TrainNet: a novel transport infrastructure for non real-time data delivery

Mohammad Zarafshan Araki
University of Wollongong

Zarafshan Araki, Mohammad, TrainNet: a novel transport infrastructure for non real-time data delivery, ME-Res thesis, School of Electricial, Computer and Telecommunications Engineering, University of Wollongong, 2009. http://ro.uow.edu.au/theses/842

TrainNet: A Novel Transport Infrastructure for Non Real-Time Data Delivery

A thesis submitted in fulfilment of the
requirements for the award of the degree

Master of Engineering (Research)

from

THE UNIVERSITY OF WOLLONGONG

by

Mohammad ZARAFSHAN ARAKI
Master of Internet Technology (with Distinction)

SCHOOL OF ELECTRICAL, COMPUTER
AND TELECOMMUNICATIONS ENGINEERING
2009

This page is blank

Abstract

To date, researchers have proposed many vehicular networks in which cars or buses act as a mechanical backhaul for transporting data. For example, a bus can be retrofitted with a computer and wireless card to automatically ferry data to/from rural villages without Internet connectivity. Alternatively, a person carrying a portable storage device can be used to link geographically disparate networks. These examples of challenged networks are characterized by frequent disruptions, long delays, and/or intermittent connectivity.

This thesis proposes TrainNet, a vehicular network that uses trains to transport latency insensitive data. TrainNet augments a railway network by equipping stations and trains with mass storage devices; e.g., a rack of portable hard disks. TrainNet has two applications. First, it provides a low cost, very high bandwidth link that can be used to deliver non real-time data. In particular, cable TV operators can use TrainNet to meet the high bandwidth requirement associated with Video on Demand (VoD) services. Moreover, TrainNet is able to meet this requirement easily because its links are scalable, meaning their capacity can be increased inexpensively due to the continual fall of hard disk price. Secondly, TrainNet provides an alternative, economically viable, broadband solution to rural regions that are reachable via a railway network. Therefore, using TrainNet, rural communities will be able to gain access to bandwidth intensive digital contents such as music, video, television programs, and movies cheaply.

A key problem in TrainNet is resource scheduling. This problem arises because stations compete for the fixed storage capacity on each train. To this end, this thesis is the first to propose three max-min scheduling algorithms, namely LMMF, WGMMF and GMMF, for use in challenged networks. These algorithms arbitrate the hard disk space among competing stations using local traffic information at each station, or those from other stations. To study these algorithms, the Unified Modeling Language (UML) is first used to construct a model of TrainNet, before a simulator is constructed using the DESMO-J framework. The resulting TrainNet simulator is then used to investigate the behavior of said max-min algorithms in scenarios with realistic traffic patterns. Results show that while LMMF is the fairest algorithm, it results in data loss and has the longest mean delay, the lowest average throughput, and the lowest hard disk utilization. Furthermore, Jain's fairness index shows WGMMF to be the least fair algorithm. However, it avoids data loss as is the case with GMMF, and achieves the best performance in terms of mean delay, averaged throughput, and hard disk utilization.

Statement of Originality

This is to certify that the work described in this thesis is entirely my own, except where due reference is made in the text.

No work in this thesis has been submitted for a degree to any other university or institution.

Signed

Mohammad Zarafshan Araki
... ., 2009

Acknowledgments

This thesis would not have possible without the assistance of several people whom I wish to acknowledge. Firstly, my two supervisors, Dr. Kwan-Wu Chin and Dr. Raad Raad for their support and kindness and for giving me this opportunity. In particular, Dr. Kwan-Wu Chin, for recommending this study area and guiding me through difficult times. Secondly, my many learned and academic friends who have provided inspiration, in particular Dr. Jason Hughes from the University Archives, without whose humor and positive outlook, none of this would have been possible. Finally to Jooey for her patience, integrity, and empathy.

Contents

List of Figures

List of Tables

List of Abbreviations

AAA	Authentication, Authorization, and Accounting
ADSL	Asymmetric Digital Subscriber Line
ARQ	Automatic Repeat Request
AS	Autonomous System
BGP	Border Gateway Protocol
CATV	Cable Television
DDoS	Distributed Denial of Service
DRR	Deficit Round Robin
DSL	Digital Subscriber Line
DTN	Delay Tolerant Networking
ECN	Explicit Congestion Notification
EIGRP	Enhanced Interior Gateway Routing Protocol
FCSF	First Come First Served
GMMF	Global Max-Min Fair
GPS	Generalized Processor Sharing
GSM	Global System for Mobile Communications
HFC	Hybrid Fiber Co-axial
HIBC	Hierarchical Identity-based Cryptography
HTTP	Hypertext Transfer Protocol
I/O	Input/Output
ICT	Information and Communication Technology
IP	Internet Protocol
IPN	InterPlaNetary Internet
IS-IS	Intermediate System to Intermediate System
ISP	Internet Service Provider

LAN	Local Area Network
LMMF	Local Max-Min Fair
MAP	Mobile Access Point
MANET	Mobile Ad hoc Network
MBF	Mobile Bundle Forwarder
MIM	Multipurpose Internet Mail Extensions
MPLS	Multiprotocol Label Switching
ONU	Optical Network Units
OSPF	Open Shortest Path First
PC	Personal Computer
PCMP	Persistent Connection Management Protocol
PDA	Personal Digital Assistant
PEP	Performance Enhancing Proxy
PKG	Private Key Generator
PKI	Public Key Infrastructure
POP	Point-of-Presence
QoS	Quality of Service
RAID	Redundant Array of Independent Disks
RMI	Remote Method Invocation
RPC	Remote Procedure Call
RTP	Real-time Transport Protocol
RTT	Round-Trip Time
SCTP	Stream Control Transmission Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UML	Unified Modeling Language
UUCP	Unix-to-Unix Copy Program
VoD	Video on Demand
VSP	Video Service Provider
WCATV	Wireless Cable Television
WFQ	Weighted Fair Queuing
WF^{2}Q	Worst-case Fair Weighted Fair Queuing

WGMMF	Weighted Global Max-Min Fair
WiFi	Wireless Fidelity

This page is blank

Chapter 1

Introduction

Recently, a class of emerging networks, known as challenged networks, has been gaining attention due to their potential application in extreme operating environments. Some of which include InterPlaNetary Internet (IPN) [2], rural Internet connectivity [13, 14, 15, 16], disaster relief [17, 18], battlefield communications [19, 20], and wireless sensor monitoring [21, 22]. In these environments, challenged networks are subjected to long propagation delays, frequent partitioning, and high error rates.

Figure 1.1 depicts a challenged network where a scientist has a communication path to a weather station located on the planet Mars. We see that the path between the scientist's workstation and the weather station comprises of multiple wired and wireless links. This means any files transfer will involve a communication path linking the scientist's workstation and the antenna complex via the Internet. The files are then transferred using relay satellites orbiting Mars. After that, the files are sent to another antenna complex using a radio link, and from there a wireless LAN is used to deliver the files to the weather station [1]. In this example, the speed of light is a limiting factor that results in large propagation delays. Furthermore, communications are intermittent due to the interposition of planetary bodies between Earth and Mars. Hence, the resulting network has round-trip times (RTTs)[1] measured in hours or day; un-

---

[1] Round-trip delay time refers to the time a signal needs to make a round-trip over a closed circuit [23].

Please see print copy for image

Figure 1.1 A deep space communication scenario demonstrating a scientist sending a software module to a weather station located on the planet Mars [1].

like the Internet which has RTTs in the order of milliseconds or seconds. As a result, TCP/IP operates poorly under such large RTTs due to the reasons to be discussed in Chapter 2 [1].

Another example of challenged networks is the InterPlaNetary Internet (IPN) project proposed by NASA's Jet Propulsion Laboratory (JPL) [24]. The project aims to interconnect the Internet with other remote Internets residing in outer space; e.g., networks located on spacecrafts or other planets. The IPN, as shown in Figure 1.2, comprises both terrestrial and interplanetary links. Communications across space experience long propagation delays and intermittent connectivities. Again, the speed-of-light is a limiting factor while the interposition of planetary objects between a sender and a receiver causes frequent disruptions to communication channels. The IPN can be viewed as *a network of disconnected Internets* rather than *a network of connected networks*, as is the case with the current Internet. As a result, existing Internet protocols face a myriad of issues in IPN [24, 2].

Please see print copy for image

Figure 1.2 The InterPlaNetary Internet using a wireless backbone. [2]

Researchers have also proposed challenged networks for tracking wild life. ZebraNet [22] is a wildlife tracking project that aims to learn zebras mobility patterns within a large geographical area in Kenya. Each animal carries a custom collar equipped with a small embedded device with a wireless transceiver. Collars collect sensor data and opportunistically exchange their logged data when zebras are in close proximity of each other. These data are stored on *non-volatile flash memories* until it can be uploaded to a mobile station carried by researchers. The data are then uploaded manually to a central repository on the Internet [22].

The last example of challenge networks is DakNet [13]; a commercial project that provides low cost Internet connectivity to villages in India and Cambodia. DakNet consists of Internet kiosks in villages, hotspots in cities, and buses regularly travel between kiosks and hotspots. A bus, also known as a mobile access point (MAP), opportunistically exchanges data between hotspots and kiosks using WiFi. Instead of public busses, other forms of MAP include a person on a motorbike carrying a flash memory [13].

1.1 Motivation

This thesis is motivated in part by the fact that video traffic will dominate Internet Protocol (IP) networks in the foreseeable future. This thesis is also concerned with the availability of Internet services in rural and remote regions of Australia.

1.1.1 Exabyte Traffic

According to Cisco [3], video traffic produced by consumers is driving IP traffic growth, and hence is likely to dominate IP traffic in the future. Moreover, [25] predicts that traffic from consumers will exceed business IP traffic in 2009, meaning consumer traffic will exceed 32 exabytes per month by 2012 and almost 90 percent of this traffic will be due to video traffic. Interestingly, at the end of 2012, consumer video will be composed of (1) Internet video, including video-to-PC and video-to-TV, (2) video files over peer-to-peer file sharing networks, and (3) cable Video on Demand, which will account for 6.5, 7.5, and 13 exabytes of monthly traffic respectively. The later form of consumer video, cable VoD, refers to traditional TV programs that are offered by cable TV operators.

The growth in cable VoD traffic will result in IP traffic increasing at a higher rate in the metro as opposed to the core part of the network. Given that IP traffic is not limited to the Internet, Cisco predicts that cable VoD traffic will start using IP networks in 2009. Hence, by the end of 2012, one third of the consumer IP traffic will be due to cable VoD traffic, where this traffic will reside at Cable TV (CATV) networks operating at the metro.

Figure 1.3 illustrates the rates at which core, metro, and access traffic will grow from 2007 to 2012 [3]. We see that,

- Core traffic will nearly quintuple, growing from 5 to 28 exabytes per month.
- Metro traffic will nearly septuple, growing from 7 to 47 exabytes per

Please see print copy for image

Figure 1.3 The growth of traffic over core, metro, and access between 2007 and 2012 [3].

month.

- Access traffic will nearly septuple, growing from 8 to 53 exabytes per month.

CATV operators have several means of responding to the challenges that arise from increasing VoD traffic. Firstly, CATV operators can increase the rate in which they upgrade their networks. Secondly, they can use *content delivery methods* [26], since much of today's video content is static and cacheable. For example, video content can be delivered at the edge, meaning that video contents can be transported to CATV head ends prior to their air time. Since CATV head ends are closer to the subscribers, the distribution process will generate much less traffic. Thirdly, video contents can be delivered using one-to-many distribution techniques. For example, traffic engineering allows one-to-many data delivery across MPLS tunnels [27]. Nonetheless, due to high maintenance cost and technical difficulties in implementing such techniques [27], unicast data transmission has remained the only practical option available to date.

1.1.2 Rural Connectivity

The number of Internet users in developing countries is far below those found in developed countries. One in 500 people in developing countries has access to the Internet, whereas this ratio increases to one in four for developed countries. Consequently, in developing regions, access to information and communication technology (ICT) is mostly limited to people living in major cities and large regional centers [28].

In a similar manner, there is a significant difference between Internet services supplied to urban and non-urban areas in Australia [29]. In the case of narrowband Internet access, most rural and remote users need to make long distance calls to an ISP, while metropolitan subscribers have access to unlimited local calls, as well as choice of various ISPs [30]. In addition, telephone services provided in rural areas are typically not robust enough to sustain a long connection time needed to transport large files [29]. On the other hand, broadband access in rural and remote regions is still in short supply [31]. As in other developed nations, all types of broadband access technologies are available in major cities. However, rural and remote regions have very limited options in terms of Internet access [31]. In October 2003, Telstra stated that DSL was accessible to 74 percent of Australia's residents [31]. Yet, only 1000 exchanges were capable of providing DSL services in 2003, and the upgrade of the remaining 4000 exchanges was subject to commercial viability [31]. In contrast, satellite services are available in all parts of Australia [31] but are expensive, therefore making it inappropriate for rural and remote communities [32]. In summary, improving Internet access, especially broadband access, in Australia is an ongoing challenge, as is the case in other countries [31].

1.2 Thesis Aims

This thesis addresses two challenges:

1. Providing a low cost, very high bandwidth solution to CATV operators

in order to meet the demand of VoD traffic.

2. Providing low cost, broadband Internet access to rural regions.

The first challenge is critical given that CATV operators are required to upgrade their networks to cope with ever increasing VoD traffic. One option is for these operators to upgrade their networks with fiber cables. However, the civil cost of laying fibers is prohibitive given that video contents must be transported from multiple video service providers (VSPs) to geographically distributed head ends. Moreover, non economical viable regions are excluded from upgrades; resulting in violation of ICT regulations that require technological advances to benefit everyone equally [31]. Lastly, fiber optic links are not scalable, where they only provide fixed bandwidth that cannot be upgraded easily.

The second challenge is important to a country like Australia where the vast majority of the population is concentrated in cities. This means people living in rural regions are excluded from high bandwidth Internet access. Existing solutions to connect these regions have low bandwidth, and increasing this bandwidth in order to provide better services is not economically viable; e.g., laying fiber. Today's access options available in regional Australian violate regulations, see [29,31], that state that rural communities should be treated as urban citizens in that they must have access to a competitive broadband market at the same price as urban regions. Furthermore, broadband access is required for enabling services such as e-Health, e-Education, and e-Government. For example, having broadband access means libraries and schools in rural regions can offer virtual lectures and tutorial presentations to rural communities. Moreover, rural communities will be able to enjoy services such as downloading movies, TV programs and music. More importantly, these services can be provided cheaply.

1.3 Contributions

This thesis has two contributions. Firstly, it proposes a novel data transport system called TrainNet. TrainNet is a delay tolerant vehicular network that uses trains as a mechanical backhaul. Specifically, TrainNet augments trains and stations with mass storage devices; e.g., a rack of portable hard disks capable of storing terabytes of data.

The augmented trains are then used to transfer data between stations. As a result, TrainNet provides a low cost, very high bandwidth path that can be used to transport latency insensitive data such as non real-time video contents. Furthermore, TrainNet's links are scalable, meaning trains' storage capacity can be upgraded easily and inexpensively by adding new hard disks. In fact, the falling cost per megabyte means TrainNet is capable of offering virtually unlimited capacity at a reasonable price. Therefore, TrainNet provides a cost effective solution as compared to laying new fiber cables or using satellites to transfer video contents.

A key advantage of TrainNet is that it is capable of providing high bandwidth Internet access to rural regions, as long as these regions are connected by a railway line. Using TrainNet, rural communities will have access to broadband applications similar to those found in urban regions. For example, downloading digital contents such as software, music, books, and movies is currently very popular among Internet users. These services can easily be made available to rural areas with access to TrainNet. Users can select the movies he/she wants to watch via a dial-up link, and then have TrainNet transport the movies to his/her location. Moreover, the downloaded movies can be cached at a central repository for other users to download.

Lastly, this thesis is the first to apply max-min fair algorithms in DTNs. Specifically, it proposes three max-min scheduling algorithms that address the resource scheduling problem in TrainNet; that is, they fairly divide a train's storage capacity among competing stations. Furthermore, the proposed max-min algorithms avoid retransmission of video contents, a desirable feature when delivering large files.

1.3.1 Publication

M. Zarafshan, K-W Chin., and R. Raad (2009), "TrainNet: A Novel Data Transport Infrastructure for Delivering Latency Insensitive Data", submitted to IEEE Transactions on Vehicular Technology.

1.4 Thesis Structure

The rest of the thesis is organized as follows:

Chapter 2 reviews the literature on challenged networks. It discusses problems arising from intermittent links and compares and contrasts existing solutions. Furthermore, this chapter gives examples of vehicular networks and reviews research efforts pertaining to routing, congestion, security, and resource scheduling. It also discusses applications of max-min fairness in packet-switched networks.

Chapter 3 introduces TrainNet, and presents two example applications. Furthermore, this chapter defines TrainNet's components as used in a distributed VoD system.

Chapter 4 presents the resource scheduling problem in TrainNet. It defines the problem and proposes one preliminary and three max-min algorithms.

Chapter 5 presents a simulation framework and the model used to evaluate TrainNet performance. Furthermore, this chapter presents simulation settings used to investigate the fairness and data loss avoidance behavior of the proposed scheduling algorithms. Lastly, it includes results and discussions arising from experiments with varying traffic scenarios.

Chapter 6 concludes the thesis, and presents future research works.

Chapter 2

Literature Review

Section 2.1 to 2.4 first review problems and solutions proposed for emerging challenged networks. This also includes routing, congestion, and security issues as well as examples of vehicular based intermittent networks. Section 2.5 and 2.6 then reviews research efforts pertaining to resource scheduling in challenged networks and the use of max-min fairness in packet-switched networks, respectively. Finally, Section 2.7 concludes the chapter.

2.1 Challenged Networks

Unlike conventional networks, challenged networks operate in environments with extreme characteristics, such as frequent connection disruption, long propagation delay, and/or intermittent connectivity. Examples of challenged networks include those found in deep space networks, wireless sensor networks, mobile ad-hoc networks, satellite networks, military networks, underwater networks, and disaster relief networks [11].

Table 2.1 summarizes key characteristics that make challenged networks different from conventional networks. These characteristics are given according to two examples: the deep space network described for Figure 1.1 and the public broadcast network [4] depicted in Figure 2.1. Public broadcast networks allow users to openly send and receive contents such as music, videos, and games with

Please see print copy for image

Table 2.1 A comparison of challenged and conventional networks [1, 11, 12].

Please see print copy for image

Figure 2.1 A public broadcast network comprising of mobile wireless device [4].

each other. As shown in Figure 2.1, mobile nodes are pedestrians carrying a wireless device such as PDAs, cell phones, notebooks, or game consoles. These mobile nodes communicate in an ad hoc manner without relying on any fixed infrastructure. A key characteristic of such networks is the frequency in which nodes move in and out of range of each other [4].

2.1.1 Problems and Issues

The Internet protocol suite, also known as TCP/IP [33], has been successfully adapted to diverse networking environments in the past two decades. The successes of TCP/IP are due to its ability to interconnect heterogeneous networks, where it provides a common communication platform to devices with different data-link and physical layer technologies.

Recently, TCP/IP is experiencing difficulties in coping with the characteristics of challenged networks [34, 1, 11]. These difficulties arise due to three fundamental assumptions of TCP/IP: (1) contemporaneous end-to-end links between senders and receivers, and that links have (2) relatively short round-trip times and (3) low packet errors [11]. Unfortunately, these assumptions are violated in challenged networks because links have large propagation delays and have intermittent connectivities. As a result, TCP/IP protocols have poor performance over such links [1].

The key issues facing TCP in challenged networks are as follows [35,1,5,36,37]:

- A TCP session starts with a three way handshake that consumes 1.5 RTTs before any application data can be exchanged. Hence, no application data can be exchanged if a challenged network has communication opportunities[1] that last shorter than 1.5 RTTs.
- Consider a sender A using TCP to send packets $1,\ldots,i,\ldots,N$ to a receiver B. At receiver B, TCP will deliver these packets in order to the application. This means if all packets except packet i is received successfully, TCP

---

[1] In the context of challenged networks, communication opportunity refers to the duration of time that a pair of nodes has end-to-end connectivity [38, 39].

stops sending packets $i+1, \ldots, N$ to the application layer until packet i is retransmitted. In a challenged network, retransmission of data over the same connection may not be possible. In such networks, TCP will discard packets $i+1$ to N.

- TCP has a generic two minute timeout where a connection is terminated if no data is sent or received after the timeout value. Thus, any communication scenario requiring a longer timeout, such as those in challenged networks, is unable to use TCP, unless the timeout value is extended.

- The throughput of TCP degrades as the round-trip latency increases, especially in challenged networks with their large RTT. This is due to TCP's congestion handling algorithm. When TCP is used to transfer bulky data, it periodically switches between two phases: *slow-start* and *congestion avoidance*. As RTT increases, TCP remains in slow-start longer and the congestion window grows at a slower rate; both of which reduces throughput.

- TCP performs retransmissions on an end-to-end basis. This is a problem for data sources that has energy and storage constraints; e.g., an outer-space robot collecting imaging sensor data. Consider three nodes A, B, and C, where A and C has a connection that spans links A–B and B–C, each of which has a signal propagation latency of 250 milliseconds. Using TCP, A can only free the buffer space allocated to a packet after one second; that is, when it receives an acknowledgment from C. However, if retransmissions were performed hop-by-hop, A could release its buffer after 500 milliseconds; that is, when it receives an acknowledgment from B.

Similar issues also exist for UDP based protocols, such as RPC [40], RMI [41], and RTP [42]. These protocols utilize end-to-end ARQ mechanisms to guarantee reliable data delivery. However, these mechanisms are tailored to operate over conventional links. Hence, in challenged networks, they are subject to limitations similar to those experienced by TCP [1, 11].

Existing Internet routing protocols are not suitable for challenged networks. The Internet employs *a hierarchical routing system* [33]. For example, the Border Gateway Protocol (BGP) [43] is used to forward packets between autonomous systems (AS). Within an AS, routers use an interior routing protocol such as OSPF [44], IS-IS [45], and EIGRP [46]. When these routing protocols are run over challenged networks, they experience the following problems. First, BGP relies on TCP to exchange routing information. Hence, in challenged networks, the performance of BGP is limited by the shortcomings of TCP. Secondly, inter-AS routing protocols are designed to find paths in changing topologies where nodes have direct or indirect connectivity all the time. To monitor connectivity, routers using these protocols are required to send HELLO messages or their routing table periodically or whenever there is a change, such as a broken link. If a node loses connectivity to other nodes, it will be excluded from routing decisions. As shown in Table 2.1, nodes in challenged networks frequently loose their connectivity. Hence, inter-AS routing protocols are ill suited to challenge networks [34, 1, 11].

2.2 Solutions

To date, four classes of approaches have been used to address the problem of intermittent connectivity in challenged networks. Firstly, *link repair approaches* [47,48,49] use performance enhancing proxies (PEPs) [50]. In these approaches, intermittent links are routed through PEPs such that they appear similar to ordinary physical links.

The mechanisms employed by PEPs can be categorized into four classes. Firstly, PEPs can acknowledge the data received from an end point, thereby fooling the end point into believing that data has arrived at its destination without delay. Secondly, PEPs can use encapsulation to carry packets across a tunnel established over an intermittent link. Thirdly, PEPs can compress packets, thereby reducing the number of bytes crossing an intermittent link. Fourthly, PEPs can send fake acknowledgements to an endpoint, thereby preventing the termination of TCP connection during link disconnections. The usability of these mechanisms depends on the characteristic of intermittent links. Unfortunately, the disadvantage of PEPs is that links have different characteristics in different operating environments. Hence, they cannot be reused in different challenged networks [11,51].

Another class of solutions, e.g., [22] and [13], use *persistent storage* to overcome intermittent connectivity. Exploiting storage enables mobile nodes to combat network disruptions, where data is stored on storage media for an extended period of time until connectivity is restored. Furthermore, mobile nodes can save their state in case of a power outage. As we shall see in Section 2.3, this approach has been used widely in vehicular networks to combat intermittent connectivity.

[52,53,54] have used a *rendezvous point* to overcome the intermittent connectivity problem. These works consider using a central entity that is used by mobile nodes as a rendezvous point for all communications. This means mobile nodes are required to establish direct or indirect connections to exchange messages with each other. As we shall see later on, this is a common approach used in the vehicular networks; see Section 2.3.1, 2.3.2, and 2.3.3.

The authors of [55], [56] and [57] consider altering the movement of nodes to address intermittent connectivity. For example, in case of a network partition, a mobile node is forced to carry messages between disjoint parts of the network. Such a solution is not possible in challenged networks where nodes have fixed movement patterns. As we shall see in Chapter 3, TrainNet exploits trains that travel according to a predefined time-table. Hence, approaches that alter the movement of nodes such as trains are beyond the scope of this thesis.

Lastly, the problem of intermittent connectivity can be addressed by means of new communication paradigms [11,58,7]; the topic of the next section.

2.2.1 General Purpose Communication Paradigms

To date, there are three general purpose architectures: Delay Tolerant Networking [11], DTN-over-HTTP [58], and Parallel Networks (ParaNets) [7].

2.2.1.1 DTN

DTN is concerned with interconnecting radically heterogeneous challenged networks suffering from frequent connection disruption, long propagation delay, and/or intermittent connectivity. In other words, DTN addresses the issues arising in challenged networks and provide interoperability between challenged networks.

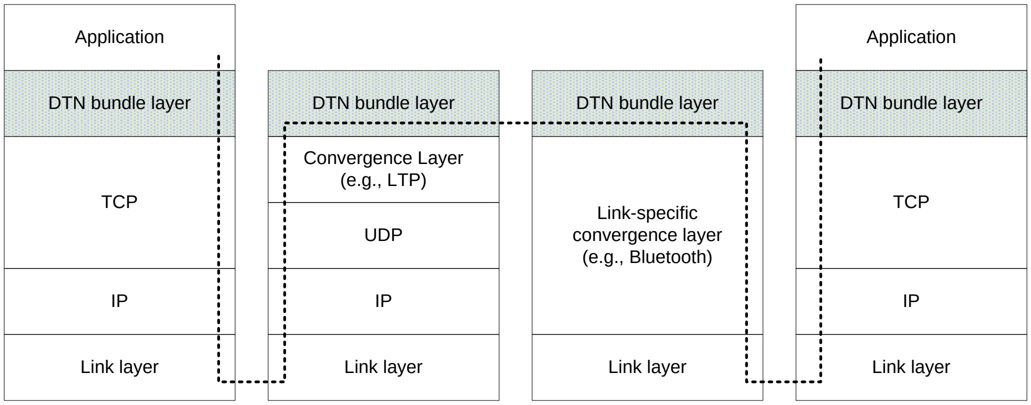

Figure 2.2 The overlay network approach where the DTN bundle protocol runs over different transport and lower layer protocols [5,6]. The dotted line shows the path in which applications exchange data.

