Performance Evaluation of Active Network-based Unicast and Multicast Congestion Control Protocols

by

Riri Fitri Sari

Submitted in accordance with the requirements
for the degree of Doctor of Philosophy.

The University of Leeds
School of Computing

August 2003

Abstract

This thesis investigates the application of the Active Networks (ANs) paradigm in congestion control. ANs provide an alternative paradigm to solving network problems by allowing the network elements to perform computation. Thus, the AN is a promising paradigm to shorten the deployment time of new protocols.

Congestion control is a vital element for the Internet to avoid undesirable situations such as congestion collapse. The complexity and importance of congestion control has attracted many researchers to approach it in different ways, i.e. queuing theory, control theory, and recently game theory (pricing).

This thesis is concerned with the performance evaluation of AN-based congestion control protocols which have been classified according to their modes of operation, i.e. unicast, multicast single rate, and multicast multirate protocols. The research phase includes modelling and simulation experiments with the ns-2 network simulator.

The first area of interest in this thesis is unicast congestion control protocols. We integrate, run, and test the novel active queue management called Random Early Marking (REM) over an AN-based unicast congestion controlled network called Active Congestion Control Transmission Control Protocol (ACC TCP). It can be concluded that the implementation of the ANs paradigm in congestion control, enhanced by the application of REM queuing policy, improves the performance of the network in terms of its low buffer occupancy and stability compared with the one using Random Early Detection (RED) queue management algorithm.

Results of simulation studies comparing the performance of conventional protocols with those of AN-based protocols are presented. We investigate the TCP-friendliness behaviour of an AN-based single rate multicast congestion control called Active Error Recovery/Nominee Congestion Avoidance (AER/NCA), which uses active services to recover from loss at the point of loss and assists the congestion control. The use of AN helps the multicast application to achieve optimal data rates. We compare the results of AER/NCA TCP-friendliness to those of a single rate multicast protocols called Pragmatic General Multicast Congestion Control (PGMCC). Our simulation revealed that AER/NCA achieves the desirable property of TCP-friendliness. We also calculated AER/NCA's fairness index.

For multicast multirate congestion control protocols we compare an adaptive AN-based layered multicast protocol called ALMA (Active Layered Multicast Adaptation) with a non active network-based one called Packet-pair Receiver-driven Layered Multicast (PLM). ALMA performs layered multicast congestion control using AN approach, whereas PLM uses packet-pair techniques and fair scheduler. Experiments results shows that ALMA reacts differently from PLM in sharing the bottleneck link with CBR and TCP flows. We found that ALMA has fast convergence properties and provides flexibility to manage the network using the price function. Our experiments show that ALMA is not a TCP-friendly protocol and has a low inter-protocol fairness index.

Declarations

Some parts of the work presented in this thesis have been published in the following articles:

R. F. Sari and M. Kara. Evaluation of TCP-friendliness in Active Services Congestion Control Protocol. In *Proceedings of 17th Annual UK Performance Engineering Workshop (UKPEW2001).* Eds. K. Djemame, M. Kara. Leeds, United Kingdom, 18-19 July 2001. pp. 239-250. ISBN: 0 9541000 0

R. F. Sari and K. Djemame. Performance Comparison of RED and REM on Active Networks-based Congestion Control Protocols. In *Proceedings of the 2002 International Symposium on Performance Evaluation of Computer and Telecommunications Systems (SPECTS 2002).* Eds. M. Obaidat, F. Davoli, I. Onyuksel, R. Bola. San Diego, USA. 14-19 July 2002. pp. 137-145. ISBN: 1-56555-252-0

R. F. Sari and K. Djemame. Performance Comparison of Active and Non-Active Networks-based Multirate Multicast Congestion Control Protocols. In *Proceedings of the 10th IEEE International Conference on Networks (ICON 2002).* Eds. L.W.C. Wong, L-Y. Lau. Singapore. 27-31 August 2002. IEEE Computer Society Press. pp. 249-254. ISBN:0-7803-7533-5

R. F. Sari and K. Djemame. On Interprotocol Fairness of Active Network-based Multicast Congestion Control Protocols. In *Proceedings of the 11th IEEE Conference on Software, Telecommunications and Computer Networks (SOFTCOM 2003).* Eds. N. Rozic, D. Begusic. Split, Dubrovnik (Croatia), Ancona, Venice (Italy), 7-10 October 2003. pp. 232-236. ISBN:953-6114-64-X.

R. F. Sari. Performance Evalution of Active Network-based Layred Multicast Congestion Control Protocols. In *Proceedings of IEEE Conference on Software, Telecommunications and Computer Networks (SOFTCOM 2003).* Eds. T. Yoon, S. Suh. Phoenix Park, Korea, 9-11 February 2004. pp. 555-560. ISBN:89-5519-119-7 93560.

Acknowledgements

This thesis exists because I have been fortunate enough to be consistently supported by my family, supervisors, and friends. This acknowledgement is not enough to present my sincere gratitude for their help.

I would like to thank Dr. Mourad Kara for getting me started with the Active Networks and Congestion Control topic at the beginning of my PhD project in April 1999, and for his feedback, advice and encouragement, until he left for industry in June 2001. I would also like to thank Dr. Karim Djemame for his advice and support that have made this thesis possible, especially during the final stage of the writing-up period. My sincere gratitude to Professor Ken Brodlie for his advice, encouragement and support. I would also like to thank Dr. Raymond Kwan, the post-graduate tutor, for his administration support and advice.

Special thanks to Dr. Suhaidi Hassan, Somnuk Puangpronpitag and Jin Wu, who have make my presence in our research group office worth the travelling from Sheffield to Leeds. Thank you all for the stimulating discussion on our research topic and getting the best from the Internet. Thanks also to all members of the academic and support staff at the School of Computing.

Thank you to my husband - Dr. Kusno Adi Sambowo and to my two lovely little baby girls for making my life fulfilled with love and understanding. Thank you to my mother Azizah who has shared my burden and made me strong constantly from day to day because she kept reminding me of the power of Allah the Almighty.

Thank you to Dr. Lidia Yamamoto of the University of Liege-Belgium, a friend I met on the ns-2 mailing list. Thank you for becoming my closest friend in the world of Active Networks. The Internet proves to be a great place to meet the right person at the right time. I would like to thank Dr. Daniela Romano, with whom I have shared the long trips commuting from Sheffield and who has made me stronger in realising completing a PhD and mothering two babies during that time is possible. Thanks to Dr. Sarah Fores, Margaret Howlett, Paul Nicholson, John Kitching and Dr. Elise Langham for their comments and their time in reading this thesis.

I would like to thanks the QUE Project at the Electrical Engineering Department of the University of Indonesia for sponsoring my research. I am also indebted to my real 'guru' in computer engineering in Jakarta, Dr. Bagio Budiardjo, for his wisdom that inspired me to enter acadaemia years ago. Finally, my heartiest gratitude goes to all my entire family and friends for their support during the difficult times.

To

my mother Azizah,

my father Mursyid,

my husband Kusno Adi Sambowo,

and my daughters :

Almira Lavina Sambowo,

Naufalia Brillianti Sambowo.

Contents

List of Figures

List of Tables

List of Abbreviations

ABONE	Active Network Backbone
ACC	Active Congestion Control
ACK	Acknowledgement packet
ACM	Association of Computing Machinery
AER/NCA	Active Error Recovery/Nominee Congestion Algorithm
AHCC	Active Hierarchical Congestion Control
AIMD	Additive Increase Multiplicative Decrease
ALMA	Active Layered Multicast Adaptation
AMP	Active Multicast Protocol
ANs	Active Networks
ANANA	Active Networks Assigned Number Authority
ANEP	Active Network Encapsulation Protocol
ANSE	Active Network Simulation Environment
ANTS	Active Network Transfer Service
API	Application Programming Interface
AQM	Active Queue Management
ARM	Active Reliable Multicast
ASCC	Application Specific Congestion Control
ASP	Active Server Pages
ADSL	Asynchronous Digital Subscriber Line
ATM	Asynchronous Transfer Mode
ATML	Active Traffic Control Mechanism for Layered Multicast
CA	Congestion Avoidance
CAIDA	Cooperative Association for Internet Data Analysis

CALM	Congestion-Aware Layered Multicast
CANES	Composable Active Network Elements
CBR	Constant Bit Rate
CBQ	Class Based Queueing
CCM	Congestion Control Message
CSM	Congestion Status Message
CWND	Congestion Window
DARPA	Defense Advanced Research Project Agency
DDoS	Distributed Denial of Service
DiffServ	Differentiated Service
DyRAM	Dynamic Replier Active Reliable Multicast
DRED	Dynamic Random Early Detection
ECN	Explicit Congestion Notification
EE	Execution Environment
FAIN	Future Active Internet Protocol Network
FIFO	First In First Out
FTP	File Transfer Protocol
FM	Frequency Modulated
FQ	Fair Queueing
IEEE	Institute of Electrical and Electronics Engineers
GSM	Global System for Mobile communications
HTTP	Hypertext Transfer Protocol
ID	Identifier
IETF	Internet Engineering Task Force
IntServ	Integrated Service
IOS	Internet Operating System
IP	Internet Protocol

IPv6	Internet Protocol version 6
ISDN	Integrated Services Digital Network
JANOS	Java Active Network Operating System
JVM	Java Virtual Machine
Kbps	Kilo bits per second
LARA	Lancaster Active Router Architecture
LBNL	Lawrence Berkeley National Laboratory
LPE	Loss Path Estimate
LFN	Long Fat Network
LW	Long Wave
MAC	Medium Access Control
MBONE	Multicast Backbone
Mbps	Mega bits per second
MLDA	Multicast Enhanced Loss Delay-based Adaptation Algorithm
MPLS	Multi Protocol Label Switching
NodeOS	Node Operating System
ns-2	Network Simulator-2
nam	Network Animator
NACK	Negative Acknowledgement
Opensig	Open Signalling
ORE-SDK	Oplet Runtime Environment Software Development Kit
OTcl	Object-oriented Tools Command Language
PAN	Programmable Active Network
PARSEC	Parallel Simulation Environment for Complex Systems
PGMCC	Pragmatic General Multicast Congestion Control
PLAN	Programmable Language for Active Networks
PLM	Packet-pair Layered Multicast

PRR	Packet Retransmission Ratio
RED	Random Early Detection
REM	Random Early Marking
RFC	Request For Comment
RLM	Receiver-driven Layered Multicast
RLC	Receiver-driven Layered Congestion Control
RMANP	Reliable Multicast Active Networks Protocol
RNG	Random Number Generator
RRSA	Resource Routing Service Access
RS	Repair Server
RTT	Round Trip Time
SACK	Selective Acknowledgement
SPM	Source Path Message
SRED	Stabilised Random Early Detection
SS	Slow Start
TCP	Transmission Control Protocol
TCL	Tool Command Language
TFRC	TCP Friendly Rate Control Protocol
TFMCC	TCP Friendly Multicast Congestion Control
UDP	User Datagram Protocol
VAN	Value Added Network
VBR	Variable Bit Rate
VINT	Virtual Internet Testbed
VPN	Virtual Private Network
WFQ	Weighted Fair Queueing
QoS	Quality of Service

List of Symbols

$U_x(X_s)$	utility
p(t)	congestion measure at the link
N	sample of size for price estimation
b(t)	queue length
c	constant > 0
y(t)	aggregate source rate
γ	step size in price adjustment > 0
α	weight of buffer in price adjustment (constant > 0)
β	weight in RTT estimation
δ	weight in aggregate input rate estimation
max	maximum dropping/marking threshold
min	minimum dropping/marking threshold
maxp	maximum packet dropping probability
wq	weight parameter
avg	average queue size
b	average throughput
loss(p)	loss probability estimation
MTU	Maximum Transfer Unit
f_i	fairness index
x_i	throughput of session i on a bottleneck link
n	number of sessions sharing the bottleneck link

Chapter 1

Introduction

The objective of the work presented in this thesis is to evaluate unicast and multicast Active Network-based congestion control protocols. We will describe the interest of the work by outlining the background, objectives, scope and contributions of the work in this chapter.

1.1 Research Motivation

The dramatic expansion of the global information infrastructure today can largely be attributed to the strength and simplicity of the Internet Protocol (IP). However, there are many problems encountered in today's Internet such as the difficulties of integrating new technologies and standards into the shared network infrastructure, poor performance due to redundant operations at several protocol layers, and difficulty in accommodating new services in the existing architectural model. As the Internet grows, the demand to provide advanced networking abilities such as rapid network service deployment, robust network security and flexible network management, and Quality of Services (QOS), has become a big challenge for the networking research community. The concept of *programmable network* technologies has been introduced to overcome this problem. This is conducted by segregating the computing and the routing capabilities of the network nodes, by means of separating the communication hardware from the control software. The programmable network technologies can be classified into *Active Networks* (ANs)[1], *open programmable interface technologies*, and *hybrid networks* (embracing both approaches). Two major

[1]Active Network is a kind of programmable network due to the capability of the intermediate nodes to be programmed. Therefore, in this thesis Active Networks and programmable networks are used to point out the same idea.

communities, DARPA[2] and IEEE[3] Communication Society Open signalling working group (Opensig) have contributed their efforts on the next generation networking technology to meet the above challenges since 1994.

Active Networks (ANs) were initiated by DARPA, whereas the IEEE Opensig group has focused more on signalling, middleware and service creation through their PIN (Programming Interface for Networks) P1520 project. Both Active Networks and open programmable networks paradigms are aiming towards enabling new and flexible Internet applications and services, as well as improving end-to-end performance for existing ones. It can be said that AN is a form of programmable network due to the programmability of the active nodes. In this thesis, our work is more focused on ANs rather than the programmable network based on the Opensig society's direction.

ANs are not just passive carriers of bits but provide the capability for the user to inject customised programs into networks [159]. The active nodes will be able to modify, store, and redirect the user data flowing through the network. ANs allow executable content to be delivered to network elements through active packets or capsules. They provide a programmable network Application Programming Interface (API). The dynamic control enabled by ANs allows services to be tailored to the current network conditions. These services will improve the performance seen by applications. In ANs in which intermediate nodes (routers/switches) are programmable, the network has a computational engine ready to be programmed and adapt to changes. New protocols can be developed to take advantage of the flexibility of the AN-based systems.

