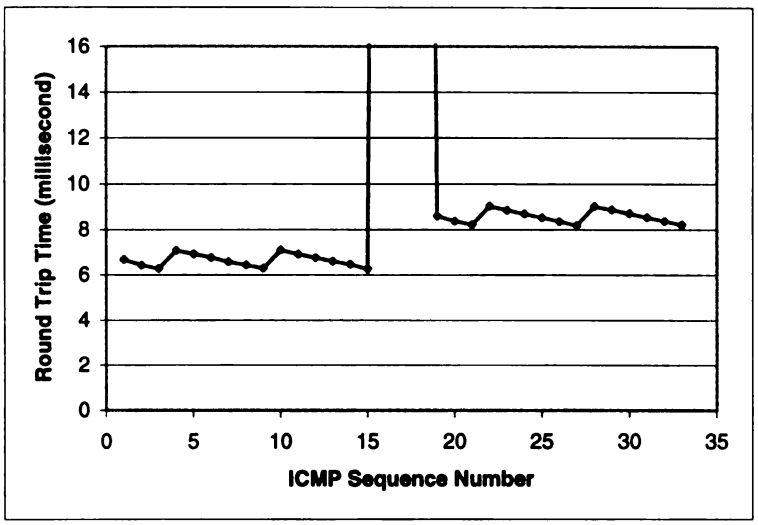

Figure 4.12 ICMP round trip time

To demonstrate how EMPOWER help examine the behavior of protocols in the target mobile ad hoc network, an ftp file transfer is performed between the two test nodes. The time-sequence graphs of the ftp data transfer connection on each VMN_A.

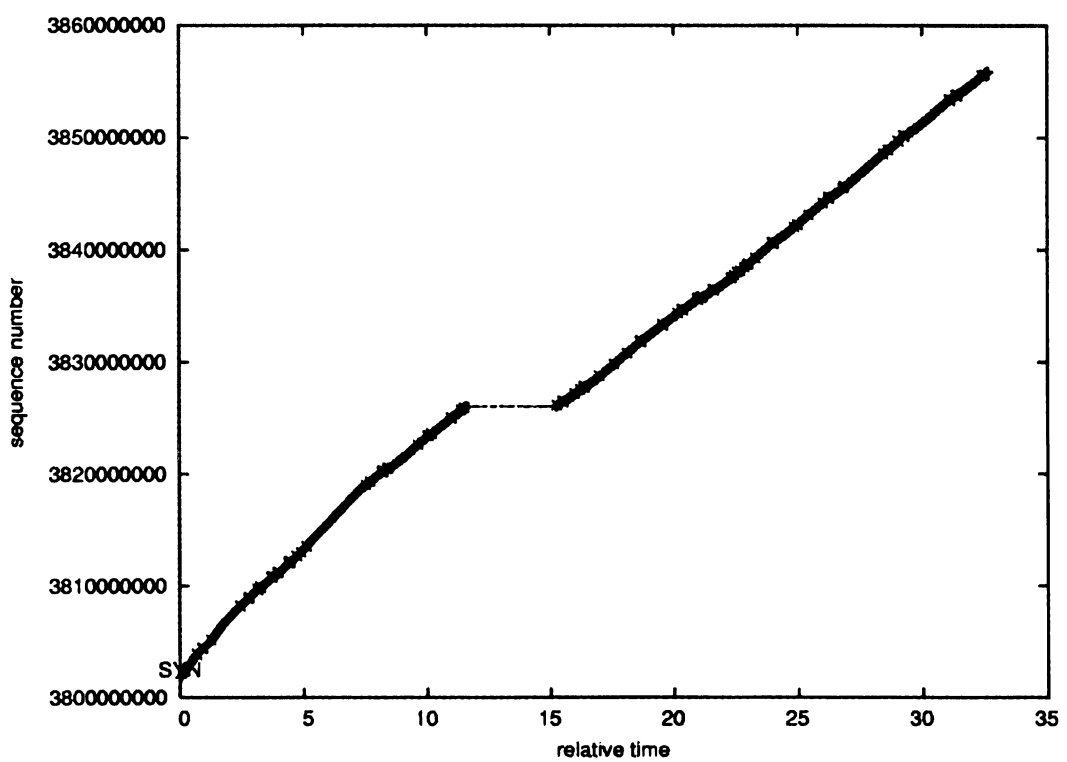

Figure 4.13 Time sequence graph of VMN_A

The time-sequence graphs suggest that the ftp data transfer begins at t=3.5s and continues for about 11.5 seconds. Then the connection was interrupted for 3 seconds, and then resume at t=18s. Thus all the network events in this scenario are accurately emulated by EMPOWER with only one emulator node. This emulation example demonstrates that EMPOWER provides an alternative for the evaluation of various TCP flavors in wireless networks.

CHAPTER 5 APPLICATIONS OF EMPOWER

In this chapter, we will present two applications of our emulator cluster EMPOWER. We show that using EMPOWER, we are able to easily setup desired emulation scenarios and generate specific network effects at will.

5.1 Performance Evaluation of Multi-Domain DiffServ Network

Traffic differentiation in wide area networks has been actively discussed in network research community. The need for scalable traffic differentiation and QoS provision in the Internet gives rise to the research of Differentiated Services [39] (DiffServ). In the Differentiated Services (DS) architecture, a DS domain consists of edge routers and core routers that are DS-compliant. All ingress traffic will be classified by marking the DSCP (Differentiated Services Code Point) in accordance with a Service Level Agreement (SLA). In the interior of a DS domain, core routers simply forward packets by applying different Per Hop Behaviors (PHBs) to each packet. Consequently, a DS domain might be capable of providing specific QoS for certain class of traffic.

Today's Internet consists of a large number of independent administrative domains or autonomous systems. Clearly not all these domains support DiffServ. In a sense, the SLA between the service provider of the DS domain and the user who accesses that DS domain may not be observed since the service provider cannot guarantee the same required service level because user's traffic may traverse a number of non-DS domains

before reaching the destination. ISPs can only guarantee the service level within its fully controlled administrative domains. Thus the problem of how to predict and evaluate the service level degradation of user's traffic in a hybrid multi-domain network needs to be addressed.

In order to cope with this issue, we propose a modeling scheme of a hybrid multi-domain DiffServ network, which can be used to analyze the service level degradation in the underlying network with respect to an All-DS network. In addition, we propose a performance metric called *Service Provision Index* to measure the service level degradation. Preliminary simulation and experimental emulation results show that a number of factors may affect the service level of a specific traffic flow. Based on the results, we make some observations on the problem to outline the key factors of the service level degradation and possible directions to improve it. Note that while the service level of premium service and assured service may be degraded in a hybrid multi-domain DiffServ network, the service level of best effort service may be improved. For simplicity we use the term "service level degradation" referring to the change of service level of all traffic classes.