DTN defines an architecture comprising of an end-to-end message-based overlay. Figure 2.2 illustrates an additional network layer, known as the *bundle layer* that operates above the transport layer and below the application layer. The bundle layer provides store-carry-and-forward service using in-network persistent storage. Devices implementing the bundle layer are called DTN nodes. The bundle layer is designed to interconnect radically heterogeneous networks. For this purpose, the architecture defines convergence layer adapters, which provides the necessary functions to carry DTN messages (i.e., *bundles*) over heterogeneous transport protocols. A convergence layer adaptor is responsible for reliably transmitting bundles and to handle connection establishment and termination over a given network. The bundle protocol also employs a flexible naming scheme based on Uniform Resource Identifiers (URI) [59]. This scheme supports a diverse range of naming and addressing syntax. In addition, the bundle layer also provides a number of management and diagnostic features to support end-to-end acknowledgment, hop-by-hop reliable delivery, and security.

To date, DTN has been widely used as the common platform for challenged networks. Example includes but not limited to those found in military networks [19,20], disaster relief networks [17,18], sensor networks [21], outer space networks [6].

2.2.1.2 DTN-over-HTTP

DTN-over-HTTP defines an overlay architecture similar to DTN. However, DTN-over-HTTP is different in that its uses a modified version of HTTP, known as HTTP-DTN, as an overlay layer that operates at the application layer. HTTP-DTN has several benefits over bundle layer. Firstly, it provides extra features such as end-to-end reliability and error detection. Secondly, when using TCP is not feasible, it can be run over other message based transport protocols such as Saratoga [6] and Stream Control Transmission Protocol (SCTP) [60]. Thirdly, it is capable of transferring contents identified by Multipurpose Internet Mail Extensions (MIME). As suggested in [58], alternative overlay architectures can be realized using Unix-to-Unix Copy Program (UUCP) [61], Netnews [62], and Fidonet [63].

2.2.1.3 ParaNets

ParaNets defines an architecture that allows network protocols to simultaneously run over heterogeneous protocols. For example, consider two DTN nodes A and B communicating using DTN networks X and Y. Suppose, network X has an end-to-end connectivity over a low bandwidth link and network Y has an intermittent connectivity over a high bandwidth link. Using ParaNets, networks X and Y can be used as a communication channel depending on the message type. For example, a transport protocol can first use network X to transmit control messages when establishing connections and use network Y to send bulky data.

Please see print copy for image

Figure 2.3 Example of a tree-like protocol stack [7].

As depicted in Figure 2.3, ParaNets extends the vertical structure of a protocol stack to a more flexible tree-like structure. This structure includes classic layers similar to those found in conventional protocol stack. However, each layer is allowed to interface with any of its underlying layers. As shown in Figure 2.3, transport protocol A is connected to network protocol A and data link protocol C simultaneously. Each time a session is established, a virtual stack that best suits the session is built by selecting a path from an application to a physical layer.

ParaNets is different from DTN and DTN-over-HTTP in that it assumes the availability of parallel networks between two DTN end-points. As suggested in [7], this assumption is reasonable given that challenged networks will increase in the foreseeable future. Furthermore, DTN has undergone a large amount of developments. However, in ParaNets, issues including transport, routing, addressing, security, and administrative remain an open issue.

2.3 Vehicular Networks

The solutions discussed in Section 2.2 have been applied to a wide range of challenged networks, in particular vehicle-based networks. The following sections describe networks that use rendezvous points, and persistent storage to combat intermittent connectivity. The networks in Section 2.3.4 and 2.3.5 are DTN-enabled, in that they use an instance of the DTN protocol stack [64].

2.3.1 Drive-thru Internet

Drive-thru Internet [52] is a distributed MANET that provides car passengers with intermittent connectivity to WiFi hot spots located along a road. This is an alternative cost-effective solution to GSM/3G-based networks. In Drive-thru, cars use WiFi to connect to hot spots, where the main goal is to provide connectivity between cars and a Drive-thru proxy hosted on the Internet. This proxy acts as the rendezvous point. The proxy and cars communicate using an application layer protocol called Persistent Connection Management Protocol (PCMP). This protocol is responsible for re-establishing connectivity to the proxy. In Drive-thru, the proxy relays application and transport layer data units on behalf of cars, and thereby provides Internet connectivity to cars that are reachable via a hot spot. Drive-thru has been reported to have an averaged goodput of more than 100 megabytes at each hotspot.

2.3.2 Taxi Radio Dispatch System

The distributed MANET proposed in [53] is a radio dispatch system for taxis. A dispatch unit is utilized heavily by a radio taxi company to allocate free taxis to passengers and to inform taxi drivers to pick up passengers. The system includes a centralized dispatcher located at the taxi company's headquarter, which acts as the rendezvous point. Using WiFi, taxis regularly report their status (i.e., occupied or free) and traffic conditions to the dispatcher. When the centralized dispatch wants to notify a driver, it replies with a pickup request. Although routing is considered beyond the scope of [53], the authors suggest the use of location-aided routing (LAR) [65] to route messages.

2.3.3 DieselNet

DieselNet [54] is a bus-based test-bed in which WiFi transceivers are deployed on 40 buses moving around the area surrounding the University of Massachusetts at Amherst campus. Each bus continuously scans its surrounding area using a WiFi access point (AP). When a bus, say A, encounters another bus B, bus A obtains an IP address from bus B and establishes a TCP connection. Afterwards, bus A continuously transfers data to bus B until the TCP connection is broken. Next, bus B uses GPS to determine its position, and makes a profile of its TCP connection with bus A using the returned position, bus A's ID, current time, the duration of the connection, and the amount of data transferred. Bus B then stores this profile on its persistent storage and transmits this profile to a central repository when it has Internet access.

In DieselNet, throwboxes [66] are also used to facilitate communications. These throwboxes are stationary, battery powered nodes equipped with WiFi and storage. When a bus passes by a throwbox, the throwbox acts as a rendezvous point where it receives data from the bus, and delivers the data when the destination bus passes by.

2.3.4 CarTel

CarTel [67] is a mobile distributed sensor network that collects and visualizes various sensor information; e.g., road traffic and Wi-Fi hotspot monitoring. In CarTel, cars are mobile DTN nodes that collect data and *opportunistically* upload it to a central portal using WiFi. The portal hosts an Intermittently Connected Continuous Query Database System (ICEDB), where all sensor data is stored. CarTel applications query ICEDB to retrieve sensor data for further analysis and visualization. The portal and cars communicate using an implementation of the DTN protocol stack called CafNet [68]. This allows cars to serve as *data mules* and hold data until Internet connectivity becomes available. In CarTel, the connection duration between cars and access points is very short; about 75 seconds. Hence, to maximize throughput during this short time, cars use Cabernet [69] to establish *fast WiFi connectivity* within 400 milliseconds.

2.3.5 KioskNet

KioskNet [16] is a bus-and-kiosk network that provides low cost Internet connectivity to rural villages in developing countries. In KioskNet, DTN nodes are deployed on buses traveling regularly between Internet kiosks located at villages to download/upload data to be deposited or retrieved from Internet gateways located at cities. Buses run according to a schedule and opportunistically exchange data between kiosks and gateways using WiFi. Each bus is capable of holding 40 gigabytes of data. Gateways are connected to the Internet via dial-up or ADSL links, through which they communicate with a proxy that relays data between the gateways and the Internet. The bandwidth of such a link is reported to be about 100 Kbps, the equivalent of 1 gigabyte per day.

2.4 Routing, Congestion, and Security

Since the advent of challenged networks, routing has remained an active research area and has received considerable attention from the community. However, security and congestion have received relatively little attention.

2.4.1 Routing

To date, several delay-tolerant routing schemes have been proposed; each of which is applicable to different types of challenged network. These routing techniques can be categorized broadly into two classes: *deterministic* and *stochastic*. Deterministic methods operate in networks where future topologies are predictable or known in advance. In addition, they aim to achieve a *particular objective*, for example, earliest delivery time or minimum hop count. On the other hand, stochastic methods deal with network topologies that randomly evolve through time. They aim to move messages closer to their respective destination one hop at a time [70].

The proposed approaches for deterministic routing include tree, modified shortest path, and space time [70]. In the tree approach [71], all nodes have a global knowledge of topology changes over time. When a node A wants to select a path to a node B, a tree is first built starting from node A, branching to intermediate nodes between A and B, and finishing at node B. Each branch of this tree includes the times during which each intermittent node encounters the next hop node. A final path is then calculated by choosing the earliest time that connects node A to node B. This approach also considers a second case where nodes have no knowledge of topology changes. Furthermore, in a third case, routing histories are recorded in messages, thereby allowing nodes to learn the network topology over time.

In the modified shortest path approach [38], routing is performed using a *knowledge oracle*. Four oracles are proposed in [38]; each of which corresponds to a different level of information corresponding to different network dynamics. Using these oracles, a modified time-sensitive Dijkstra is used to calculate the best path with the shortest waiting time. Space time approach [72] assumes that topology changes can be predicted accurately. Based on this assumption, a *space time-graph* is created, which is then converted to a static graph. The Floyd-Warshall algorithm [73] is then applied on the static graph to calculate the shortest path between a source and destination pair.

The approaches proposed for stochastic routing can be categorized into four classes: epidemic spray approach [74], predication-based approach [22], model-based approach [75], and control movement approach [55]. In epidemic spray, messages are flooded to nodes. This means, when a node receives a message, it forwards the message to all or a subset of its neighbors. Unfortunately, flooding results in duplicated messages. Given that nodes have limited buffer size, the overall amount of duplicated messages must be controlled. Algorithms proposed to limit duplicated messages include those found in [76, 77, 78, 79, 80].

In predication-based approaches, messages are not blindly flooded to neighbors. Instead, a node first estimates the chance in which a message may reach its destination if the message is forwarded via a given neighbor. It then decides to forward the message or wait for a better chance. Algorithms that make predications using next hop information include those found in [81, 82, 83, 84] and algorithms that make predications using end-to-end metrics include those found in [81, 85, 86].

In model-based approaches, mobile nodes' movements are not random, as is the case in predication-based approaches. Instead, mobile nodes' movement patterns are exploited to select low cost paths without flooding the network. The algorithm proposed in [4] exploits pedestrians' motion patterns and the algorithms in [87] and [88] utilize vehicle mobility patterns. Unlike model-based approach, control movement algorithms are active, meaning that they alter the movement of mobile nodes [56,57,89]. For example, when routing is not possible due to a network partition, a mobile node is forced to carry messages between disjoint parts of the network.

Routing in challenged networks is still an on-going research area. Open issues requiring further investigations are as follows [70, 39]:

- Deterministic routing approaches use objectives such as earliest delivery time or minimum hop count. These objectives result in achieving different network performances such as low delay or high throughput. Investigation into the relationship between routing objectives and network performance is an open issue.

- Stochastic routing algorithms create multiple copies of a message in order to increase the probability of delivery. However, these duplicated messages consume precious resources. Therefore, designing techniques that balance high delivery probability and resource consumption is an open issue.

- In predication-based algorithms, simplified and accurate estimates must be developed. For example, estimates can be calculated according to mobile nodes' movements. In such a scheme, a message is passed from node A to node B when the mobility pattern of node B guarantees a high delivery probability.

- Multicast routing protocols must be developed for challenged networks. Such protocols must operate over tree rather than mesh topologies to support custody transfer; see Section 2.4.2. Furthermore, multicast protocols need security mechanisms which are more complex than those proposed in existing unicast protocols.

2.4.2 Custody and Congestion

Custody transfer is an optional service for reliable delivery of bundles. In a DTN network, intermediate nodes along a path can accept custody of a bundle message. These nodes are known as *custodians*. When a DTN node accepts the custody of a bundle message, it stores that message on a persistent storage until another custodian acknowledges that it has custody of the message [90].

A node facing continuous demands for custody transfer will run out of storage resources and become congested. A congested node has a few means of responding to network congestion. Firstly, it may discard expired bundles or forward bundles to other nodes with free storage space. Secondly, it may cease accepting regular bundles or bundles requiring custody transfer. Finally, the node may discard unexpired bundles or bundles for which it has custody of them [90, 91].

Storage routing (SR) [91] is a stochastic routing scheme that resolves congestion by moving bundles to nodes with free storage space. Using SR, an intermediate node along a path shares its storage resources with its adjacent nodes. Consider DTN node A and E sending a message m via three intermediate nodes B, C, and D. Suppose moving message m from node B to node C results in storage exhaustion on node C. Recall that stochastic routing moves messages closer to their destinations one hop at a time. However, to avoid congestion, SR allows node C to move message m to some node F further from node D and E.

Another approach proposed in [92] is to adopt an economic model that enables

DTN nodes to autonomously decide whether to accept or reject a message. In this approach, each message carries some credit that is used to 'buy' storage. To date, methods proposed in [91] and [92] are the only schemes for controlling congestion in DTNs [39]. Nonetheless, these methods are not useful for controlling congestion in deterministic networks. Hence, designing techniques that can handle congestion in deterministic networks is an open issue.

2.4.3 Security

To date, two approaches have been taken to address security in challenged networks. Firstly, Hierarchical Identity-based Cryptography (HIBC) [93] is used to address issues such as secure channels, mutual authentication, and key revocation [94]. Unlike PKI, public keys in HIBC are chosen freely and private keys are generated by trusted authorities called Private Key Generators. PKGs build a hierarchy of trust where at least one PKG is trusted by all PKGs. Secondly, [95] defines two security mechanisms for the bundle layer: (1) data integrity and (2) data confidentiality. A DTN node is cable of encrypting, decrypting, signing and verifying bundles. These mechanisms can be performed on an end-to-end basis or on a hop-by-hop basis [39].

The aforesaid approaches are time based in that they perform security algorithms using a synchronized clock. Although such a synchronized clock is provided in the bundle layer, it has been shown that achieving accurate timing is not feasible [58]. Hence, proposing non time based techniques is an open issue. Furthermore, bundle security protocol lacks proper mechanisms for key management as well as authentication, authorization, and accounting (AAA) management [39].

2.5 Resource Allocation and Scheduling

Resource limitation is a key problem in challenged networks. Specifically, resource constraints imposed on storage and bandwidth [96]. To date, two works have focused on resource scheduling in DTN networks.

2.5.1 RAPID

RAPID [96] is a routing protocol recently proposed for DieselNet. This protocol is a variant of stochastic routing that models routing as a utility-driven resource allocation problem. RAPID includes three components: an *interface algorithm*, a *selection algorithm*, and a *control channel*.

The *Interface algorithm* estimates the utility of packets using a routing objective. Example objectives include (1) average delay, (2) deadlines, and (3) minimizing maximum delay. Objective (1) gives the highest priority to a packet that has the minimum expected delay among other packets. Using objective (2), the priority of a packet is calculated as the probability that the packet will be delivered within its time life. Objective (3) allocates the highest priority to a packet that has the maximum expected delay.

Given two nodes A and B, the *selection algorithm* replicates packets from node A to node B with respect to packets' utilities. The packet with the highest utility is first replicated, the packet with the second highest utility is then replicated, and so forth until the communication opportunity between A and B ends. Furthermore, in case of storage congestion, the selection algorithm discards the packet with the lowest utility.

The use of *control channel* is motivated by knowledge oracles described in Section 2.4.1. That is, it is used to acquire global knowledge, where nodes exchange metadata over the control channel. A node then uses this metadata to estimate the expected delay of packets. Furthermore, replicas of a delivered packet are eliminated by propagating acknowledgments over the control channel.

2.5.2 Scheduling in KioskNet

A key problem in KioskNet is to balance the aggregate load experienced by gateways. This problem manifests itself in the following traffic paths: (1) kiosk-to-gateway-to-proxy and (2) proxy-to-gateway-to-kiosk. As an example of (1), consider a scenario in which a bus dumps 40 gigabytes of data onto a single gateway, resulting in a queuing delay that is at least 40 days long. As a result, the link from the gateway to the relay proxy becomes busy for a long period of time. As an example of (2), consider a scenario in which the relay proxy consecutively choose a single gateway to deliver data to several kiosks. This causes the link from the proxy to that gateway becomes busy for a long period of time. Note that the links in the given examples are different from each other.

The solution to problem (1) is simple [97]. First, messages are flooded from each kiosk to multiple gateways, and then, a hand-shake mechanism at the proxy prevents gateways from uploading messages that have been downloaded by the proxy. The solution for problem (2) is more complex since the proxy cannot flood the gateways. For this problem, the bus-schedule aware policy proposed in [97] aims to minimize proxy-to-kiosk delay while simultaneously provides some level of fairness to the kiosks. To do this, the algorithm first assigns priorities to kiosks where the highest priority is given to the kiosk serviced by the closest bus to a gateway; the second highest priority is given to the kiosk serviced by the second closest bus to a gateway, and so forth. It then serves the kiosks based on their priorities.

KioskNet is the closest work to ours in that it considers fairness. However, the notion of fairness defined in KioskNet is not max-min. The above solutions achieve fairness during a long period of time; e.g. a day. This is reasonable given that bundles have to wait for their buses if they arrive at the gateways early [98].

2.6 Max-Min Fairness

Max-min fairness is the notion of fairness widely accepted within packet-switched networks [99, 100, 101, 102]. Max-min fair schedulers, when used to allocate the bandwidth of a link to multiple flows, protect each flow from *misbehaving* flows. Furthermore, they can lead to *efficient congestion control* and *better quality of service* (QoS) such that fair throughput and delay are guaranteed [99, 103, 104]. As we shall see in Chapter 4, this thesis is the first to apply max-min fairness in the context of DTN networks.

Briefly, a max-min fair algorithm divides a scarce resource among a group of participants. The participants are considered to have equal rights to the resource but their demands are intrinsically different from each other. The algorithm first satisfies the participant who has the *smallest* demand, and then evenly distributes unused share of the resource among other participants. After that, the participant who has the second smallest demand will be satisfied, and so forth, until no share of the resource is left unallocated. Upon termination of the algorithm, no participant will receive a share larger than his/her demand, and unsatisfied participants have an equal share of the scarce resource [102, 101].

Generalized Processor Sharing (GPS) [102, 101, 104] is the first algorithm proposed to achieve QoS in integrated services networks. GPS cannot be implemented in practice because it assumes traffic is divisible infinitely, and that a single link can be used to simultaneously transmit different flows [99]. Hence, many packet based algorithms have been proposed to emulate GPS.

Weighted fair queuing (WFQ) [105] and Worst-case fair weighted fair queuing (WF^2Q) [106] are among the fist packet-based versions of GPS. These algorithms are computationally expensive [99]. Deficit Round Robin (DRR) [107] is another packet-based version of GPS that is less complex than WFQ and WF^2Q. However, DRR approximates GPS less accurately than WFQ and WF^2Q.

MSF^2Q [103] is a multi-server version of GPS that extends the notion of max-min to an aggregate of resources. MSF^2Q is useful when similar resources can be aggregated to form a single logical resource; e.g., Ethernet links or I/O paths [103]. For example, MSF^2Q can be used to manage Ethernet links where a server is connected to a switch via multiple network cards. Furthermore, it can be used to manage I/O paths where a storage host is attached to a RAID server via multiple I/O channels.

2.6.1 Max-Min Fair Applications

Max-min fairness has been applied to different networking areas. It has found application in the allocation of bandwidth in wireless ad-hoc networks [108] and sensor networks [109], where resource allocation constraints are different from wired networks. For example, in wired networks, only nodes sharing a link compete for link bandwidth. On the other hand, in wireless networks, transmission of a node reaches all nodes in its transmission range, and therefore nodes compete with each other even when they do not send data over a single link. The method presented in [110] achieves max-min by assigning weight to flows according to the congestive state of a neighborhood. The idea is to first satisfy the most congested flow, and then the second most congested flow, and so forth, until no bandwidth is left. Apart from congestion, other resource constraints such as energy [111,112], buffer [113,114], and processing power [115] have been considered as weights.

Secondly, network routing and load balancing are fused together to optimize network utilization and throughput. The method proposed in [116] performs routing over a collection of transmission routes where more than one path exists between a pair of nodes. To maximize throughput, an algorithm is needed to first select the path spanning the link with the lowest capacity, then the path spanning the link with the second lowest capacity, and so forth. The authors of [116] showed that finding such an algorithm is an NP-complete problem. Hence, the problem is attacked using an approximate algorithm running in polynomial time.

Thirdly, max-min fairness has been used to prevent distributed denial of service (DDoS) attacks [117,118,119]. The method presented in [118] isolates legitimate traffic from an attacker's traffic to protect the availability of web services. To do this, max-min fairness is used to allocate the capacity of a web server among routers connected to the server. In this scheme, a leaky bucket rate controllers, also known as a router throttle, is deployed at each router that proactively regulates the rates at which routers forward packets to the server. Hence, if a

DDoS attack occurs, the web server is still able to provide services to a large percentage of clients.

2.7 Conclusion

The fundamental problem in challenged networks is intermittent links, which cause large communication delays. To this end, many approaches have been developed to ensure hosts are able to communicate in the absence of a contemporaneous path. For example, DieselNet utilizes the combination of persistent storage and rendezvous points to achieve higher delivery ratio than those networks using only persistent storage or rendezvous points. However, DieselNet does not take advantage of buses' fixed schedule. On the other hand, KioskNet takes advantage of bus time tables to balance the load experienced by gateways.

To date, trains have not been used as a backhaul in challenged networks. Trains have a number of advantages over cars and buses. Specifically, trains movements are deterministic, meaning that they travel over fixed railway paths according to an accurate time table. Trains also cover longer distances than buses and cars. Furthermore, trains can accommodate mass storage devices such as a rack of portable hard-disks. Moreover, a train can carry far more hard disks, and hence have much higher network capacity, than a car or a bus. This means a train is capable of transporting bulky data over long distances. The implication here is that large video files can be transported over a DTN that uses trains. Moreover, much of the today's video contents are static, meaning that they are not sensitive to delay. Lastly, given that storage is available at very low cost, the storage capacity of trains can be increased cheaply.

As mentioned in Section 2.6, max-min fairness is used widely in packet switched networks. However, in challenged networks, fairness has received no attention. To this end, this thesis is the first that proposes a max-min scheme for a DTN network that uses trains as the backhaul. The proposed DTN based max-min schemes avoid retransmission of video contents; a desirable feature because video files consume excessive bandwidth. Furthermore, the proposed scheme provides a deterministic delay bound.

In the next chapter, the thesis describes TrainNet, a DTN network that uses trains to carry latency insensitive video data. TrainNet can also be used to provide store-and-forward Internet access to rural regions reachable via a railway network.

Chapter 3

TrainNet

Section 3.1 first gives an overview of TrainNet. After that, Section 3.2 outlines two example applications of TrainNet. Section 3.3 then defines TrainNet components as used to augment a large-scale CATV network. Finally, Section 3.4 summarizes and concludes the chapter.

3.1 System Overview

TrainNet is a delay-tolerant data transport system that augments an existing railway network with the ability to carry non real-time data. In TrainNet, trains and stations are equipped with high-capacity hard-disks to store, carry, and forward data across a railway network. The resulting network is then capable of offering a low-cost, very high-capacity link to network operators looking to expand their network capacity. Hence, TrainNet is a low cost solution to network operators seeking to expand their capacity without having to lay expensive fiber cables. Moreover, TrainNet allows these networks operators to provide broadband services to remote areas that are reachable via a rail network.

TrainNet is well suited for delivering delay-tolerant audio/video contents. For example, a video service provider (VSP) can exploit TrainNet to transport video files to the head ends of a cable network. A network operator can choose to utilize TrainNet by establishing new point-of-presences (POPs)[1] at railway stations. Alternately, the network operator can connect its existing POPs to train stations via fiber-optic cables. These POPs then allow network operators to send and receive data to and from train stations, and exploit TrainNet's very high bandwidth link.

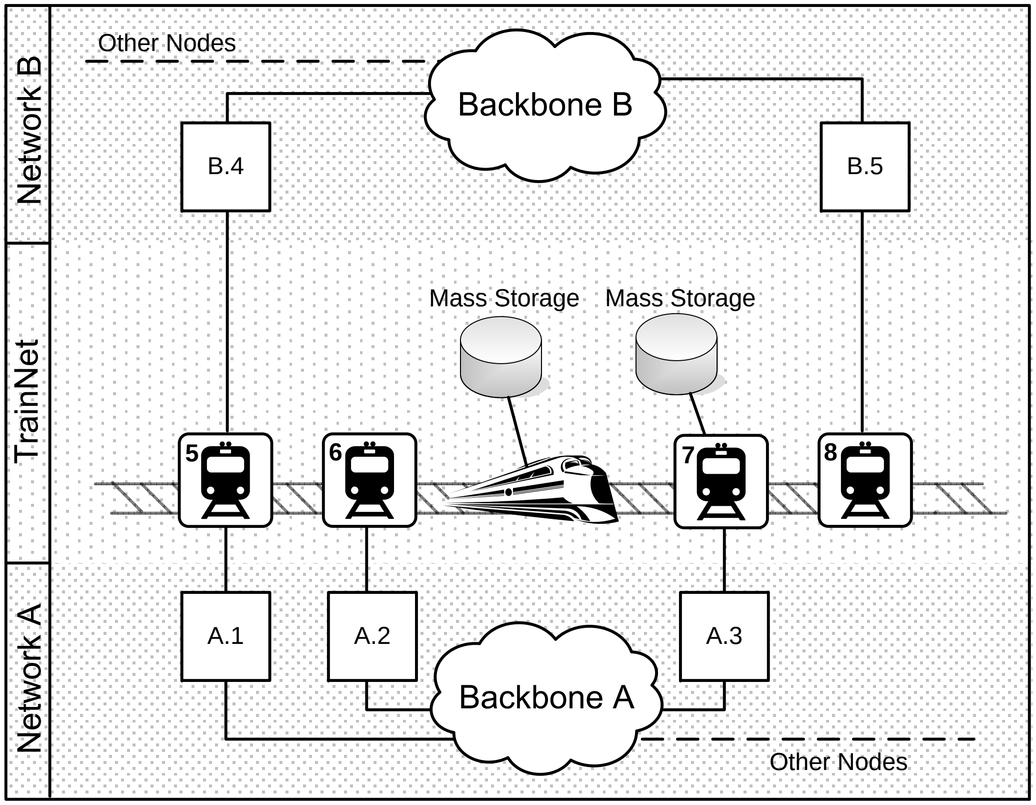

Figure 3.1 TrainNet being used to provide a low cost, very high capacity link to network A and B. POPs are represented by A.1, A.2, A.3, B.4, and B.5.

Figure 3.1 shows TrainNet being used to provide a secondary high-capacity path for two networks: A and B. These networks are connected to station 5 to 8 via POP A.1 to A.3 and B.4 to B.5 respectively. The operators of network A and B can utilize TrainNet by sending data to each station via these POPs. For example, an operator sending three terabytes of data from station 5 to 8 that is 40 minutes away by train results in a communication link that exceeds 10

---

[1] A POP is a physical location where a network operator company houses a collection of servers, routers, and switches [120].