Examples of *applications* that can benefit from such AN approaches are multicast video distribution, quality of service support, caching and network management [69]. Web proxy caches and firewalls are the most familiar real life ANs applications that can be found today.

In the current Internet Protocol (IP), all components such as packet format and addressing scheme have been standardised. Changing anything in IP would normally require hardware replacement (router and IP switches). As opposed to the current network (Internet), AN nodes can execute different programs provided by the users or the network managers. The new paradigm of AN permits packets to carry their own network service and to compute that service within the infrastructure [150]. ANs have an abstract layer supported on all devices. Incorporating innovation

[2]US Defense Advance Research Project Agency
[3]Institute of Electrical and Electronics Engineers

means downloading those new services into the infrastructure. In conjunction with the existing infrastructure, active nodes can coexist with legacy nodes by tunnelling, in the same way as Multicast Backbone (MBONE) tunnels through 'non-multicast aware' routers.

During the last few years, ANs have been one of the state of the art topics in networking. The AN architecture, which includes the active node operating system, execution environment, and active applications, has become mature. More AN prototypes have been proposed and tested for inclusion in the AN testbed.

The problem of *congestion* has been considered a high priority topic in computer networks, and it became more urgent to be solved with the growth of the Internet and its applications, as well as the limitation of the shared resources. Congestion control is necessary when the resources in a network need to be allocated such that the network operation performance can reach an acceptable level when the demand exceeds or is near the capacity of the network resources [82]. Figure 1.1 shows the throughput as a function of offered load, in which we can see how congestion avoidance, congestion recovery, and congestion collapse happen. *Congestion collapse* is a state where any increase in the offered load leads to a decrease in the useful work done by the network [68]. *Congestion avoidance* mechanism is implemented when packet loss is detected. Generally congestion control algorithms use packet loss to detect congestion.

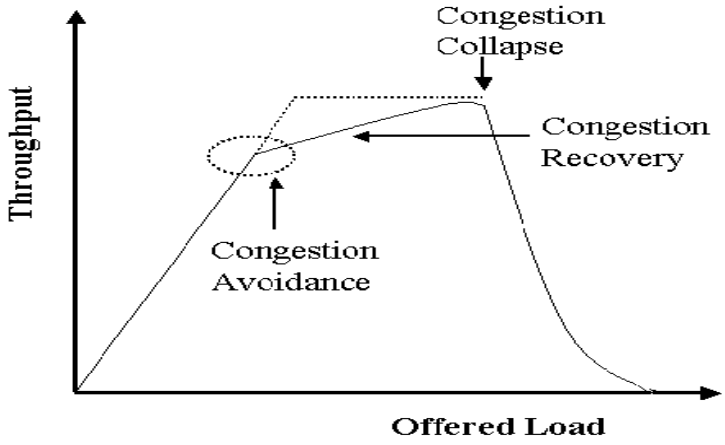


Figure 1.1: Throughput as the function of the offered load (Overview to *congestion* collapse, recovery and avoidance) [68]

Congestion control is necessary to avoid congestion collapse happening in the Internet. If the current Internet traffic which is based on best effort continues to

grow, measures must be taken to ensure a congestion control mechanism exists in every protocol. ANs paradigm will help the congestion control mechanism implementation due to the availability of computational power in the intermediate nodes, in which some kind of intelligence can be installed.

The *Internet performance* perceived by the users depends largely on the performance of the *Transmission Control Protocol (TCP)*, due to the fact that TCP is the most widely used transport layer protocol in the Internet. Although the performance dynamics of TCP over traditional networks are relatively well understood, the research community is only beginning to explore the TCP performance implications for the emerging and future networking environments [76]. This thesis will explore the dynamic performance of TCP over ANs, both for unicast and multicast congestion control protocols.

In recent years, a few AN-based congestion control protocols have been proposed in the literature. The effectiveness of these proposed active services has to be validated, and there are tools considered for this purpose, i.e. mathematical analysis and computer simulation. Mathematical analysis of the ANs can be very complex. This approach requires the assumption of simplification in some system's characteristics, which can lead to inaccuracy and unrealistic models. Therefore many new protocols are proposed in the form of network simulations or prototypes.

Simulation plays a vital role in characterising the behaviour of the current Internet and the possible effect of the proposed changes. Simulation is beneficial in exploring new protocol proposals for the Internet that have not yet been realised. This thesis will focus on the evaluation of AN protocols, specifically unicast and multicast congestion control protocols.

The focus of this thesis is to evaluate several ANs and congestion control protocols. The modes of operations of the Internet can be classified into *unicast, multicast single rate,* and *multicast multirate* protocols. All *unicast* protocols are obviously single rate, in which data is sent point-to-point at a single rate. Multicast protocols enables efficient data transmission from a sender to many receivers. Multicast protocols can be either *single rate* or *multirate* (layered multicast). A *layered multicast* protocol enables multiple multicast groups to transmit data at different rates for a diverse set of receivers.

We evaluate one protocol for each kind of stream. In our work on unicast protocols, we implement a new Active Queue Management (AQM) mechanism called Random Early Marking (REM) [6] in an AN environment. The work on multicast single rate was focused on the protocol's dynamic behaviour and the TCP-

friendliness issue. The work on multicast multirate compared the protocol to the non-active one. We also address the pricing issue in the context of AN-based layered multicast congestion control protocols. Pricing has been used in the AN-based layered multicast protocol studied in this work, in which the bandwidth is considered as a scarce resource. Congestion control is performed by comparing the price with the budget in making layer pruning and subscribing decisions.

1.2 Thesis Objectives

The goal of this research is to use *performance evaluation* techniques in investigating the application of ANs to overcome the congestion control problem. The performance evaluation and extension of the current Internet protocols with the inclusion of advanced issues in networking will contribute to the general body of knowledge on the Internet protocols' behaviour.

It is the objective of this research programme to study the ANs advances in architectures, implementations and deployment of new services, in attempting to solve the problem of congestion control. We will use network simulation as a research methodology to evaluate the performance of the AN-based unicast and multicast congestion control protocols.

In order to address the issue faced by the next generation unicast and multicast protocols based on the ANs paradigm, this work will address several key areas:

[K1] Survey of the existing ANs and congestion control research to identify key problem areas.

[K2] The evaluation of unicast, multicast single rate, and multicast multirate protocols and identification of the problems which exist in each congestion control protocol.

[K3] The simulation of several AN-based congestion control protocols to cover advanced issues such as TCP-friendliness, active queue management, layered multicast protocols, and pricing.

1.3 Scope of Thesis

The scope of the research includes the ANs paradigm advances and their implications in solving the network congestion control problem. The applications found in

literature to solve the network congestion problem (such as TCP congestion control) are among the interests of this research. The provision of an evaluation framework of AN-based congestion control protocols is also within the scope of this research. The applications that benefit from the flexibility provided by ANs are investigated.

The scope of this thesis is:

1. the evaluation of an AN-based unicast congestion control protocol called Active Congestion Control Transmission Control Protocol (ACC TCP) [47, 49]. In this work we extend the existing AN-based protocol by modifying it to support the implementation of a new active queue management algorithm called Random Early Marking (REM).

2. the evaluation of an AN-based single rate multicast congestion control protocol called Active Error Recovery/Nominee Congestion Avoidance (AER/NCA) protocol [88]. In this work we focus on the TCP-friendliness of the protocol and its comparison with a non-AN-based congestion control protocol such as Pragmatic General Multicast Congestion Control Protocol (PGMCC) [135].

3. the evaluation of an AN-based multirate multicast congestion control protocol called Active Layered Multicast Adaptation (ALMA) [186, 187]. In this work we compare the ALMA protocol with a non-AN one such as Packet-pair Layered Multicast (PLM) [102].

Figure 1.2 illustrates our research scenario in which the relationship between the three modes of operations in the Internet (unicast, single rate multicast, and multirate multicast), ANs, congestion control, and ns-2 simulator, are shown. In this work we considered two Active Queue Management algorithms called Random Early Detection (RED) [63] and Random Early Marking (REM). More details on ns-2 simulator, RED and REM can be found in Chapter 2.

1.4 Methodologies

The research methodology is based on the following:

1. *Modelling and experimental design.* It is necessary to perform modelling in order to create a framework in which to evaluate the characteristics of AN-based protocols in the form of simulations. Experimental design is required to plan precisely the experiments that will be conducted according to the objectives of the research.

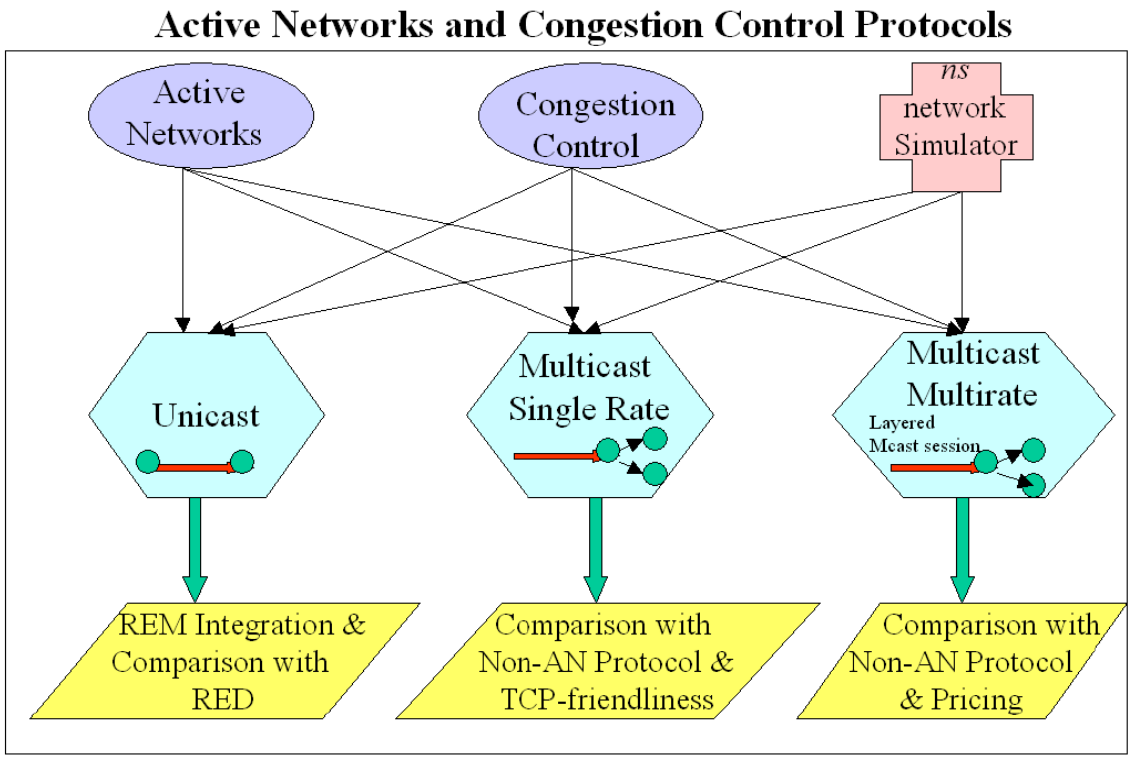


Figure 1.2: Research scenario

2. *Simulation.* In this step, the simulation model is implemented using a network simulator. Multiple scenarios of traffic behaviour in an AN environment are simulated and the congestion control approaches are evaluated.

3. *Performance evaluation and validation.* The performance of the implementation of the ANs paradigm's simulation to different traffic flows is evaluated. The model and simulation scenarios need to be validated to ensure that the obtained results are valid. The validation stage is conducted after completion of each step.

1.5 Major Contributions

The major contributions of this work include:

[C1] Introduction and implementation of the REM active queue management algorithm in unicast AN-based congestion control protocols. Since the introduction of REM in 1999, this queue management scheme has generated a lot of research interest due to its novel approach to queue manage-

ment. The comparison of the results of RED and REM queue management schemes in an ANs environment is one of the contributions of this thesis. We discover that REM can complement ACC TCP and performs better than RED in an AN-based environment.

[C2] Evaluation of an AN-based single rate multicast multirate protocol in terms of its dynamic microscopic behaviour and TCP-friendliness. The comparison of the inter-protocol fairness of AN-based and non-AN-based single rate multicast protocols considered is also one of the contributions. We discover that the AN-based unicast protocol can achieve TCP-friendliness, just as the non-AN-based protocol does.

[C3] Performance comparison of AN-based and non-AN-based multirate multicast protocol in terms of its layered multicast characteristics, i.e. speed, accuracy and stability in achieving convergence to subscribe to the optimal layer; the behaviour in sharing the bottleneck link with Constant Bit Rate (CBR) and TCP flows; and fairness indices. We discover that AN can help in enabling a layered multicast protocol, and the use of a pricing function provides flexibility for the network manager to shape the distribution of bandwidth.

1.6 Thesis Overview

This thesis presents the work on the performance evaluation of the AN-based congestion control protocols, both for unicast and multicast streams. Several main issues in congestion control protocols such as TCP-friendliness, active queue management impact, and bandwidth estimation-related techniques are highlighted. In particular, this work describes the necessity of congestion control protocols to be fair and friendly towards the same kind of flows, as well as towards TCP. TCP-friendly congestion control protocols can be complemented with the appropriate use of active queue management mechanisms to attain performance gain. A new protocol of this kind can be deployed in an AN-based environment, to ensure fast and flexible deployment in the intermediate routers without having to wait for a standardisation stage.

This thesis is structured as follows:

Chapter 2 surveys the background and the related work on all major research issues related to ANs and their application. First we describe the development of

ANs. Subsequently, current development of the architectures and prototypes of ANs are presented, as well as the ANs application areas. We describe the background of using simulation as a research methodology in networking. In the last part of the chapter, we also address the simulation tool used in this research.