We define a Hybrid Multi-Domain Network (HMDN) as a set of numerous independent DS domains or non-DS domains with a certain network topology. From a specific traffic flow's perspective, an HMDN can be simplified as a domain chain, as shown in Figure 5.1. These domains may be maintained by different ISPs. Note that in the case when all the domains are DS domains, the ISPs of two adjacent domains can substantially determine the inter-domain SLAs between those two domains. Specifically, a downstream domain is responsible for providing certain QoS guarantee to the

underlying traffic flow coming from the upstream domain. By repeating this kind of inter-domain service provision, the ISP of the first domain in the chain (user's access network) is able to assure an SLA to the user. In addition to this static-SLA solution, ISPs might use Bandwidth Brokers [40] between DS domains to dynamically allocate network resources for the traffic flow. The problem becomes more challenging when there are certain number of non-DS domains along the traffic path. These non-DS domains can only provide best effort service for all incoming traffic. Thus the SLA signed in pervious domains may not be observed. For ISPs, the service level degradation needs to be determined such that further traffic engineering and pricing schemes could be applied. We intend to examine and evaluate the difference of the end-to-end performance of the underlying traffic flow between an HMDN and its corresponding All-DS network. The metrics of the end-to-end performance we used are average packet delay and packet loss rate. We expect our approach can be extended to other performance metrics.

Let D(i) be the domains along the traffic path in the HMDN, where i ranges from 1 to m; m is the total number of domains in the HMDN. The first domain on the traffic path is called the *head* domain, while the last one is named the *tail* domain. Note that D(i) is mixture of DS domains and non-DS domains. We assume there are two classes of traffic: preferred traffic and non-preferred traffic. Preferred traffic corresponds to Premium Service, while non-preferred traffic may receive Assured Service or common best effort service. Assume both traffic flows arrive at the head domain according to a Poisson process. Let λ_1 and λ_2 denote the packet arrival rates of these two types of traffic respectively. We assume the SLAs between two adjacent DS domains have been established such that the routers in DS domains have been properly configured with

certain PHBs. Hence the priority of the preferred traffic, more specifically the corresponding DSCP, will be used to trigger EF PHBs or AF PHBs inside each DS domain. The non-DS domain simply treats the preferred traffic the same as the non-preferred traffic. Although we only introduce two traffic classes for simplicity, our approach can be extended to solve the problem that has more than two traffic classes.

5.1.1 HMDN Modeling

In today's Internet, the number of non-DS domains and the network topology can be arbitrary. From the perspective of a specific flow, an HMDN is simply a domain chain with a certain number of DS domains and non-DS domains. To clearly present our model, we use an example HMDN as shown in Figure 5.1 (a). In this example, only one non-DS domain exists. To explore the impact of the non-DS domain on the service of traffic flow, we study a corresponding All-DS network, as shown in Figure 5.1 (b). The models of these two networks are shown in Figure 5.2. We model a DS domain as a group of two FCFS queues with non-preemptive priority scheduling among them. On the other hand, a simple M/M/1/K queue is used to model a non-DS domain. The service time of each server in the queueing model is μ_i, whereas i is the domain number. The queues in all D(i) are numbered sequentially from the head domain to the tail domain as $q(j), j \in [1, s]$, where s is total number of queues in the system. Note that in each DS domain or non-DS domain, a large number of routers may exist. Though the performance of each router may vary in a wide range, they almost employ the same packet handling mechanism to process each traversing packet. In this sense, we are able to theoretically model a domain using the M/M/1/K queue. We expect that this simplification will not affect our quantitative result. We conduct quantitative analysis in

an effort to explore the fundamentals of the underlying problem in the early stage of the research. Without this simplicity we cannot obtain any quantitative and meaningful result.

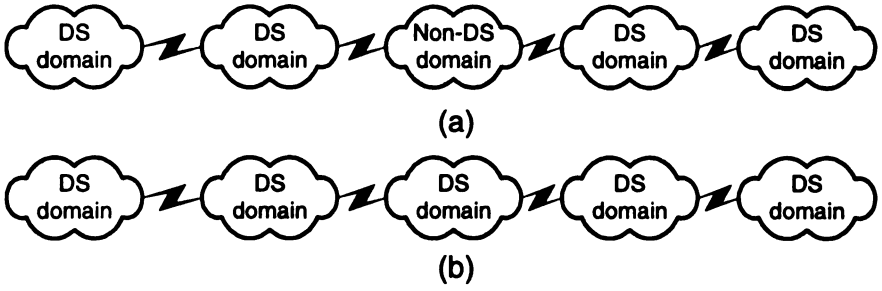

Figure 5.1 An HMDN and an All-DS domain

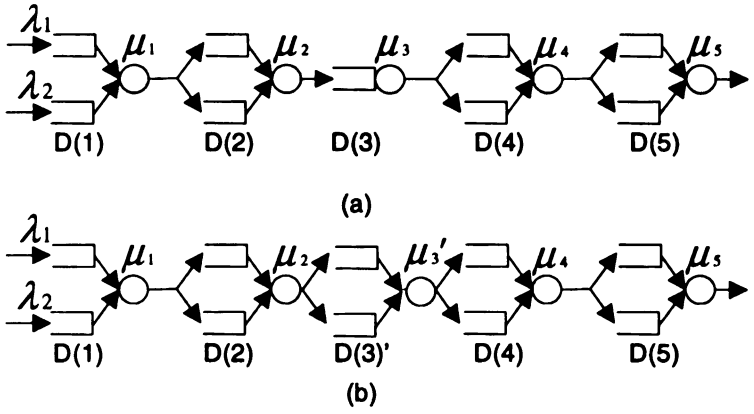

Figure 5.2 An HMDN model with five domains

It can be shown that this system follows a Markov process. We define the state variable of the system as a vector:

$$\xi = (k1, k2, k3, \cdots k9)$$

where $k_i, 1 \le i \le 9$ is the number of packets in each queue. Let $p(\xi)$ denotes the steady state probability of the state $\xi \in S, S$ is the state space. Let

matrix generator of the Markov chain. With $\pi \mathbf{P} = \mathbf{P}$ and $\sum_{\xi \in S} p(\xi) = 1$, the steady state probability of each state can be solved. In particular, **P** can be obtained from a state transition diagram.

5.1.2 An HMDN Example

As an example, we show a trivial HMDN with only one DS domain and one non-DS domain, as shown in Figure 5.3. The Markov chain of the example system can be solved by first reordering the three-dimension states to one-dimension states, and then apply GTH algorithm [41, 42].