Gbps! Furthermore, a TrainNet operator can increase this link capacity easily and cheaply by adding new hard disks.

3.2 Example Applications of TrainNet

This section gives two example applications of TrainNet. In the first example, a distributed VoD system is deployed over a large-scale CATV network. In the second example, TrainNet enables a wireless CATV (WCATV) network to supply VoD and broadband Internet access to rural subscribers.

3.2.1 Overview of CATV Networks

Cable television (CATV) networks were originally designed to broadcast television and radio programs to residential communities. With the advent of Hybrid Fiber Co-axial (HFC) technology [9], many CATV operators have upgraded their networks to deliver a variety of digital TV programs originating from different video service providers.

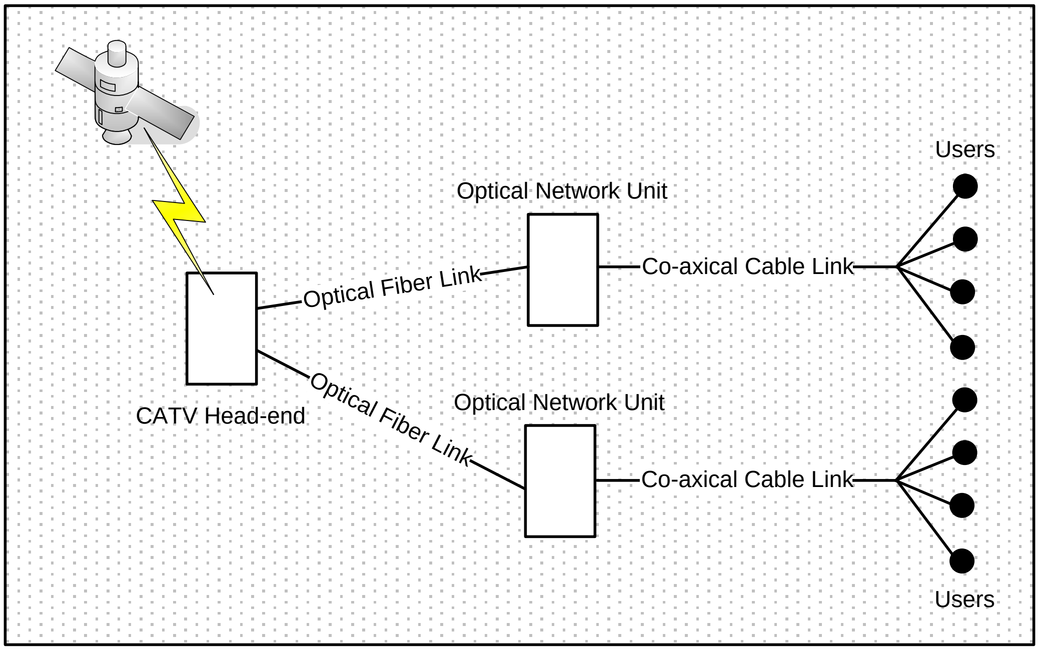

Figure 3.2 A typical CATV head-end based on HFC access technology [8, 9].

Figure 3.2 depicts a CATV head end that is based on HFC access technology. A CATV network includes one or more number of head ends depending on its size and coverage. Here, the CATV head-end receives TV signals through a satellite downlink, and then uses optical fibers to transmit the signals to optical network units (ONU). Each ONU then uses co-axial cable links to transmit signals to the cable modems in subscribers' premises. As Figure 1.3 illustrates, the HFC network utilizes a tree like topology, which allows traffic to flow in one direction from the head end to the user. HFC access technology benefits from high bandwidth links because this allows cable television operators to provide additional broadband services such as VoD, telephone and Internet services.

The key issues facing a cable TV operator in terms of offering the aforementioned broadband services are as follows. Firstly, the HFC networks must be upgraded to handle bidirectional information flows. Secondly, new high speed cable modems are needed to simultaneously handle data, voice, and video. Thirdly, high speed broadband links are needed to transport data from service providers to CATV head-ends [121, 9, 8].

A CATV operator is required to employ many broadband links to connect service providers to geographically distributed individual head ends. These service providers range from telephony service providers, Internet service providers, and video service providers. Fiber optic links is one broadband solution that is available to CATV operators. However, the civil cost of laying fibers is huge considering the distance between service providers and head ends.

This issue can be easily addressed using TrainNet. This is because TrainNet's high-capacity links are ideal for delivering latency insensitive video contents from VSPs to CATV head ends, thereby allowing cable TV operators to easily upgrade their networks to carry VoD services to subscribers.

3.2.2 Distributed VoD System using TrainNet

Figure 3.3 depicts a CATV-based distributed VoD system that uses TrainNet. This system enables geographically distributed users to have on-demand access

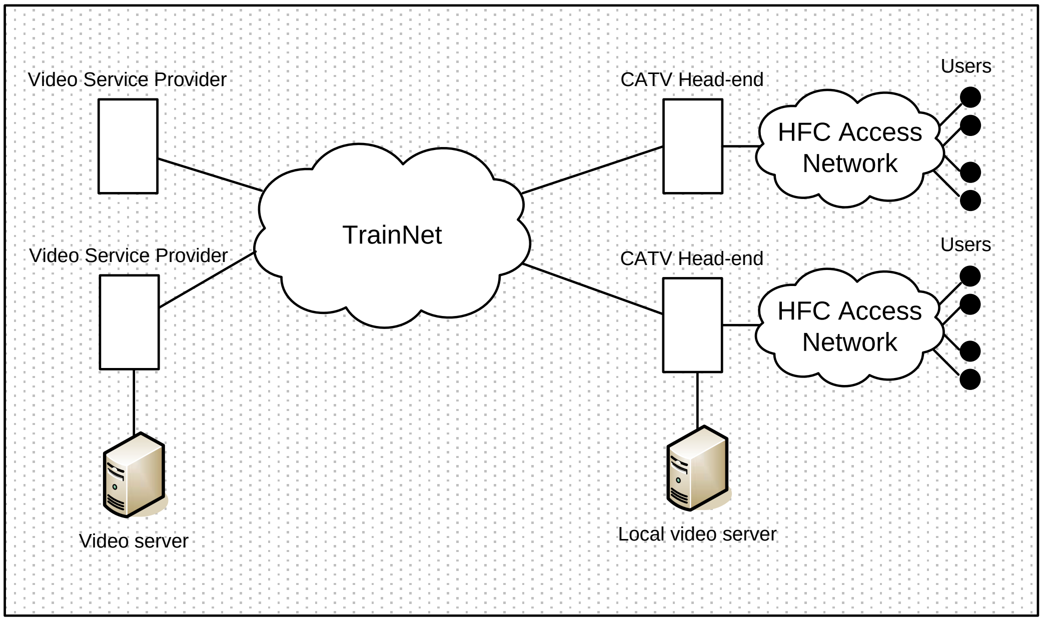

Figure 3.3 Deployment of a distributed VoD system using TrainNet.

to video contents originating form different VSPs. In such a system, VSPs are connected to CATV head-ends via high-speed broadband links provided by TrainNet. A VSP can use TrainNet by sending video files to a railway station. TrainNet then transports these files to another station that is connected to a CATV head-end, and from there delivers the files to the head-end. The head-ends then store these video contents and deliver them on demand to the end-users via HFC access networks [122, 8, 10].

3.2.3 Broadband Services over Rural WCATV

WCATV system is the technology of choice for cable television broadcasting in rural regions, where laying and maintenance of cables is financially infeasible. In such a system, a wireless access network is deployed to deliver television programs from a CATV head-end to the broadband wireless modems residing at subscriber premises. Apart from the delivery of television programs, WCATV systems allow the deployment of VoD and broadband Internet access to rural subscribers [121].

A key issue facing WCATV operators is the need for affordable methods for transporting data from service providers to rural regions. This issue can be easily addressed using TrainNet as long as WCATV networks are reachable via a railway network. A WCATV operator can install local servers at the CATV head-end, each of which replicates the services being provided by a service provider. These services are provided with delay whenever the information requested by a user is not available locally.

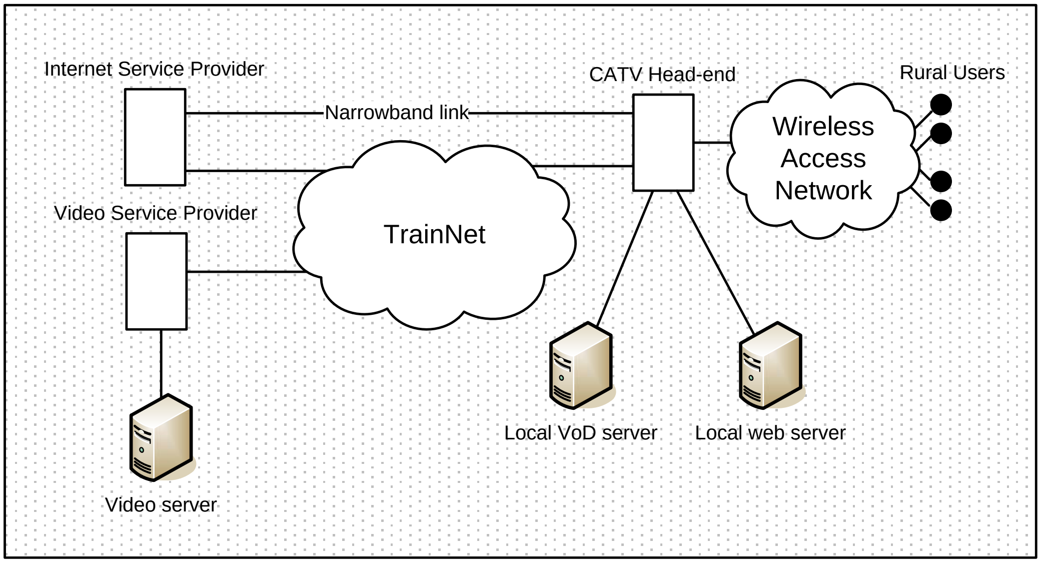

Figure 3.4 Deployment of broadband video services and Internet access over a wireless CATV.

Figure 3.4 illustrates a WCATV network offering video services and broadband store-and-forward Internet access to rural subscribers with the help of TrainNet. We see that TrainNet connects the CATV head-end to a VSP and an ISP. The VSP can utilize TrainNet to transport video contents to the VoD server located at the head-end. These video contents then become accessible to end-users, which can be delivered to them on-demand. As shown in Figure 3.4, the CATV head-end can access the ISP via TrainNet and a secondary path that is a dial-up or ADSL link. This secondary path allows the WCATV operator to offer narrowband applications without any delay. For example, a user can use this link for googling or sending a text message. Moreover, the ISP can exploit TrainNet to transmit static contents to the web server located at the head-end.

This allows WCATV operator to offer broadband applications such as:

- Web surfing services where users have access to locally cached web contents available on the web server. This service allows users to browse popular web contents, such as videos on YouTube^{TM}. Note that a user can request for web contents that are not available locally via the narrowband link. The ISP can then use TrainNet to deliver the requested content to the local web server.
- Web-based email services with attached voice and video messages.
- Access to multi-media content servers offering software, music, books, and movies. These repositories are hosted on local servers and their contents can be updated periodically using TrainNet.
- Remote education, where users have access to multimedia contents and tutorial presentations which are already available on servers. Note that a user can request for more information to be delivered from libraries around the world.
- Electronic commerce where users explore virtual malls and carry out their shopping online [8].

3.3 TrainNet Model

This thesis only considers trains travelling on a single railway track. That is, given stations 1 to N, the train stops at each station once when travelling from station 1 to N; similar to trains in reality. Trains travel according to an accurate timetable. Each train starts traveling from the first station; i.e., station 1 in the earlier example. Each station can deposit data onto a train for stations located downstream from it. For example, in the scenario exhibited in Figure 3.5, station 6 can use a train to send data to station 7 and 8 but it cannot send data to station 5. This means that the first station has access to all stations on a rail track, while the last one has access to no station. Note, the TrainNet model used herein can be applied to trains going in the opposite direction, thereby facilitating bi-directional communications. In other words, station 8 and 5 becomes the first and last station respectively.

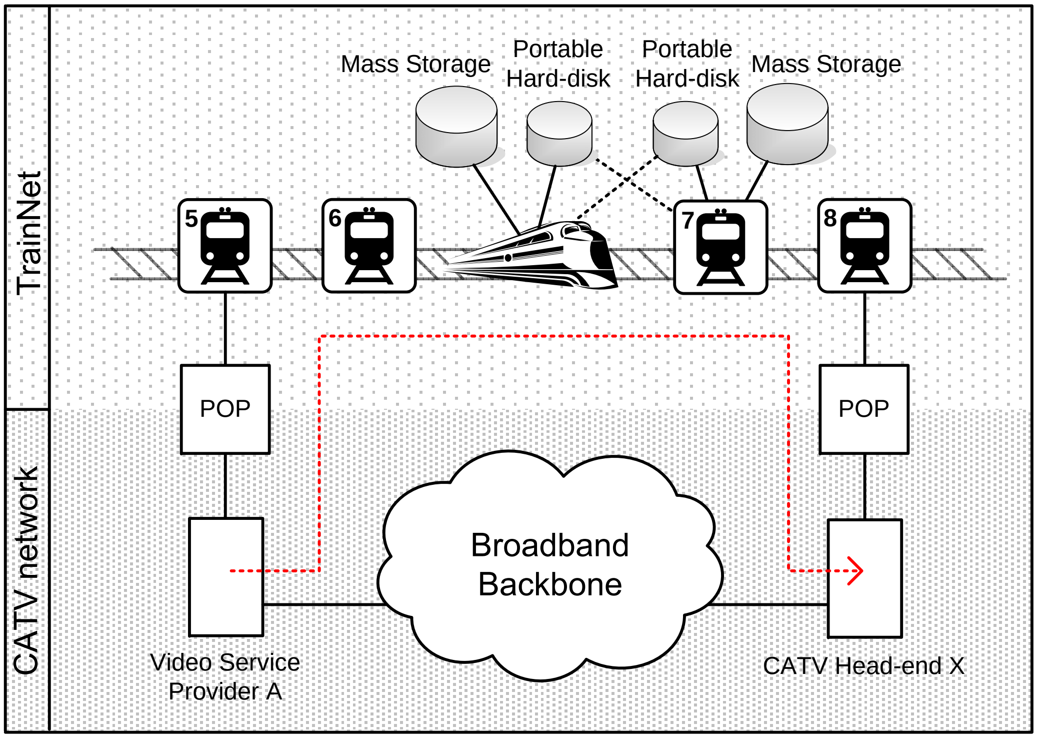

Figure 3.5 TrainNet in association with a distributed VoD system deployed over a large-scale CATV network. The dotted line shows the path through which video files are forwarded from VSP A to CATV head-end X via TrainNet.

TrainNet has three types of DTN nodes, namely *source stations*, *destination stations*, and *train nodes*. A source station is connected to a VSP via a POP, thereby allowing the VSP to send video files to the station. A train node is a mobile bundle forwarder (MBF) and its role is to forward data from source stations to destination stations. A destination station is connected to a CATV head-end via a POP, thereby enabling the head-end to receive video files from the station. For example, in Figure 3.5, station 5 is a source station and station 8 is a destination station. Note, a single station may exchange data with more than one VSP or head-end.

For simplicity, the path from a source station to a destination station is modeled as a one-hop virtual link. This means that the data from a source station is never stored at an intermediate station. For example, in Figure 3.5, consider service provider A using TrainNet to deliver video files to head-end X. A train carrying video files from station 5 is not allowed to deposit service provide A's files at stations 6 or 7 to be later forwarded to station 8 by another train. Without this assumption, there will be four routing possibilities to consider between station 5 and station 8. This routing problem is beyond the scope of this thesis.

From Figure 3.5, we see that stations and trains are equipped with *mass-storage* devices, i.e., a rack of portable hard-disks. The mass-storage devices utilize Redundant Array of Independent Disks (RAID) technology [123] which allows a damaged hard-disk to be replaced without data loss.

Data is exchanged between a station and a train as follows. Before the arrival of a train, the station fills its outgoing traffic onto one or more *portable hard-disks*. At the same time, the incoming train fills data destined for the station onto one or more portable hard-disks. Upon arrival at the station, these hard-disks are exchanged via one of the following methods:

- A staff member of the rail network is assumed to be available to swap the hard-disk at the station with those on the train. This thesis only considers this case as the method of data exchange.

- The station has a fiber cable that can be plugged into a switch on the train upon arrival. This too requires a staff member to plug and unplug the fiber cable as the train arrives and departs. Alternatively, TrainNet can use free space optical links [124]. Such links are capable of carrying up to 1.25 Gbps of data. Note that there is no need for portable devices in this case and data can be directly exchanged between mass storage devices at the station and on the train.

Figure 3.6 shows how TrainNet is connected to a global *dispatcher*. The dispatcher is responsible for managing the space on the portable hard-disks being

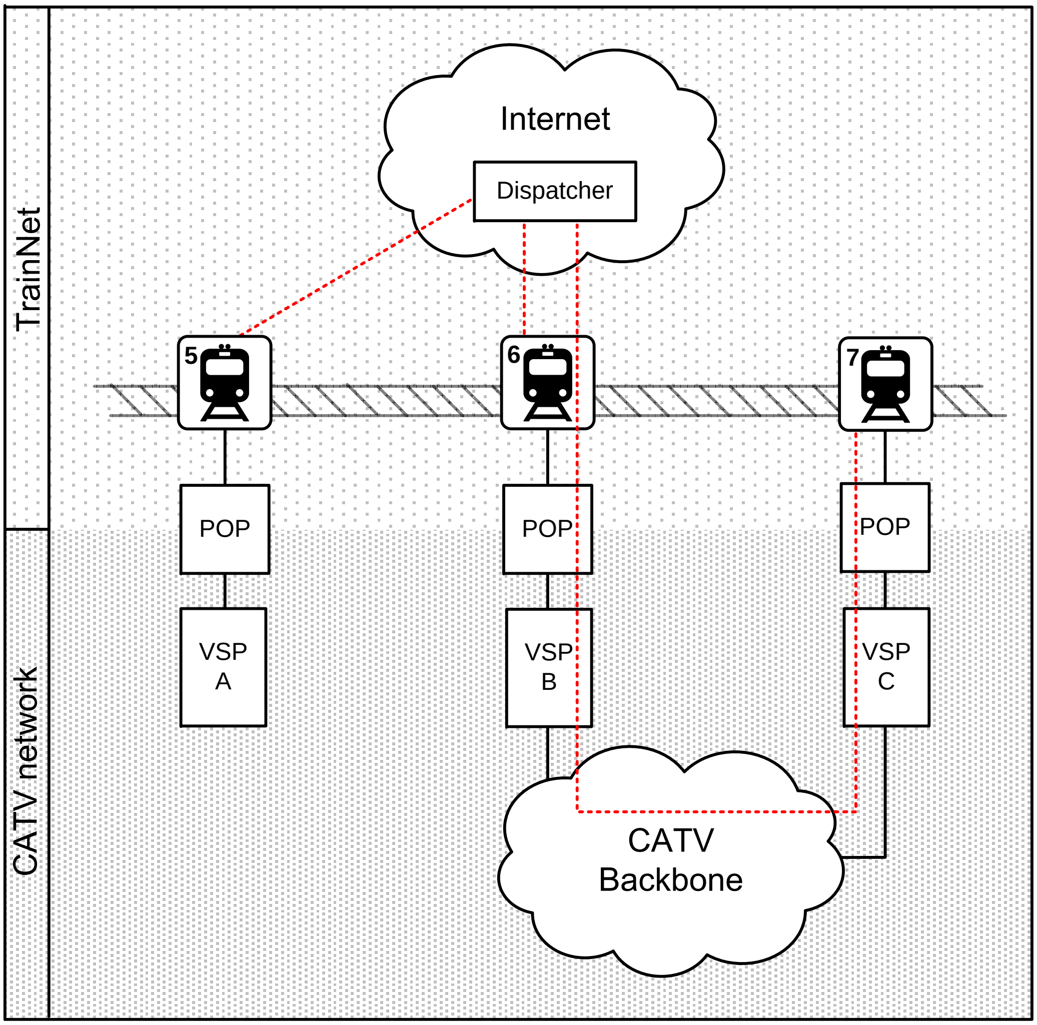

Figure 3.6 TrainNet being used to augment a large-scale CATV network. Here, the global dispatcher is deployed over the Internet. The dotted lines linking station 5 and 6 to the dispatcher illustrates connections established over ADSL links. Moreover, the dotted line going from station 7 to the dispatcher demonstrates a connection that is established through the CATV backbone and the ADSL link between station 6 and the dispatcher.

loaded onto trains using a *scheduling algorithm*. Specifically, each source station is required to ask the dispatcher for space on these out-going hard disks as a train approaches. To access the amount of data at each station, the global dispatcher assumes it has a connection to each station via a CATV backbone, ADSL or even dial-up. These connections are depicted in Figure 3.6. In Chapter 4, we will first outline a scheduler that does not use this global dispatcher before showing how the allocation of hard disk space can be made fairly and without data loss using a global dispatcher.

A key issue in TrainNet is the space constraint on hard disks; at each station and on a train. Specifically, the mass-storage at a source station may become overloaded by a VSP sending data to the station. In this thesis, the VSP is assumed to have a flow control mechanism that prevents this from happening; for example, the station can employ ECN [120] to indicate it is about to experience congestion, thereby informing a VSP to reduce its transmission rate. Apart from that, the mass-storage on a train may become overloaded by source stations. Hence, each train is equipped with a storage capacity equivalent to the aggregate capacity of all portable hard-disks being used to deliver data to the train. In other words, assuming each hard disk being loaded onto a train is 250 gigabytes in size, and there are four source stations, the train will then have 1000 gigabytes of storage capacity.

The portable hard-disks are the bottlenecks of the system, given their fixed capacity and the number of hard disks that a staff member can load on/from the train within the time period in which the train is at a station. This means when multiple source stations have traffic for the same destination station, they will have to compete for the limited space on the portable hard disks that are used to transfer data from the train to the destination station. Conversely, traffic at each source station needs to compete with each other for space on the portable hard-disks that are loaded onto a train. In other words, the competition for space on the portable hard disks being loaded and unloaded on/from a train constitutes a resource schedule problem. This problem is the focus of Chapter 4.

3.4 Summary

This chapter presents TrainNet, a novel architecture that uses a rail network to provide a very high bandwidth, low cost link that can be used to carry non real-time data. It also describes the components of TrainNet and explains how data is transported from source to destination stations using trains.

TrainNet uses hard disks and a global dispatcher to overcome the problem of intermittent links. The global dispatcher is a rendezvous point that allows a scheduler to learn about incoming and outgoing traffic of stations. As we shall see in Chapter 4, this information is used to fairly divide trains' storage capacity among stations. Unlike existing vehicular networks, TrainNet is a deterministic network. This is reasonable because trains travel across fixed railway routes and they follow an accurate timetable. This allows communications to be scheduled in advance.

Chapter 4

Resource Scheduling

Section 4.1 first gives a formal definition of the resource scheduling problem in TrainNet. Afterwards, Section 4.2 gives a preliminary scheduling algorithm for the said problem in Section 4.1. This is then followed by three max-min scheduling algorithms in Section 4.3. Finally, Section 4.4 gives a brief summary and then concludes the chapter.

4.1 The Resource Scheduling Problem

The resource scheduling problem is defined as follows. Consider a single unidirectional railway track linking N stations. Let station i sends messages to stations $i+1, i+2, \ldots, N$ using a train T. Recall that the path from station i to station j, where $i < j$, is modeled as a one-hop virtual link. Furthermore, station i performs a route look up using a routing table to determine the destination of a message. Here, a TrainNet operator is assumed to run the Border Gateway Protocol [43], where peering relationships are established between each station and POPs in order to exchange reachability or routing information.

Let $H^{j \to T}$ be the set of portable hard disks that is loaded onto train T at station j, and let $H^{T \to j}$ be the set of portable hard-disks that is unloaded from train T at station j. Without loss of generality, assume the total capacity of $H^{j \to T}$and $H^{T \to j}$ to be C gigabytes. The scheduler is then responsible for

1. Dividing the space on hard disk $H^{i \to T}$ fairly amongst traffic leaving station i.
2. Dividing the space on hard disk $H^{T \to j}$ fairly amongst the traffic leaving train T or station j.
3. Avoiding data loss caused by the capacity constraint of hard disk $H^{T \to j}$.

4.2 Scheduler

The First Come First Served (FCSF) [73] algorithm can be implemented in TrainNet to address the aforementioned resource scheduling problem. Specifically, each source station i has a single queue where incoming messages are queued according to their arrival time. Station i then fills each hard-disk $H^{i \to T}$ with messages from the queue. Furthermore, a train T traveling from station 1 to N has $N-1$ queues corresponding to each destination station. Within each queue, messages are ordered according to their arrival time. Once train T approaches station j, the train fills hard-disk $H^{T \to j}$ with messages that are waiting to be off loaded at station j.

Despite its simplicity, the FCFS algorithm results in data loss as well as unfair allocation of hard disk space. Consider station 1 to $N-1$, each of which has C gigabytes of data for station N, and a train T that has $C \times (N-1)$ gigabytes of storage capacity. Train T then collects C gigabytes from station 1 to $N-1$ using hard disk $H^{1 \to T}, \ldots, H^{N-1 \to T}$, respectively. Once train T approaches station N, it fills hard-disk $H^{T \to N}$ with the C gigabytes of data that it has received from station 1. Notice that the data from station 2 to N-1 are not delivered because hard-disk $H^{T \to N}$ is filled with data from station 1. Hence, the $C \times (N-2)$ gigabytes of data from station 2 to $N-1$ will be discarded by train T, and must be delivered again by other trains.

FCFS is also unfair because of the following two reasons:

- Consider station i and $i+1$ sending data to station $i+2$ using a train T traveling from station i to $i+2$. The train first receives data from station i and then from station $i+1$. Hence, once train T fills hard disk $H^{T \to i+2}$, it gives messages from station i a higher priority at the expense of downstream stations because these messages were loaded first.

- At a station i, traffic/flows headed to different stations are not given space fairly on hard disk $H^{i \to T}$. In other words, the traffic going from station i to station j and k is not given equal rights to the space on hard disk $H^{i \to T}$. This is due to the use of a single queue at station i where traffic is served in a FCFS manner. For example, a train T approaching station i, where the first C gigabytes in the queue are headed for station j, would prevent other traffic leaving station i for a station k, where $j \neq k$, from using hard disk $H^{i \to T}$.

To address the aforementioned problems, the next section presents three variants of the max-min fair algorithm.

4.3 Max-Min Fair Algorithms

This section describes three variants of the well known max-min fair algorithm: *Local Max-Min Fair* (LMMF), *Global Max-Min Fair* (GMMF), and *Weighted Global Max-Min Fair* (WGMMF). These algorithms aim to address the resource scheduling problem given in Section 4.1 such that:

1. All stations have an equal right to the space on the portable hard-disks.
2. Data loss is eliminated.

4.3.1 Local Max-Min Fair

The first variant, called LMMF, is run when a train T is about to arrive at station i. Assume that train T has messages from station $1, 2, \ldots, i-1$ for station i, and station i has messages for station $i+1, i+2, \ldots, N$. At station i, messages are placed into $N-i$ queues based on their respective destination, and within each queue, messages are served in FCFS order.