Chapter 3 reviews the ANs application to congestion control in unicast, single rate multicast, and multirate multicast. First, we describe the congestion control advances and the current congestion control protocols found in literature. Subsequently, we consider advanced issues in networking, i.e. active queue management, TCP-friendliness, pricing and layered multicast protocols. Further, different approaches to unicast and multicast congestion control protocol using ANs are shortly discussed. In this context, we also review the three protocols evaluated in this work, i.e. Active Congestion Control Transmission Control Protocol (ACC TCP), Active Error Recovery/Nominee Congestion Avoidance Algorithm (AER/NCA), and Active Layered Multicast Adaptation (ALMA).

Chapter 4 presents the performance study of AN-based unicast congestion control protocols. We integrate the REM algorithm into ACC TCP and evaluate its performance compared with the RED active queue management mechanism. First we introduce the general approaches used in all experiments in this work, and then we present the simulations and results for different network experiments.

In Chapter 5 we focus on the fairness issues on a multicast single rate protocol called AER/NCA. The inter-protocol fairness of AER/NCA is also compared with that of the Pragmatic General Multicast Congestion Control (PGMCC) protocol (non-AN-based).

Next, Chapter 6 presents the performance evaluation of the Active Layered Multicast Adaptation (ALMA) protocol. This chapter describes the comparison of the AN-based layered multicast adaptation protocol (ALMA) with a traditional layered multicast protocol called Packet-pair Layered Multicast (PLM).

Finally, Chapter 7 concludes the thesis and outlines some future work.

Chapter 2

Active Networks

This research focuses on the evaluation of Active Networks (AN)-based congestion control protocols and on providing a framework of the multi modes of operations of the Internet, i.e. unicast, multicast single rate, and multicast multirate. This chapter provides the background to active networks which will be covered in detail in the thesis. Section 2.1 provides an overview of ANs and the development of this new paradigm. Section 2.2 reviews simulation as a research methodology used in this work and describes the simulation environment. Finally, Section 2.3 summarises the chapter.

2.1 Active Networks

Active Networks (ANs) are networks that allow intermediate nodes to perform computations up to the application layer. Users can program the network by injecting their programs into it. These programs travel inside network packets and are executed in intermediate nodes, resulting in the modification of their state and behaviour. The packets of an active network can be regarded as programs, which are called active packets (capsules), to distinguish them from 'traditional' network packets, in which processing within the network is limited to only a few functions such as routing, congestion control, and simple Quality of Service (QoS) mechanisms.

Routers and switches within the network can perform computations on user data flowing through them. Users can program the network by supplying their own programs to perform these computations [159].

There are some pros and cons with regard to the deployment of the idea of ANs to the current 'traditional' networks. The argument for ANs states that a variety of useful network services that involve processing at intermediate nodes will be made

possible. On the contrary, the argument against ANs argues that the Internet is successful today because of its simplicity, whereas in making the networks 'active' things may get complicated [128].

The success of ANs would mean that intermediate node functions may be programmed and deployed through simple, open, and rapid processes that do not require standards committees' or vendors' resources [128]. The following subsection reviews the development of ANs as a new paradigm in networking, ANs architecture, prototypes, applications and commercial efforts.

2.1.1 Development of New Paradigm in Networking

The AN terminology was first coined by David Tennenhouse in 1996 [159, 158]. The idea behind ANs has been discussed in the United States' Defense Advanced Research Project (DARPA) since late 1994, and many researchers have built up their work on top of the paradigm [116, 101]. In the last half decade, ANs have become one of the main keywords in the refereed network conferences and publications. DARPA ANs conference and exposition in May 2002 republished all US-sponsored projects in ANs and highlighted the achievement in this field[1].

Recall that we have introduced active networks and programmable networks in Chapter 1. There are two ways of looking at active and programmable networks, i.e. the Active Networks (ANs) and the Open signalling (Opensig). The primary technology for active networks is the Internet, whereas Opensig is based on telecommunications technology. The Opensig community argues that open access to switches and routers can be provided by modelling communication hardware using a set of open programmable network interfaces. Open signalling is the approach taken by the telecommunications sector to make these networks programmable by means of programmable switches. The IEEE 1520 project on Programmable Interfaces for Networks (PIN) is pursuing the Opensig approach to standardise programming interfaces [36]. The ANs projects are basically focused on IP networks, where control and data paths are combined. The level of dynamic deployment of new services is restricted within the IP networks, using capsules and node primitives, which is more dynamic than what is proposed by the Opensig community.

Both the ANs and Opensig community include a broad spectrum of projects with diverse architectural approaches in order to reach the same goal, which is to improve the approaches and technologies to construct, deploy, and manage new services

[1]http://www.darpa.mil/ato/programs/activenetworks/actnet.htm

in telecommunication networks. Campbell et al. [36] produced a good survey of programmable networks in which they show the programmable networking model in terms of communication and computation, node kernel, network programming environment, and programming network architecture.

Tennenhouse et al. [159] in their early work distinguished two approaches to ANs, *discrete* and *integrated*, depending on whether programs and data are carried discretely, i.e. within separate messages, or in an integrated fashion. A *discrete approach (out-band)* is performed by programmable switches, where the users send their packets through such a 'programmable' node. The *integrated approach (in-band)* uses the term capsules (active packets) for every message, and *each of them* is considered as a program. When a capsule arrives at an active node, its content is evaluated and then dispatched to an execution environment [34].

2.1.2 Architecture

During the advancement of ANs, there has been no clear consensus on their architecture. However, most of the development of the ANs is based on the DARPA architecture. The architecture for an active node divides the functionality between the Execution Environments (EEs) and Node Operating Systems (NodeOS). Each EE provides an interface through which end-to-end services are provided to users. Multiple EEs could be present in a single active node. The Node OS manages the resources of the active node that include transmission, computing, and storage. When an active node receives a packet over a physical link, it classifies the packet based on its header, which can be assigned to a channel or discarded. [35] describes the ANs architecture in detail. Figure 2.1 shows the resource managers and active applications over the DARPA architecture in which the application programming interface can be seen between the EE and the NodeOS.

EEs define a particular interface to resources available within the network. There are several kinds of EE that can be useful in an active node, i.e. programming language interpreter (strong AN), traditional network layer protocol (without an AN), customisable services and active-enabled traditional network layer protocols (intermediate - between strong and without ANs)[20, 21].

Active Networks Transfer System (ANTS) [171, 172], Composable Active Network Elements (CANES) [1], Messenger-0 (M0) [11], Active Server Pages (ASP) [15], Internet Protocol version 6 (IPv6) [44] and Tamanoir [67] are some examples of EEs. A backbone framework called Active Network backbone (Abone) has been

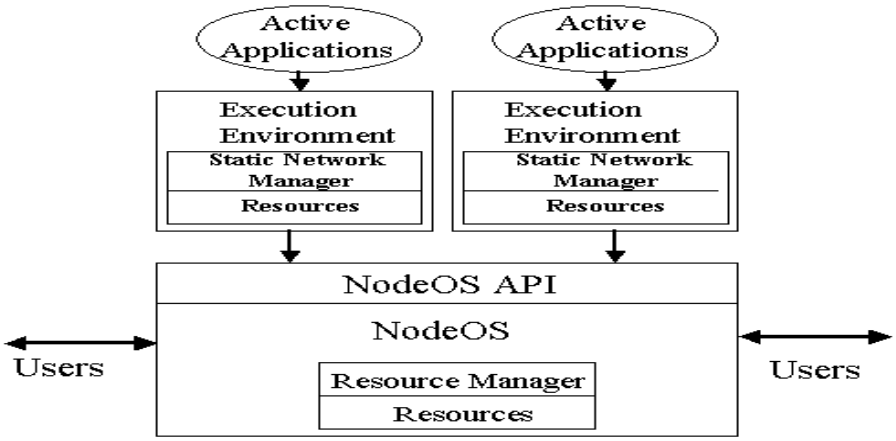


Figure 2.1: Resource managers and active applications over DARPA architecture [187]

created to be a testbed for many new AN protocol proposals [17]. The Abone is a virtual testbed for DARPA sponsored ANs projects, in which a set of computer systems has been configured into a virtual active network. These systems execute under UNIX, Linux, Solaris, or other AN-specific operating systems. During the last five years, the number of new applications and protocols which reached the Abone deployment stage has not been as high as expected, mainly due to the complexity of deploying new mechanisms into the EE and adhering to the testbed mechanism.

In order to allow the use of the existing infrastructures for ANs, the Active Networks Encapsulation Protocol (ANEP) has been created as the result of the work performed by the work group initiated by US DARPA [2, 35]. This protocol allows the use of the protocol calls for an ANs frame format. Therefore the programs to be executed could be integrated into its payload. This protocol is a near term interoperability solution for active packets. In the ANEP terminology, a packet consists of an ANEP header and a payload. An active node is a network element that can evaluate active packets. ANEP is responsible for routing packets to a particular EE. The ANEP header includes a type identifier field. In this protocol, well-known Type IDs are assigned to specific EEs by the Active Network Assigned Number Authority (ANANA). In order to be processed by an EE, a packet does not need to contain an ANEP header.

2.1.3 Prototypes

At least eleven network prototypes of ANs have been developed, which include Active Networks Transfer System (ANTS) [173], Alien [1], Router Plugins [42], Tempest [43], Netscript [148], Switchware [2], Active Service, Smart Packet [140, 94, 34, 167], Programmable Active Networks (PAN) [121], Composable Active Networks Elements (CANES) [1], and Lancaster Active Router Architecture (LARA++) [139]. Some of them provide both EE and Node OS, for example (Canes [1], Bowman operating systems [115]) and (Switchware, SANE operating systems [2]). [38] provides a good review of these ANs prototypes. An interesting study which compares those prototypes in terms of their architecture domain, networking technology, programmable communication abstraction, level of programmability, and composition language can be found in [36].

In the following discussion we will take a particular example of an execution environment (ANTS) and will describe it in detail for the reasons of its popularity. As an early work in prototyping ANs, Active Networks Transport System (ANTS) Application Programming Interfaces (APIs) have been used by many researchers. ANTS was built by Wetherall, one of the initiators of ANs at MIT [174, 175, 176]. In ANTS, a small Tool Command Language (Tcl) code is inserted into the IP option field and the version of the Tcl interpreter is then incorporated into the Linux kernel. The ANTS toolkit is an EE for protocol and service deployment which is built on top of the Java Virtual Machine (JVM). User code can be transmitted in packets (called capsules) in the form of Java byte code. In a simple form, a capsule only contains a reference to a routine to be executed at the active node, whereas more complicated codes must be loaded by a code distribution mechanism.

The APIs provided by ANTS can be categorised into capsule manipulation, control operation, as well as environment and storage calls. ANTS leverages the mobile code facility of Java to provide the deployment of new protocols in ANs [172]. The widespread use of ANTS to build their AN services, such as in multicast application, can be found in [33]. ANTS has also been used as the basis of API in [136], and ANTS EE is being ported to Janos (Java Active Networks Operating System) at the University of Utah [162]. An extension of ANTS called PANTS (Programmable Active Network Transport System) has been proposed in [53] in which a node's runtime execution environment can be changed dynamically. In this system, capsules are also allowed to rewrite their code dynamically as well.

ANTS has been chosen to be the EE for many ANs services because it enables users to introduce new services using the capsule programming model. Capsules

are special packets that combine data with a customised forwarding program. The type of forwarding is selected by end user software. For this purpose, a mobile code technique is used to transfer corresponding programs along with the data as the capsule passes through the network. Therefore the system is more flexible and allows topology, loss and load information to be used to construct a variety of new routing, set-up, congestion handling, and measurement services [175]. ANTS's flexibility, rapid development, performance, and security are some major competitive advantages of this execution environment compared with others.

Modelling and simulation of the ANs have been discussed in detail in [144], which illustrated different environments for ANs. In fact, most concerns relate to the safety and security issues in ANs. These issues have been addressed from a programming point of view, for example in Packet Language for Active Network (PLAN) [78]. PLAN uses lightweight programs with limited functionality to call the other program's routine. These services require authentication and authorisation before allowing access to the resource they protect. Active Networks Simulation Environment (ANSE) [133], for example, provides an integrated environment for modelling and parallel simulation of ANs and supports PLAN EE.

2.1.4 AN Applications

In this section, AN-based applications are highlighted and their advantages are presented. We can categorise AN-based solutions into the following categories: wireless access and mobile computing, telephony, Virtual Private Networks (VPN), network management, security, web caching, multicast and congestion control.

2.1.4.1 Wireless Access and Mobile Computing

ANs would be of tremendous value if they facilitated the widespread deployment of even a small number of services such as multicast and mobile IPv6[2]. An active packet can be used to upgrade the service of a wide area network such as the Internet because no standard should be selected previously [173]. The wireless network architecture is one of the ANs applications that is currently being developed. The work of Wetherall and Van Bose [24], for example, proposed RadioActive networks to utilise the spectrum efficiently. RadioActive networks is an adaptable wireless network architecture to create communication networks that can be dynamically

[2]Mobile IPv6 is one of the key technologies for realising seamless communication between fixed and mobile access network.

re-programmed to optimise their use for the environmental conditions, traffic characteristics and user requirements.

The work of Chin [38] proposed AN-based routing protocols for unicast and multicast mobile computing networks. The advantages of those AN-based protocols are shown in their adaptation to mobility, lower signalling overhead, high reuse of allocated network states, extensibility and scalability.

2.1.4.2 Telephony

In telephony, ANs can be applied to implement adaptive coders and utilising idle lines. Maxemchuck [113] shows the impact of applying the active router network concept to telephony, such as the creation of a commodity other than bandwidth, for equipment manufacturers and network providers, the provision of local access with a distinct advantage, the integration of telephone networks with the Internet, as well as the increase of the rate of change in networks by reducing the need for a standard protocol. A user layer AN provides a means of interconnecting the Internet and telephone network.

2.1.4.3 Virtual Private Network (VPN)

A user-defined VPN is an example of active services to create an open and distributed infrastructure for the Internet to transmit data between corporate sites. Some work to project advanced enterprise network and service management to ANs has been addressed in [25]. [111] implemented ANs to help in configuring virtual private networks in a modular way and with a high degree of flexibility.