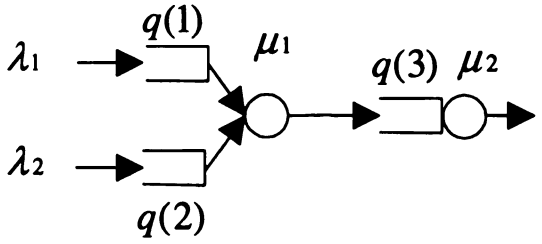

Figure 5.3 An example of trivial HMDN modeled

Let $\pi(k_1, k_2, k_3)$ be the steady state probability of state $\xi = \{k_1, k_2, k_3\}$. We aim to find the performance of the system for each class. For any packet, it can be possibly delayed while waiting in any queues because another packet is in service in the server. Thus the mean delay of preferred packets is:

$$W_1 = \frac{1}{\lambda_1} \sum_{k_3=0}^{K_3} \sum_{k_2=0}^{K_2} \sum_{k_1=0}^{K_1} k_1 \pi(k_1, k_2, k_3) + \frac{1}{(\lambda_1 + \lambda_2)} \sum_{k_3=0}^{K_3} \sum_{k_2=0}^{K_2} \sum_{k_1=0}^{K_1} k_3 \pi(k_1, k_2, k_3) \quad (1)$$

Similarly, the mean delay of non-preferred packets is:

$$W_2 = \frac{1}{\lambda_2} \sum_{k_3=0}^{K_3} \sum_{k_2=0}^{K_2} \sum_{k_1=0}^{K_1} k_2 \pi(k_1, k_2, k_3) + \frac{1}{(\lambda_1 + \lambda_2)} \sum_{k_3=0}^{K_3} \sum_{k_2=0}^{K_2} \sum_{k_1=0}^{K_1} k_3 \pi(k_1, k_2, k_3) \quad (2)$$

Generally, we intend to find the service level an HMDN can achieve for a specific traffic class. Let r be the number of traffic classes available in a DiffServ network. Traffic class r with a rate of λ_r has the highest priority. Let $\phi_i = \frac{\lambda_i}{\sum_{j=1}^{r} \lambda_j}, i \in [1, r]$ denote the probability of an arrival packet that belongs to class i. In (1) and (2), the mean packet delay varies with different traffic loads. Let W_i^H and W_i^{DS} denote the mean packet delays of class i packets in an HMDN and in an All-DS network, respectively. We define *Service Provision Index (SPI)* of packet delay:

$$\Omega_i^{delay} = \left| \frac{W_i^H - W_i^{DS}}{W_i^{DS}} \right| * \frac{W_i^H - W_i^{DS}}{W_i^{DS}}, i \in [1, r] \quad (3)$$

as the level of service level degradation achieved in the underlying HMDN with respect to packet delay. $\Omega^{delay} = 0$ means the HMDN is able to provide the same level of service in terms of mean packet delay. $\Omega^{delay} < 0$ indicates that the mean delay in an HMDN is smaller than that of the All-DS network. The service level is improved if $\Omega^{delay} < 0$. The service level is degraded if $\Omega^{delay} > 0$. The larger the *SPI*, the lower the service level an HMDN can provision for the traffic class. The overall *SPI* of an HMDN can be defined as:

$$\Omega^{delayy} = \sum_{i=1}^{r} \phi_i \Omega_i^{delay} \quad (4)$$

5.1.3 HMDN Performance Evaluation

Quantitative analysis of an HMDN described above is based on some assumptions such as the Poisson process of the traffic and the model of M/M/1/K queue for a router. Its major disadvantage is that when more DS domains or non-DS domains are added to the HMDN, the number of state in an HMDN will be quite large. However, quantitative analysis is still useful in that it presents a preliminary relationship between the service level and its major factors for small HMDNs. Indeed, before the study of large HMDNs, we need the analysis of small HMDNs to find out non-critical factors of performance criteria. For the example HMDN, let $\lambda_1 = \lambda_2 = 100$, $\mu_1 = \mu_2 = 300$, the average delay of two traffic classes with various buffer sizes are shown in Figure 5.4. The series "Pre" denotes preferred traffic. The series "Non-Pre" denotes non-preferred traffic. When the buffer size is small, preferred traffic demonstrates a clear higher priority over non-preferred traffic. In Figure 5.5, the average delay of two traffic classes are depicted with various packet arrival rates and a fixed buffer size of 20.

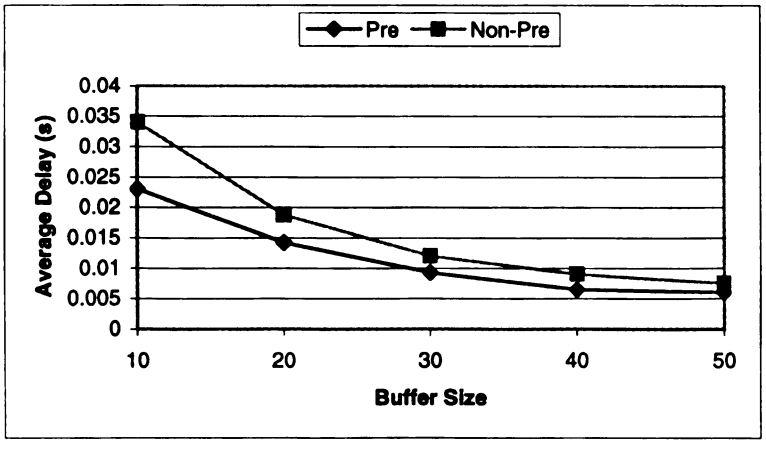

Figure 5.4 Average delay with various buffer sizes

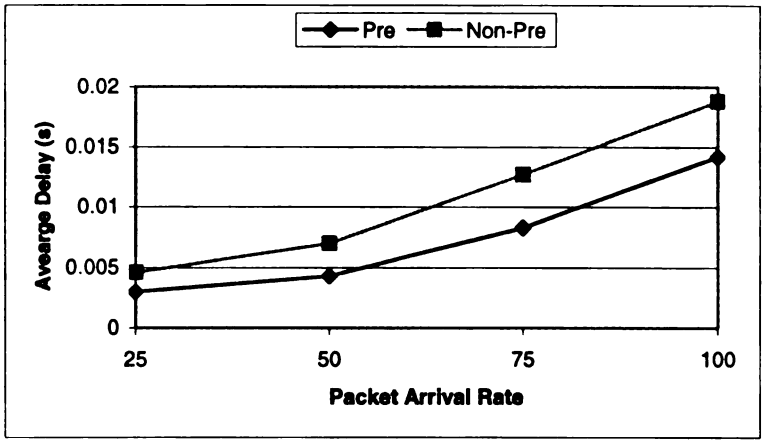

Figure 5.5 Average delay with various packet arrival rates

We use network simulator ns-2 [1] to simulate the HMDN in Figure 5.1. The simulation layout is shown in Figure 5.6. All the routers in this scenario are DS-compliant routers except R_3. To compare the performance of an HMDN and a related All-DS domain, after each simulation of an HMDN the non-DS router will be replaced by a DS router, as shown by the dot line in Figure 5.6. Two UDP traffic sources are generated to traverse this simple topology. The detailed simulation parameters are shown in Table 5.1. We choose RED as the queue management scheme for the non-DS router such that the results of an HMDN and All-DS network could be compared. Packets from S_1 will be marked as preferred traffic by the policer of the ingress routers (R_1 and R_4). The average end-to-end delay and drop rate of each traffic class are measured.