The goal of LMMF is to divide $H^{i \to T}$ and $H^{T \to i}$ fairly amongst the traffic leaving station i and train T, respectively. To do this, LMMF determines the amount of data being queued at station i and on train T. Note that LMMF is run at each station and not at the dispatcher, see Figure 3.6. Hence, LMMF uses only *local information* such as queue length of a given station, when making decisions.

At station i, LMMF allocates space on hard-disk $H^{i \to T}$ as follows:

1. Let $D_{i,k}$ denotes station i's demand for space on $H^{T \to k}$, where $k \in \{i+1, \ldots, N\}$. Without loss of generality, $D_{i,i+1}$ to $D_{i,N}$ is sorted in increasing order, so that $D_{i,i+1} \leq D_{i,i+2} \leq \cdots \leq D_{i,N}$.
2. Let $Q_{i,k}$ be the share on hard-disk $H^{i \to T}$for messages going from station i to station k. The value of $Q_{i,i+1}$ to Qi, N is initially set to $\frac{C}{N-i}$. Recall that C is the capacity of $H^{i \to T}$.
3. LMMF then determines whether $D_{i,i+1}$ is less than $Q_{i,i+1}$. If it is, $Q_{i,i+1}$ is set to $D_{i,i+1}$. The remaining hard-disk space $\frac{C}{N-i} - D_{i,i+1}$ is then evenly distributed to $Q_{i,i+2}, \ldots, Q_{i,N}$. If $D_{i,i+1}$ is more than $Q_{i,i+1}$, LMMF retains the value of $Q_{i,i+1}$.
4. At this point, the value of $Q_{i,i+1}$ is final, and the algorithm repeats Step 3 to determine $Q_{i,i+2}, \ldots, Q_{i,N}$.

Figure 4.1 shows how LMMF divides hard disk $H^{5 \to T}$ fairly amongst traffic going from station 5 to stations $k = 6, \ldots, 9$. We see that station 5's demands for space on hard-disk $H^{T \to 6}$ to $H^{T \to 9}$ are 0.05, 0.50, 0.15, and 1.00 respectively; i.e., $D_{5,6} = 0.05$, $D_{5,7} = 0.50$, $D_{5,8} = 0.15$, and $D_{5,9} = 1.00$. Here, the hard disk capacity has been normalized to one. As depicted in the figure, LMMF yields $Q_{5,6} = 0.05$, $Q_{5,7} = 0.40$, $Q_{5,8} = 0.15$, and $Q_{5,9} = 0.40$ for hard disk $H^{5 \to T}$. These values are computed as follows.

Following Step 1, the algorithm first sorts $D_{5,k}$ in increasing order, so we have $D_{5,6} \leq D_{5,8} \leq D_{5,7} \leq D_{5,9}$. In Step 2, $Q_{5,6}$ to $Q_{5,9}$ are initially set to $\frac{1}{4}$, given there are four stations. Step 3 then determines whether $D_{5,6}$ is less than $Q_{5,6}$,

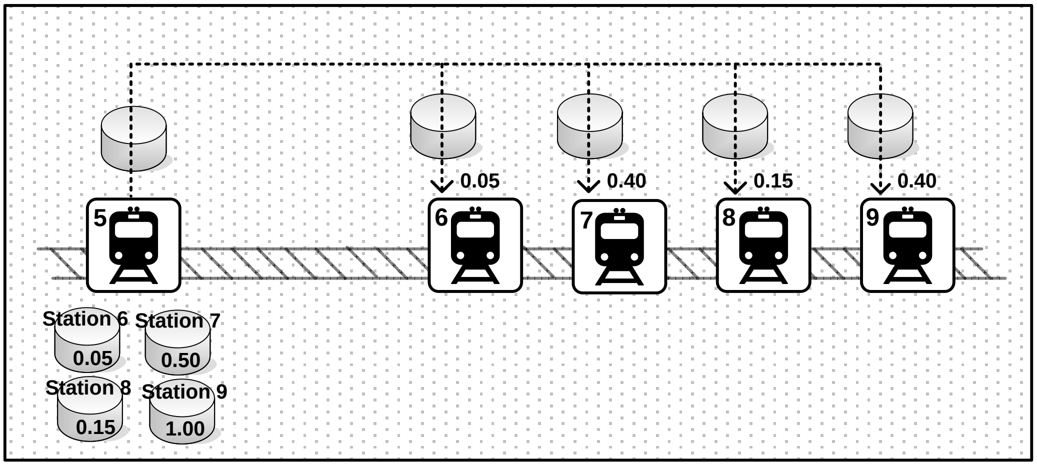

Figure 4.1 An example showing how the LMMF algorithm divides hard-disk $H^{5 \to T}$among the traffic leaving station 5.

and consequently, $Q_{5,6}$ is set to 0.05. Next, the unused allocation of $D_{5,6}$, i.e., 0.20, is evenly distributed to $Q_{5,8}$, $Q_{5,7}$ and $Q_{5,9}$, resulting in each of them receiving a share of $0.25 + \frac{0.20}{3}$. With $Q_{5,6}$ finalized, LMMF goes back to Step 3 and compares $Q_{5,8}$ with $D_{5,8}$. This yields a value of $Q_{5,8} = 0.40$ and a remainder of 0.167 for distribution among station 7 and 9. Repeating Step 3 for $D_{5,7}$ and $D_{5,9}$ yields the value 0.40 for both $Q_{5,7}$ and $Q_{5,9}$.

LMMF works similarly on a train, i.e., when it is used to allocate space on hard disk $H^{T \to i}$. The only difference is that the capacity constraint of hard-disk $H^{T \to i}$ will result in data loss. For example, consider station 1 to $i-1$, each of which sends C gigabytes of data for station i using train T. The train traveling from station 1 to $i-1$ collects C gigabytes of data from each station using hard disks $H^{1 \to T}$,...,$H^{i-1 \to T}$, respectively. Once train T approaches station i, it runs LMMF, which fills $\frac{C}{i-1}$ gigabytes of data from each station onto $H^{T \to i}$. As a result, $C - \frac{C}{i-1}$ gigabytes of data from each station 1 to $i-1$ are discarded by train T, and must be retransmitted by other trains.

In the example above, LMMF results in data loss because, at each station 1 to $i-1$, it does not consider other stations' demands for space on hard disk $H^{T \to i}$. This can be prevented from happening by evenly dividing hard-disk

$H^{T\to i}$ amongst stations 1 to $i-1$ and then allowing each of them to use $\frac{C}{i-1}$ gigabytes of space on hard disk $H^{1\to T}$,...,$H^{i-1\to T}$ only. We will see how this key observation is applied in the next max-min variant.

4.3.2 Global Max-Min Fair

The next variant, called GMMF, is run before the arrival of train T at station i. Consider station 1 to N, where station i receives data from station $1,\dots,i-1$ and sends data to station $i+1,\dots,N$ using train T. Moreover, assume that station j, where $i<j$, uses train T to send data to station $j+1,\dots,N$ and receive data from station $1,\dots,j-1$.

Let $f^{i\to j}$ denote the traffic flow going from station i to j using train T. Furthermore, $F^{i\to T}$ denotes the set of flows going from station i to station $i+1,\dots,N$ using hard disk $H^{i\to T}$; i.e., $F^{i\to T}=\left\{f^{i\to i+1},\dots,f^{i\to N}\right\}$, and $F^{T\to j}$ denotes the set of flows going from station *1,...,j-1* to station j using hard-disk $H^{T\to j}$; i.e., $F^{T\to j}=\{f^{1\to j},\dots,f^{j-1\to j}\}$.

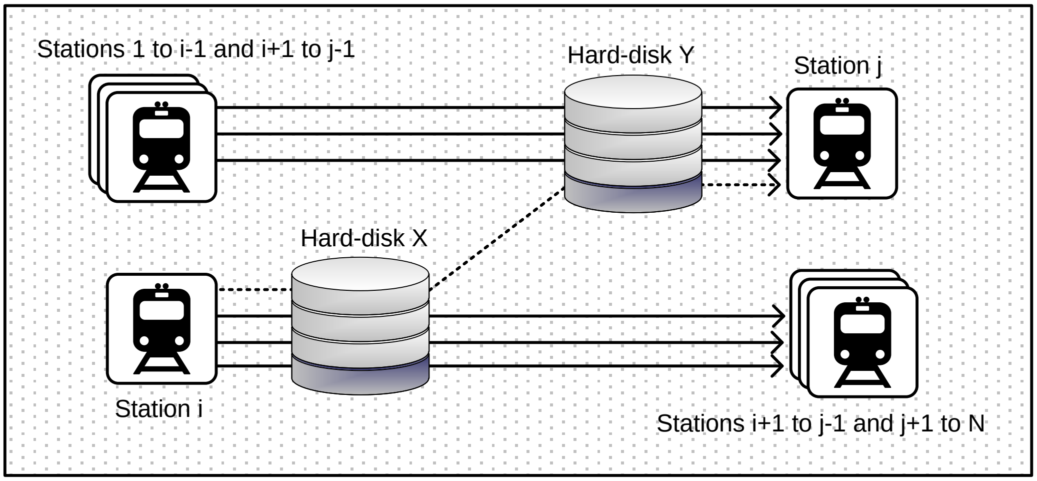
Stations 1 to i-1 and i+1 to j-1
Hard-disk Y
Station j
Hard-disk X
Station i
Stations i+1 to j-1 and j+1 to N

Figure 4.2 The dotted line shows the virtual traffic flow going from station i to j. Here, hard disk X and Y corresponds to $H^{i\to T}$ and $H^{T\to j}$ respectively.

The goal of GMMF is to divide the space on hard disk $H^{i\to T}$ fairly amongst the flow $f^{i\to i+1}$,...,$f^{i\to j}$,...,$f^{i\to N}$. From Figure 4.2, we see that GMMF needs to consider how hard disk Y ($H^{T\to j}$) is allocated to traffic going to station j, which in turn determines how much space traffic from station i should occupy hard disk X ($H^{i \to T}$). In other words, if only p bytes are entering TrainNet, where p bytes correspond to the share allocated to $f^{i \to j}$, the share on hard-disks $H^{i \to T}$ and $H^{T \to j}$ are equal. Hence, a train T will never hold more data than the capacity of hard disk $H^{T \to j}$. As a result, train T will never have to discard data.

Let $r^{i \to j}$ be the share allocated to flow $f^{i \to j}$ on hard-disk $H^{i \to T}$ and $H^{T \to j}$. Furthermore, let $R^{i \to T}$ denotes the vector of shares on hard-disk $H^{i \to T}$; i.e., $R^{i \to T} = \left\{r^{i \to i+1}, \ldots, r^{i \to j}, \ldots, r^{i \to N}\right\}$, and $R^{T \to j}$ denotes the vector of shares on hard-disk $H^{T \to j}$; i.e., $R^{T \to j} = \{r^{1 \to j}, \ldots, r^{i \to j}, \ldots, r^{j-1 \to j}\}$. The total occupied space on hard-disks $H^{i \to T}$ and $H^{T \to j}$ is then,

$$U^{i \to T} = \sum r^{i \to j}, \ for\ all\ r^{i \to j} \in R^{i \to T} \tag{4.1}$$

$$U^{T \to j} = \sum r^{i \to j}, \ for\ all\ r^{i \to j} \in R^{T \to j} \tag{4.2}$$

GMMF calculates the share $r^{i \to i+1}, \ldots, r^{i \to j}, \ldots, r^{i \to N}$ as follows:

1. Vector $R^{i \to T}$ is first initialized to zero.
2. Let $n^{T \to j}$ and $n^{i \to T}$ denote the cardinality of the set $R^{i \to T}$ and $R^{T \to j}$ respectively. The maximum allocation allowed for all shares in $R^{i \to T}$ is then calculated as follows:
   $m = min(\frac{C-U^{i \to T}}{n^{i \to T}}, \frac{C-U^{T \to i+1}}{n^{T \to i+1}}, \ldots, \frac{C-U^{T \to j}}{n^{T \to j}}, \ldots, \frac{C-U^{T \to N}}{n^{T \to N}}), \ for\ all\ r^{i \to j} \in R^{i \to T}$
3. Next, the algorithm increments all shares $r^{i \to j} \in R^{i \to T}$ by m, and saturate the hard-disk $H^{i \to T}, H^{T \to i+1}, \ldots, H^{T \to j}, \ldots, H^{T \to N}$ if one of the following conditions is satisfied:
   - $m \times n^{i \to T} + U^{i \to T} = C, \ for\ all\ r^{i \to j} \in R^{i \to T}$

- $m \times n^{T\to j} + U^{T\to j} = C,\ for\ all\ r^{i\to j} \in R^{T\to j}$

4. Each $r^{i\to j}$ crossing a saturated hard-disk is then removed from $R^{i\to T}$ and $R^{T\to j}$, and the allocation for $r^{i\to j}$ is considered finalized.
5. The algorithm repeats the process from Step 2 again until $R^{i\to T}$ becomes empty.

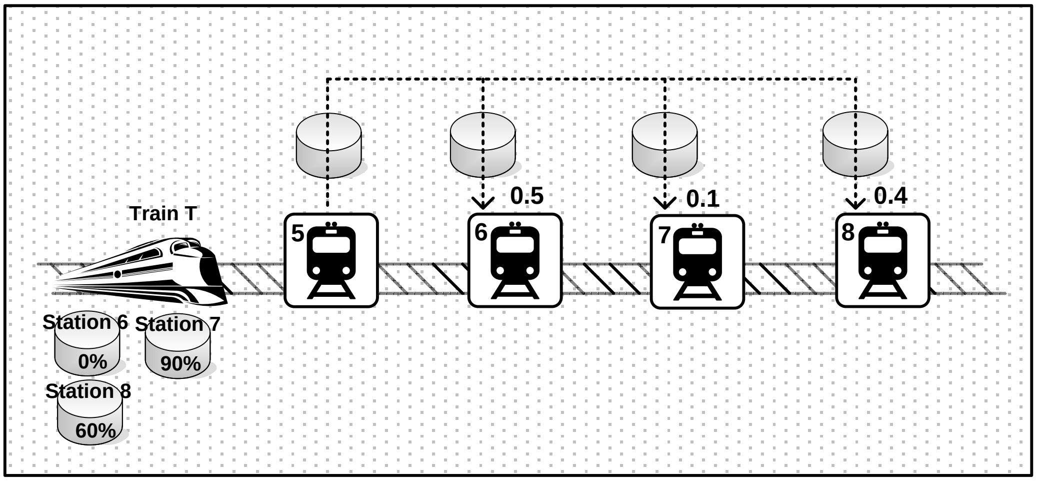

Figure 4.3 An example in which the hard-disk $H^{5\to T}$ is divided between flow $f^{5\to 6}$, $f^{5\to 7}$, and $f^{5\to 8}$ using GMMF.

Figure 4.3 shows how hard-disk $H^{5\to T}$ is divided between the flow $f^{5\to 6}$, $f^{5\to 7}$, and $f^{5\to 8}$. We see that the total occupied space on hard disk $H^{T\to 6}$, $H^{T\to 7}$, and $H^{T\to 8}$ is 0.0, 0.9, and 0.6, respectively. This means the hard disk to be unloaded at station 8 is already at 60% of its storing capacity before the train arrives at station 5. The algorithm first initializes the share $r^{5\to 6}$, $r^{5\to 7}$, and $r^{5\to 8}$ to zero. Next, according to Step 3, the algorithm determines $m = 0.1$ with respect to hard-disk $H^{5\to T}$, $H^{T\to 6}$, $H^{T\to 7}$, and $H^{T\to 8}$, resulting in each flow receiving 0.1 share on $H^{5\to T}$. This causes $H^{T\to 7}$ to become full, meaning $r^{5\to 7}$ can be finalized. Next, the algorithm removes $r^{5\to 7}$ from $R^{5\to T}$ at Step 4, goes back to Step 2 and determines $m = 0.3$ with respect to hard-disk $H^{5\to T}$, $H^{T\to 6}$, and $H^{T\to 8}$. Repeating the process for the rest of the flows yields a value of 0.4 and 0.5 for $r^{5\to 8}$ and $r^{5\to 6}$ respectively.

GMMF avoids data loss by allocating an equal share to each flow $f^{i\to j}$ on hard-disk $H^{i\to T}$ and $H^{T\to j}$. This behavior results in better performance compared to LMMF. For example, consider station 1 to 4, where there are four virtual flows going from station $1 \to 2$, $1 \to 4$, $2 \to 4$, and $3 \to 4$. In this example, the flow $f^{1\to 2}$ and $f^{1\to 4}$ will compete for $H^{1\to T}$ whereas the flow $f^{1\to 4}$,$f^{2\to 4}$, and$f^{3\to 4}$ will compete for $H^{T\to 4}$. Using LMMF, the flow $f^{1\to 2}$ and $f^{1\to 4}$ will receive $\frac{1}{2}$ of hard disk $H^{1\to T}$, and the flow $f^{1\to 4}$,$f^{2\to 4}$, and $f^{3\to 4}$ will receive $\frac{1}{3}$ of hard disk $H^{T\to 4}$. This means that the flow $f^{1\to 4}$ is subjected to a data loss rate of $\frac{1}{2}$ - $\frac{1}{3}$. GMMF prevents this from happening by allocating $\frac{2}{3}$ and $\frac{1}{3}$ share of hard disk $H^{1\to T}$ to the flow $f^{1\to 2}$ and $f^{1\to 4}$ respectively. Hence, no data loss occurs and the flow $f^{1\to 2}$ receives $\frac{1}{6}$ more share on hard disk $H^{1\to T}$.

A key limitation of GMMF is that it assumes all flows in $F^{i\to T}$ have equal rights to the space on hard-disk $H^{i\to T}$. This assumption can be modified in favor of achieving greater utilization of space on hard disk $H^{T\to i+1}, \ldots, H^{T\to N}$. This modification can be done by assigning weights to flows and then giving each flow a priority proportional to its weight; this is the topic of Section 4.3.3.

4.3.2.1 GMMF is Max-Min Fair

A key concern is whether GMMF is max-min fair. To prove this property, we will need to show that if $r^{i\to j}$ is considered finalized at Step 4, where $r^{i\to j}$ is the share allocated to $f^{i\to j}$, flow $f^{i\to j}$ is bottlenecked by hard-disk $H^{T\to j}$ or $H^{T\to j}$. After that, we apply the following proposition [101]:

Proposition 4.1 *A feasible vector of shares $R^{i\to T}$ is max-min fair, if and only if, every traffic flow in $F^{i\to T}$ crosses a bottleneck hard disk.*

The proof to be presented adopts the following definitions from [101]:

Definition 4.1 A vector of shares, $R^{i\to T} = \{r^{i\to j} | f^{i\to j} \in F^{i\to T}\}$, is considered *feasible* if it satisfies the following constraints:

$$r^{i\to j} \geq 0, \ \textit{for all } r^{i\to j} \in R^{i\to T} \tag{4.3}$$

$$U^{i\to T} \leq C \tag{4.4}$$

$$U^{i\to T} \leq C \tag{4.5}$$

Definition 4.2 A feasible vector $R^{i\to T}$ is said to be *max-min fair* if given two flows $f^{i\to j} \in F^{i\to T}$and $f^{i\to k} \in F^{i\to T}$, where $j \neq k$ and $r^{i\to j} \leq r^{i\to k}$, $r^{i\to j}$ cannot be increased without causing a decrease in $r^{i\to k}$.

Definition 4.3 $H^{i\to T}$ and $H^{T\to j}$ are defined as a *bottleneck hard-disk* for $f^{i\to j}$ if they fulfill the following conditions:

$$r^{i\to j} \geq r^{i\to x},\ for\ all\ r^{i\to x} \in R^{i\to T} \tag{4.6}$$

$$U^{i\to T} = C \tag{4.7}$$

or

$$r^{i\to j} \geq r^{y\to j},\ for\ all\ r^{y\to j} \in R^{T\to j} \tag{4.8}$$

$$U^{T\to j} = C \tag{4.9}$$

Theorem 4.1 *GMMF is max-min fair.*

Proof. At each iteration, Step 3 increments each $r^{i\to j}$ remaining in $R^{i\to T}$ by an equal share, i.e., m. This means that each traffic flow $f^{i\to j}$ has at least an equal share to any other traffic flow crossing hard disk $H^{i\to T}$ and $H^{T\to j}$. Thus, as per Definition 4.3, if Step 4 considers share $r^{i\to j}$ finalized, where $r^{i\to j}$ is the share allocated to $f^{i\to j}$, flow $f^{i\to j}$ is bottlenecked by hard-disk $H^{T\to j}$ or $H^{T\to j}$. Upon termination of the algorithm, no share remains in $R^{i\to T}$, and therefore all traffic flows are bottlenecked by at least one hard disk. Hence, according to Proposition 4.1, $R^{i\to T}$ is max-min fair. □

4.3.3 Weighted Global Max-Min Fair

The WGMMF algorithm is similar to GMMF except that it gives a flow say $f^{i\to x}$ a higher priority than the flow $f^{i\to y}$ if $n^{T\to x} < n^{T\to y}$, where $n^{T\to x}$ and $n^{T\to y}$ denote the cardinality of the set $F^{T\to x}$ and $F^{T\to y}$ respectively. In other words, a flow is given a higher priority or weight if it uses a hard disk with less utilization. Hence, if $H^{T\to x}$ is less utilized than $H^{T\to y}$, flows using $H^{T\to x}$ will receive a higher weight than those using $H^{T\to y}$.

WGMMF assigns a weight to each flow $f^{i\to j} \in F^{i\to T}$ as follows:

$$W^{i\to j} = \prod n^{T\to k},\ \textit{for all}\ f^{i\to k} \in F^{i\to T}\ \textit{where}\ j \neq k \tag{4.10}$$

At a source station i, WGMMF gives the highest priority to the flow $f^{i\to j}$ if $n^{T\to j} = \min(n^{T\to i+1}, \dots, n^{T\to j}, \dots, n^{T\to N})$. For example, consider six flows going from station 1 → 2, 1 → 3, 1 → 4, 2 → 3, 2 → 4, and 3 → 4. Using Equation 4.10, at station 1, $W^{1\to 2}$, $W^{1\to 3}$, and $W^{1\to 4}$ is calculated as 6, 3, and 2 respectively. If GMMF is used in this example, flow $f^{1\to 2}$, $f^{1\to 3}$, and $f^{1\to 4}$ will be given equal rights to hard-disk $H^{1\to T}$, and thereby providing $f^{1\to 4}$ at least $\frac{1}{3}$ of hard disk $H^{1\to T}$ and $H^{T\to 4}$. This means that GMMF scarifies efficiency in favor of fairness. For example, if $f^{1\to 4}$ were given no share on $H^{T\to 4}$, flow $f^{2\to 4}$ and $f^{3\to 4}$ were entitled to $\frac{1}{2}$ of hard disk $H^{T\to 4}$, while the space on $H^{1\to T}$could be divided evenly among $f^{1\to 2}$ and $f^{1\to 3}$. Hence, the overall hard-disk utilization would be better. To address this problem, WGMMF uses $W^{1\to 4}$ to decrease the right of $f^{1\to 4}$ to hard-disk $H^{1\to T}$ and $H^{T\to 4}$, and thereby allowing flow $f^{2\to 4}$ and $f^{3\to 4}$ to obtain a greater share of $H^{T\to 4}$. We will see later how WGMMF results in better hard disk utilization.

To apply $W^{i\to j}$ to flow $f^{i\to j}$, WGMMF represents flow $f^{i\to j}$ as $W^{i\to j}$ virtual flows, each of which is denoted as $f_1^{i\to j}, \dots, f_W^{i\to j}$. These virtual flows, thus, effectively increases the demand of $f^{i\to j}$ by $W^{i\to j}$. Let vector $R_W^{i\to T}$ denotes the vector of weighted shares corresponding to flow $f^{i\to i+1} \dots, f^{i\to j}, \dots, f^{i\to N}$, including each of their virtual flows. This means for each flow $f^{i\to j}$, $R_W^{i\to T}$ includes $W^{i\to j}$ shares, each of which is denoted by $r_1^{i\to j}, \dots, r_{W^{i\to j}}^{i\to j}$. The resulting vector is thus, $R_W^{i\to T} = \{r_1^{i\to i+1}, \dots, r_{W^{i\to i+1}}^{i\to i+1}, \dots, r_1^{i\to j}, \dots, r_{W^{i\to j}}^{i\to j}, \dots, r_1^{i\to N}, \dots, r_{W^{i\to N}}^{i\to N}\}$.

WGMMF then calculates the share$r^{i\to i+1}, \dots, r^{i\to j}, \dots, r^{i\to N}$ as follows:

1. WGMMF runs GMMF on $R_W^{i\to T}$ rather than$R^{i\to T}$. The result is a max-min fair allocation of shares $r_1^{i\to i+1}, \dots, r_{W^{i\to i+1}}^{i\to i+1}, \dots, r_1^{i\to j}, \dots, r_{W^{i\to j}}^{i\to j}, \dots, r_1^{i\to N}$ $, \dots, r_{W^{i\to N}}^{i\to N}$.
2. The algorithm then calculates the final share of each flow $f^{i\to j}$ as $r^{i\to j} =$

$r_1^{i\to j} + \cdots + r_{W^{i\to j}}^{i\to j}$ or $r^{i\to j} = W^{i\to j} \times r_{W^{i\to j}}^{i\to j}$. This results in flow $f^{i\to j}$ receiving a share $r^{i\to j}$ that is *proportionally fair* with respect to its weight $W^{i\to j}$.

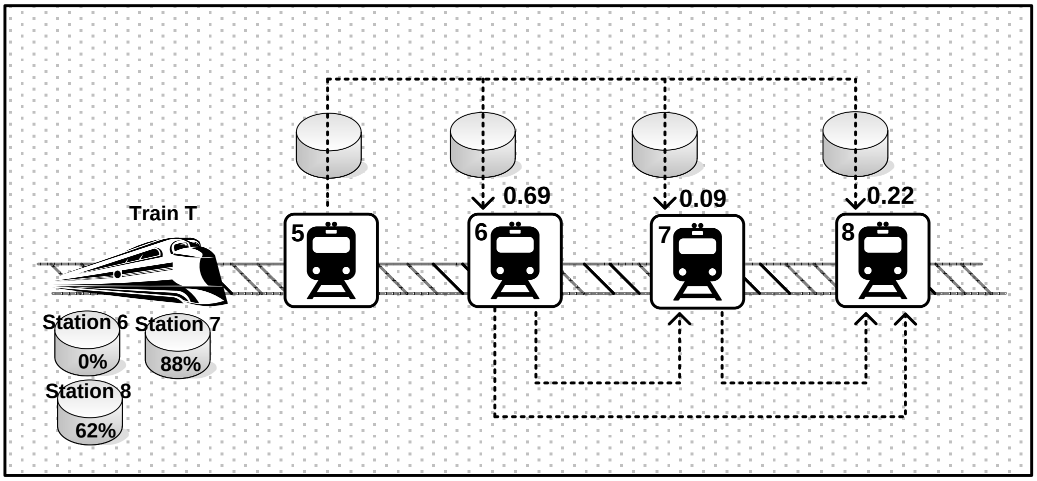

Figure 4.4 An example in which $H^{5\to T}$ is divided between the flow $f^{5\to 6}$, $f^{5\to 7}$, and $f^{5\to 8}$.