2.1.4.4 Network Management

ANs can help a Network Management system to expand, shrink, and change its function and scope as necessary. Active packets can be used to reconfigure the system as necessary by changing or rewriting the capsules. An example of the use of ANs for service and network management is found in [30]. Brunner and Stadler's work on telecommunication environments covers service management and virtualising ANs [31]. Barone et al. [15] proposed a network management framework which allows the exploration of ANs that can interoperate with any EE such as ANTS, PLAN, and ASP. ANs deployment in the Grid[3] computing environment

[3] "Grid is a type of parallel and distributed system that enables the sharing, selection and aggregation of resources distribution across 'multiple' administrative domains based

has also been performed in [67, 96]. Some European Union-based institutions are working on a common project, called Future Active IP Network (FAIN) [105], to incorporate ANs policy-based management issues on the Grid.

2.1.4.5 Security

ANs' deployment means a massive increase in the degree, sophistication, and complexity of software development in the network. Security threat is a major issue in an open programmable network environment. In an attempt to solve this, the work of Brown [29] deals with building an ANs system that maintains end-to-end security without degrading the functionality it provides.

In [164] Van shows how ANs can be used for defending against address spoofing. Seraphim framework presented in [118] uses active capabilities to compute the authorisation decision. [147] includes a mechanism to use ANs to overcome Distributed Denial of Service (DDos) attacks by arranging AN-based detection, coordination, and reaction mechanisms. Active routers and switches could help trace DDoS attack by executing code based on traffic flow, destination and source address. Active routers can also report anomalous activity and decide on the appropriate reaction. This prototype has been tested in the Abone (Darpa's Active Network backbone).

2.1.4.6 Caching

Caching is an area in which ANs application has been widely investigated. The use of AN-based caching and filtering can help online auctions, in which a filtering program drops bids that are lower than the current price for a given item [101]. Web caching using active services has been performed by Wetherall in his work in [173]. The study on caching and active video agents led Najafi [120] to propose a new cost model which separates node and link cost for ANs. He also proposed an agent routing and agent positioning mechanism in ANs and implemented it using a Java applet.

2.1.4.7 Multicast

ANs support for reliable multicast such as in [33, 88] and active router architecture for multicast video distribution [89] are some of the applications in multicasting. ANTS has also been used in some active multicast-based protocols such as active

on their (resources) availability, capability, performance, cost, and users' QoS requirement" [http://www.gridcomputing.com]

multicast core migration in [93]. The problem domain of multicast transport protocol has also been attempted by employing the ANs paradigm in order to implement a router-assisted scheme. Detailed discussion of an AN-based multicast congestion control algorithm will be presented later on in Chapter 3.

2.1.4.8 Congestion Control

The application of ANs in congestion control has been performed in different ways and in targeting different network traffic. One way of applying ANs to congestion control is by allowing applications to request specific node algorithms to be invoked during congestion periods, for example selective discard, transcoding and lossless compression algorithms. We will discuss the ANs application in congestion control in detail in Chapter 3.

2.1.5 Commercial Efforts

Hitachi, Bell Labs, NEC, Nortel Networks, and British Telecom are some companies that have put research and development efforts into ANs. The work of Brunner [30, 31] is an example of work done in viewing the ANs technology from the telecommunication environment point of view. This work is based on an assumption that ANs technology will mature and can be deployed commercially. It includes investigation of service provisioning and service management based on ANs, with respect to customer-provider interactions.

The trend of incorporating programmability into network devices such as firewalls, routers, and remote access is increasing. Cisco and Microsoft have already made some effort to consider network programmability to be a commercial commodity. The Internet Operating System (IOS) router of Cisco provides an API for protocol development on Cisco routers. Microsoft Routing and Resource Service Access (RRSA) provides an API for protocol development on the Microsoft Windows NT platform. IOS and RRAS do not provide a standardised interface for dynamic incorporation of new programmability [18]. Netboost is another company that provides a programmable network application engine with an open API and hardware support for packet processing operations. The goal is to allow third parties to develop their own network services such as virus scan [170].

Nortel Networks, a network technology and equipment provider, supports active and programmable networks by having a product called Accelar Routing Switch with network programming ability. Their Oplet Runtime Environment Software

Development Kit (ORE SDK) enables the injection of customised software, including the ANs Execution Environment into the network device. They argued that ANs need Accelar because the next generation network must have programmability with open API which could be provided by this high performance, open system, and reprogrammable equipment [157].

There are some pros and cons for telecommunication providers to deploy ANs. Some benefits, such as rapid and flexible service provisioning, improved network management, lower roll-out and maintenance cost are appealing. On the other hand the increased potential of compromising system stability is one factor which might prevent them from implementing ANs. For most router vendors the use of ANs is not an appealing option, for the reason that they will lose control due to open architecture and that they can not optimise proprietary hardware [45, 151].

There are several paths undertaken in the future trends of ANs. ANs in telephony, wireless networking, programming abstraction and interfaces for ANs, programming for Quality of Service (QoS), modelling and implementation of new network services, service creation platforms, enabling technologies and languages are some of the trends of research in this area to deploy the concept of ANs in the Internet [69, 128, 108]. However, the requirement for an open architecture and signalling provided by the service provider, in order to enable the deployment of ANs, needs to be driven by a large network community [154].

2.2 Research Methodology

In this subsection, we address the research methodology used in this thesis, which is simulation. First, we explain the reason for choosing simulation to evaluate AN-based protocols instead of using an analytical approach or prototyping. Subsequently, the confidence and credibility issues in using network simulation are addressed. We introduce some available network simulators and describe our reason for choosing ns-2 [51] as our simulation tool. The performance evaluation conducted in this thesis is also discussed in the final part of this chapter before we present the summary.

2.2.1 Research Methodologies in Computer Networks

Research on proposed changes to the Internet is ongoing, including for example to develop new link-level technologies such as wireless and satellite links, new infras-

tructure such as web caching and content distribution networks, and new applications such as streaming multimedia. Research on these proposed changes can be done with mathematical analysis, experiments in testbeds and small-scale deployment in the current Internet (prototyping), or simulation.

Mathematical analysis provides the possibility of exploring a model of the Internet in which one has complete control. This analysis is beneficial to understand the factors affecting the network. In some cases, it has the risk of using a simplified model in which Internet behaviour has been lost [65].

In computer networks there are a lot of issues related to resource sharing. Mathematical analysis for resource-sharing systems is based on queuing theory, in which a queuing model can be classified into the shape of its arrival process/service time/ servers/maximal occupancy. For example the M/M/1/K model illustrates the typical delay and loss performance of a data multiplexer with exponentially distributed (Markovian - M) arrival process, exponentially distributed service time, 1 server, and K customers. Both arrival process and service time can be in the form of exponential (M), deterministic (D), and general (G) distributions [66, 182]. The drawback here is that the analytical approach can only be used for a small-scale network, and the network behaviour which can be analysed is very limited.

Prototyping is another research methodology used to test a new protocol. However, this approach is very expensive and requires a lot of effort, as well as accumulated experience, to build the new system and run it on the testbed. Moreover, it is also time consuming. In addition, in most cases the performance evaluation results of the system can not be immediately generated.

In addition to mathematical analysis and prototyping, simulation has emerged as an established research methodology in computer networks. Simulation could be used to see how an existing system works, to study a non-existing system without building it, and to analyse the effect of different design parameters. Simulation which has been used as a research methodology in this thesis will be discussed in detail in the remaining subsections.

2.2.2 Simulation

Due to the complexity of the network, simulation plays a vital role in characterising the behaviour of the current Internet and the possible effect of the proposed changes. Simulation is essential to explore proposals that have not yet been realised in the Internet, in diverse environments or in large- scale topologies.

Simulation can be used to check the correctness of analysis and allow exploration of complex scenarios. Furthermore, simulation has been known to be very flexible and can be used to validate an analytical model. Simulation can specifically be used to simulate computer networks. Therefore, a network simulator can be a very beneficial tool to model and evaluate new protocols. This leads us to the need to use a widely acceptable and common network simulator.

Law and Kelton [100] classified simulation models as *static* vs *dynamic*, *deterministic* vs *stochastic*, and *continuous* vs *discrete*. A static or dynamic simulation model is based on time in the simulation. A deterministic or stochastic model is determined by the set of inputs and outputs and the specification of relationships in the model. The decision whether to use a discrete or a continuous model for a particular system depends on the importance of individual characteristics and movement of the model (discrete) or it can be treated in aggregate. Most simulation models are dynamic, stochastic and discrete. A discrete event simulator models a system as it evolves over time by a representation in which the state variables change instantaneously at separate points in time, for example the queuing of customers in a bank [12, 100].

Simulation has emerged as a main research methodology used by many scientists working on the core of the Internet development. Recent credible publications such as in *ACM Sigcomm, IEEE Infocom, IEEE/ACM Transactions on Networking,* and *Performance Evaluation Journal*, have shown that simulation has been accepted as a firm research methodology in networking [127]. This is due to the heterogeneity and rapid change in the Internet. Scalability forces us to address questions of topology, traffic generation, and multiple layers of protocol, as well as to pay more attention in picking the underlying models to be explored.

Floyd argued in [64] that Internet research needs specific simulation models for the research question being investigated. For AQM and scheduling as a research problem a typical model would be a dumbbell topology with aggregate traffic. For a unicast congestion control, a single path with competing traffic is a typical model. The supporting measurements such as characteristics of links, queue management along paths, packet-reordering behaviour, packet corruption on a link, variability of delay, and bandwidth asymmetry are also important factors to be considered. For multicast congestion control, a single multicast group in a large topology with router level topology is sufficient. In this system, loss pattern and traffic generation by group members can be used as a supporting measurement [65].

Currently, the core scientists working on the development of the Internet Protocol

(IP) set the common procedure for the research methodology in computer networks. This area has developed at a tremendous speed, so that no common standard can be settled. An example of a case is as follows. In [64] Floyd shows the case of previous research which introduced a very well-known queue management scheme (RED). Floyd admitted that, due to the lack of effort to address some key issues such as different performance scenarios with high packet drop rates and the effect of parameter setting, a lengthy literature was spawned to address that problem and show the limitations of that proposal.

2.2.3 Confidence and Validation in Network Simulation

Validation is the process of assuring that a model provides meaningful answers to the questions being investigated [77]. Validation can also be defined as a process of establishing that the model available is a sufficient representation of the real system [16]. In general, it is easier to verify the correctness of mathematical analysis than the correctness of the software implementation of a complex underlying model.

Pawlikowski et al. [127] argued that a credible simulation study includes two important necessary conditions, i.e. the use of an appropriate Pseudo-Random Number Generator (PRNG) of independent uniformly distributed numbers, and appropriate analysis of simulation output data. They proposed the terminology Terminating State (TS) and Steady State (SS). A researcher can state whether his/her system is terminating or finite time horizons' simulation, in which the simulation needs to be repeated an appropriate number of times, using different statistically independent sequences of pseudo-random numbers as sources of elementary randomness in different replications of the simulation. The aim is to ensure that the sample of collected output data can represent independent and identically distributed random variables. Confidence intervals can also be calculated using standard statistics. Another alternative is to study the behaviour of networks in steady state, which is theoretically reachable by a network after an infinitely long period of time. Data collection at the beginning of a simulation and during initial warm-up periods is not used to calculate steady-state estimates.

Law and Kelton [100] describe the use of two approaches, i.e. the *replication method* in which simulation is conducted by n independent runs, and *batch means* in which one long run is performed and n performance measures are conducted. The results are then used to calculate the mean, co-variance and confidence interval of the data to ensure that they are statistically represented.

The use of a common simulator in network research has been identified to yield substantial benefits, such as improved validation of the existing protocols [28]. It provides a rich protocol development infrastructure and an opportunity to study large-scale protocol interaction, as well as enabling easier comparison of results across research efforts.

2.2.4 Simulation Tools

Simulation is used to obtain quantitative results or to explore more general relationships between network parameters and network dynamics. A large number of network simulators are available to use for Internet research. Each of them is designed for a specific area of research interest for a particular network type or protocol, for example for Asynchronous Transfer Mode (ATM) or multicast protocols.

Many network simulators use a discrete-event simulator as their engine. The availability and support for particular protocols is a major concern in choosing the appropriate network simulator to be used.

REAL [90], National Institute of Standard and Technology Asynchronous Transfer Mode simulator (NIST ATM) [180], Comnet [73], Scalable Simulation Framework (SSF) [41], Opnet [192], and x-sim [27] are among the existing network simulators. Comnet and Opnet are two successful commercial network simulators widely available in many academic institutions, in which simulation languages with network protocol libraries have been provided. SSF emphasises support for routing protocol simulation and targeting scalability. Other simulators are targeted primarily at protocol design and networking research communities.

The advantages of using a supported common network simulator are also a driving force to choose a particular simulator. We have chosen the ns-2 network simulator of the Virtual Internet Testbed (VINT) project at the Lawrence Berkeley National Laboratory (LBNL) for the reasons of free availability and support [51, 28]. In addition to that, the ns-2 network simulator is well validated and widely accepted by the research community. ns-2 has been extensively used by many networking research groups worldwide and has been acknowledged as the framework of many refereed published works.

2.2.5 ns-2 Network Simulator

ns-2 [51] is a powerful, flexible and versatile network simulation environment which consists of C++ core methods and uses Object-oriented Tool Command Language

(OTcl) shells. Additional functionality and network topologies can be written as methods and processes in OTcl scripts. ns-2 has a split programming model in which packet processing is performed by a systems language, whereas simulation setup is done using a scripting language. C++ is used for data per packet events, whereas OTcl is used for period or triggered events. The Tcl interpreter called TclCL is the language used to link C++ and OTcl. Protocols simulated in ns-2 cover all layers from the application layer to the physical layer.