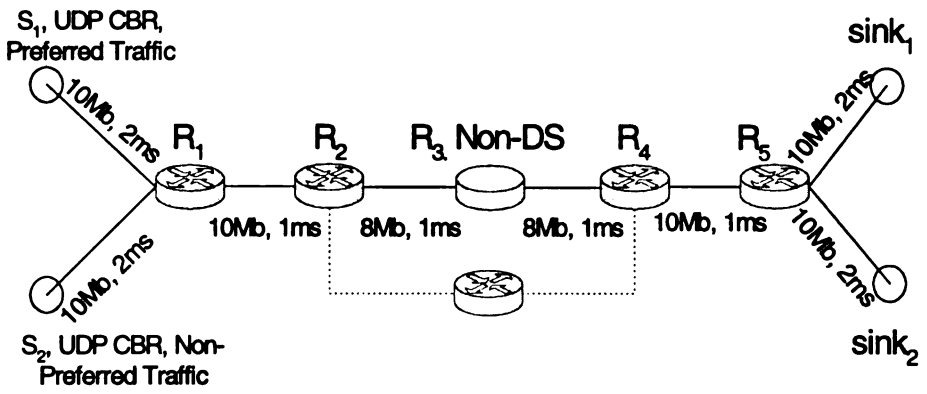

Figure 5.6 Simulation configuration

Table 5.1 Simulation parameters

Link bandwidth and delay between two DS routers	10Mbps, 1ms
Link bandwidth and delay between a non-DS router (R_3) and R_2 or R_4	8Mbps, 1ms
Link bandwidth and delay between traffic sources and the ingress DS router (R_1)	8Mbps, 2ms
Link bandwidth between the sinks and its adjacent DS egress router (R_5)	8Mbps, 2ms
DS routers queueing disciplines	RED with priority scheduling
Non-DS router (R_3) queueing discipline	RED
Policer on the edge DS routers	Token bucket
Traffic source	CBR with various rate and fixed packet size (1000 bytes)
Simulation Time	100s

Although simulation is easy to configure and setup, emulation and a testbed are still needed due to the drawbacks of simulation such as considerably long simulation time

with complicated topologies and too many assumptions. The comparison of simulation, emulation and testbed can be found in [30]. Emulation can be regarded as a real-time simulator. The major advantage of emulation is that the emulation time is real time, in contrast to simulation where simulation time is largely affected by the complexity of the simulated network.

The emulation experiments are conducted in EMPOWER [30], as described in the previous chapter.

The Differentiated Services implementation for Linux [37] has been modified to support multiple virtual routers in one Linux kernel. We use the same test scenarios and parameter settings as in the simulation experiments. The emulation configuration in EMPOWER is depicted in Figure 5.7. The dash lines represent the logical packet flows between two adjacent virtual routers. A traffic generator program is executed on test node A to send UDP packets to test node B, where some measurements and calculations are conducted.

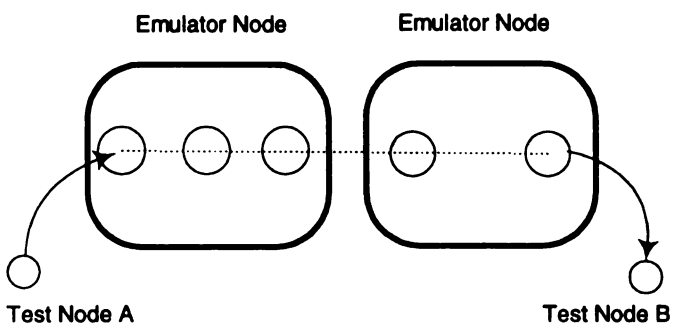

Figure 5.7 Logic view of the emulation configuration

Two scenarios are setup for the same simulation layout in Figure 5.6. In Scenario 1, the rates of both traffic classes will be increased equally. In Scenario 2, only the rate of preferred traffic will be increased. The simulation results of scenario 1 are shown in

Figure 5.8, Figure 5.9, and Figure 5.10. The impact of the non-DS router on preferred and non-preferred traffic can be clearly seen from those figures. As in Figure 5.8, when the overall traffic load does not cause any congestion along the links, the end-to-end delays of both traffic classes are almost the same in the All-DS network and in the HMDN network. While the traffic load increases, the preferred traffic in an All-DS network experience considerably low end-to-end delays (less than 20ms), as compared with that of non-preferred traffic (larger than 90ms) in the same network. In addition, non-preferred traffic in the All-DS network suffers even higher packet drop rate than in the HMDN due to the service differentiation on the congested links. Significant change of *SPI* can be seen for preferred traffic when the non-DS router is introduced. However, for non-preferred traffic, the change is not that much. Note that the service level of non-preferred traffic has been improved because of the non-DS router.

In Scenario 2, in order to observe the effect of the ratio of one traffic class in the overall traffic load, we vary the rate of preferred traffic. The *SPI* in Scenario 2 for preferred traffic is a slightly smaller than that of Scenario 1. In addition, we need to examine the impact of link bandwidth of the non-DS router on the server degradation. In previous simulation scenarios, link bandwidth of the non-DS router is 8Mbps, slightly smaller than bandwidth of other DS routers. Accordingly, when the traffic rate increases, the non-DS router becomes the bottleneck. Now with a fixed overall traffic load of 8Mbps, the average packet delay under various link bandwidth of the non-DS router is shown in Figure 5.11. For comparison we also set link bandwidth of the corresponding DS router in the All-DS domain to be the same of the non-DS router.

To find the impact of multiple non-DS domains in an HMDN, multiple non-DS routers are placed on the traffic path. Each non-DS router has the same configuration described in Table 5.1. As shown in Figure 5.12, there is a sharp increase of *SPI* for preferred traffic when the first non-DS is introduced. After that, no significant changes of *SPI* can be observed. Similarly, non-preferred traffic experiences a clear gain in service level when the first non-DS router is introduced. Additional simulations show that some "key" non-DS domains play important roles to determine the trend of *SPI*, which will be detailed in the next section.