Figure 4.4 shows how WGMMF divides $H^{5\to T}$ among flow $f^{5\to 6}$,$f^{5\to 7}$, and $f^{5\to 8}$. According to Equation 4.10, these flows have a weight value of 6, 3, and 2, respectively. This means $F_W^{5\to T}$ has 11 flows $f_1^{5\to 6}, \ldots, f_6^{5\to 6}, f_1^{5\to 7}, \ldots, f_3^{5\to 7}, f_1^{5\to 8}$, and $f_2^{5\to 8}$. Using, GMMF on $F_W^{5\to T}$ yields a share of 0.03 for $r_1^{5\to 7}, \ldots, r_3^{5\to 7}$, which results in flow $f^{5\to 7}$ obtaining 0.03×3 or 0.09 share of hard-disk $H^{5\to T}$. The final value of $r^{5\to 6}$ and $r^{5\to 8}$ is 0.69 and 0.22 respectively.

WGMMF has better performance than GMMF when allocating space on $H^{T\to i+1}, \ldots, H^{T\to N}$. For example, consider station 1 to 3 and three virtual flows going from station1 → 2, 1 → 3, and 2 → 3. In this example, flow $f^{1\to 2}$ and $f^{1\to 3}$ will compete for $H^{1\to T}$ and $f^{1\to 3}$ and $f^{2\to 3}$ will compete for $H^{T\to 3}$. Using GMMF, the flow $f^{1\to 2}$ and $f^{1\to 3}$ will receive $\frac{1}{2}$ of hard disk $H^{1\to T}$, and $f^{1\to 3}$ and $f^{2\to 3}$ will receive $\frac{1}{2}$ of hard disk $H^{T\to 3}$. GMMF gives flow $f^{1\to 3}$ and $f^{2\to 3}$ equal rights to the space on hard disk $H^{T\to 3}$. However, flow $f^{1\to 3}$ has to compete with flow $f^{1\to 3}$ for space on $H^{1\to T}$, while flow $f^{1\to 3}$ is entitled to the entire space of hard disk $H^{2\to T}$. Here, the idea is to decrease $f^{1\to 3}$'s right to the space on $H^{1\to T}$, and thereby allowing flow $f^{2\to 3}$ to receive a larger share on hard disk $H^{T\to 3}$. Hence, WGMMF uses $W^{1\to 2} = 2$ and $W^{1\to 3} = 1$ to allocate $\frac{2}{3}$ and $\frac{1}{3}$ share of hard disk $H^{1\to T}$ to $f^{1\to 2}$ and $f^{1\to 3}$ respectively. As a result, the flow $f^{2\to 3}$ receives $\frac{1}{6}$ more share on hard disk $H^{T\to 3}$, and the overall utilization of hard-disk $H^{T\to 2}$ and $H^{T\to 3}$ is increased compared to GMMF.

The goal of WGMMF is to increase hard-disk utilization of GMMF using the weight function given in Equation 4.10. However, this weight function can be replaced in favour of other goals, such as providing differentiated services.

4.4 Summary

This chapter presents the resource scheduling problem that arises from bottlenecked hard disks. To address this problem, this chapter presents four scheduling algorithms: FCFS, LMMF, GMMF, and WGMMF.

The proposed scheduling algorithms are different from those used in DieselNet and KioskNet; see Chapter 2. The resource allocation in RAPID focuses on optimizing routing objectives and not on achieving fairness. Apart from that, the bus-schedule-aware algorithm in KioskNet only focuses on balancing the load at Internet gateways. While both of these algorithms achieve some level of fairness, their fairness is not max-min.

Chapter 5

Simulation and Results

Section 5.1 first introduces DESMO-J and gives an overview of its features. After that, it describes the process based approach; a modeling style which is used to implement the TrainNet simulator according to the system model described in Chapter 3. Section 5.2 outlines the key components of the TrainNet simulator. Afterwards, Section 5.3 explains the scenario used for all experiments. Next, Section 5.4 discusses traffic models and Section 5.5 explains the metrics used to evaluate the performance of the proposed scheduling algorithms. Section 5.6 presents simulation results, and Section 5.7 concludes the chapter.

5.1 Simulation Platform

DESMO-J [125] is an object-oriented framework for developing simulation models. It extends the Java [126] programming language to support the discrete-event simulation paradigm. Specifically, it adds the following features to the standard Java platform:

- A simulation infrastructure consisting of scheduler, event list, simulation time clock, report generator, and debugging tools.
- Classes for developing common modeling components, such as queues, random number generators, stochastic distributions, and data collectors.

- Abstract classes to define a simulation model, including model, entity, event, and simulation process.

DESMO-J supports the *process based* modeling style [127]; an object-oriented approach for developing discrete-event simulators. Specifically, all activities owned by a component of the system are grouped into a *process*, which can be seen as the lifecycle of that component. During its lifecycle, each component is either in the *active* or *passive* state. Only a component that is active is permitted to cause a change in the model's state, where the component is allowed to modify its properties or that of other components. Moreover, it can create new components, activate other passive components, and deactivate itself for a certain period of time [127].

5.2 Simulation Model

The simulation model of TrainNet comprises of six component types: TrainNetSim, TrafficGenerator, Station TrainGenerator, Train, and Global Dispatcher. The UML[1] activity diagram shown in Figure 5.1 to 5.6 depicts the lifecycle of each component respectively, where each diagram indicates a component's activities, the sequence in which these activities take place and their relationship to other components in the model.

The simulation model is implemented using the abstract classes of DESMO-J. TrainNetSim is a static component, and thus its class is derived from the *model* abstract class. Other components are dynamic, and therefore their classes are derived from the *simulation process* abstract class. The source code of TrainNet simulator is accessible via a Subversion repository [129].

The following sections elaborate each component further.

---

[1] For readers who are not familiar with the Unified Modeling Language, an introductory book [128] is recommended.

5.2.1 TrainNetSim

TrainNetSim is the main component of the model. It creates and activates all other components except the Train processes. TrainNetSim holds global references to all components, thereby allowing each component to access and modify other components. TrainNet also holds global data stores used by components to exchange information.

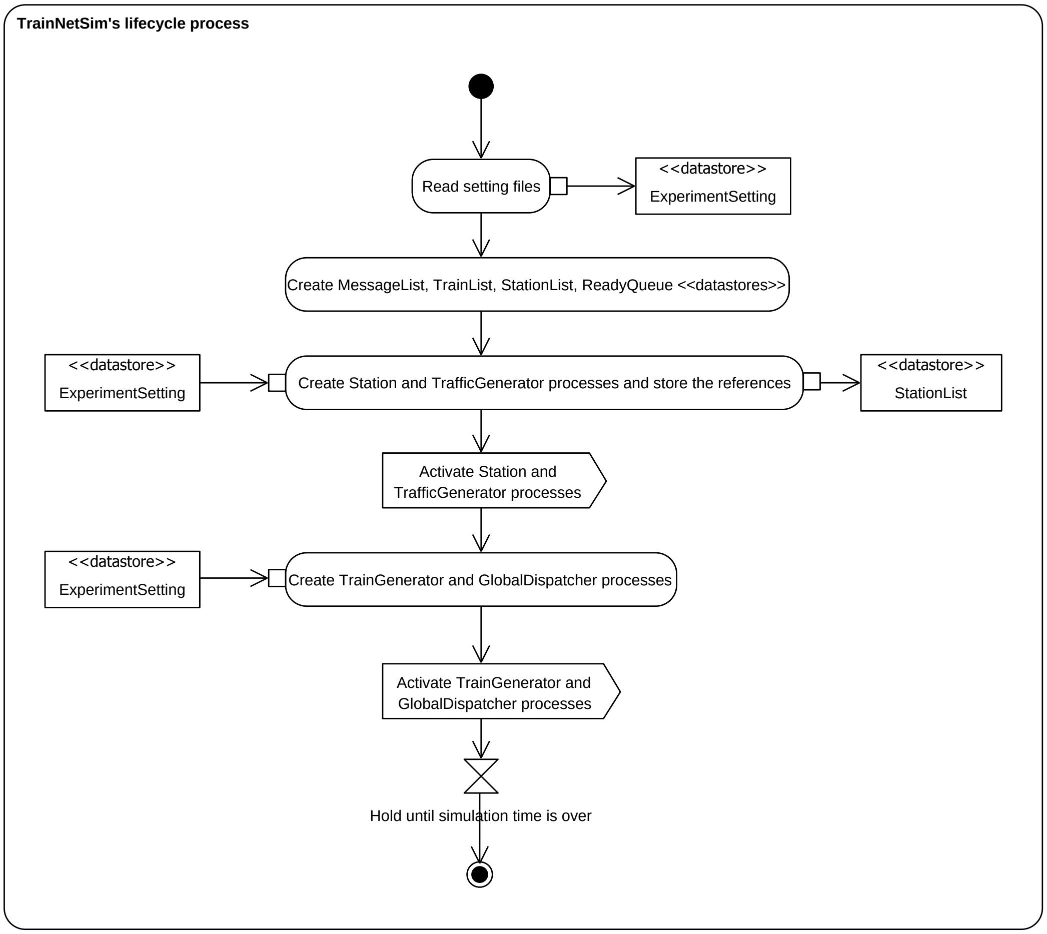
TrainNetSim's lifecycle process
Read setting files
<<datastore>> ExperimentSetting
Create MessageList, TrainList, StationList, ReadyQueue <<datastores>>
<<datastore>> ExperimentSetting
Create Station and TrafficGenerator processes and store the references
<<datastore>> StationList
Activate Station and TrafficGenerator processes
<<datastore>> ExperimentSetting
Create TrainGenerator and GlobalDispatcher processes
Activate TrainGenerator and GlobalDispatcher processes
Hold until simulation time is over

Figure 5.1 TrainNetSim process lifecycle.

Figure 5.1 shows that TrainNetSim first reads the experiment's settings from configuration files and stores them in a shared data store called `ExperimentSetting`. The `ExperimentSetting` object includes the parameters described in Section 5.3. TrainNetSim then creates three shared data stores: `MessageList`, `StationList`, and `TrainList`. These objects hold global references to Message objects, Station processes, and Train processes, respectively. Moreover, TrainNetSim creates `ReadyQueue`, a shared data object which holds a global reference to a train that is arriving at a station. Next, TrainNetSim instantiates and then activates the Stations, TrafficGenerators, TrainGenerator, and GlobalDispatcher processes according to the settings given in `ExperimentSetting`. Finally, TrainNetSim waits until the simulation time is over before terminating itself.

5.2.2 TrafficGenerator

Each TrafficGenerator component is associated with a Station process, and it is responsible for generating the outgoing messages of that Station process. The outgoing messages are generated according to the traffic models described in Section5.4. The TrafficGenerator process is activated whenever a message needs to be created.

The TrafficGenerator, as seen in Figure 5.2, first creates a bundle message based on the configurations given in `ExperimentSetting`. Next, the TrafficGenerator adds the message to a shared data store called `NewMessage` and awakes the Station process to fetch the message. Note that each user generates a single request. After that, the TrafficGenerator process calculates the next user arrival time according to the user arrival model specified in the `ExperimentSetting` object. Finally, it deactivates itself until the next user arrives.

5.2.3 Station

Each Station component is responsible for managing its outgoing messages in a number of queues according to the respective destination of each message. Note, in our experiments, the queues have infinite length. The number of Station processes created in the model depends on the experiment configurations given in `ExperimentSetting`.

From Figure 5.3, the Station component is first activated by the TrafficGenerator, which generates outgoing messages. Then, it fetches the new message

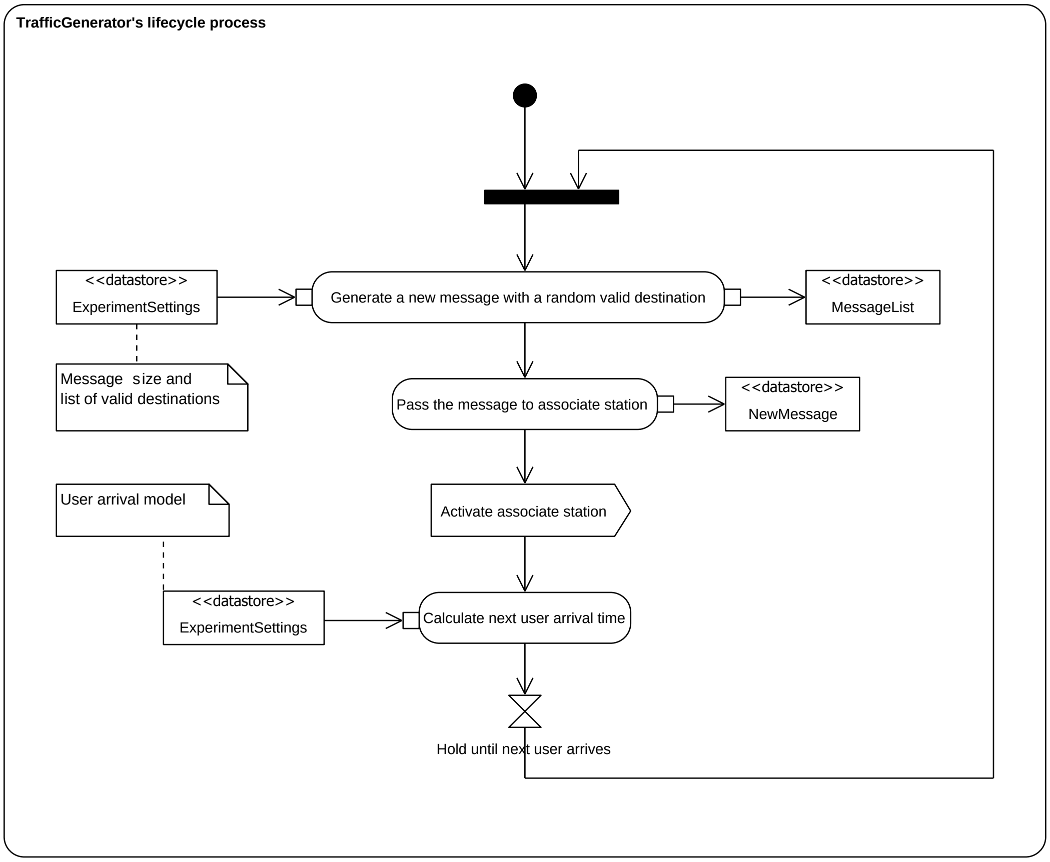
TrafficGenerator's lifecycle process
<<datastore>>
ExperimentSettings
Generate a new message with a random valid destination
<<datastore>>
MessageList
Message size and
list of valid destinations
Pass the message to associate station
<<datastore>>
NewMessage
User arrival model
Activate associate station
<<datastore>>
ExperimentSettings
Calculate next user arrival time
Hold until next user arrives

Figure 5.2 TrafficGenerator process lifecycle.

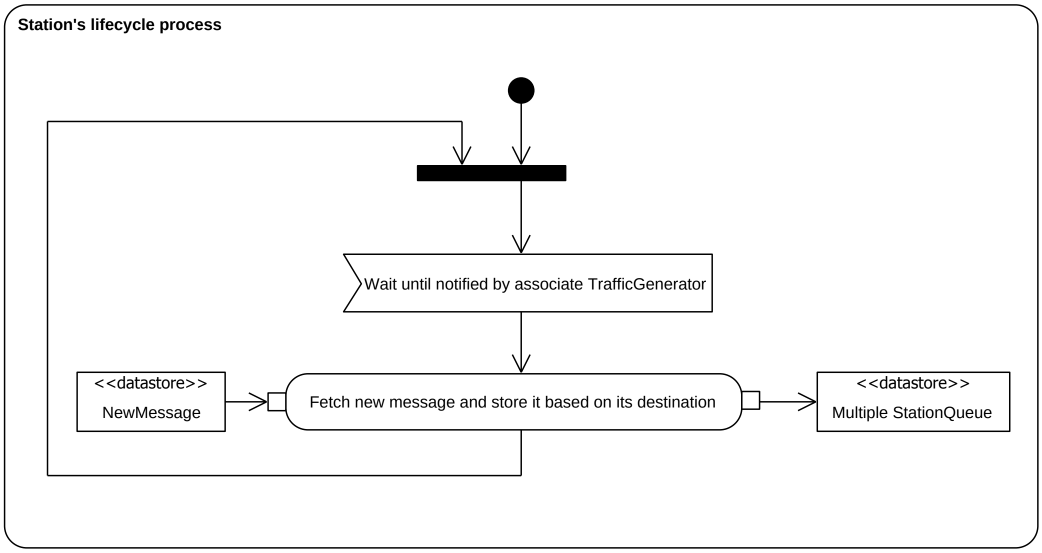
Station's lifecycle process
Wait until notified by associate TrafficGenerator
<<datastore>>
NewMessage
Fetch new message and store it based on its destination
<<datastore>>
Multiple StationQueue

Figure 5.3 Station process lifecycle.

produced by the TrafficGenerator and stores the message in its outgoing queues based on the message's destination. Finally, the station process deactivates itself and waits until the TrafficGenerator awakens it again.

5.2.4 TrainGenerator

The model includes one TrainGenerator component that generates Train processes according to a common timetable given in the `ExperimentSetting` object. The TrainGenerator is activated whenever a new Train process is due to enter the simulator.

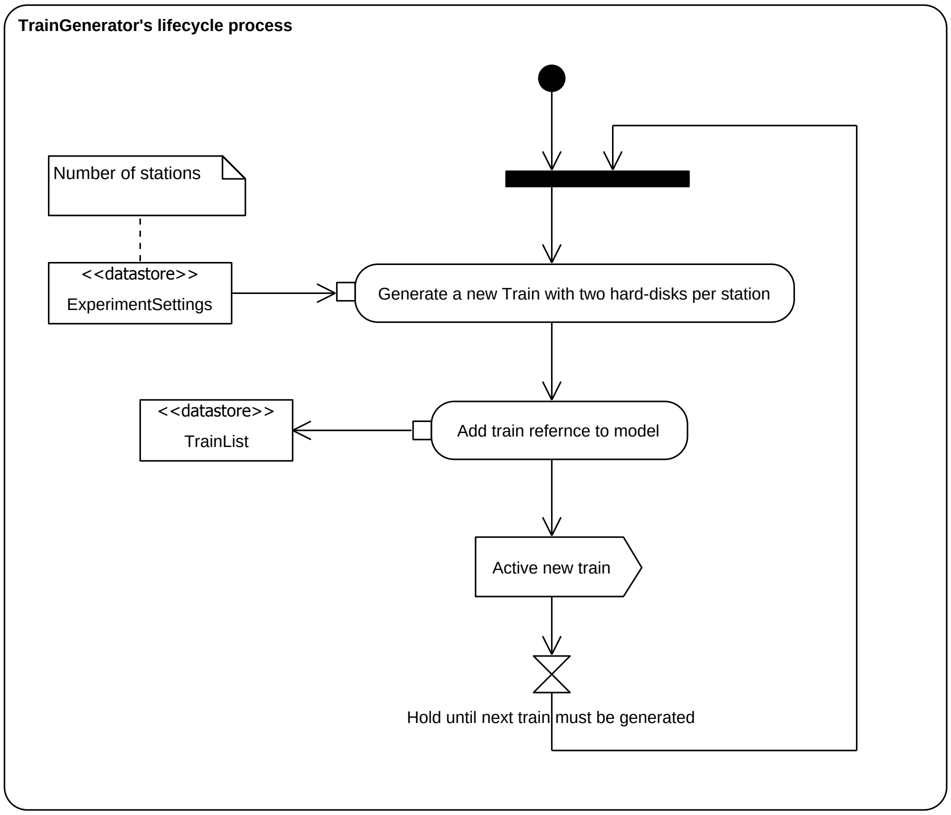
TrainGenerator's lifecycle process
Number of stations
<<datastore>>
ExperimentSettings
Generate a new Train with two hard-disks per station
<<datastore>>
TrainList
Add train refernce to model
Active new train
Hold until next train must be generated

Figure 5.4 TrainGenerator process lifecycle.

As displayed in Figure 5.4, the TrainGenerator component first creates a Train processes, and it then adds that Train to the shared data store generated by TrainNetSim. After that, TrainGenerator activates the new Train process before deactivating itself.

5.2.5 Train

Each Train component travels between stations according to a common timetable given in `ExperimentSetting`. The number of Train processes created in the model depends on a common timetable. Train processes are responsible for holding all hard-disk objects but they do not manage the messages on these hard-disks. Each Train process T holds one incoming and one outgoing hard-disk object per each Station process S. The incoming hard-disk object is used to store messages Train T receives from Station S. The outgoing hard-disk object holds messages Train T sends to Station S. The hard-disk objects have equal storing capacities specified in the `ExperimentSetting` object.

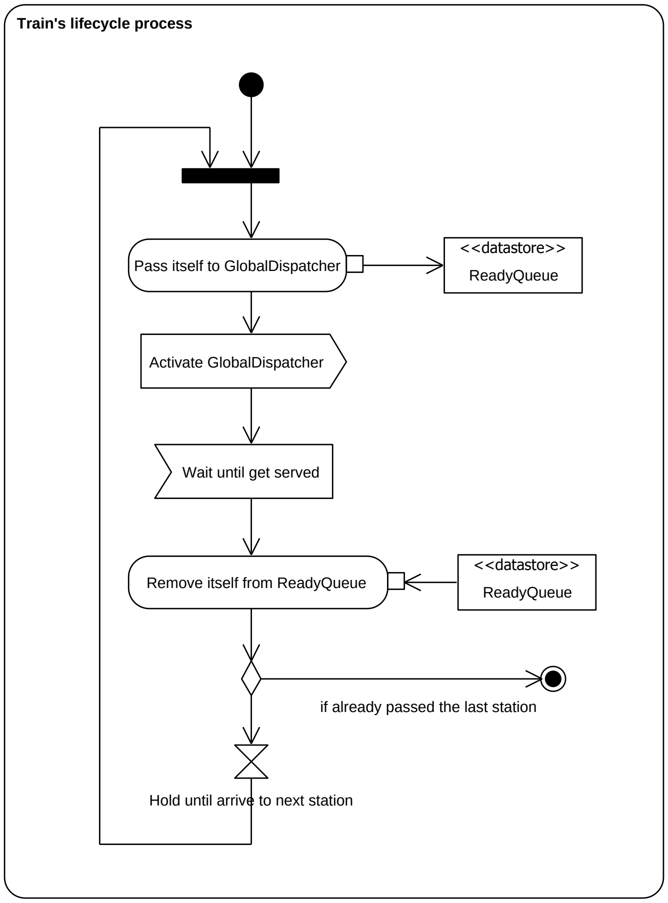
Train's lifecycle process
Pass itself to GlobalDispatcher
<<datastore>>
ReadyQueue
Activate GlobalDispatcher
Wait until get served
Remove itself from ReadyQueue
<<datastore>>
ReadyQueue
if already passed the last station
Hold until arrive to next station

Figure 5.5 Train process lifecycle.

Referring to Figure 5.5, the Train process is activated every time it arrives at a station. Upon activation, the Train adds itself to the `ReadyQueue` object, and then activates the GlobalDispatcher process and starts waiting. Afterwards, the GlobalDispatcher exchanges messages between the Train and the Station, and then activates the Train. Upon reactivation, the Train removes itself from the `ReadyQueue` object and leaves the Station. It then checks whether it has already passed the last station. If yes, it terminates itself, otherwise, it deactivates.

5.2.6 GlobalDispatcher

The model contains one GlobalDispatcher component that is activated every time a train T arrives at a station S. The GlobalDispatcher exchanges messages between the `MultipleStationQueues` of Station S and the `IncomingHardDisks` and `OutgoingHardDisks` of Train T. The scheduling algorithm used by the GlobalDispatcher process is given in the `ExperminetSetting` object.

From Figure 5.6, we see that GlobalDispatcher is activated by Train T arriving at Station S. GlobalDispatcher first fetches the Train process from the `ReadyQueue` object. Recall that Train T holds one incoming hard disk and one outgoing hard disk for each station. Next, the GlobalDispatcher fills Station S's `OutgoingHardDisk` with messages stored on incoming hard-disks corresponding to Stations 1 to S-1. Messages are considered delivered to Station S if they are successfully transferred to `OutgoingHardDisk` that Train T holds for Station S. Note that hard-disk objects have limited storing capacity, and therefore some messages may not be transferred to Station S. Such messages are considered lost. Lost messages are then added to their respective source station to be retransmitted by another Train. Next, the GlobalDispatcher fills Station S's `IncomingHardDisk` with messages stored on the `MultipleStationQueues` of Station S. Finally, the GlobalDispatcher reactivates Train T to continue its journey, and it then deactivates itself until another train arrives at another station.