There are two types of applications in ns-2, sources and traffic generators. Sources are used to drive stream transports (e.g. TCP), and enable Telnet (simulates characters typed by user) and File Transfer Protocol (bulk data transfer). On the other hand traffic generators are used to drive connectionless transport (e.g. UDP) to provide exponential on/off times, Pareto on/off times, Constant Bit Rate (CBR) (deterministic rate). ns-2 provides router queue management mechanisms to define how to hold and mark packets such as Droptail, Random Early Detection (RED), and Class Based Queuing (CBQ). It also support various networks such as for Medium Access Control (MAC) layer protocols for LAN simulations, wireless network, and Multi Protocol Label Switching (MPLS). As shown in Figure 2.2, the ns-2 objects can be divided into connector and classifier, which in turn can be used to define queue and agent for example. Figure 2.3 shows the routing mechanism between nodes in ns-2. This figure shows that a node consists of classifiers and multiplexers, and also shows some variables which constitute packet routing mechanism such as queue, dequeue, and time to live (TTL).

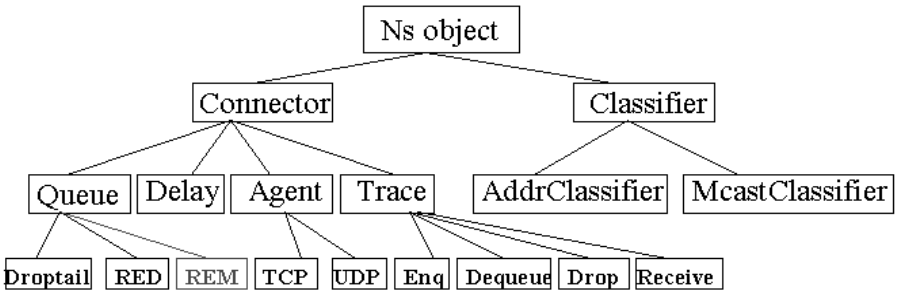


Figure 2.2: ns-2 object [39]

ns-2 has a mathematical support in which a Random Number Generator (RNG)

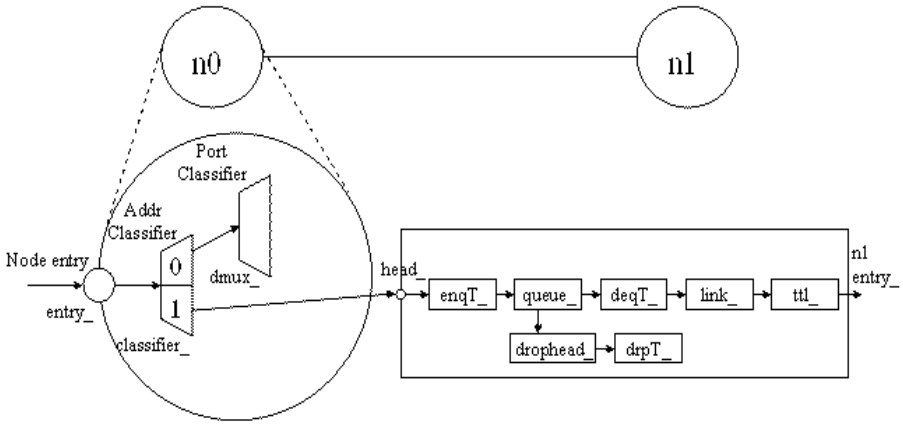


Figure 2.3: ns-2 routing mechanism [39]

is implemented, based on S. Park and K. Miller's work [124]. Distributions of random variables are applied to RNG streams, which can be uniform, exponential, pareto, constant, hyper-exponential, normal, and log-normal. The random variables are generated empirically, driven from trace files or list of values. ns-2 can also collect samples and provide means, variance, sum and count. In order to run independent simulations, it is necessary to re-initialise all simulator objects (simulator clock, event scheduler, traffic objects, etc) or re-execute simulation scripts. As re-initialising simulator objects in difficult in ns-2, it is necessary to make repeated executions of ns-2 scripts. For this purpose we can have a main process (for parameterisation and processing) that can spawn a child process (for simulation and output results to file) written in Tcl scripts.

Network topology and components can be configured with scripts. The trace data for every detail of network behaviour can be collected, processed and visualised. Figure 2.4 illustrates ns-2 trace files which consist of 12 fields describing the event ('r': receive, '+': enqueue, '-': dequeue, and 'd': drop), time of event, origin of the packet, destination of the packet, type of the packet, packet size, flags, flow identity, source address, destination address, sequence number of the packet and the packet identity. ns-2 provides visualisation support in the form of a network animator called nam which has proved to be very helpful in assessing the behaviour of the network [46]. nam is used to verify the topologies generated by Tcl scripts and observe the operations of ns-2 agents. A capture of a nam instance is shown in Figure 2.6.

ns-2 simulation has been used to evaluate the AN-based protocols throughout this thesis. The ns-2 simulation model of ACC TCP is implemented in C++ and

Event	Time	From node	To node	Packet Type	Packet size	Flags	Flow id	Scr Addr	Dest Addr	Seq Num	Packet Id

Figure 2.4: ns-2 trace file [39]

OTcl. It enables us to trace important variables used to evaluate the Random Early Marking (REM) algorithm in the ACC TCP. In AER/NCA, an ns-2 extension in C++ and OTcl has also been used to do error recovery and congestion control. OTcl has been used in the entire extension of ALMA protocol's ns-2 extension, except for a few extended C++ files to calculate the average queue occupancy. Figure 2.5 shows the ns-2 simulation and the results generation mechanism in which trace files are used for analysis. The nam network animator is used to characterise the dynamic behaviour of the protocol. The programming environment for our ns-2 simulator comprises the Linux Operating System 2.2.15.

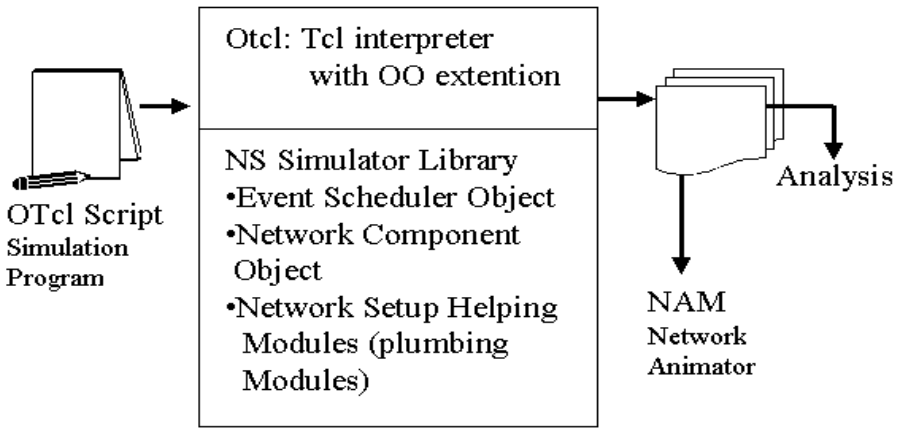


Figure 2.5: ns-2 simulation environment and result generation procedure [39].

ns-2 is a well-validated simulator and the validation is performed as part of the program construction. In this case, detailed sets of tests, applications and protocols are included, and the results are validated by comparing the input with known output. The validation stage in ns-2 expands confidence in network simulations [77], as the customisation validity of the simulation studies in computer and telecommunication networks is of utmost importance [127].

Performance evaluation of a network protocol can be approached from different ways. Network protocol evaluation can be done with a testbed network such as Dummynet [134]. Dummynet is a link emulator executable on FreeBSD machines

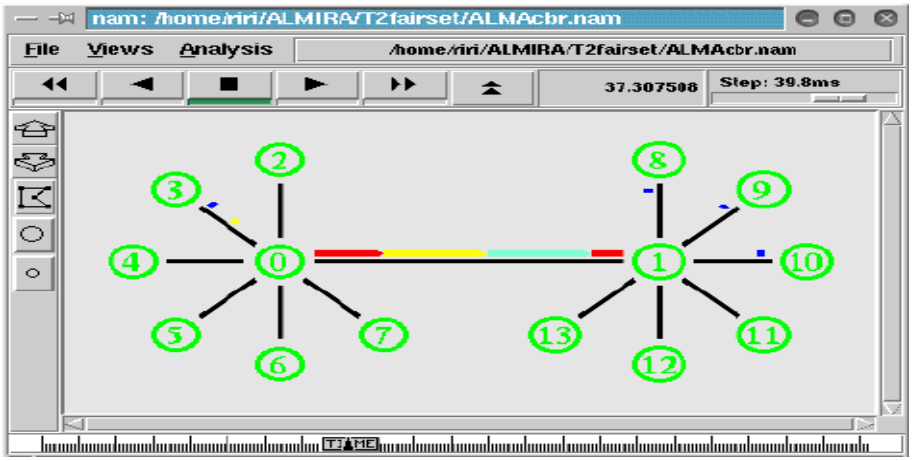


Figure 2.6: nam visualisation

acting as a bridge between two routers. The queue and the congestion can be emulated on the link. However, in this work, we do not use emulation for reasons of lack of resources and infrastructures, as well as experience within the research environment to do the emulation and prototyping.

There are different ways to test a new protocol in relation to our main goal to approach the congestion control problem using ANs. AN-based protocols can be prototyped into the ANTS execution environment which can be run on top of the Abone AN testbed. However, among all of the proposed unicast and multicast protocols evaluated in this work, only AER/NCA has an ANTS implementation due to the extensive resources needed to perform the prototyping. In addition, the scale of the work to deploy new protocols into their ANs prototype is beyond the scope of this research.

We believe that performance evaluation using the ns-2 network simulator is a part of a common standard procedure in network protocols development today. The main objective of the network simulation is to ensure that the functional requirements of the new protocols can be achieved and are feasible. Therefore, after considering the available resources for our research, we decided to use ns-2 as the performance evaluation platform.

2.2.6 Performance Metrics

A wide range of performance metrics have been used throughout this thesis. The following is the list of the metrics used:

1. *Throughput.* Throughput is defined as the number of data packets received at each sink endpoints in a unit of time (in Kbps). For our multicast multirate experiments, rate gained by a multicast flow indicated the throughput and bandwidth used by that flow. High throughput is a desirable characteristic of a protocol.

2. *Average queue length.* The average queue length of a buffer indicates the capacity of the queue management scheme to absorb packets in order to prevent congestion.

3. *Normalised throughput.* Normalised throughput is defined as the sum of throughput gained over the maximum possible throughput. The maximum possible throughput is the bandwidth of the bottleneck link. The closer the normalised throughput to one, the better the bandwidth utilisation of the protocol.

4. *Normalised delay.* Normalised delay is defined as the sum of the total delay of each link divided by the mean packet transmission time (delay) [72]. The packet transmission time (per packet delay) is the function of packet length (packet size * 8 bits) over the maximum available bandwidth.

5. *Packet Loss Ratio (PLR).* Packet Loss Ratio is defined as the number of total packet drop/loss over the total number of transmitted packets during the simulation time. A low PLR is expected from a protocol in order to provide a good quality of service.

6. *Packet Retransmission Ratio (PRR).* Packet Retransmission Ratio is defined as the number of retransmitted packets over the number of transmitted packets during the simulation time. The higher the PRR, the lower the bandwidth utilisation, as the number of packet drop and subsequently need to be retransmitted is high. Retransmitted packets cause a waste of bandwidth.

7. *Jitter.* The time from when a packet is generated at the source until it is received at the receiver can fluctuate from packet to packet due to the varying queuing delays within the network. This phenomenon is called jitter (the variation of delays). In this thesis, jitter is shown by the graph of packet sequence number versus time. First, for each packet sequence number, the difference between the event time of that packet and the last packet received time is calculated. Next, the result is divided by the difference between the

current packet sequence number and the last packet received sequence number. Hence, the result is used to plot the jitter for each packet sequence number.

8. *Bandwidth Usage.* Bandwidth usage is defined as the ratio of total number of AN packets over the total number of packets transmitted.

9. *Inter-protocol Fairness.* A quantitative description of fairness used in this thesis is based on Jain Fairness index [76, 84]. It is calculated based on the throughput of each flow. The Jain Fairness equation is explained further is Chapter 3 (Equation 3.3).

10. *Layer Subscription.* In multirate multicast protocol we use layer subscription level to evaluate the bandwidth utilisation of each receiver in conjunction with the number of layer it can subscribe. In this case, we can also analyse the convergence time which is the time taken by receiver to adjusts its reception rate to a stable state.

2.3 Summary

This chapter introduced ANs as an enabling technology to provide solutions to networking problems. From the literature, it can be summarised that ANs are a promising network paradigm to accelerate the pace of innovation and to support further deployment of services due to user-driven customisation of infrastructure. Work published by Tennenhouse et al. [159] introduced the ANs approach to improve the Internet in terms of shortening the protocol stardardisation period. Many prototypes and works have emerged and been developed since the inception of ANs in 1996.

Several aspects of ANs development have been reviewed in this chapter. We reviewed the architecture, prototypes, applications, and commercial effort for ANs. The deployment of varieties of network services using ANs has been performed by researchers all over the world [4, 33, 117], taking into account the issues of policy, safety, security, scheduling, resource allocation between flows [131], global resource management, and evolutionary improvement to the current network infrastructures.

We have described our reasoning for choosing simulation as our research methodology. The advantages of network simulation, rationale, as well as the driving force in choosing a common network simulator for research, have been described in detail. The confidence and validity of using a network simulator have been addressed. ns-2 as a discrete event simulator used in this work has been presented in detail.

We reviewed the ns-2 network simulator of the Lawrence Berkeley National Laboratory [51, 10] as the simulation tool. Indeed, another reason for us to choose this simulation tool is the availability of the ns-2 extension for ACC TCP, AER/NCA, and ALMA protocols, which will be discussed in detail in Chapters 4 to 6, respectively. The use of a common network simulator on which all the protocols to be evaluated and extended are based will ensure the confidence and validity in the performance comparison and evaluation. We do not perform prototyping during the course of our research, considering the complexity of the protocols, and the breadth of the problem.

This background research on ANs and the use of simulation as a research methodology leads us to the next chapter in which we will discuss congestion control as a problem domain. A coherent research programme that investigates the AN-based unicast and multicast protocols available based on the ns-2 network simulator will be discussed in Chapter 3. Furthermore, we incorporate networking issues such as active queue management, TCP-friendliness, and layered multicast into the evaluation of AN-based protocols. The rest of Chapter 3 describes the protocols investigated in this research in detail. The results of our simulation experiments along with traffic topology and classification are subsequently presented in Chapters 4, 5 and 6.