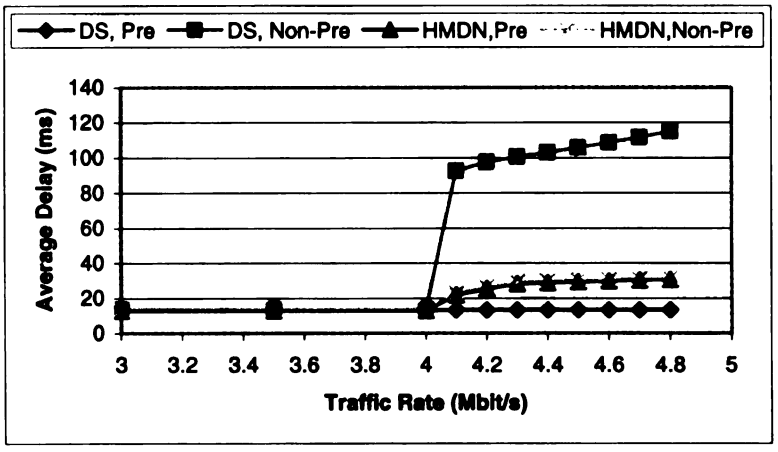

Figure 5.8 Average delay with various traffic rates

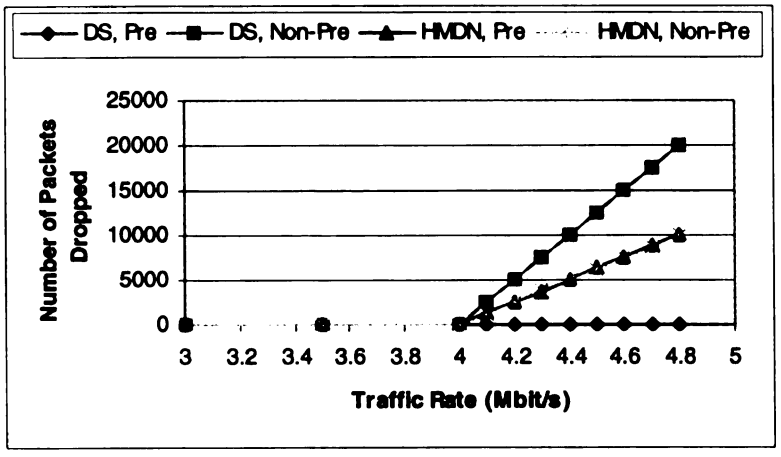

Figure 5.9 Number of packets dropped with various traffic rates

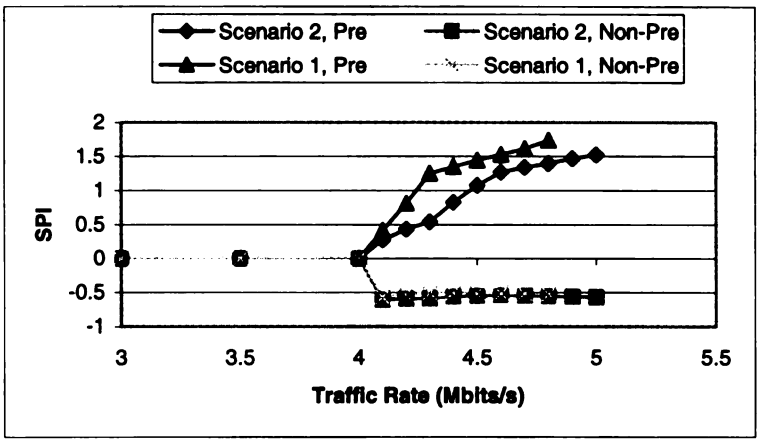

Figure 5.10 *SPI* in two scenarios

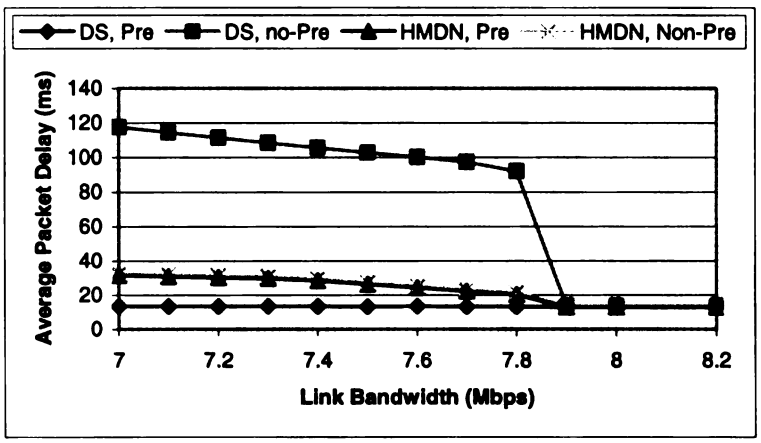

Figure 5.11 Average packet delay with various link bandwidths and fixed traffic load

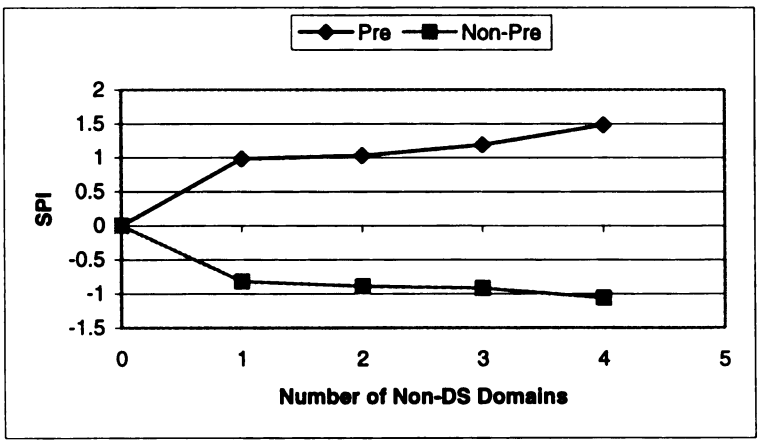

Figure 5.12 *SPI* with number of Non-DS domains

Emulation results are shown in Figure 5.13 and Figure 5.14. As shown in Figure 5.13, while the packet arrival rate of non-preferred traffic increases in the All-DS domains network, preferred traffic is able to maintain a low average packet delay. When a non-DS domain with a link whose bandwidth is slightly smaller than other links is introduced to the traffic path, preferred traffic suffers a dramatic delay increase, while non-preferred traffic achieves a clear delay decrease. Similar to the simulation result shown in Figure 5.8, the DS domains are account for the service differentiation of preferred and non-preferred traffic in the HMDN. Figure 5.14 illustrates the *SPI* with various number of non-DS domains sharing the same configuration as described in previous single non-DS domain case. It is clear that, given some non-DS domains with potential congestion links, the number of non-DS domains barely affects the *SPI* of either preferred or non-preferred traffic.