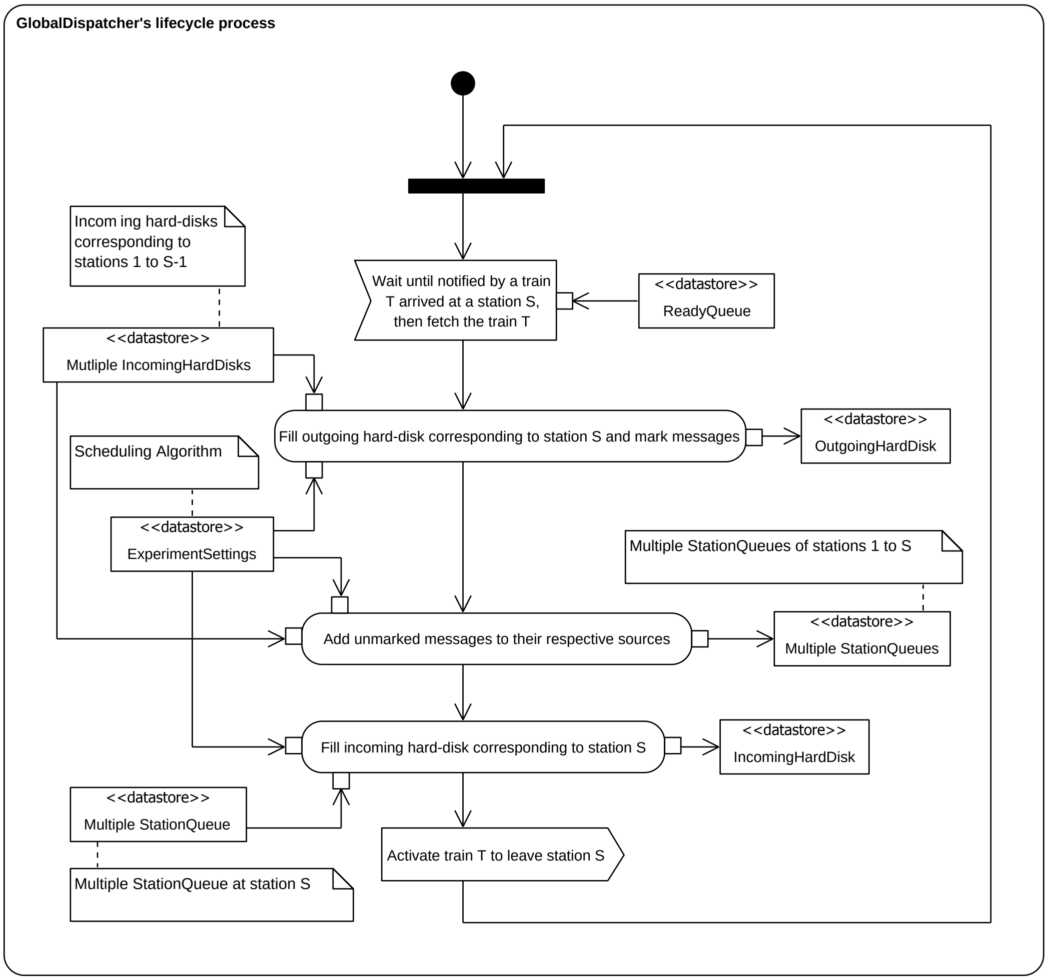
GlobalDispatcher's lifecycle process
Incom ing hard-disks corresponding to stations 1 to S-1
<<datastore>> Mutliple IncomingHardDisks
Wait until notified by a train T arrived at a station S, then fetch the train T
<<datastore>> ReadyQueue
Fill outgoing hard-disk corresponding to station S and mark messages
<<datastore>> OutgoingHardDisk
Scheduling Algorithm
<<datastore>> ExperimentSettings
Multiple StationQueues of stations 1 to S
Add unmarked messages to their respective sources
<<datastore>> Multiple StationQueues
Fill incoming hard-disk corresponding to station S
<<datastore>> IncomingHardDisk
<<datastore>> Multiple StationQueue
Multiple StationQueue at station S
Activate train T to leave station S

Figure 5.6 GlobalDispatcher process lifecycle.

5.3 Experiment Scenarios

Figure 5.7 shows a scenario in which TrainNet delivers video content from three service providers A, B, and C to three head-end X, Y, and Z. This scenario is used for all experiments. It includes four stations 1 to 4 and five virtual flows going from station $1 \rightarrow 2$, $1 \rightarrow 4$, $2 \rightarrow 3$, $2 \rightarrow 4$, and $3 \rightarrow 4$. Note, from here on $f^{i \rightarrow j}$ denotes the virtual flow going from station i to j. Moreover, $H^{i \rightarrow T}$ and $H^{T \rightarrow j}$ denote the portable hard disks that are loaded onto train T at station i and unloaded from train T to station j respectively. Table 5.1 shows the competition for space on hard disk $H^{1 \rightarrow T}$, $H^{2 \rightarrow T}$, $H^{3 \rightarrow T}$, $H^{T \rightarrow 2}$, $H^{T \rightarrow 3}$, and $H^{T \rightarrow 4}$. For example, the flow $f^{1 \rightarrow 2}$ is entitled to the entire space on hard disk $H^{T \rightarrow 2}$, however, it must compete with $f^{1 \rightarrow 4}$ for the space on hard disk $H^{1 \rightarrow T}$.

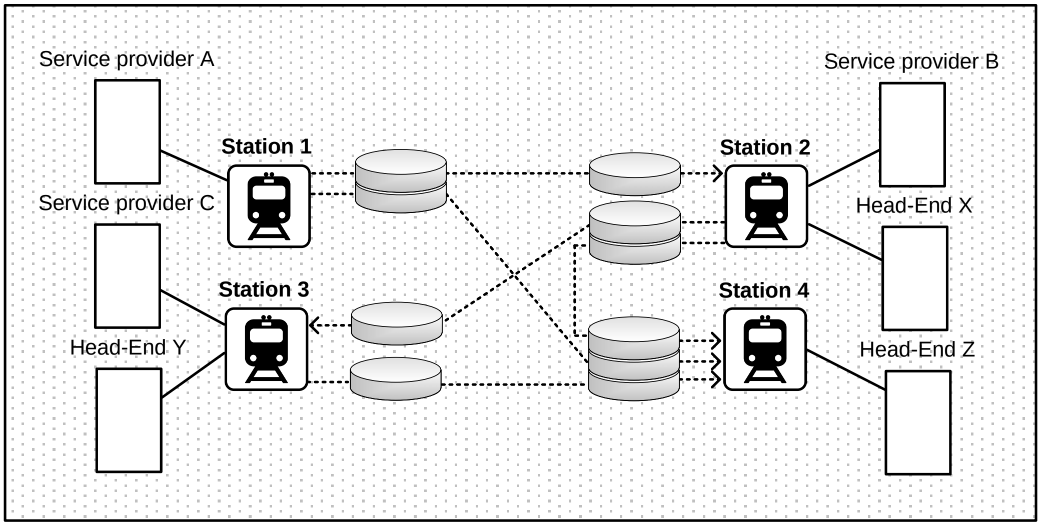

Figure 5.7 A scenario comprising of four stations, three service providers, three head-ends, five flows, and six hard disks.

The scenario's parameters used for the trains are as follows. There are 48 trains per day. Train arrival has a constant rate of 0.5 per hour (i.e., every 30 minutes). A train takes 20 minutes to travel from one station to the next. Each train has a buffer size of 360 gigabyte, and portable hard disks with 120 gigabyte capacity.

Given the above scenario, experiments are then conducted on the four schedul-

Flow	**Competition for hard disks**					
	$H^{1\to T}$	$H^{2\to T}$	$H^{3\to T}$	$H^{T\to 2}$	$H^{T\to 3}$	$H^{T\to 4}$
$f^{1\to 2}$	×			×		
$f^{1\to 4}$	×					×
$f^{2\to 3}$		×			×	
$f^{2\to 4}$		×				×
$f^{3\to 4}$			×			×

Table 5.1 Competitions among virtual flows.

ing algorithms described in Chapter 4. Specifically, these experiments seek to answer the following questions:

- Which algorithms are fair?
- What are the data loss rates caused by each algorithm?
- What are the averaged aggregate throughput and mean aggregate delay achieved by each algorithm?

5.4 TrafficModels

Two traffic models are used, both based on the analysis of a CATV distribution network [10]. In the first traffic model, the demands of a VoD system are assumed to be similar to the demands of traditional broadcast programs. That is, with a *prime peak* during the evening when more users are likely to watch television. The user arrival rate is considered to be a function of the time of the day, and it is modeled as follows [10].

The requesting process experienced by each service provider is assumed to be a Poisson process with a time-varying rate $\lambda(t)$. Thus, the inter-arrival delays between requests are modeled as an exponential distribution with mean $\frac{1}{\lambda(t)}$. Each user is characterized by the following two parameters [10]:

- The average number of movies requested during a certain period of time, which is set to λ= 2 movies per week. Each movie is assumed to be 3.5 Gigabyte in size. This file size is equivalent to 100 minutes of high definition video content [3].
- To model the prime peak during the evening of each day, the experiments used the following normal distribution; see Table 5.2 for corresponding parameters. Note that weekdays and weekends are assumed to have the same peak [10].

$$\lambda\left(t\right)=\frac{N\frac{\lambda}{7}}{\sigma\sqrt{2\pi}}e^{\frac{-(t-\mu)^2}{2\sigma^2}} \tag{5.1}$$

Symbol	**Definition**	**Value**
μ	Prime peak time over a day	72000 seconds (i.e., 8 P.M.)
σ	Standard deviation	5400 seconds (i.e., 90 minutes)
N	Total user population	3000

Table 5.2 Parameters and values for Equation 5.1 [10].

The second traffic model is similar to the first one, except that the normal distribution defined by Equation 5.1 is replaced by the following function of time:

$$\lambda\left(t\right)=10+\left\lfloor\frac{t}{3600}\right\rfloor\times 10 \tag{5.2}$$

Figure 5.8 exhibits the number of requests generated by the two user arrival models over a 24 hour period. The figure shows that the first model generates a large portion of the requests within a few hours. Specifically, the aggregate number of requests generated between the hours 1600 and 2400 is equal to 94 percent of the total requests per a day. This behavior causes the system to fluctuate between two levels of traffic load, i.e., being idle or saturated. On the other hand, Figure 5.8 indicates that the second model yields a steady growth in the number of requests. Hence, the stations' demand for hard disk space ranges from zero to infinity.

Please see print copy for image

Figure 5.8 Number of requests generated by the two user arrival models based on [10].

The first traffic model is used to measure the averaged aggregate throughput and mean aggregate delay, where a realistic traffic model is desirable. The second model is used to investigate fairness, data loss performance, and hard disk utilization, where varying load at stations is desirable.

5.5 Metrics

After each run, the following metrics are calculated and collected every 3600 simulation seconds:

- Capacity shares per flow (MB): The capacity share obtained by $f^{i \to j}$ is calculated by summing the size of messages that are delivered successfully from station i to station j. This metric measures the share an algorithm allocates to each flow. We will later see how this metric is used in the calculation of Jain's fairness index.

- Data loss rate per flow (MB): The data loss rate experienced by $f^{i\to j}$ is equal to the aggregate size of messages that are not transferred from train to station j.
- Hard disk utilization (MB): The utilization of hard disk $H^{T\to j}$ is equal to the aggregate size of messages that have been successfully delivered via that hard disk.
- Averaged aggregate throughput (Mbps): The average aggregate throughput is calculated by dividing the aggregate size of messages that are successfully delivered during a given period by the number of seconds in the period.
- Mean aggregate delay (seconds): The mean aggregate delay is calculated by averaging the difference between the arrival time and delivery time of each individual message.

The above said metrics are averaged over 100 simulation runs per experiment. The average capacity shares, data loss rates, and hard disk utilization are normalized to one by dividing them by the aggregate size of the hard disks used over a 3600 second period.

Jain's fairness index [130] is used to quantify the degree of fairness. This index is defined according to Equation 5.3, where N denotes the number of flows and $X^{i\to j}$ denotes the capacity share obtained by $f^{i\to j}$.

$$J = \frac{(\sum X^{i\to j})^2}{N(\sum X^{i\to j^2})} \tag{5.3}$$

Jain's fairness index ranges from $\frac{1}{N}$ to 1, where $J = \frac{1}{N}$ indicates no fairness and $J = 1$ shows absolute fairness.

5.6 Results

In Section 5.6.1 to 5.6.4, the capacity shares and data loss rates are first used to investigate the fairness and data loss avoidance of each scheduling algorithm. In Section 5.6.5, hard disk utilization is then used to compare the algorithms against each other. Finally Section 5.6.6 compares the algorithms in terms of the average aggregate throughput and mean aggregate delay.

5.6.1 First Come First Served

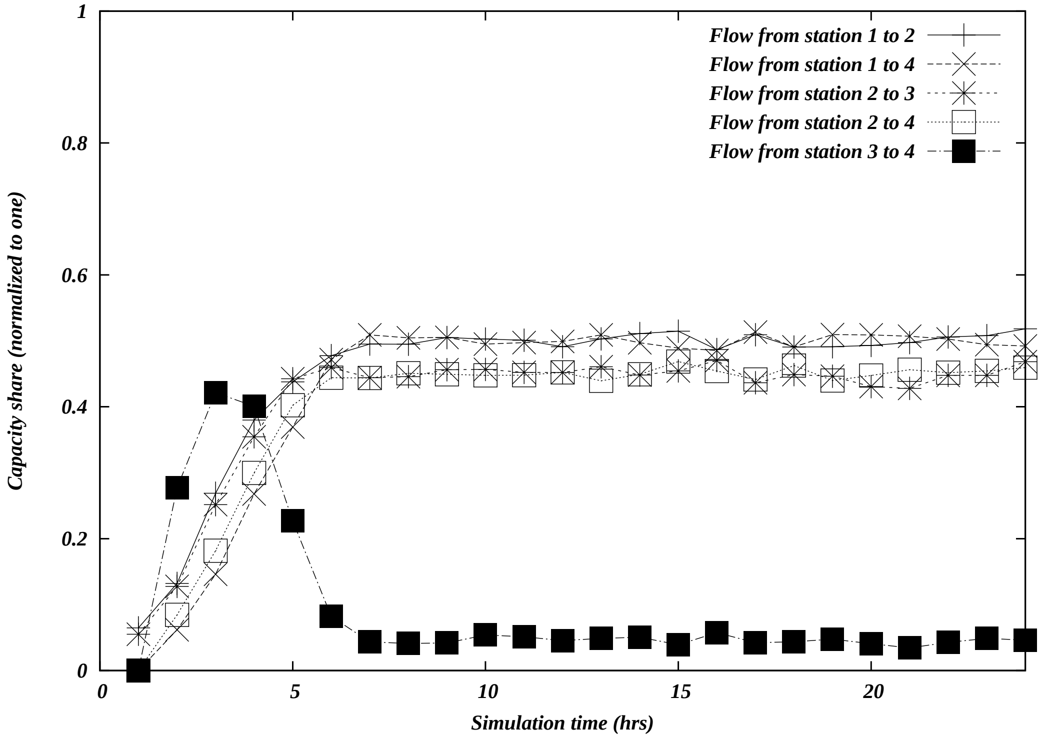

Figure 5.9 Capacity share allocated to each flow when using FCFS.

This section examines the fairness and data loss performance of FCFS. Figure 5.9 and 5.10 shows the capacity shares and data loss rates of flow $f^{1\to2}$, $f^{1\to4}$, $f^{2\to3}$, $f^{2\to4}$, and $f^{3\to4}$ over a 24 hour period. Form Figure 5.9, we see that FCFS is not fair to station 3, and cause data starvation at station 3. Specifically, between the hours 7 to 24, $f^{1\to2}$, $f^{1\to4}$, $f^{2\to3}$, and $f^{2\to4}$ are allocated approximately 0.5 of the hard disk capacity, while the flow $f^{3\to4}$ is given

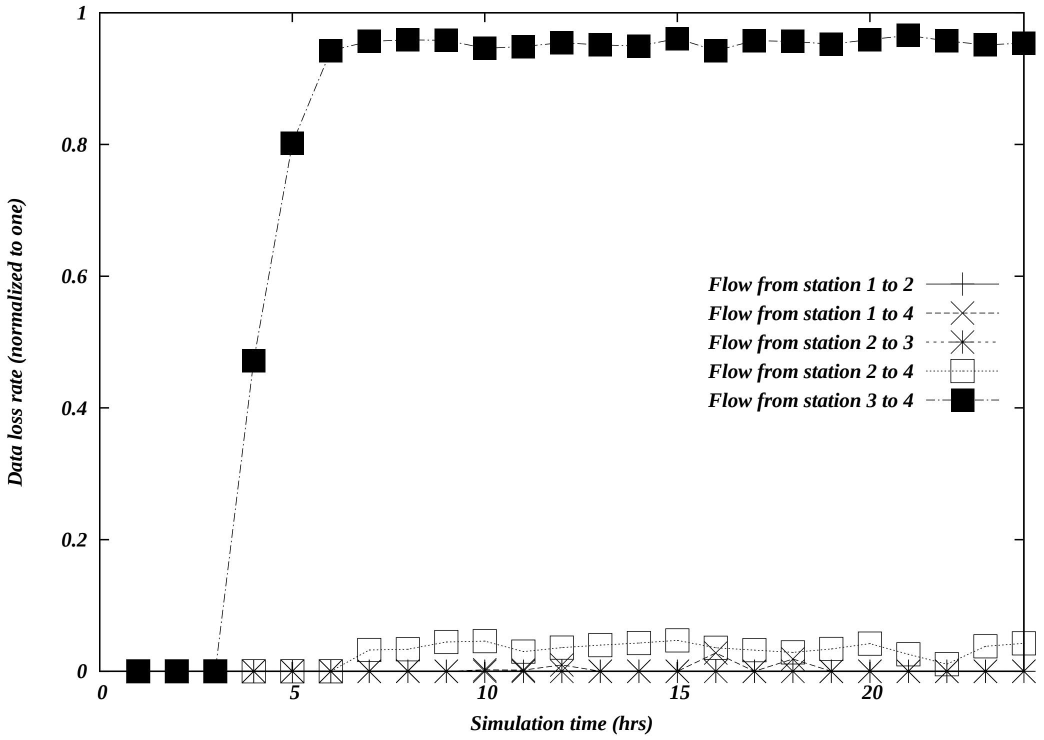

Figure 5.10 Data loss rate per flow for the FCFS algorithm.

no share. As a result, FCFS only has a Jain's fairness index of 0.8. Figure 5.10 confirms that during the same period, the traffic going from station 3 → 4 is completely lost. This happens because FCFS serves messages according to their arrival times. Hence, messages that originate from station 1 and 2 have a higher priority than the ones transmitted by station 3. After hour 7, station 1 and 2 consumes the entire space on $H^{T\to 4}$ by filling $\frac{1}{2}$ of hard disk $H^{1\to T}$ and $H^{2\to T}$ with messages for station 4. Thus, messages loaded onto train by station 3 are discarded by FCFS.

5.6.2 Local Max-Min Fair

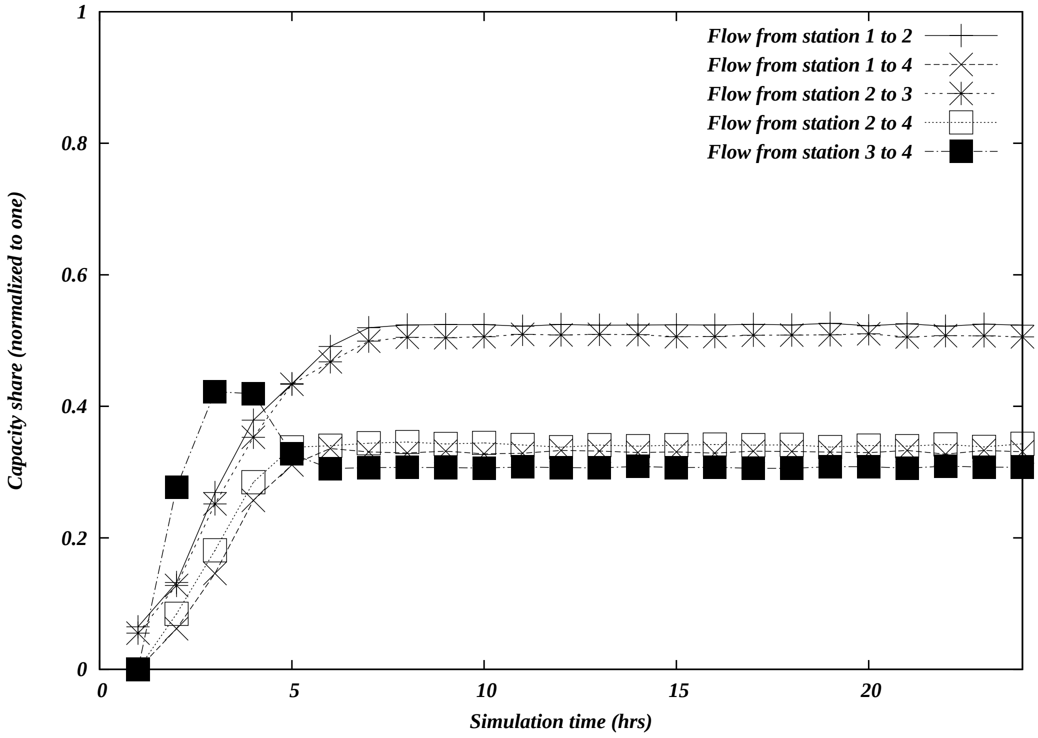

Figure 5.11 Capacity share allocated to each flow when using LMMF.

This section investigates the behavior of LMMF in terms of fairness and data loss rate. Figure 5.11 presents the capacity shares achieved by flow $f^{1\to 2}$, $f^{1\to 4}$, $f^{2\to 3}$, $f^{2\to 4}$, and $f^{3\to 4}$ over a 24 hour period. The graph indicates that between the hours 7 to 24, $f^{1\to 2}$, $f^{1\to 4}$, $f^{2\to 3}$, $f^{2\to 4}$, and $f^{3\to 4}$ are allocated $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{2}$, $\frac{1}{3}$, and $\frac{1}{3}$ of the hard disk capacity, respectively. Therefore, LMMF has

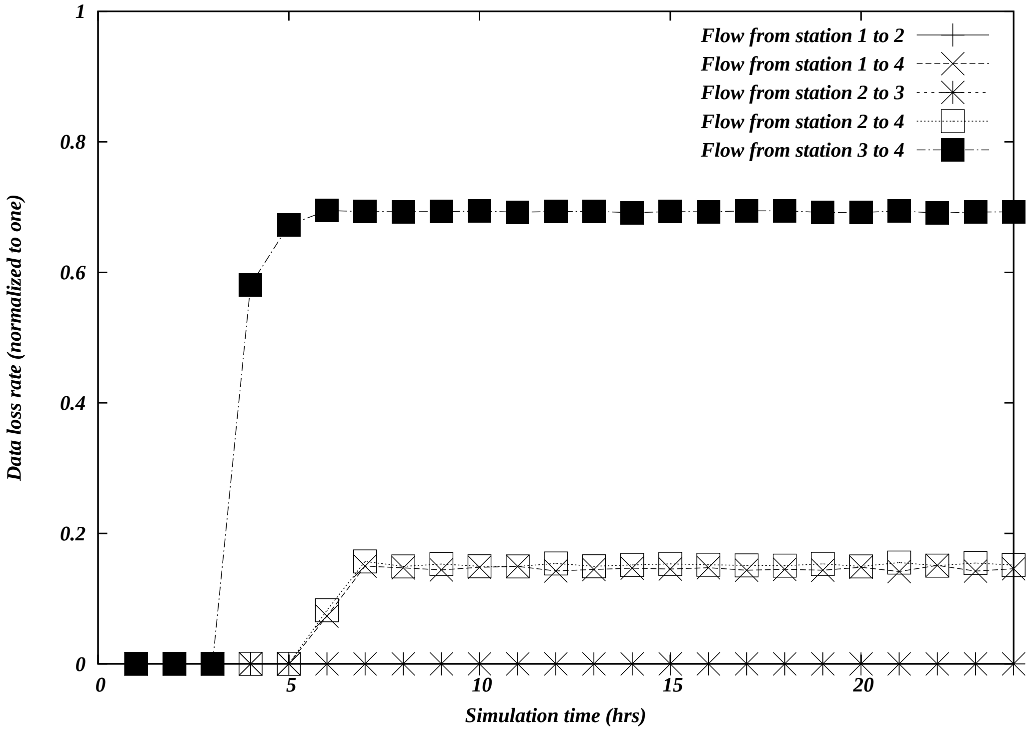

Figure 5.12 Data loss rate per flow for the LMMF algorithm.

a Jain's fairness index of 0.96. After hour 7, the space on hard disk $H^{T\to 4}$ is evenly divided among $f^{1\to 4}, f^{2\to 4}$, and $f^{3\to 4}$. Hence, these flows have an equal capacity share of $\frac{1}{3}$. During the same period, the capacity share of $f^{1\to 2}$ and $f^{2\to 3}$ is 0.5 because $f^{1\to 2}$ competes with $f^{1\to 4}$ for the space on $H^{1\to T}$ and $f^{2\to 3}$ competes with $f^{2\to 4}$ for the space on $H^{2\to T}$.

Figure 5.12 shows the data loss rates caused by LMMF over a 24 hour period. LMMF divides the hard disk $H^{1\to T}$ and $H^{2\to T}$ evenly among flow $f^{1\to 2}$ and $f^{2\to 3}$, respectively. No competition exists for the space on hard disk $H^{T\to 2}$ and $H^{T\to 3}$. Therefore, the data loss rate for the flow $f^{1\to 2}$ and $f^{2\to 3}$ is always zero; see Figure 5.12. On the other hand, the hard disk $H^{T\to 4}$ is subject to competition between flow $f^{1\to 4}, f^{2\to 4}$, and $f^{3\to 4}$, each of which demands 0.5, 0.5, and 1 of the hard disk space respectively. LMMF allocates the same $\frac{1}{3}$ share of hard disk $H^{T\to 4}$ to each flow. Thus, the data loss rate of $f^{1\to 4}, f^{2\to 4}$, and $f^{3\to 4}$ is $\frac{1}{2}-\frac{1}{3}, \frac{1}{2}-\frac{1}{3}$, and $1-\frac{1}{3}$ respectively.

5.6.3 Global Max-Min Fair

Figure 5.13 and 5.14 shows the data loss rates and capacity shares of flow $f^{1\to 2}, f^{1\to 4}, f^{2\to 3}, f^{2\to 4}$, and $f^{3\to 4}$ over a 24 hour period. As shown in Figure 5.13, GMMF completely avoids data loss. Figure 5.14 indicates that after hour 7, the share of flow $f^{1\to 2}, f^{1\to 4}, f^{2\to 3}, f^{2\to 4}$, and $f^{3\to 4}$ is $\frac{2}{3}, \frac{1}{3}, \frac{2}{3}, \frac{1}{3}$, and $\frac{1}{3}$, respectively. This results in a Jain's fairness index of 0.8909; i.e., GMMF is less fair than LMMF. GMMF divides hard disk $H^{T\to 4}$ evenly amongst flow $f^{1\to 4}, f^{2\to 4}$, and $f^{3\to 4}$, and thereby resulting in $f^{1\to 4}, f^{2\to 4}$, and $f^{3\to 4}$ receiving $\frac{1}{3}$ of hard disk $H^{1\to T}, H^{2\to T}$, and $H^{3\to T}$, respectively. This leaves $\frac{2}{3}$ of hard disk $H^{1\to T}$ and $H^{2\to T}$ to flow $f^{1\to 2}$ and $f^{2\to 3}$, each of which is entitled to the entire space on $H^{T\to 2}$ and $H^{T\to 3}$ respectively.