Chapter 3

Survey on Active Network-based Congestion Control

This chapter reviews congestion control and the use of Active Networks (ANs) to solve the problem of congestion. AN-based congestion control protocols are presented according to the classification of the modes of operations, i.e. unicast, multicast single rate, multicast multirate. This chapter also provides the background to the AN-based congestion control protocols evaluated in this thesis. We review the congestion control advancement and the current Internet congestion control which includes the TCP/IP protocol, Active Queue Management (AQM), TCP-friendliness, and pricing mechanism. Section 3.1 introduces the congestion control and the advancement of congestion control schemes. Section 3.2 discusses the deployment of ANs into unicast and multicast congestion control. Section 3.3 reviews the protocols evaluated in this thesis, which include ACC TCP, AER/NCA and ALMA protocols. Finally, in Section 3.4, a summary of the chapter is presented.

3.1 Congestion Control

Congestion control is a vital element to the Internet to avoid undesirable situations such as congestion collapse. In simple terms, a resource is congested when the total sum of demands is more than its available capacity. Congestion control is necessary when the resources in a network need to be allocated so that the network operation performance level can reach the acceptable level when the demand exceeds or is near the capacity of the network resources [82]. Yang and Reddy [189] classified the congestion control algorithms into closed loop and open loop schemes, which mostly covered the mechanisms found in the literature.

The problem of congestion has been considered a high priority topic in computer networks and became more urgent with the growth of the Internet and its applications, as well as the limitation of the network resources to share. The problem has been addressed in different ways to overcome the congestion in different network platforms, i.e. IP-based, ATM-based[1], and in Integrated (IntServ) and Differentiated Service (DiffServ) environments. In this thesis, we consider a congestion control protocol for TCP, considering that it is the predominant traffic in the Internet and constitutes 90% of its total traffic[2].

Recent publications on Internet congestion control protocols [57, 107] consider key advances which assume the explicit modelling of the congestion measure that communicates back to data sources the information on congestion in the network resources being used. The congestion measure for each network link is called price, and sources have an aggregate price of links in their path. Pricing became a promising technique to control the congestion in communications networks, as it directly provides the information needed to allocate scarce resources during times of congestion.

There are several possible views on the future of congestion control in the Internet [59], i.e. 1) availability of infinite bandwidth causing no congestion; 2) the continuity of the current Internet which is best effort traffic having co-operative end-to-end congestion control; 3) ubiquitous per-flow scheduling (the game theory view) with users optimising their own utility functions; 4) congestion-based pricing for differentiated services in which control is performed through pricing; 5) back to use virtual circuits; and 6) the occurrence of congestion collapse. In this section, we address congestion control issues in the current Internet, in which we consider TCP congestion control, active queue management, TCP-friendliness, and pricing. The pricing schemes to control congestion are covered in this thesis to introduce the third and fourth views to the future of congestion control. We believe that some intelligence and computational power in the routers, such as the one proposed by ANs, will be able to help the deployment of pricing to a competitive and non-cooperative environment such as the Internet.

3.1.1 Congestion Control Advances

The original version of TCP did not contain any congestion control mechanism. In [81], Van Jacobson described a congestion collapse prevention mechanism by detect-

[1] http://www.atmforum.org

[2] http://www.caida.org - The Cooperative Association for Internet Data Analysis

ing and controlling congestion, and introducing the congestion window (CWND)[3], slow-start, and the Additive Increase Multiplicative Decrease (AIMD)[4] mechanism. He added a fast retransmit mechanism to reduce dependency on the retransmit time out, and called it TCP Tahoe. The fast recovery mechanism was subsequently included in TCP Reno. In 1994, Brakmo [26] introduced TCP Vegas, in which he proposed a new congestion detection and avoidance mechanism. In 1996, selective acknowledgements was proposed for TCP as a new flavour called TCP Selective Acknowledgements (SACK). In addition, TCP New Reno, TCP Daytona [138], and TCP Santa Cruz [125] are some of the new flavours of TCP in which the increase in performance of the protocol is the target of the design. Recent studies in [65] identified more that 400 different implementations and versions of TCP. The dominant family of TCP congestion control in the Internet today is New Reno and SACK.

3.1.2 Current Internet Congestion Control

Paxson and Floyd [65] describe the invariant properties of the current Internet. They state that the Internet has 24-hour traffic pattern cycles, log-normal connection sizes, heavy-tailed distribution of connection sizes, Poisson arrivals for start times of user sessions, self-similarity in traffic pattern[5] , invariants in topology, and heterogeneous characteristics. These factors lead to a complexity in analysing and simulating the whole Internet, and difficulties in deploying new protocols.

Our discussion on Internet congestion control covers the basics of TCP congestion control mechanisms, active queue management, TCP-friendliness, and pricing. Internet traffic, 90% of which is TCP-based, shows burstiness due to the heavy-tailed ratio of file size. Most TCP connections are 'mice' with a short burst and require low latency, but a few are 'elephants' which are bandwidth consuming but tolerate latency. By flooding the network and causing queue overflow, the elephants undermine the mice resulting in unnecessary loss and long queue delay [107].

The essence of current TCP congestion control is that the TCP sender adjusts its sending rate according to the rate (probability) of packets being dropped. TCP Tahoe and Reno use packet loss to measure congestion. On the other hand, TCP Vegas uses the queuing delay as a measure of congestion. This queuing delay in-

[3]Congestion window is the number of bytes the sender may transmit. CWND is used to control the packet transmission from the sender.

[4]AIMD is a TCP congestion avoidance mechanism in which *packet loss is used as an indicator of congestion.* On packet loss, CWND is divided by 2 (multiplicative decrease). On the other hand, on successful acknowledgment, increase the CWND by 1 (additive increase).

[5]A good discussion on congestion control and self-similar traffic can be found in [73]

formation is updated by First In First Out (FIFO) buffer management. Queue management[6] has also long been used as a congestion control measure [54], in which queue size-based congestion control has been employed as a feedback congestion control system.

Considering that not all Internet applications use TCP, we have to note that they do not follow the same concept of fairly sharing the available bandwidth in a common bottleneck. The emerging audio/video streaming applications and other types of real-time applications would compete with the TCP-flows unfairly upon congestion if a definition of a TCP-friendly requirement is not introduced to new Internet protocols. In other words, a desirable situation in which fair distribution of bandwidth can be achieved would only be possible if the TCP-friendliness mechanism is considered when designing new protocols.

Gevros et al. [68] pointed out that historically the congestion problem has been viewed from different points, i.e. applying queuing theory, control theory, and economic theory (game theory). Market solutions are used to solve the congestion problem in which the rules of the game can be changed (for example, price changes). In order to address this, *pricing* has been an interest of the research community since the late 1990's.

In the following subsections, we will address TCP congestion control, active queue management, TCP-friendliness, and pricing, in order to introduce the issues related to the current Internet congestion control.

3.1.2.1 TCP Congestion Control

Since the introduction of TCP, several improvements have been made to its congestion control algorithm such as using fast retransmit (TCP Tahoe) and fast recovery (TCP Reno). Fairness and stability of TCP congestion control mechanisms are crucial for the overall network performance [75]. The development of congestion control mechanisms in the Internet enabled it in becoming today's best effort network model [68]. The modelling, validation, and estimation of the performance of TCP are also becoming a major research concern [14, 191]. The need to evaluate TCP performance in future networking environments is covered in [13, 95, 76].

TCP was originally designed to provide reliable data delivery over conventional networks [50]. Generally, congestion control algorithms use packet loss to detect congestion. The packet losses cause retransmission, oscillatory behaviour of the source input rate, and link under-utilisation [112]. Most congestion control schemes

[6] In this work, we use the terms queue management and buffer management interchangeably.

consist of a feedback mechanism and a control mechanism. In designing congestion schemes it is important to apply the principle that if the control is slower than the feedback, the system will be slow to respond to changes. Therefore the control interval should be carefully selected.

The TCP congestion control can be described as follows. There are two distinct phases in TCP congestion control, i.e. Slow Start (SS) and Congestion Avoidance (CA). The TCP initialisation during the start-up is called SS phase in which the congestion window (CWND) is set to 1 TCP segment (usually 512 bytes). The SS phase happens both in the beginning of the connection or when a time-out occurs. During this SS phase and when the acknowledgement of a transmitted segment is received, the CWND is doubled once every Round Trip Time (RTT). When the CWND reaches the value of a parameter called THRESHOLD (normally 64 Kb), then the CA phase is reached. TCP uses the Additive Increase Multiplicative Decrease (AIMD) mechanism at this stage to react to congestion, in which the CWND is increased linearly by one unit (segment) per RTT. On the occurrence of time-out at the sender, the THRESHOLD value is set to half of the current CWND, and subsequently CWND is set to 1 segment [86]. Subsequently, TCP re-enters the slow start phase.

When the offered load increases above the network capacity, packets are eventually dropped. TCP detects a packet loss through the receipt of a number of duplicate acknowledgements (ACKS) from the receivers. After receiving a number of duplicate ACKS (usually 3), TCP infers the occurrence of a packet loss and immediately decreases its transmission rate by adjusting its THRESHOLD value to half the current CWND value and retransmits the missing segment. The window size is set to THRESHOLD + 3 segments. This halves the CWND and compensates for the TCP segments that have already been buffered by the receiver. For each duplicate ACK, CWND is incremented by 1, and a segment is transmitted if allowed by the value of CWND. When the (non-duplicated) ACK arrives that acknowledges new data, the CWND is set equal to the THRESHOLD value, and continues with the linear increase (congestion avoidance). This mechanism is called *fast retransmit* which is implemented together with the *fast recovery* algorithm. *Fast retransmit* happens when TCP receives duplicate ACKS and it decides to retransmit the segment, without waiting for the segment timer to expire. *Fast recovery* happens once the lost segment has been transmitted, hence TCP tries to maintain the current data flow by not going back to slow start and adjusts the window for all segments that have been buffered by the receiver [153, 52]. Figure 3.1 shows the evolution of the TCP

congestion window.

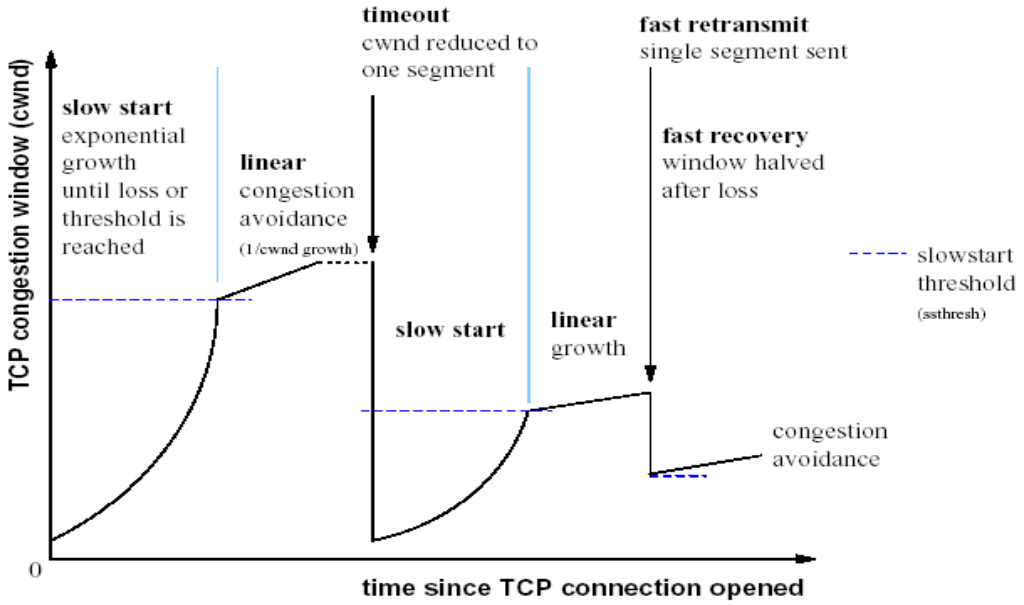


Figure 3.1: Evolution of TCP congestion window [181]

Several proposals have been prepared to overcome the problem of congestion in TCP, for example using Random Early Detection (RED) gateways, and Explicit Congestion Notification (ECN) [63, 132]. ECN uses routers to notify endpoints of congestion, but applies corrective actions from the endpoints. Some efforts to solve congestion related to transport protocols such as TCP can be seen in [63, 55]. Recent developments in TCP congestion control can be found in [58].

3.1.2.2 Active Queue Management (AQM)

A queue management algorithm traditionally manages the length of packet queue in the router by dropping packets only when the buffer overflows. The technique called 'Droptail' enables the router to accept packets until the maximum size is exceeded. The Droptail queuing mechanism treats all traffic equally and is the simplest queuing mechanism. Droptail buffer management forces network operators to choose between high utilisation (large buffers) and low delay (small buffers).

In addition to this traditional queuing algorithm, several active queue policies such as Random Early Detection (RED) [63] and Random Early Marking (REM) [6] have been introduced, in which the congestion problem is addressed in a proactive manner. Some other adaptive [5] and efficient [155] queue management schemes have also been introduced such as Blue [52], Stabilised RED (SRED) and Dynamic

RED (DRED) [192]. Blue uses the instantaneous queue length and link utilisation as indicators of traffic load and congestion. In SRED the estimated number of active flows and the instantaneous queue size are used to compute the probability of packet drop. DRED employs a feedback control approach to randomly discard packets in the queue in order to stabilise the actual queue size at a predetermined target.

The main purpose of AQM is to provide congestion information for sources to set their transmission rates. RED controls its average queue size to manage its transmission rate. It drops packets randomly prior to the period of congestion, and informs the routers to reduce their transmission rate.