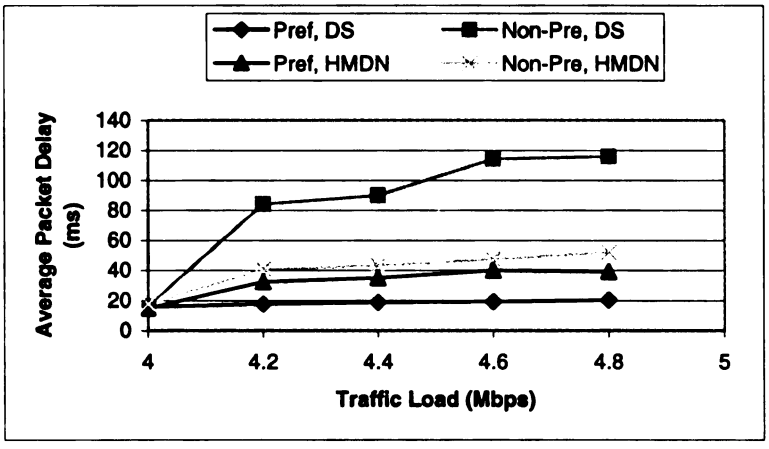

Figure 5.13 Average packet delay with various traffic rates in an emulation experiment

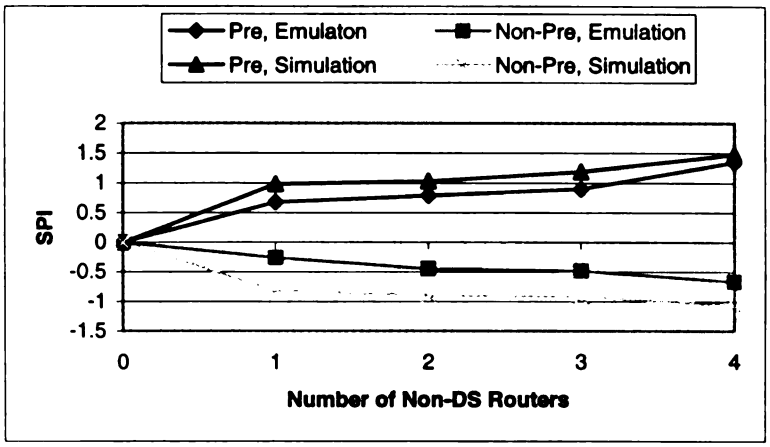

Figure 5.14 *SPI* with number of non-DS routers in an emulation experiment with EMPOWER

5.1.4 Discussion of the Results

We make the following observations based on the quantitative analysis, the simulation and emulation results. First, the link bandwidth of a non-DS domain is critical to the performance of preferred traffic. Given that there is no bottleneck in other part of an HMDN, this link is mostly responsible for the service degradation in which no distinct service differentiation between different traffic classes are observed. Its may result in an *SPI* that is much larger than that in the case when the bottleneck link belongs to any other DS domains. When the bandwidth is sufficiently large along the traffic path, we cannot expect any service degradation.

Second, in a multiple non-DS domains case, there always exists some number of "key" DS domains and non-DS that plays much important role than the other DS domains. The "key" DS domains may provision the lowest service level for certain class of traffic than other DS domains, or impose strict traffic policing mechanisms on preferred traffic. The "key" non-DS domains may have some links with the lowest

bandwidth than other domains. These "key" domains largely affect the benefit of service differentiation in the head domain.

Third, the number of non-DS domains in an HMDN will not strongly affect the service level degradation given that they have the same link configurations, as is the case in our simulation. However, their locations in the HMDN may make significant difference on the service level degradation. If the non-DS domains have different link configurations, the "key" non-DS domain becomes the major factor of service degradation of the HMDN. Indeed, we consider a non-DS domain as a specific packet handling mechanism that attempts to decrease or eliminate the effect of service differentiation in previous DS domains, whereas a DS domain's major functionality can be regarded as another packet handling mechanism that intends to realize some predefined service differentiation on traffic classes. A non-DS domain's capability to change the service level degradation depends on a variety of parameters such as its link bandwidth, its location on the traffic path, its congestion control mechanism and its performance. A DS domain's capability to change the service level degradation is determined by its service differentiation configuration, its link bandwidth and its performance. The overall effect of an HMDN is the sum of each domain's impact on the service level degradation, which can be either positive or negative.

5.2 A Highly Available Storage System using Cluster

5.2.1 CoStore Overview

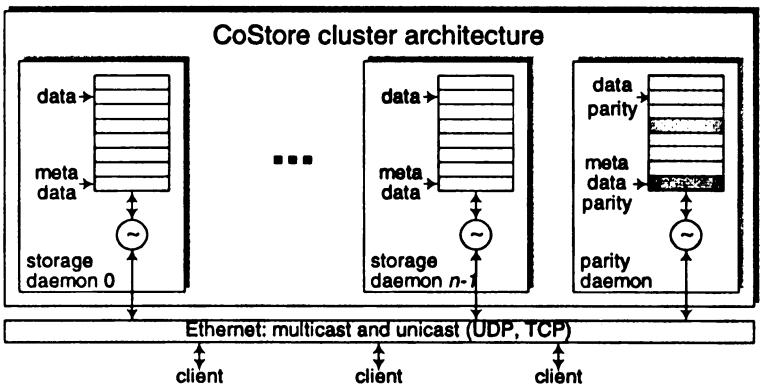

Figure 5.15 CoStore cluster architecture with RAID level 4

CoStore architecture [44] was initially proposed to solve the increasingly pervasive disk-space wasting problem in many organizations [45]. The authors have conducted a feasibility study of building storage systems utilizing the idle disk space on desktop workstations and deploying the proposed architecture in existing desktop computing infrastructure. The basic idea of CoStore has been extended to support mirroring among a number of remote sites in wide area networks. Compared with a traditional point-to-point clustered pair, a multi-entity CoStore cluster with mirroring has several advantages: larger volume size, higher aggregated network bandwidth and performance. Due to the number of active members serving in a cluster, there are multiple members in a cluster each with local disk resources and there exist multiple network connections between storage and mirror clusters.