5.6.4 Weighted Global Max-Min Fair

Figure 5.15 and 5.16 illustrates the data loss rates and capacity shares of flow $f^{1\to 2}, f^{1\to 4}, f^{2\to 3}, f^{2\to 4}$, and $f^{3\to 4}$. Figure 5.15 shows WGMMF results in no data loss. Figure 5.16 shows that after the hour 7, the share of flow

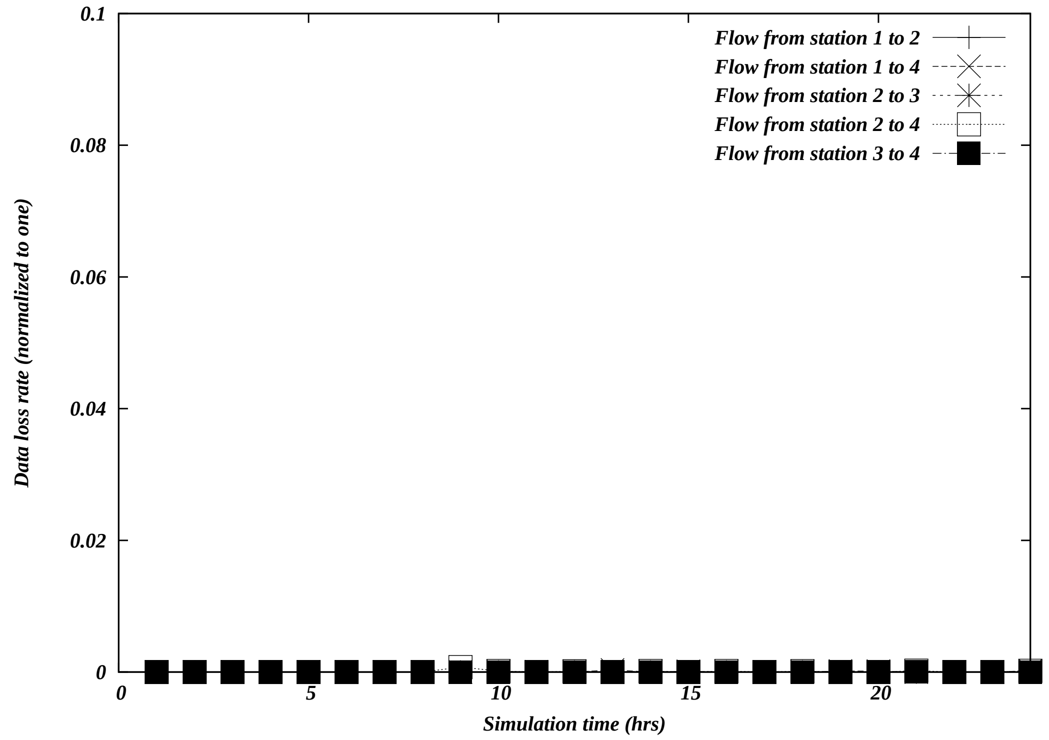

Figure 5.13 Data loss rate per flow for the GMMF algorithm.

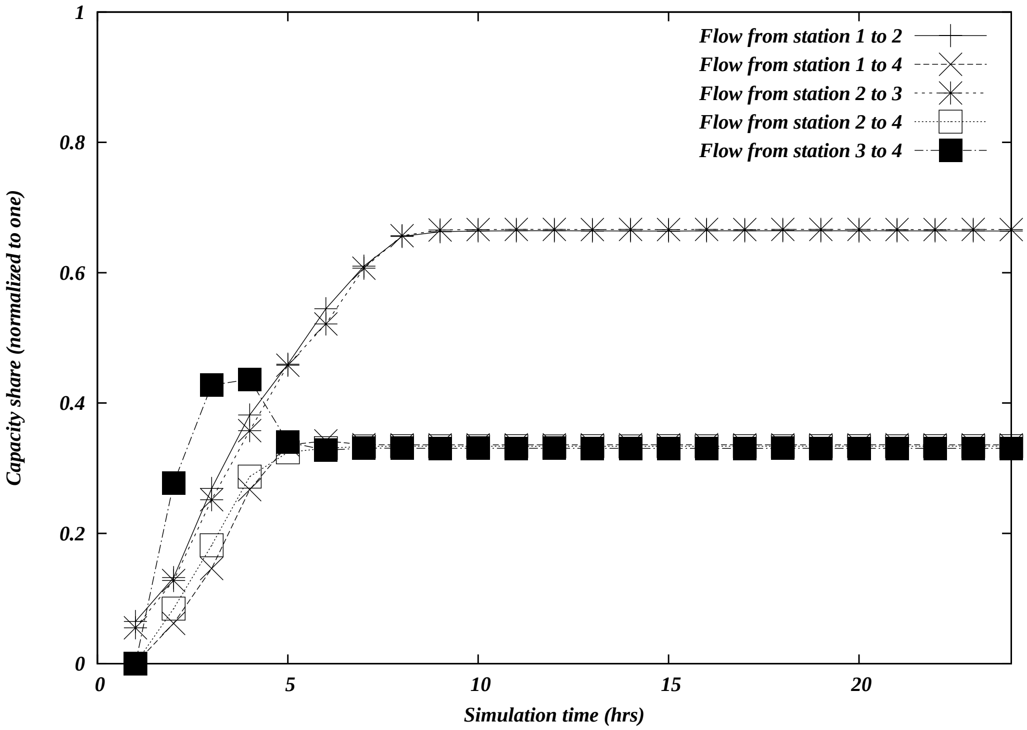

Figure 5.14 Capacity share allocated to each flow when using GMMF.

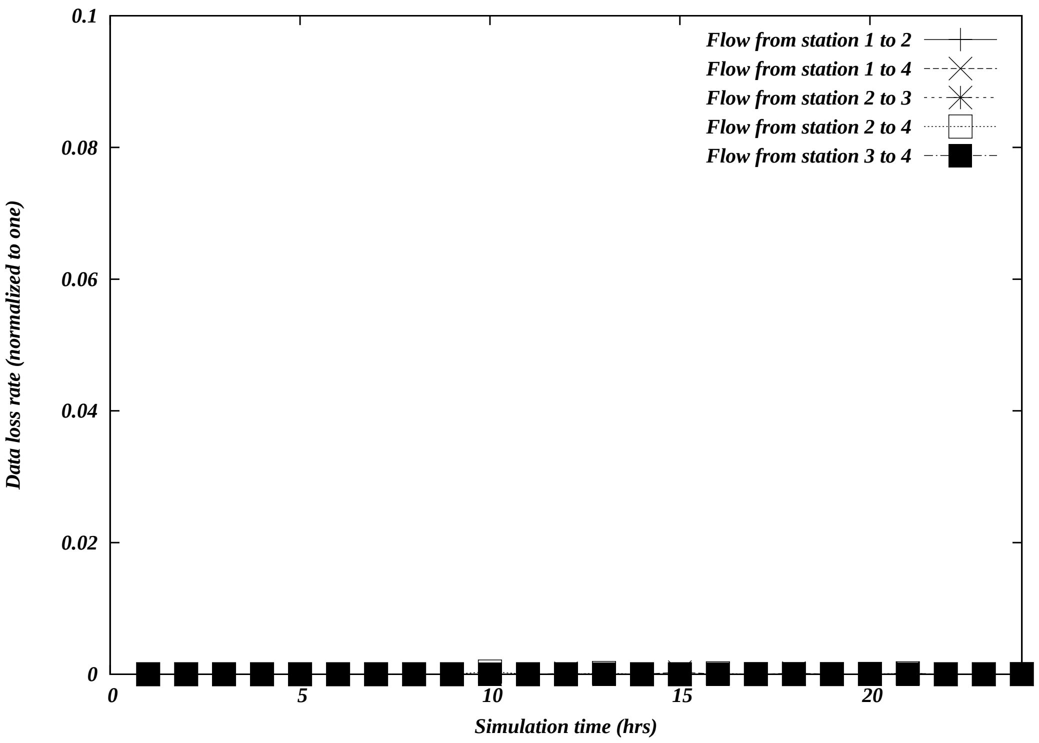

Figure 5.15 Data loss rate per flow for the WGMMF algorithm.

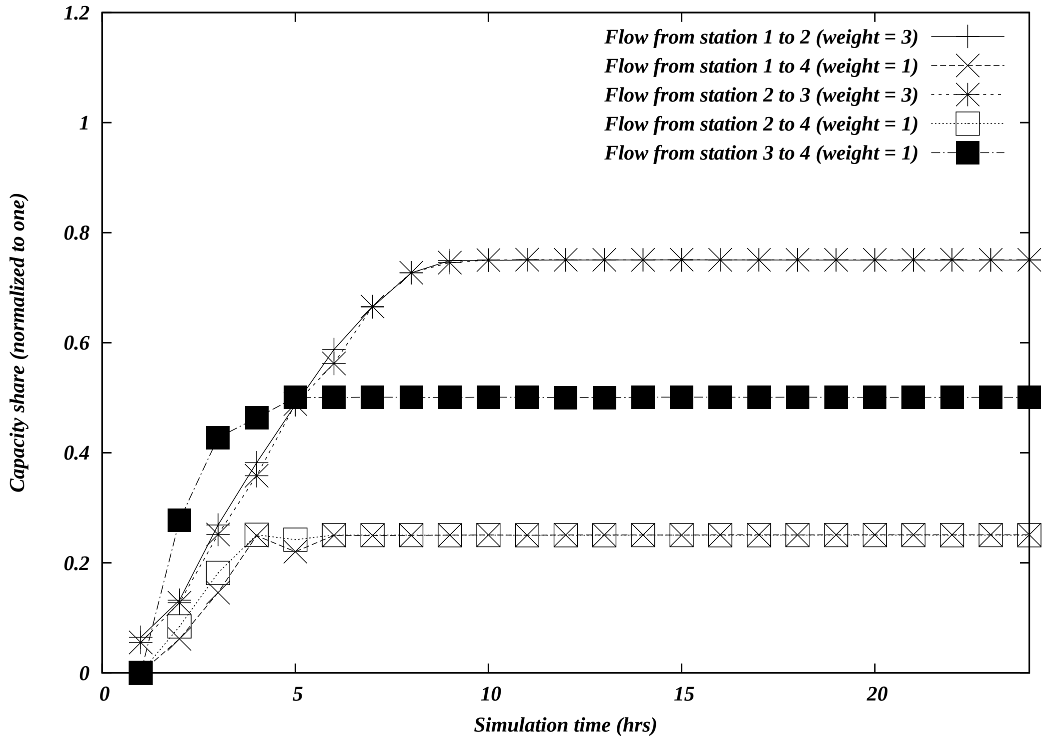

Figure 5.16 Capacity share allocated to each flow when using WGMMF.

$f^{1\to2}, f^{1\to4}, f^{2\to3}, f^{2\to4}$, and $f^{3\to4}$ is $\frac{3}{4}, \frac{1}{4}, \frac{3}{4}, \frac{1}{4}$, and $\frac{2}{4}$, respectively. Hence, WGMFF has a Jain's fairness index of 0.8333. This means WGMMF is less fair as compared to LMMF and GMMF.

5.6.5 Comparison of All Algorithms

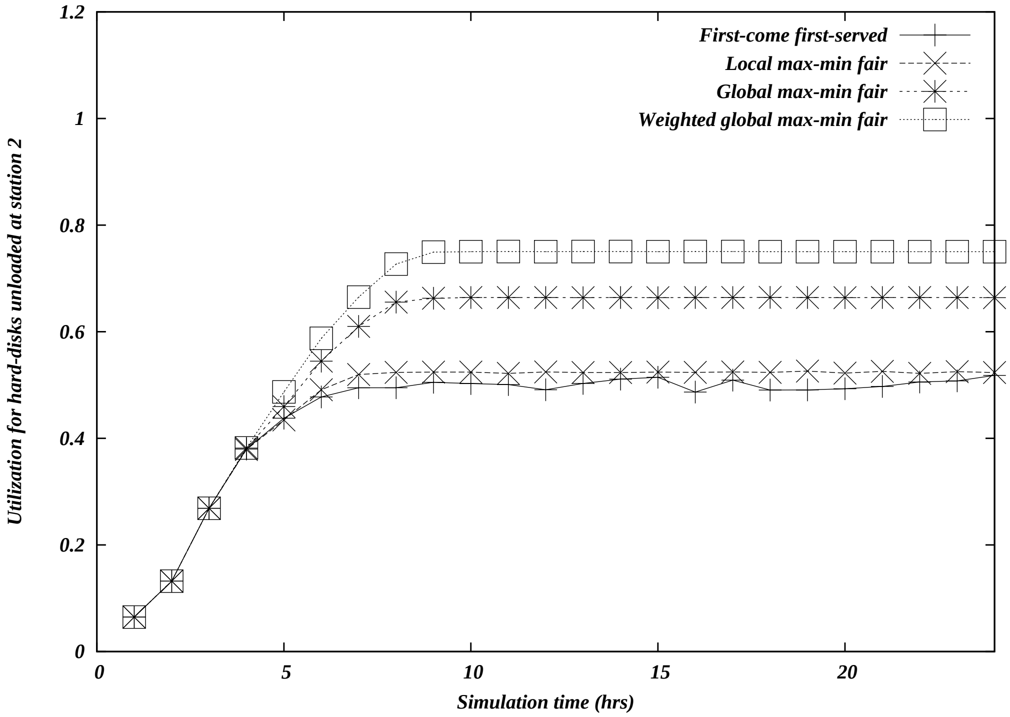

Figure 5.17 Utilization of hard disk $H^{T\to2}$ for all algorithms.

This section compares the hard disk utilization of all the proposed algorithms. Figure 5.17 to 5.19 shows the hard disk utilization of $H^{T\to2}, H^{T\to3}$, and $H^{T\to4}$, respectively. Here, hard disk utilization achieved by FCFS is used as a baseline for comparison purposes. The figures indicate that all variants of max-min fair algorithm have better hard disk utilization as compared to FCFS. WGMMF has the highest utilization and LMMF has the lowest utilization.

Figure 5.17 plots the utilization of hard disk $H^{T\to2}$ over a 24 hour period. Between the hours 7 to 24, the utilization achieved by WGMMF, GMMF, and LMMF algorithms is $\frac{3}{4}$, $\frac{2}{3}$, and $\frac{1}{2}$, respectively. Flow $f^{1\to2}$ is entitled to the entire

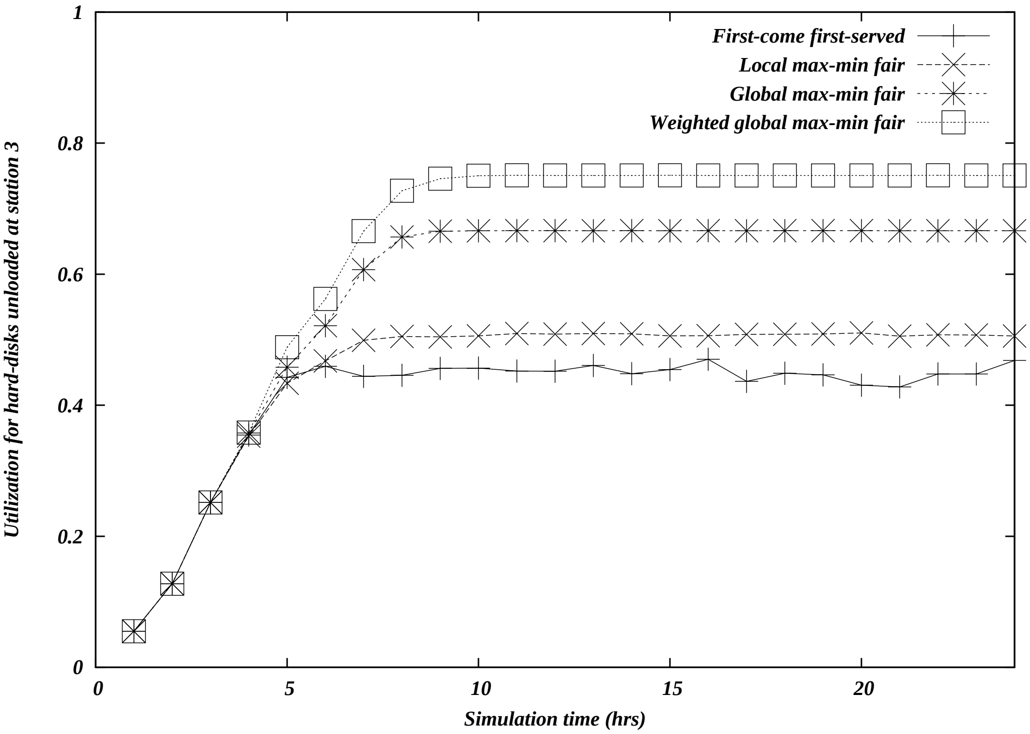

Figure 5.18 Utilization of hard disk $H^{T\to 3}$ for all algorithms.

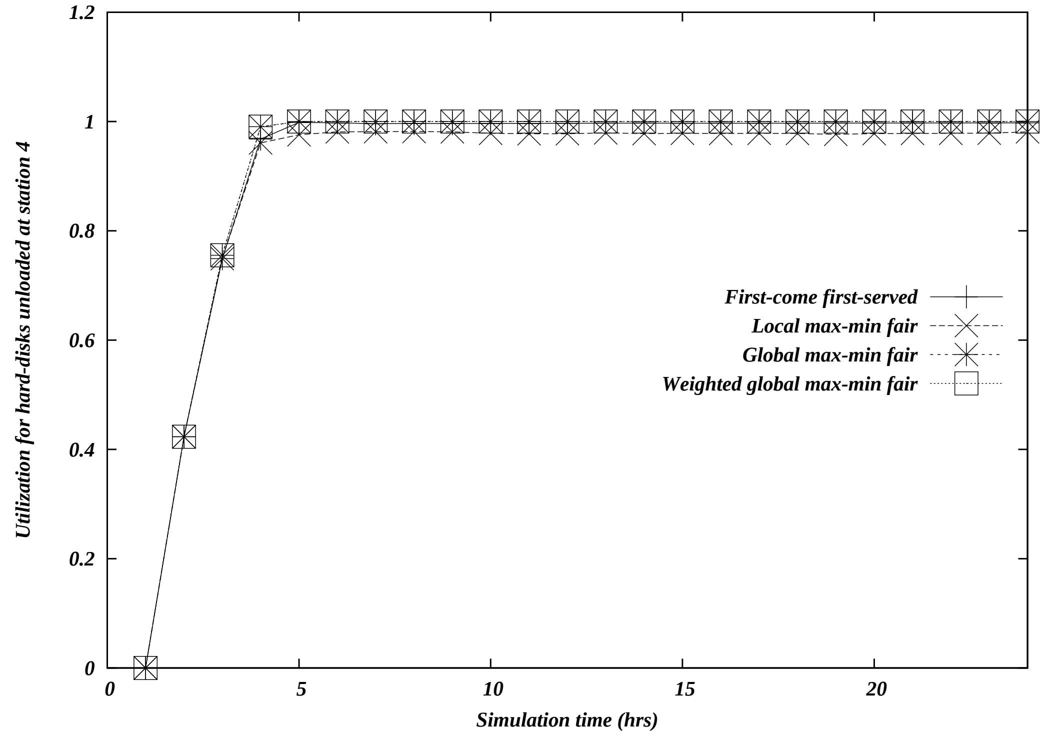

Figure 5.19 Utilization of hard disk $H^{T \to 4}$ for all algorithms.

space on $H^{T\rightarrow 2}$, while $f^{1\rightarrow 2}$ has to compete with $f^{1\rightarrow 4}$ for the space on hard disk $H^{1\rightarrow T}$. Thus, the utilization achieved by each algorithm is equivalent to the share of flow $f^{1\rightarrow 2}$ on hard disk $H^{1\rightarrow T}$. LMMF divides the hard disk $H^{1\rightarrow T}$ evenly, and thereby resulting in the flow $f^{1\rightarrow 2}$ receiving $\frac{1}{2}$ of hard disk $H^{T\rightarrow 2}$. Using LMMF, flow $f^{1\rightarrow 4}$ is subject to the data loss rate of $\frac{1}{6}$ because this flow receives $\frac{1}{3}$ of hard disk $H^{T\rightarrow 4}$. GMMF and WGMMF eliminate this data loss by allocating equal shares of hard disk $H^{1\rightarrow T}$ and $H^{T\rightarrow 4}$ to flow $f^{1\rightarrow 4}$. GMMF allocates $\frac{2}{3}$ and $\frac{1}{3}$ of hard disk $H^{1\rightarrow T}$ to the flow $f^{1\rightarrow 2}$ and $f^{1\rightarrow 4}$, respectively. Hence, the utilization achieved on hard disk $H^{T\rightarrow 2}$ by the GMMF algorithm is $\frac{2}{3}$. The WGMMF permits the flow $f^{1\rightarrow 4}$ to use $\frac{1}{4}$ of hard disk $H^{1\rightarrow T}$. Thus, the flow $f^{1\rightarrow 2}$ obtains $\frac{3}{4}$ of hard disk $H^{1\rightarrow T}$ and $H^{T\rightarrow 2}$.

Figure 5.18 plots the simulation time versus the utilization rate of hard disk $H^{T\rightarrow 3}$. After hour 7, WGMMF, GMMF, and LMMF utilize $\frac{3}{4}$, $\frac{2}{3}$, and $\frac{1}{2}$ of hard disk $H^{T\rightarrow 3}$, respectively. The rational behind how each algorithm utilizes hard disk $H^{T\rightarrow 3}$ is similar to that explained for each algorithm utilizing hard disk $H^{T\rightarrow 2}$. Again, WGMMF has the highest utilization and LMMF has the lowest utilization.

Figure 5.19 shows the utilization of hard disk $H^{T\rightarrow 4}$ over a 24 hour period. The figure indicates that all algorithms fully utilize the hard disk during the hours 7 to 24. LMMF and GMMF divide hard disk $H^{T\rightarrow 4}$ evenly amongst flow $f^{1\rightarrow 4}$, $f^{2\rightarrow 4}$, and $f^{3\rightarrow 4}$, resulting in each flow to receive $\frac{1}{3}$ of hard disk $H^{T\rightarrow 4}$. Using WGMMF, flow $f^{1\rightarrow 4}$,$f^{2\rightarrow 4}$, and $f^{3\rightarrow 4}$ receives $\frac{1}{4}$,$\frac{1}{4}$, and $\frac{2}{4}$ of hard disk $H^{T\rightarrow 4}$, respectively.

5.6.6 Throughput and Delay

This section compares the mean aggregate delay and averaged aggregate throughput achieved by all algorithms. Figure 5.20 plots the simulation time against the aggregate throughput attained by the algorithms. The figure indicates that LMMF, GMMF, and WGMMF have better throughput than FCFS. Between the hours 43 and 50, the throughput values achieved by WGMMF, GMMF, and LMMF are 1400, 1300, and 1100 Mbps, respectively. These values

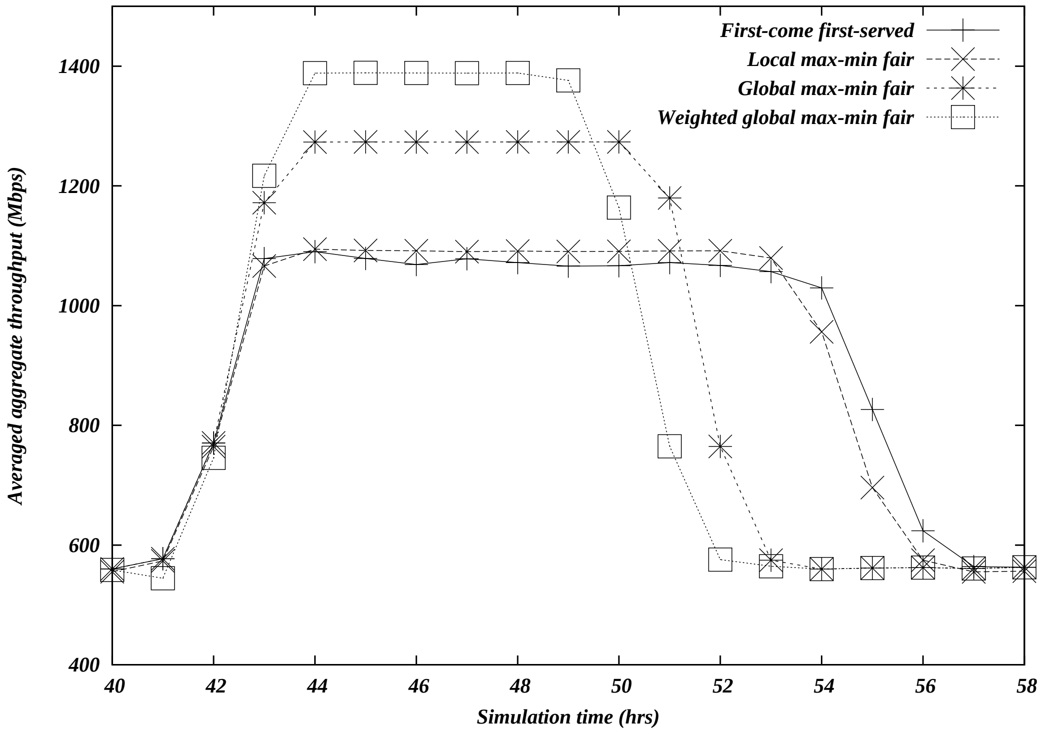

Figure 5.20 Averaged aggregate throughput achieved by all algorithms.

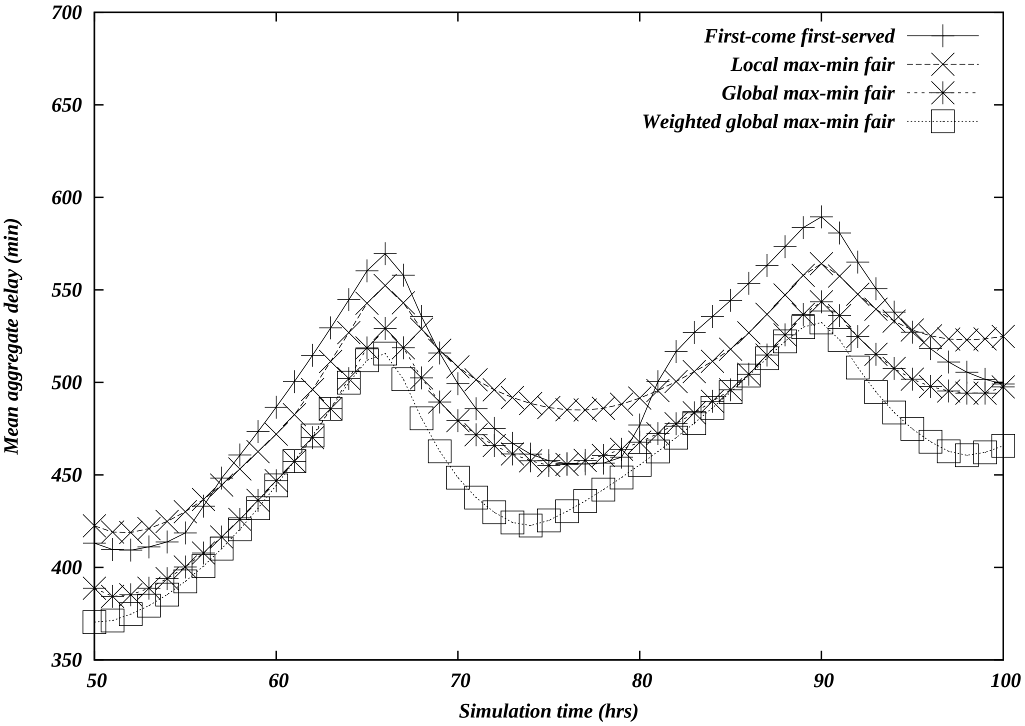

Figure 5.21 Mean aggregate delay achieved by all algorithms.

confirm the results in Section 5.6.5. WGMMF avoids data loss, and it utilizes hard disk $H^{T\to 2}$ to $H^{T\to 4}$ more efficiently than the other algorithms. Thus, WGMMF has the highest rate of successful message delivery among the proposed variants.