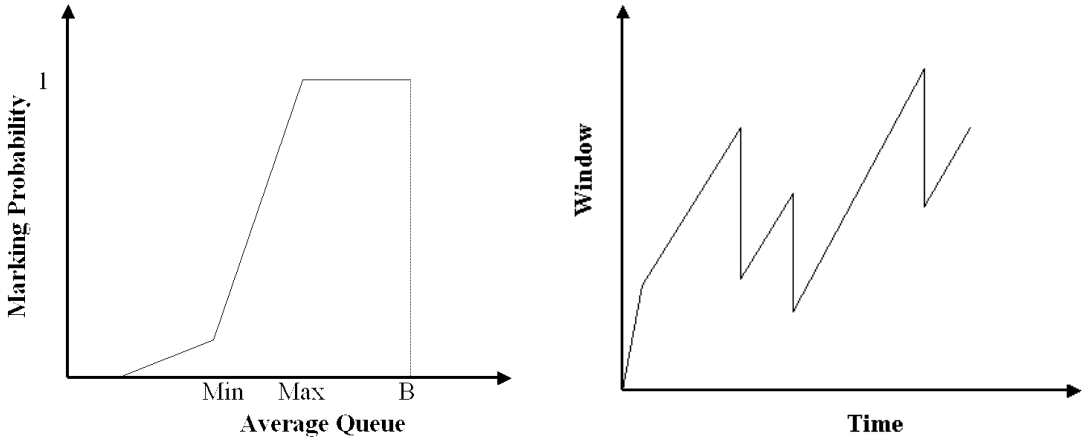
RED and REM are primarily different in the way they detect and measure the network congestion. Note that marking in RED means a slow-down request to anticipate congestion somewhere in the path to its destination, whereas in REM, the marking allows a source to estimate the aggregate shadow price of its path. In the following subsections we will describe RED and REM in more detail.

RED

The RED algorithm warns sources of congestion by probabilistically marking and dropping packets. RED promotes fairness by dropping random packets before the queue becomes full. RED performance could be enhanced by sensitive parameter setting and desynchronisation, which prevent bursty loss and buffer overflows [63]. RED has been widely implemented in commercially available routers and designed to replace the traditional Droptail queuing management. This is due to their capacity in avoiding the global synchronisation problem in which Droptail tends to drop packets simultaneously, causing low throughput and high delay jitter [80].

RED's parameter set {`avg, min, max, maxp,` and `wq`} has been used to probabilistically drop packets arriving at the output port. RED maintains an average queue size (`avg`) which is computed as an exponential weighted moving average of instantaneous queue (`q`) and weight parameter `wq`. In other words,`avg` can be computed as $avg \Leftarrow (1-wq)avg+wq*q$. `max` and `min` show the maximum and minimum dropping/marking threshold, whereas `maxp` is the maximum packet dropping probability. In the condition in which the `avg` exceeds minimum preset threshold value (`min`), RED drops or marks packets randomly. All the arriving packets are dropped when the average length reaches the maximum preset threshold `max`. Figure 3.2 shows the link and source algorithm of the RED AQM scheme. The link algorithm describes what to feedback (using RED) and shows the packet dropping probability in RED as a function of the average queue size. On the other hand, the source

algorithm decides how to react to congestion (based on TCP Reno). Figure 3.2(b) depicts the evolution of TCP Reno's congestion window over the time which shows its congestion control mechanism such as fast recovery.



a) Link b) Source (TCP Reno)

Figure 3.2: Link and source algorithm of RED [6]

The RED algorithm can be simplified as follows:

For every packet arrival {

Calculate avg
if (`avg = max`) {
Drop the packet
}
else if (`avg > min`) {
Calculate the dropping probability pa
$pb = \frac{maxp*(avg-min)}{max-min}$
$pa = \frac{pb}{(1-count*pb)}$ [note : count is the number of packets accepted since the last packet drop]
Drop the packet with probability pa' otherwise forward it
}
else {
Forward the packet
}
}

REM

REM was introduced by Low et al. [106, 190] which views the feedback congestion control system as a duality model: the interaction between the end system and the network. REM is a new AQM scheme that has features to match user rate to

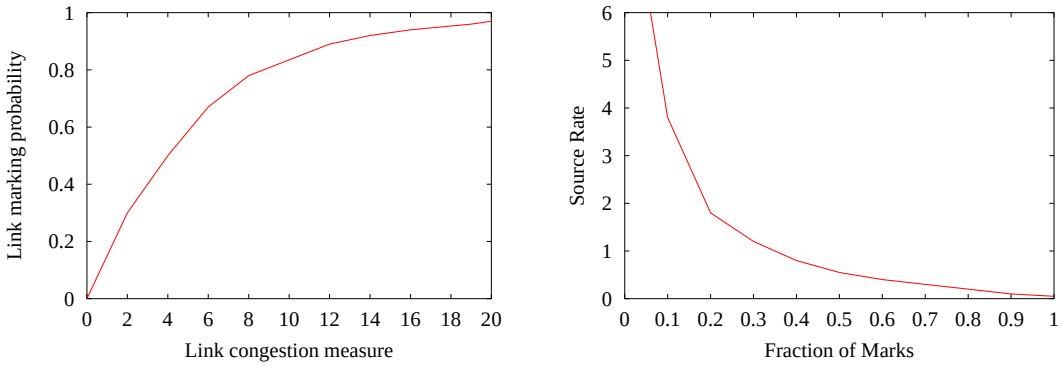
network capacity while clearing buffers, in addition to the sum prices which are the end-to-end marking (or dropping) probabilities observed by a user [6]. It depends largely on the sum of link prices. REM attempts achieving high utilisation as well as negligible packet loss and packet delay. The behaviour of REM-based networks has been designed to ensure fairness and stabilise the network, for the reason that constant bandwidth probing is not necessary in REM.

REM measures congestion in *price,* which is a periodically updated value based on *rate mismatch* (difference between input rate and link capacity) and *queue mismatch* (difference between queue length and target). This is to enable the first idea of REM *(match rate clear buffer)* which is to stabilise both the input rate around the link capacity and queue around a small target (regardless of the number of users). Figure 3.3 exhibits the link algorithm of the REM AQM scheme which shows the exponential link marking probability as the function of link congestion measure. This is to underline the second idea of REM *(sum prices)* which is *to use the sum of the link prices along a path* as a measure of congestion in the path, and to embed it in the end-to-end marking probability that can be observed at the source [6]. The output queue marks each arrival packet not already marked at an upstream queue, with a probability that is exponentially increasing in the current price. Considering that the sum of prices which is a precise measure of congestion in the path is embedded in the end-to-end marking probability, it can easily be estimated by sources from the fraction of their own packets that are marked (Figure 3.3(b)), and used to design their rate adaptation.

If the number of users increases, the mismatches in rate and queue grow pushing up price and marking probability. This sends a stronger congestion signal to the sources, which then reduce their rates. If the source rates are too small, the mismatches will be negative, pushing rates, until eventually the mismatches are driven to zero, yielding high utilisation and negligible loss and delay in equilibrium. The buffer will be cleared in equilibrium if the target queue is set to zero [6].

In order to see the impact of queue management models, we will take into account an example in which a set of S sources share a network of links l. Upon a transmission at rate $x_s(t)$, which is a function of period t, each source s attains a utility function $U_s(x_s)$. A congestion measure $p_l(t)$ is updated by each link, and each source updates its rate $x_s(t)$ according to the sum of $p_l(t)$. The following vector forms represent the relationship:

$$x(t+1) = F(x(t), p(t)) \tag{3.1}$$



a) Link b) Source

Figure 3.3: Link and source algorithm of REM [6]

$$p(t+1) = G(x(t), p(t)) \tag{3.2}$$

The function F models the source algorithm such as TCP Reno or SACK [50], whereas function G models queue management. Each TCP/AQM scheme that attempts to optimise the network usage can be characterised by the function of (F, G, U) to describe the source rates, congestion measure, and the utility function. For the purpose of comparison of RED and REM, only function G (link algorithm) is elaborated here, due to the fact that function F only models the source algorithm.

Table 3.1 shows RED and REM from the perspective of the duality model, where the parameters mean the following: *p(t)*: congestion measure at the link; γ=stepsize > 0; *x(t)*: aggregate source rate at bottleneck link; *b(t)*: queue length; α is a constant > 0 [106]; and c is a constant (true link capacity). The $\alpha b(t)$ part of the equation for REM refers to the *clear buffer* mechanism, whereas the $x(t)$-c part refers to the *match rate* mechanism. The expression shows a periodically updated value based on rate mismatch (input rate ($x(t)$) minus link capacity (c)), and queue mismatch (difference between queue length and target buffer ($\alpha b(t)$)). In equilibrium, the buffer must be empty ($b(t)$=0 (zero buffer)) and the link is full ($x(t)$=c (full utilisation)). Further discussion on the derivation and explanation of this comparison can be found in [106].

REM proved to be capable of detecting network congestion more accurately than RED [106], and therefore maintains a smaller queue occupancy at the gateway. Consequently, REM is believed to provide a more robust and a better end-to-end performance. The comparison of RED and REM on TCP Reno performed in [7] has shown that REM's key feature is low buffer occupancy at steady state, while

achieving high utilisation at the same time as the number of connections changes.

Queue Management	Congestion Measure p(t)	Function G
RED	Queue	$[p(t) + x(t)-c]^+$
REM	Price	$[p(t)+\gamma(\alpha b(t)+x(t)-c)]^+$

Table 3.1: Comparison of RED and REM [106]

Recall that many AQM proposals are found in the literature such as Blue and Adaptive RED. However, in our work in Chapter 4, we only compare the RED and REM performance on an AN-based congestion control protocol for the reason that REM is a recently introduced promising AQM mechanism. RED has been chosen because it is a prominent and deployed example of an AQM scheme.

3.1.2.3 TCP-friendliness

Many protocols have been designed with the aim to overcome the reliability and scalability problem. Their 'peaceful' coexistence with TCP (TCP-friendliness) and their fairness characteristics will ensure smooth deployment of new protocols into the Internet. Deploying non-congestion-controlled applications in the Internet might result in extreme unfairness/unfriendliness towards competing TCP traffic due to its rapid back-offs during the congestion control period.

Non-TCP flows should be *TCP-friendly* when they are competing with TCP flows for bandwidth, in which the arrival rate of non-TCP flows should not exceed that of a conformant TCP flow in the same circumstances [60]. In other words, they should be responsive to network congestion as well as being friendly to the competing TCP application in terms of bandwidth sharing.

Many proposed unicast and multicast congestion control schemes try to conform to these TCP-friendliness criteria. In addition, TCP-friendliness is a desirable characteristic for both unicast and multicast flows. A *unicast flow* is defined as *TCP-friendly* when it does not reduce the long-term throughput of any co-existent TCP flow. A *multicast* flow is considered *TCP-friendly* when, for each sender-receiver pair, the multicast flow has the property of being unicast TCP-friendly [177]. TCP-friendliness is a subset of *fairness*, as it deals with the *inter-protocol fairness* with TCP[7].

[7]Throughout this thesis, the TCP-friendliness term is used interchangeably with inter-protocol fairness. There are different approaches to explain TCP-friendliness and its definition which can be found elsewhere in literature. However, in our work the TCP-friendliness evaluation is performed only to evaluate the congestion control mechanism in AN-based protocols.

TCP-friendliness is a desirable characteristic that needs to be achieved in unicast and multicast protocols. *Equation-based congestion control* protocols have been proposed to ensure TCP-friendliness in many works [122, 62, 61, 165]. TCP-friendly Multicast Congestion Control (TFMCC) [74, 178] and Pragmatic General Multicast Congestion Control (PGMCC) [135] protocols are prominent recent examples of single rate multicast congestion control protocols. We consider TCP-friendliness in our work in relation to the purpose of evaluating the congestion control mechanism in some AN-based protocols. A pure end-to-end approach can not easily promote TCP-friendliness, and therefore network elements and ANs support have been used in achieving this desired property. Inter-protocol fairness and bandwidth allocation have also been widely researched, such as in [3, 37, 103].

There are some definitions of fairness available in the literature for transport protocols, for example basic max-min fairness, proportional fairness and the *Jain fairness index* [76]. Basic max-min fairness underlines that a protocol should make maximal usage of available network resources, but be able to share such resources equally amongst data streams. Proportional fairness aims to maximise the use of network resources, in which every network resource has a 'price' associated by the network administrator and a routing priority is assigned to a given stream. The Jain fairness index is based on Equation 3.3 [76, 84]. In Chapters 5 and 6, we will use Jain's fairness index equation to evaluate the AN-based multicast protocols.

$$fairnessindex = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2} \tag{3.3}$$

in which x_i is the throughput of session i on a bottleneck link, and n is the number of sessions sharing the bottleneck link. The fairness index lies between $1/n$ to 1. Locally equal partitioning of bandwidth achieves an index of 1. If only k of n flows receive equal bandwidth (and others get none) then the index is k/n. Some properties of the index include the population size independent, scale and metric independent, bounded on [0..1], and continuous.

Many fairness studies were conducted by using the well-known single bottleneck topology in which the average throughput of the new protocol should closely match the average TCP throughput. A further study has also been conducted [184] which proposed the three mechanisms, i.e. rate-based controller, selection mechanism of representative, and the switchover mechanism of the flow and congestion control schemes, in achieving fairness in their reliable multicast protocol. Their inter-session fairness with RED and Droptail routers is promisingly high, as it presents a co-

operation between a flow control and a congestion control scheme.

Most fully reliable multicast protocols use the slowest receiver's rate to be the average group rate in order to provide an advantage of TCP-friendly mechanisms. This is performed by having a TCP rate estimation function using the loss rate and RTT estimation. In single rate multicast protocols, the round trip time and loss information are very important for the sender to determine its rate. If the sender does not exceed this rate for any receiver, then the protocol should be TCP-friendly. For multirate multicast protocol, the detail of mathematical analysis and impact of multicast layering on network fairness can be found in [137].

3.1.2.4 Pricing

Pricing has become an important topic in network congestion control due to the exponential growth of the Internet and the pervasive diffusion of the TCP/IP paradigm to the transport of data and real time traffic. Pricing directly provides the information needed to allocate scarce resources during times of congestion to those users who value them most. Pricing is a promising technique to control the congestion in communications networks.