A CoStore cluster implements a virtual storage server with a file interface as in other storage appliance products using a network attached storage (NAS) approach. Members

in a cluster form an IP multicast group. As shown in Figure 5.15, a CoStore cluster consists of a group of network attached storage devices and clients that access the storage resources in the cluster. All members in a CoStore cluster work collaboratively to construct a storage system with a unified file namespace, i.e. one root directory. One multicast IP address for a CoStore cluster gives clients an impression of a single server initially. This virtual CoStore server is a serverless design because no central file manager is required to maintain the consistency of the overall namespace in the cluster. Both file system metadata (and hence local file system management) and distributed file system responsibilities are evenly distributed across all participating members.

5.2.2 Test Experiment

A CoStore prototype has been implemented on Windows 2000. Currently the fault-tolerance subsystem supports RAID level 0/0+1 and 4/4+1. A number of experiments in a LAN and campus network have been conducted. However, as we do not have access to computers that reside coast-to-coast, we are not able to further expand the distance between cluster members in high-bandwidth, high packet delay networks such as in a data center. Note that our network emulator is able to emulate multiple routers and links that form a network topology. An emulated link may have bandwidth up to 100Mbps, packet delay of several hundreds of milliseconds. Hence, EMPOWER is used to evaluate the performance of mirror daemons located far away from storage daemons. In this case, a single node EMPOWER node acts as a router between storage daemons and mirror daemons. The packet delay parameter is set to 30ms in either direction and the measured TCP throughput is 32.8Mbps using TTCP. EMPOWER can also be used to generate a

network topology, in which the reliability and disaster recoverability of CoStore is tested against some link failure or node failure.

Case studies in the chapter show that EMPOWER can be used to leverage protocol test in various aspects. The distributed nature of EMPOWER would allow the emulation of more complex scenarios that are very difficult to generate and control in field test. Emulation result from EMPOWER is believed to be more close to filed deployment than network simulation.

CHAPTER 6 CONCLUSION REMARKS

6.1 Summary of the Work

A scalable and flexible network emulation system is highly needed for the research of network protocols, systems, and applications. In particular, network topology should be emulated such that topology-related network protocols could be tested and evaluated against emulated network conditions and traffic dynamics. Existing network emulators either fail to emulate network topology, or are mostly non-scalable in terms of emulated network capacity. In this research project, we proposed a distributed network emulator cluster EMPOWER, which provides such an emulation environment with a number of commodity computers as the infrastructure and software modules to map a target network to an emulation configuration and impose predefined network conditions and traffic dynamics to traversing traffic. Each emulator node in EMPOWER can possibly emulate multiple virtual routers, making the whole emulator cluster highly scalable in terms of emulation capacity. The resource competition problem of EMPOWER can be substantially solved by the virtual-router mapping scheme, in which the maximum number of active network ports in an emulator node can be determined and the target network is then partitioned accordingly to avoid resource competition in an emulator node. In addition, EMPOWER provides a mechanism to emulate the mobility of a wireless network in a wireline network, thus mobile ad hoc networks can be emulated in a wireline network. EMPMOWER is flexible and easy-to-use for emulation configuration in that the whole system can be configured by a script that defines network

traffic and topology dynamics of a test scenario. Our preliminary results show that EMPOWER is quite accurate in emulating network effects and network topology, as well as node mobility in wireless networks. The EMPOWER prototype has been used by a number of research projects such as performance evaluation of hybrid multi-domain DiffServ Network and a high availability storage cluster.

6.2 Future Work

6.2.1 High Bandwidth Emulation

The problem of bandwidth emulation scalability remains an open research issue in EMPOWER. Ethernet channel bonding or Gigabit Ethernet might be used to provide a higher bandwidth emulation infrastructure. However, this will in turn reduce the number of virtual routers in an emulator node and bring another factor of resource competition to the system. Another approach to this problem is to scale down link bandwidth by some factor such that the resulted bandwidth can be emulated in a local area network. However, such a bandwidth scaling will also influence emulation accuracy and validity. In addition, should traffic rate in the emulated network be scaled down at the same time? How can we derive emulation result for the target network based on that from scaled emulated network? What is the relationship between the scaling factor and result? These interesting problems are yet to be investigated in the future.

6.2.2 Emulation versus Simulation

In this research, we have identified the advantages and disadvantages of two existing approaches for performance evaluation and implementation testing: network simulation

and network emulation. Indeed, most research issues that require performance evaluation will be first evaluated by simulation in order to get fast and comprehensive result due to the flexibility of simulation. As we focus on providing a "real" network environment in which protocols and applications perform exactly the same as in a target network, we choose to work on the research of network emulation as an effort to address some fundamental problems of existing network emulation solutions. As part of our future research, we need to compare emulation and simulation in various aspects by establishing the same scenario in the emulator cluster and a network simulator. The comparison will give us more insight for the improvement of our emulator cluster.

6.2.3 Wireless Network Emulation

In addition, we plan to incorporate hybrid emulation and a simulated physical layer of wireless network into EMPOWER such that the wireless emulation facility of EMPOWER can be enhanced to support the emulation of physical layer and data link layer. We are also in the process of using EMPOWER to explore a variety of research issues about routing protocols and QoS provisioning in both wireless and wireline networks.

BIBLIOGRAPHY

[1] The Network Simulator: ns-2, http://www.isi.edu/nsnam/ns/

[2] S. Park, A. Savvides, and M. B. Srivastava, "Simulating Networks of Wireless Sensors," In *Proceedings of the 2001 Winter Simulation Conference*, 2001.

[3] R. Bagrodia, R. Meyer, M. Takai, Y. Chen, X. Zeng, J. Martin, B. Park, and H. Song, "Parsec: A Parallel Simulation Environment for Complex Systems," *IEEE Computer*, vol. 31, pp. 77-85, October 1998.

[4] GloMoSim, http://pcl.cs.ucla.edu/projects/glomosim/

[5] R. M. Fujimoto, "Parallel Discrete Event Simulation," *Communications of the ACM*, vol. 33, pp. 30-53, October 1990.

[6] J. Flynn, H. Tewari, and D. O'Mahony, "JEmu: A Real Time Emulation System for Mobile Ad Hoc Networks," In *Proceedings of the Firsit Joint IEI/IEE Symposium on Telecommunications Systems Research*, Dublin, Ireland, November 2001.

[7] M. Kojo, A. Gurtov, J. Manner, P. Sarolahti, T. Alanko, and K. Raatikainen, "Seawind: a Wireless Network Emulator," In *Proceedings of 11th GI/ITG Conference on Measuring, Modeling and Evaluation of Computer and Communication Systems(MMB 2001)*, RWTH Aachen, Germany, 2001.

[8] I. Yeom and A. L. N. Reddy, "ENDE: An End-to-end Network Delay Emulator," Master's thesis, Texas A&M University, 1998

[9] NIST NET, http://snad.ncsl.nist.gov/itg/nistnet/

[10] M. Allman, A. Caldwell, and S. Ostermann, "ONE: The Ohio Network Emulator," Ohio University, Tech Report TR-19972, August 1997.