Figure 5.21 shows the mean aggregate delay incurred by the four algorithms over a 50 hour period. The graph experiences a periodic growth every 24 hours at 8 pm. This growth is driven by the peak hour arrival rate described in Section 5.4. Between the hours 60 and 90, WGMMF, GMMF, LMMF have a mean aggregate delay of 530, 545, and 565 minutes, respectively. Figure 5.21 indicates that WGMMF has the shortest mean delay amongst the proposed variants. GMMF and WGMMF have shorter mean delay than LMMF because they avoid data loss and utilize hard disk $H^{T\to 2}$ to $H^{T\to 4}$ more efficiently. On the other hand, WGMMF gives a higher priority to a flow, if it uses a hard disk with less utilization , thereby allowing it to have a shorter mean delay as compared to GMMF.

5.7 Conclusion

This chapter presents a process based simulation model of TrainNet which is then used to implement a TrainNet simulator in DESMO-J. This model is used to evaluate FCFS, LMMF, GMMF, and WGMMF, in terms of fairness, data loss performance, and hard disk utilization.

The max-min scheduling algorithms addresses the resource scheduling problem defined in Chapter 4. Jain's fairness index shows that LMMF is the fairest algorithm among max-min variants. However, LMMF results in considerable data loss. GMMF is less fair than LMMF. Yet, it prevents data loss and achieves higher hard disk utilization, shorter mean delay, and higher average throughout than LMMF. WGMMF is less fair than GMMF. However, it avoids data loss and achieves a better performance in terms of delay, throughput, and hard disk utilization.

Chapter 6

Conclusion

This thesis presents TrainNet, a novel transport system that uses trains equipped with hard disks. In particular, the proposed system has the following characteristics:

- Trains travel according to a predefined time table.
- TrainNet has three types of DTN node: source station, destination station, and train.
- TrainNet exchanges data with conventional networks via POP connected to source and destination stations.
- TrainNet exploits the falling cost of hard disks, and thereby is capable of providing virtually unlimited transport capacity.
- Portable hard disks are used to exchange data between a train and a station.
- Stations are connected to a global dispatcher that runs a scheduling algorithm to arbitrate the hard disk space among competing source stations.

TrainNet addresses two key challenges in conventional networks. Firstly, it provides a low cost, high bandwidth link that is capable of delivering video contents from VSPs to CATV head ends, thereby allowing CATV operators to meet the demands of VoD traffic. Secondly, TrainNet allows remote regions connected by a railway network to gain broadband access to the Internet.

TrainNet has several advantages over conventional data transport systems. First, TrainNet is affordable and scalable, meaning its capacity can easily and inexpensively be increased by adding new hard disks. Moreover, CATV operators can use TrainNet to offer VoD services without violating ICT regulations that require operators to price services equally in urban and non urban areas.

This thesis is the first to propose a challenged network that uses trains, and has considered fundamental design principles required to realize TrainNet. However, several issues remain. Some of which are as follows:

- The thesis assumes data are never stored at intermediate stations before being forwarded to its destination at a later time. An immediate future work is to relax this assumption, and develop a routing algorithm that allows intermediate stations to accept custody of bundle messages.
- Currently, TrainNet does not support inter-train communications, where trains are equipped with a wireless access point that allows them to exchange messages as they pass each other. This is particularly useful for carrying feedback signal to/from stations. In addition, trains can be used to collect and deliver data from/to Internet kiosks located near a rail line.
- This thesis has only considered trains travelling on a single railway track. Another immediate future work is to consider multiple railway tracks that intersect each other. For example, at a central station with trains arriving and departing on different lines. In such a scenario, it is important to study the impact of varying hard disk capacities, train schedules, and the topology of the rail network on fairness, throughput, and data loss.
- GMMF and WGMMF rely on a global dispatcher for traffic information at each station. However, the possibility of developing distributed versions of GMMF and WGMMF remains an open issue. This is particularly important when a global dispatcher is not available.

Bibliography

[1] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss, "Delay-tolerant networking: an approach to interplanetary internet," *IEEE Communications Magazine*, vol. 41, no. 6, pp. 128–136, 2003.

[2] I. Akyildiz, Ö. Akan, C. Chen, J. Fang, and W. Su, "Interplanetary internet: state-of-the-art and research challenges," *Computer Networks*, vol. 43, no. 2, pp. 75–112, 2003.

[3] Cisco, "Approaching the zettabyte era," White Paper, June 2008.

[4] G. Karlsson, V. Lenders, and M. May, "Delay-tolerant broadcasting," in *Proceedings of the 2006 SIGCOMM workshop on Challenged networks*. ACM New York, NY, USA, 2006, pp. 197–204.

[5] S. Farrell, V. Cahill, D. Geraghty, I. Humphreys, and P. McDonald, "When tcp breaks: Delay-and disruption-tolerant networking," *IEEE Internet Computing*, vol. 10, no. 4, pp. 72–78, 2006.

[6] L. Wood, W. Eddy, W. Ivancic, J. McKim, and C. Jackson, "Saratoga: a Delay-Tolerant Networking convergence layer with efficient link utilization," in *Third International Workshop on Satellite and Space Communications (IWSSC07)*, 2007.

[7] K. Harras, M. Wittie, K. Almeroth, and E. Belding, "ParaNets: A Parallel Network Architecture for Challenged Networks," in *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications, IEEE Computer Society Press, Los Alamitos*, 2007.

[8] C. Bisdikian, K. Maruyama, D. Seidman, D. Serpanos, I. Center, and Y. Heights, "Cable access beyond the hype: on residential broadband dataservices over HFC networks," *IEEE Communications Magazine*, vol. 34, no. 11, pp. 128–135, 1996.

[9] R. Gupta and H. Pirkul, "Hybrid fiber co-axial CATV network design with variable capacity optical network units," *European Journal of Operational Research*, vol. 123, no. 1, pp. 73–85, 2000.

[10] J. Nussbaumer, B. Patel, F. Schaffa, I. Center, and Y. Heights, "Multimedia delivery on demand: capacity analysis and implications," in *Local Computer Networks, 1994. Proceedings., 19th Conference on*, 1994, pp. 380–386.

[11] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications.* ACM New York, NY, USA, 2003, pp. 27–34.

[12] O. Mukhtar, "Design and implementation of bundle protocol stack for delay-tolerant networking," Master's thesis, Helsinki University of Technology, 2006.

[13] A. Pentland, R. Fletcher, and A. Hasson, "Daknet: Rethinking connectivity in developing nations," *Computer*, vol. 37, no. 1, pp. 78–83, 2004.

[14] A. Rabagliati, "Wizzy digital courier," Online, April 2008, `http://www.wizzy.org.za/`.

[15] U. Berkeley, "TIER Project," Online, April 2008, `http://tier.cs.berkeley.edu/`.

[16] S. Guo, M. Falaki, E. Oliver, S. Rahman, A. Seth, M. Zaharia, U. Ismail, and S. Keshav, "Design and implementation of the kiosknet system," in *Proceedings of the International Conference on Information and Communication Technologies and Development (ICTD 2007)*, 2007, pp. 300–309.

[17] G. Sollazzo, M. Musolesi, and C. Mascolo, “TACO-DTN: a time-aware content-based dissemination system for delay tolerant networks,” in *Proceedings of the 1st international MobiSys workshop on Mobile opportunistic networking.* ACM New York, NY, USA, 2007, pp. 83–90.

[18] K. Hanna, B. Levine, and R. Manmatha, “Mobile distributed information retrieval for highly-partitioned networks,” in *11th IEEE International Conference on Network Protocols, 2003. Proceedings*, 2003, pp. 38–47.

[19] R. Nichols and A. Hammons, “Performance of DTN-Based Free-Space Optical Networks with Mobility,” in *IEEE Military Communications Conference, 2007. MILCOM 2007*, 2007, pp. 1–6.

[20] R. Krishnan, P. Basu, J. Mikkelson, C. Small, R. Ramanathan, D. Brown, J. Burgess, A. Caro, M. Condell, N. Goffee *et al.*, “The spindle disruption-tolerant networking system,” in *IEEE Military Communications Conference, 2007. MILCOM 2007*, 2007, pp. 1–7.

[21] M. Corner, E. Berger, and B. Levine, “UMass TurtleNet,” Online, October 2008, `http://prisms.cs.umass.edu/dome/turtlenet`.

[22] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein, “Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet,” *COMPUTER ARCHITECTURE NEWS*, vol. 30, no. 5, pp. 96–107, 2002.

[23] M. Weik, *Computer science and communications dictionary.* Kluwer Academic Publishers, 2000.

[24] I. I. S. I. Group, “InterPlanetary Internet (IPN),” Online, April 2008, `http://www.ipnsig.org/`.

[25] Cisco, “Visual networking index - forecast and methodology, 2007-2012,” White Paper, June 2008.

[26] M. Hofmann and L. Beaumont, *Content Networking: Architecture, Protocols, and Practice.* Morgan Kaufmann, 2005.

[27] N. Wang and G. Pavlou, "Traffic engineered multicast content delivery without MPLS overlay," *IEEE Transactions on Multimedia*, vol. 9, no. 3, pp. 619–628, 2007.

[28] C. Sarrocco, "Improving IP connectivity in the least developed countries: breaking the vicious circle," *Info Cambridge Camford*, vol. 4, no. 3, pp. 14–28, 2002.

[29] W. Wilde and P. Swatman, "Toward virtual communities in rural Australia," *International Journal of Electronic Commerce*, vol. 2, pp. 43–60, 1997.

[30] R. T. I. (Australia) and R. Estens, *Connecting Regional Australia: The Report of the Regional Telecommunications Inquiry.* Commonwealth Department of Communications, Information Technology and the Arts, 2002.

[31] W. Policies, "The development of broadband access in rural and remote areas," 2004.

[32] G. Goggin, "Rural Communities Online: Networking to link Consumers to Providers," *A research project commissioned by the Telstra Consumer Consultative Council. Centre for Critical and Cultural Studies, University of Queensland, Brisbane*, 2003.

[33] D. Comer, *Internetworking with TCP/IP vol. 1: Principles, Protocols, and Architecture.* Prenctice Hall, 1995, vol. 1.

[34] J. Ott and D. Kutscher, "Why seamless? Towards exploiting WLAN-based intermittent connectivity on the road," in *Proceedings of the TERENA Networking Conference, TNC 2004, Rhodes*, 2004.

[35] L. Wood, C. Peoples, G. Parr, B. Scotney, and A. Moore, "TCPs protocol radius: the distance where timers prevent communication," in *Third International Workshop on Satellite and Space Communications (IWSSC07)*, 2007.

[36] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, “Modeling TCP throughput: A simple model and its empirical validation,” *ACM SIGCOMM Computer Communication Review*, vol. 28, no. 4, pp. 303–314, 1998.

[37] S. Patil and M. Kumar, “TCP Enhancement Using Active Network Based Proxy Transport Service,” *Lecture notes in computer science*, pp. 103–114, 2004.

[38] S. Jain, K. Fall, and R. Patra, “Routing in a delay tolerant network,” *SIGCOMM*, Aug/Sep 2004.

[39] K. Fall and S. Farrell, “DTN: an architectural retrospective,” *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 5, pp. 828–836, 2008.

[40] R. Srinivasan, “RFC 1831: Remote Procedure Call Protocol,” August 1995.

[41] I. Sun Microsystems, “Remote method invocation,” November 1998, `http://java.sun.com/products/jdk/1.2/docs/guide/rmi`.

[42] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RFC 1889: A transport protocol for real-time applications,” January 1996.

[43] Y. Rekhter, T. Li, S. Hares *et al.*, “RFC 1771: A border gateway protocol,” March 2003.

[44] J. Moy *et al.*, “RFC 2328: Open Shortest Path First,” April 1998.

[45] D. Oran, “RFC1142: OSI IS-IS Intra-Domain Routing Protocol,” February 1990.

[46] B. Albrightson and J. Boyle, “EIGRP–A fast routing protocol based on distance vectors,” *Proc. Networld/Interop 94, Las Vegas*, 1994.

[47] D. Velenis, D. Kalogeras, and B. Maglaris, “SaTPEP: A TCP performance enhancing proxy for satellite links,” *Lecture notes in computer science*, pp. 1233–1238, 2002.

[48] C. Caini, R. Firrincieli, and D. Lacamera, "PEPsal: a Performance Enhancing Proxy for TCP satellite connections," *IEEE Aerospace and Electronic Systems Magazine*, vol. 22, no. 8, pp. 7–16, 2007.

[49] M. Meyer, J. Sachs, and M. Holzke, "Performance evaluation of a TCP proxy in WCDMA networks," *IEEE Wireless Communications*, vol. 10, no. 5, pp. 70–79, 2003.

[50] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, "RFC3135: Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations," *Internet RFCs*, 2001.

[51] B. Bakshi, N. Krishna, D. Padhan, and N. Vaidya, "A comparison of mechanism for improving TCP performance over wireless links," in *Proceedings of ACM SIGCOMM*, 1996.

[52] J. Ott and D. Kutscher, "Drive-thru Internet: IEEE 802.11 b for" automobile" users," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 1, 2004.

[53] E. Huang, W. Hu, J. Crowcroft, and I. Wassell, "Towards commercial mobile ad hoc network applications: a radio dispatch system," in *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing.* ACM New York, NY, USA, 2005, pp. 355–365.

[54] X. Zhang, J. Kurose, B. Levine, D. Towsley, and H. Zhang, "Study of a bus-based disruption-tolerant network: mobility modeling and impact on routing," in *Proceedings of the 13th annual ACM international conference on Mobile computing and networking.* ACM New York, NY, USA, 2007, pp. 195–206.

[55] B. Burns, O. Brock, and B. Levine, "MV routing and capacity building in disruption tolerant networks," in *Proceedings IEEE INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 1, 2005.

[56] B. Burns, O. Brock, B. Levine, and M. Amherst, "Autonomous enhancement of disruption tolerant networks," in *Proc. 2006 IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2006, pp. 2105–2110.

[57] W. Zhao, M. Ammar, and E. Zegura, "Controlling the mobility of multiple data transport ferries in a delay-tolerant network," in *Proceedings IEEE INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, 2005.

[58] L. Wood, W. Eddy, and P. Holliday, "A bundle of problems," in *IEEE Aerospace Conference.* Monata: Big Sky, March 2009.

[59] T. Berners-Lee, R. Fielding, and L. Masinter, "RFC2396: Uniform Resource Identifiers ," April 1998.

[60] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson, "RFC2960: Stream Control Transmission Protocol," October 2000.

[61] D. Nowitz, *Uucp implementation description.* Bell Laboratories. In UNIX Programmers Manual,, 1978.

[62] W. Hsu and S. Lang, "Classification algorithms for NETNEWS articles," in *Proceedings of the eighth international conference on Information and knowledge management.* ACM New York, NY, USA, 1999, pp. 114–121.

[63] R. Bush, "FidoNet: technology, tools, and history," *Communications of the ACM*, vol. 36, no. 8, pp. 31–35, 1993.

[64] D.-T. N. R. Group, "Dtn2:the dtnrg reference implementation," Online, November 2008, `http://www.dtnrg.org/wiki/Code`.

[65] Y. Tseng, S. Wu, W. Liao, and C. Chao, "Location awareness in ad hoc wireless mobile networks," *Computer*, vol. 34, no. 6, pp. 46–52, 2001.

[66] N. Banerjee, M. Corner, and B. Levine, "An energy-efficient architecture for DTN throwboxes," in *Proc. IEEE Infocom*, 2007.

[67] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden, "Cartel: a distributed mobile sensor computing system," in *Proceedings of the 4th international conference on Embedded networked sensor systems.* ACM New York, NY, USA, 2006, pp. 125–138.

[68] H. Balakrishnan and K. Chen, "CafNet: a carry-and-forward delay-tolerant network," Master's thesis, Massachusetts Institute of Technology, 2007.

[69] J. Eriksson, H. Balakrishnan, and S. Madden, "Cabernet: A content-delivery network for moving vehicles," in *Proc. ACM Mobicom*, 2008.

[70] Z. Zhang, "Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: Overview and challenges," *IEEE Communications Surveys & Tutorials*, vol. 8, no. 1, pp. 24–37, 2006.

[71] R. Handorean, C. Gill, and G. Roman, "Accommodating transient connectivity in ad hoc and mobile settings," *Lecture Notes in Computer Science*, pp. 305–322, 2004.

[72] S. Merugu, M. Ammar, and E. Zegura, "Routing in space and time in networks with predictable mobility," Georgia Institute of Technology, Tech. Rep., 2004.

[73] H. Thomas, E. Charles, L. Ronald, and S. Clifford, *Introduction To Algorithms.* MIT press, 1989.

[74] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Technical Report CS-200006, Duke University, Tech. Rep., 2000.

[75] C. Becker and G. Schiele, "New Mechanisms for Routing in Ad Hoc Networks," in *Proceeding of the Fouth CaberNet Plenary Workshop*, 2001.

[76] M. Grossglauser and D. Tse, "Mobility increases the capacity of ad hoc wireless networks," *IEEE/ACM Transactions on Networking (ToN)*, vol. 10, no. 4, pp. 477–486, 2002.

[77] T. Small and Z. Haas, “The shared wireless infostation model: a new ad hoc networking paradigm (or where there is a whale, there is a way),” in *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing.* ACM New York, NY, USA, 2003, pp. 233–244.

[78] D. Nain, N. Petigara, and H. Balakrishnan, “Integrated routing and storage for messaging applications in mobile ad hoc networks,” *Mobile Networks and Applications*, vol. 9, no. 6, pp. 595–604, 2004.

[79] F. Tchakountio and R. Ramanathan, “Tracking highly mobile endpoints,” in *Proceedings of the 4th ACM international workshop on Wireless mobile multimedia.* ACM New York, NY, USA, 2001, pp. 83–94.

[80] J. Su, A. Chin, A. Popivanova, A. Goel, and E. De Lara, “User mobility for opportunistic ad-hoc networking,” in *Sixth IEEE Workshop on Mobile Computing Systems and Applications, 2004. WMCSA 2004*, 2004, pp. 41–50.

[81] J. Davis, A. Fagg, and B. Levine, “Wearable computers as packet transport mechanisms inhighly-partitioned ad-hoc networks,” in *Wearable Computers, 2001. Proceedings. Fifth International Symposium on*, 2001, pp. 141–148.

[82] A. Lindgren, A. Doria, and O. Schelen, “Probabilistic routing in intermittently connected networks,” *Lecture Notes in Computer Science*, pp. 239–254, 2004.

[83] M. Musolesi, S. Hailes, and C. Mascolo, “Adaptive routing for intermittently connected mobile ad hoc networks,” in *Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks, 2005. WoWMoM 2005*, 2005, pp. 183–189.

[84] C. Shen, G. Borkar, S. Rajagopalan, and C. Jaikaeo, “Interrogation-based relay routing for ad hoc satellite networks,” in *IEEE Global Telecommunications Conference, 2002. GLOBECOM’02*, vol. 3, 2002.

[85] K. Tan, Q. Zhang, and W. Zhu, "Shortest path routing in partially connected ad hoc networks," in *IEEE Global Telecommunications Conference, 2003. GLOBECOM'03*, vol. 2, 2003.

[86] E. Jones, L. Li, J. Schmidtke, and P. Ward, "Practical routing in delay-tolerant networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 8, pp. 943–959, 2007.

[87] Z. Da Chen, H. Kung, and D. Vlah, "Ad Hoc Relay Wireless Networks over Moving Vehicles on Highways," in *ACM Mobihoc*, 2001.

[88] J. Burgess, B. Gallagher, D. Jensen, and B. Levine, "Maxprop: Routing for vehicle-based disruption-tolerant networks," in *Proc. IEEE Infocom*, 2006.

[89] Q. Li and D. Rus, "Communication in disconnected ad hoc networks using message relay," *Journal of Parallel and Distributed Computing*, vol. 63, no. 1, pp. 75–86, 2003.

[90] K. Fall, W. Hong, and S. Madden, "Custody transfer for reliable delivery in delay tolerant networks," *IRB-TR-03-030, July*, 2003.

[91] M. Seligman, K. Fall, and P. Mundur, "Alternative custodians for congestion control in delay tolerant networks," in *Proceedings of the 2006 SIGCOMM workshop on Challenged networks.* ACM New York, NY, USA, 2006, pp. 229–236.

[92] S. Burleigh, E. Jennings, and J. Schoolcraft, "Autonomous congestion control in delay-tolerant networks," Pasadena, CA: Jet Propulsion Laboratory, National Aeronautics and Space Administration., Tech. Rep., 2006.

[93] C. Gentry and A. Silverberg, "Hierarchical ID-based cryptography," *Lecture notes in computer science*, pp. 548–566, 2002.

[94] A. Seth and S. Keshav, "Practical security for disconnected nodes," in *1st IEEE ICNP Workshop on Secure Network Protocols, 2005.(NPSec)*, 2005, pp. 31–36.

[95] S. Symington, S. Farrell, H. Weiss, and P. Lovell, “Bundle security protocol specification,” 2007.

[96] A. Balasubramanian, B. Levine, and A. Venkataramani, “Dtn routing as a resource allocation problem,” in *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications.* ACM New York, NY, USA, 2007, pp. 373–384.

[97] S. Guo and S. Keshav, “Fair and efficient scheduling in data ferrying networks,” in *Proceedings of the 2007 ACM CoNEXT conference.* ACM New York, NY, USA, 2007.

[98] S. Guo, “Algorithms and design principles for rural kiosk networks,” Master’s thesis, University of Waterloo M. Math Thesis, 2007.

[99] H. Zhang *et al.*, “Service disciplines for guaranteed performance service in packet-switching networks,” *PROCEEDINGS-IEEE*, vol. 83, pp. 1374–1374, 1995.

[100] M. Hosaagrahara, “A generalized framework for achieving max-min fairness: theory and applications,” Ph.D. dissertation, Drexel University, 2006.

[101] D. Bertsekas and R. Gallager, *Data networks.* Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1992.

[102] S. Keshav, *An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network.* Addison-Wesley, 1997.

[103] J. Blanquer and B. Ozden, “Fair queuing for aggregated multiple links,” *Proc. ACM SIGCOMM*, p. 189197, 2001.

[104] A. Parekh, R. Gallager, I. Center, and Y. Heights, “A generalized processor sharing approach to flow control inintegrated services networks: the multiple node case,” *IEEE/ACM transactions on networking*, vol. 2, no. 2, pp. 137–150, 1994.

[105] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp. 1–12, 1989.

[106] J. Bennett, H. Zhang, and F. Syst, "WF2Q: Worst-case fair weighted fair queueing," in *Proceedings IEEE INFOCOM'96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation*, vol. 1, 1996.

[107] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round-robin," *IEEE/ACM Transactions on networking*, vol. 4, no. 3, pp. 375–385, 1996.

[108] X. Huang and B. Bensaou, "On max-min fairness and scheduling in wireless ad-hoc networks: Analytical framework and implementation," in *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing.* ACM New York, NY, USA, 2001, pp. 221–231.

[109] A. Sridharan and B. Krishnamachari, "Maximizing network utilization with max-min fairness in wireless sensor networks," *Wireless Networks*, pp. 1–16, 2007.

[110] L. Tassiulas and S. Sarkar, "Maxmin fair scheduling in wireless networks," in *IEEE INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*, vol. 2, 2002.

[111] L. Zhang, Y. Ge, and J. Hou, "Energy-efficient real-time scheduling in IEEE 802.11 wireless LANs," in *Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on*, 2003, pp. 658–667.

[112] V. Raghunathan, S. Ganeriwal, M. Srivastava, and C. Schurgers, "Energy efficient wireless packet scheduling and fair queuing," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 3, no. 1, pp. 3–23, 2004.

[113] V. Bonin, F. Cerdan, and O. Casals, "A simulation study of differential fair buffer allocation," in *High Performance Switching and Routing, 2000. ATM 2000. Proceedings of the IEEE Conference on*, 2000, pp. 365–372.

[114] J. Heinanen and K. Kilkki, “A fair buffer allocation scheme,” *Computer Communications*, vol. 21, no. 3, pp. 220–226, 1998.

[115] Y. Zhou and H. Sethu, “On achieving fairness in the joint allocation of processing and bandwidth resources: principles and algorithms,” *IEEE/ACM Transactions on Networking*, vol. 13, no. 5, pp. 1054–1067, 2005.

[116] J. Kleinberg, Y. Rabani, and E. Tardos, “Fairness in routing and load balancing,” in *Foundations of Computer Science, 1999. 40th Annual Symposium on*, 1999, pp. 568–578.

[117] J. Xu and W. Lee, “Sustaining availability of web services under distributed denial of service attacks,” *IEEE Transactions on Computers*, vol. 52, no. 2, pp. 195–208, 2003.

[118] D. Yau, J. Lui, F. Liang, and Y. Yam, “Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles,” *IEEE/ACM Transactions on Networking*, vol. 13, no. 1, pp. 29–42, 2005.

[119] D. Cohen and K. Narayanaswamy, “A fair service approach to defenses against packet flooding attacks,” Online White Paper.

[120] L. Peterson and B. Davie, *Computer Networks: A Systems Approach*. Morgan Kaufmann, 2007.

[121] T. Kos, M. Grgic, and L. Mandic, “CATV broadband technologies,” in *Video/Image Processing and Multimedia Communications, 2003. 4th EURASIP Conference focused on*, vol. 2, 2003.

[122] C. Vassilakis, M. Paterakis, and P. Triantafillou, “Video placement and configuration of distributed video servers on cable TV networks,” *Multimedia Systems*, vol. 8, no. 2, pp. 92–104, 2000.

[123] M. Shooman, *Reliability of computer systems and networks*. Wiley New York, 2002.

[124] L. Communications, “Free space optics :: Technology,” Online, November 2008, `http://www.freespaceoptics.org`.

[125] D. o. C. S. University of Hamburg, “Desmo-j: A framework for discrete-event modelling and simulation,” Online, December 2008, `http://www.desmoj.de`.

[126] S. Microsystems, “Developer resources for java technology,” Online, May 2008, `http://java.sun.com/`.

[127] B. Page, W. Kreutzer, and B. Gehlsen, *The Java Simulation Handbook: Simulating Discrete Event Systems with UML and Java*. Shaker, 2005.

[128] M. Fowler, *UML distilled: a brief guide to the standard object modeling language*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2003.

[129] M. Zarafshan-Araki, “Svn: Trainnet simulator code,” Online, November 2008, `https://svn2.assembla.com/svn/TrainNetSim`.

[130] R. Jain, *The Art of Computer Systems Performance Analysis*. John Willy and Sons, 1991.