[11] D. B. Ingham and G. D. Parrington, "Delayline A Wide-Area Network Emulation Tool," Department of Computing Science, University of Newcastle.

[12] L. Rizzo, "Dummynet: a simple approach to the evaluation of network protocols," *ACM Computer Communication Review*, vol. 27, 1997.

[13] J. S. Ahn, P. B. Danzig, Z. Liu, and L. Yan, "Evaluation of TCP Vegas: Emulation and Experiment," In *Proceedings of SIGCOMM*, 1995.

[14] X. W. Huang, R. Sharma, and S. Keshav, "The ENTRAPID Protocol Development Environment," In *Proceedings of IEEE INFOCOM*, March 1999.

[15] R. Simmonds, R. Bradford, and B. Unger, "Applying Parallel Discrete Event Simulation to Network Emulation," In *Proceedings of the 14th Workshop on Parallel and Distributed Simulation*, May 2000.

[16] http://www.shunra.com/products.htm

[17] B. D. Noble, M. Satyanarayanan, G. Nguyen, and R. Katz, "Trace-Based Mobile Network Emulation," In *Proceedings of Proceedings of SIGCOMM '97*, September 1997.

[18] M. Satyanarayanan and B. Noble, "The Role of Trace Modulation in Building Mobile Computing Systems," In *Proceedings of the 6th Workshop on Hot Topics in Operating Systems*, Cape Cod MA, USA, May 1997.

[19] K. Fall, "Network Emulation in the VINT/NS Simulator," In *Proceedings of 4th IEEE Symposium on Computers and Communications*, July 1999.

[20] F. Kubinszky, "Emulation of ad-hoc networks on IEEE 802.11," Budapest University of Technology and Economics, Budapest, Hungary, MS Thesis 2000.

[21] Packetstorm IP Network Emulators, http://www.packetstorm.com

[22] EMPIRIX PacketSphere, http://www.empirix.com/empirix/voice+network+test/products/_hammer+packetsphere.html

[23] Emulab: The Utah Network Testbed, http://www.emulab.net

[24] J. Flynn, H. Tewari, and D. O'Mahony, "A Real Time Emulation System for Ad hoc Networks," In *Proceedings of Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS2002)*, San Antonio, Texas, January 2002.

[25] The CMU Monarch Project's Wireless and Mobility Extensions to NS, http://www.monarch.cs.cmu.edu

[26] N. Davies, G. S. Blair, K. Cheverst, and A. Friday, "A Network Emulator To Support the Development of Adaptive Applications," In *Proceedings of 2nd USENIX symposium on Mobile and Location Independent Computing*, Ann Arbor, U.S.A., April 1995.

[27] S. Bajaj, L. Breslau, D. Estrin, K. Fall, S. Floyd, P. Haldar, M. Handley, A. Helmy, J. Heidemann, P. Huang, S. Kumar, S. McCanne, R. Rejaie, P. Sharma, K. Varadhan, Y. Xu, H. Yu, and D. Zappala, "Improving Simulation for Network Research," Department of Computer Science, USC, Tech Report 99-702b, 1999.

[28] W. S. Cleveland and D. X. Sun, "Internet Traffic Data," Statistics and Data Mining Research, Bell Labs.

[29] K. K. Dam and L. M. Ni, "Design and Implementation of a Network Emulator," Department of Computer Science, Michigan State University, East Lansing, Michigan, Tech Report MSU-CPS-ACS-98-16, 1998.

[30] P. Zheng and L. M. Ni, "EMPOWER: A Scalable Framework for Network Emulation," In *Proceedings of the 2002 International Conference on Parallel Processing (ICPP '02), In Press*, Vancouver, Canada, 2002.

[31] J. E. G. Coffman, M. R. Garey, and D. S. Johnson, "Approximation Algorithms for Bin Packing: A Survey," in *Approximation Algorithms for NP-Hard Problems*, D. Hochbaum, Ed. Boston: PWS Publishing, 1997.

[32] V. Paxson and S. Floyd, "Wide Area Traffic: The Failure of Poisson Modeling," *IEEE/ACM Transaction on Networking*, vol. 3, pp. 226-244, 1995.

[33] W. Willinger, M. Taqqu, R. Sherman, and D. Wilson, "Self-Similarity Through High Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level," *IEEE/ACM Transaction on Networking*, vol. 5, 1997.

[34] S. S. Kang and M. W. Mutka, "Provisioning Service Differentiation in Ad Hoc Networks by the Modification of Backoff Algorithm," In *Proceedings of Int'l Conference on Computer Communication and Network (ICCCN) 2001*, Scottsdale, Arizona, October 2001.

[35] B. Bensaou, Y. Wang, and C. C. Ko, "Fair Media Access in 802.11 based Wireless Ad-hoc Networks," In *Proceedings of Mobile and Ad Hoc Networking and Computing 2000*, 2000.

[36] ttcp, http://www.cisco.com/warp/public/471/ttcp.html

[37] Sourceforge Differentiated Services on Linux, http://diffserv.sourceforge.net/

[38] P. Zheng and L. M. Ni, "The Impact of Non-DS Domains in a Multi-Domain DiffServ Network," In *Proceedings of ICCCN'02*, Miami, Florida, In Press, 2002.

[39] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," *RFC 2475*, December 1998.

[40] V. Jacobsen, K. Nichols, and L. Zhang, "A two-bit Differentiated Services Architecture for the Internet," *Internet Draft*, June 1999.

[41] W. K. Grassmann, M. I. Taskar, and D. P. Heyman, "Regenerative Analysis and Steady State Distributions for Markov Chains," *Operations Research*, vol. 33, pp. 1107-1116, 1985.

[42] W. Stewart, "Introduction to the Numerical Solution of Markov Chains," Princeton University Press, 1994, pp. 84-86.

[43] P. Zheng and L. M. Ni, "EMPOWER: A Scalable Framework for Network Emulation," Department of Computer Science and Engineering, Michigan State University, Tech Report MSU-CSE-02-5, 2002.

[44] Y. Chen, L. M. Ni, M. Yang, C. Xu, J. F. Kulser, and P. Zheng, "CoStore: A Reliable and Highly Available Storage System Using Cluster," In *Proceedings of the 16th Annual International Symposium on High Performance Computing Systems and Applications (HPCS'02)*, 2002.

[45] Y. Chen, L. M. Ni, M. Yang, and P. Mohapatra, "CoStore: a Distribued File System Utilizing Disk Space on Workstation Clusters," In *Proceedings of 21st IEEE International Performance, Computing, and Communications Conference (IPCCC02)*, 2002.