Some researchers consider pricing as an idea in which users pay a price in order to obtain some bandwidth. The development of the pricing methodology and its evaluation has become important in achieving a global optimum in utility and inter-protocol fairness.

A user x will assume that a network is congested if x's satisfaction reduces due to a modification of the performance of his/her network connection. The pricing and cost approach to Internet congestion control is aimed at heterogeneous receivers, in which the players are the users and the network manager, and a special price function can be adopted to shape the distribution of the scarce resources in order to achieve global optimum that is close to Nash equilibrium [145, 146]. This is where mathematics and game theory meet the Internet [123, 183] in which the optimisation problem in congestion control is encountered [87].

In computer networks, the discussion of pricing concentrates on usage-based pricing in achieving optimal efficiency. The network resources are used efficiently if they maximise the total user satisfaction in which the charges must equal the marginal cost of usage. This is the cost of congestion in the form of performance penalties that one user's traffic imposes on other users. Pricing becomes related to congestion control in the network to determine the optimal rate of the user's flows which maximise the aggregate source utility.

Gibbens et al. [70] also stated that the user's utility is maximised by the network allocating scarce network resources according to a weighted proportional fairness criterion. The network shares resources in proportion to how much the users are willing to pay. Shenker et al. [40] showed that it is possible to set graduated prices so that every user is more satisfied with the combined cost and performance of a network. Thus prices allow network managers to distribute the resources according to the service needed by the user. In a particular model, prices can be set to encourage high utilisation of network resources as well as a fair distribution (reactive flow control approach).

3.1.3 Congestion Control for Unicast and Multicast Protocols

Congestion control protocols can be classified with respect to many characteristics. The congestion control taxonomy of Yang and Reddy [189] divided the approaches in to non-feedback (open loop) and feedback congestion control (closed-loop). Non-feedback congestion control is based on the good design of the network protocol, such as when to accept new traffic, when to discard packets, and scheduling. Feedback congestion control methods can be applied, for example by monitoring the system to detect when and where congestion occurred, passing the information to places where action can be taken, and adjusting the system operation [156]. Propagation delays could have an adverse impact on the stability of feedback control algorithms in high speed communications.

A nomenclature has been devised in [18] for congestion control mechanisms in which the presence of network support and the type of data transfer is the parameter. When the network support is present and the data transfer is reliable, DECbit [82], Explicit Congestion Notification (ECN), Active Congestion Control (ACC), and the exponential backoff scheme can be applied. Bhattacharjee [18] proposed Application Specific Congestion Control (ASCC) for unreliable data transfer. For the current Internet, where the network support is absent, both RED and FIFO (Droptail) algorithms are used, whereas exponential backoff is used for reliable data transfer. This nomenclature is shown in Table 3.2.

In this subsection, we review the congestion control protocols with respect to their type of modes of operations and whether they operate in single rate or use a multirate approach. The unicast congestion control protocol has been defined by evolutionary enhancement to TCP, in which the connection-oriented network

Network Support/Data Transfer	Reliable	Unreliable
Present	DecBit, ECN, ACC, Exponential Backoff	ASCC
Absent	RED, Droptail, Exponential Backoff	RED, Droptail

Table 3.2: Congestion control space [18]

transports data from one sender to one receiver. Multicast congestion control deals with one to many connections.

Protocols' classification can be based on whether they are operating at a single rate or multirate. All unicast protocols are obviously single rate, in which data are sent to all receivers at a single rate, whereas multicast protocols can be either single rate or multirate.

The following are some example of protocols which can be found today in the literature for unicast, single rate multicast, and multirate multicast flows:

1. *Unicast*: Active Network-based Congestion Control Transmission Control Protocol (ACC TCP) [47, 49], TCP Friendly Congestion Control Protocol (TFRC) [62, 177].

2. *Single Rate Multicast*: Active Reliable Multicast (ARM) [104], Pragmatic General Multicast Congestion Control (PGMCC) [135], TCP friendly Multicast Congestion Control (TFMCC) [178], Reliable Multicast Active Networks Protocol (RMANP) [33], Active Error Recovery/Nominee Congestion Algorithm (AER/NCA) [88], Dynamic Replier Reliable Multicast Protocol (DyRAM) [109], Active Multicast Protocol (AMP) [11, 174], Active Hierarchical Congestion Control for Large-scale Multicast (AHCCM) [9].

3. *Multirate Multicast*: Receiver-driven Layered Multicast (RLM) [114], Receiver-driven Layered Multicast Congestion Control (RLC) [166], Packet-pair Receiver-driven Layered Multicast (PLM) [102], Active Traffic Control Mechanism for Layered Multicast (ATML) [85], Multicast Enhanced Loss Delay-based Adaptation (MLDA) algorithm [149], Coding Independent Fair Layered Multicast (CIFL) [91], Active Layered Multicast Adaptation (ALMA) protocol [187].

Next, we will discuss multicast congestion control protocols and their advances from single rate to multirate modes of operations. Then, we will present two examples of multicast protocols, namely PGMCC and PLM for single rate and multirate congestion control respectively.

3.1.3.1 Multicast Protocols

Multicast protocols address the problem of efficiently sending one message to multiple receivers, for example for conferencing and replicating information faster than multiple unicasting. Multicast applications have different focus functions and performance requirements, which can be classified into receiver-driven and sender-driven. In sender-based techniques a single rate is sent to all receivers. PGMCC [135] is an example of a TCP-friendly sender-based protocol for applications that can cope with larger variations in the sending rate. Receiver-based techniques are based on the capability to generate the source data in a layered format and send the layers as different multicast groups. Layer management can be divided into cumulative and non-cumulative approaches. In cumulative layering, the receivers could join and leave the layers based on sequential order. In contrast, in a non-cumulative approach, receivers can join any subset of the layers or only one layer. An example of a receiver-driven algorithm which is based on layered communication using AN is found in [187]. A layered multicast protocol is a multirate protocol in which a multicast session can be divided into different layers with different rates. The data rate of different layers of multicast and cumulative layered transmission is a mechanism that supports the delivery of live video content and bulk data transfer to heterogeneous receivers.

McCanne et al. [114] introduced the Receiver-driven Layered Multicast (RLM), which is the first layered multicast protocol to address heterogeneous receivers in packet video transmission. This protocol enforces a mechanism for receivers to add and drop network connections while adhering to their bandwidth restrictions. The RLM protocol utilises a layered compression scheme. An example of data layering is as follows: a single video signal is split into multiple segments, which are subsequently being spread across multiple multicast channels. The video encoding and decoding functions must be available to segment the video streams into multiple components. The lower layers consist of the lowest quality image, and higher layers will improve the quality of the image. Vicisano et al. [166] proposed their Receiver-driven Layered Congestion Control (RLC), an improvement to RLM, in which a receiver would join a higher layer only if there was no packet loss to ensure that the bandwidth is sufficient. Granularity of the layer in layered multicast is addressed in [32]. Several problems are encountered in the implementation of some proposed layered multicast mechanisms, i.e. loss induced by the join experiments, slow convergence time, and lack of inter-protocol fairness.

In the following subsection, we discuss non-AN-based single rate and multirate

layered multicast protocols which will be compared with the AN-based ones in Chapters 5 and 6, respectively.

3.1.3.2 Example of a Single Rate Multicast Congestion Control Protocol

Pragmatic General Multicast Congestion Control Protocol (PGMCC) [135] is a prominent example of a single rate multicast congestion control protocol. PGMCC selects the receiver with the worst network condition as a group representative, and calls it an acker. The acker's process of selection determines the fairness of the protocol. Each receiver tracks the Round Trip Time (RTT) and the smoothed loss rate, and communicates these values with a Negative Acknowledgement (NACK) to the sender using normal randomised feedback timers to avoid NACK implosion. PGMCC can also use network elements support to aggregate feedback once an acker is selected. A TCP style window-based congestion control is run between the sender and the acker. This close mimicking to TCP's window behaviour makes PGMCC suited for applications that can bear larger sending rate variation.

3.1.3.3 Example of a Layered Multicast Congestion Control Protocol

Packet-pair receiver-driven Layered Multicast (PLM) [102] introduces a packet-pair technique to infer which layers to join (to discover the available bandwidth), enabling fast convergence for cumulative layered multicast multirate protocols. The convergence time takes into account the issue that the time needed to subscribe to the higher layers in the multicast session is longer than the subscription to the lower layers. Accordingly, PLM uses the fair scheduler paradigm to ensure the fast convergence of layers.

PLM assumes that the routers are multicast capable, regardless of the multicast routing protocol used. It is receiver-driven, and the sender is capable of sending data via cumulative layers as well as emitting packets in pairs for each layer. Coding into cumulative layers means that each layer has the same content, but with an increase in quality in the higher layers. The receiver-driven approach enables congestion control to be performed by the join and leave actions of the receiver (adding and dropping layers). The cumulative layered congestion control requires the layers to follow the same multicast routing tree.

Different from other multicast layered congestion control schemes which are based on the information about each layer's throughput to estimate the available bandwidth, a signal of congestion in PLM is a Packet Pair (PP) estimate of the

available bandwidth. The first PP that leaves the queue after congestion occurs is a signal of congestion. The PP estimate is the packet size divided by the inter-arrival gap. The PP bandwidth inference mechanism does not induce losses when discovering the available bandwidth.

PLM drops a layer each time the sample bandwidth estimation value is lower than the current layer subscription, and adds layers in conjunction with the minimum bandwidth estimation value received during a period C. In order to avoid high congestion that prevents packet-pairs arriving, if after a predefined timeout period no packet is received, then a layer is dropped. Losses are identified by the packet sequence numbers, and use a one bit field that marks the first packet of a PP burst.

In PLM, there are three particular parameters that influence its behaviour: the granularity of the layers, the check value C, and the burst size. The layer granularity will affect the stability of the layer, in which the smaller bandwidth per layer will lead to higher stability. The check value also affects the stability, in which a larger value of C will also lead to higher stability. The larger the burst size, the more accurate the estimate.

The PP estimate used in PLM helps provide some ideal congestion control properties such as stability, simplicity, efficiency, and fairness. However, its fair scheduler network cannot feasibly be implemented on the whole Internet architecture.

The following sections review the deployment of AN in congestion control and AN-based unicast and multicast protocols.

3.2 Deployment of AN in Congestion Control

ANs offer the promise of improving congestion control due to their flexibility. Congestion is an intra-network event and is potentially far removed from the application, therefore it is a potential problem domain to be solved by ANs. The speed with which an application can self-regulate to reduce congestion or can ramp-up when congestion has cleared is limited by the time that is required for congestion notification information to propagate back to the sender [21].

In terms of congestion control, Psounis [128] stated some examples of situations where ANs functionality can help:

1. An active node can monitor the available bandwidth and control the rate of a data flow. Instead of putting the buffer in the switch, it can be put in the active node.

2. An active node can control the relevant rate of each flow in addition to the total rate.

3. Transformation of data at a congestion point can be done at the right place, although this may be costly.

4. Selective dropping of units, packets or cells can be held efficiently with ANs.

The following subsection will discuss some existing AN-based congestion control protocols according to whether they are based on feedback or non-feedback.

3.2.1 Feedback vs Non-Feedback Approach

Non-Feedback Congestion Control

The work of Bhattacharjee et al. [18, 19, 21] has applied AN to non-feedback Asynchronous Transfer Mode (ATM) congestion control. They introduced the Application Specific Congestion Control (ASCC) protocol to perform congestion control, without specifying any ATM service. This protocol processes application data during congestion with some schemes such as 'unit' level dropping, media transformation, and multi-stream interaction. This work has presented a mechanism that would allow bandwidth reduction to occur in a manner that preserves as much application-level useful data as possible.

An application-specific state can be installed inside the network such that the adaptation the sender would have applied can be performed at the congestion point. ASCC exploits the interface provided by ANs to tailor the network's behaviour during the period of congestion. A mechanism can be placed in the network to allow packet processing, e.g. discarding or transferring, to proceed according to the advice supplied by the application.

The ASCC implementation to apply AN to congestion control specified that only packets that carry Active Processing Control Information (APCI) can be actively processed or switched by a non-active nodes. This enables both active and non-active nodes to coexist in the same network. This work is important as it is among the first on ANs application in congestion control and the only AN-based non-feedback mechanism found so far in the literature.

Feedback Congestion Control Approach

The work of Faber [47] applies ANs to feedback congestion control, specifically to TCP. Active Congestion Control (ACC) exploits state and programmability to

reduce the delay when congestion is signalled to the sending systems. The ACC control packets contain 4 to 8 byte characterisations of the state of the endpoint's congestion feedback. When congestion occurs at a router, particularly when a packet is dropped, the router determines the congestion window size, deletes packets from the sender that violate the newly established window value (that would not be sent with this new window size), and informs the sender of the new window size. Faber argued that based on simulation experiments using the ns-2 network simulator, the throughput of ACC TCP is claimed to have improved by 18% compared to the basic TCP (endpoint) for uncontrolled traffic. Although this performance gain is not appealing for some researchers, the work is among the first publication on ANs and congestion control.

Under ACC TCP, active routers generate a congestion signal in the form of a TCP acknowledgement for the TCP source. This will shorten the time required for congestion notification to reach the sender. The TCP acknowledgement (congestion signal) advertises a TCP window that is half the size of the current window. A similar protocol which uses a forward active congestion control algorithm has also been reported in [168].

Most AN-based multicast congestion control protocols are based on the feedback approach, due to the fact that the congestion control mechanisms are referring to the information received from the multicast receivers.

Other Approaches

Some other works on ATM and active congestion control include [130, 180]. The work of Pung et al. [130] uses a mechanism to implement an efficient congestion control of multi-layered encoded video multicast streams. This is a design of an ATM multicast protocol with a dynamic code injection mechanism for service customisation. The work of Williamson et al. [180] reviewed the ATM Available Bit Rate (ABR) service as an active service whereby congestion control functions are user deployed.

3.2.2 AN-based Unicast and Multicast Congestion Control Protocols

The following subsections address the AN-based unicast and multicast congestion control protocols.

3.2.2.1 Unicast

Active Network Congestion Control Transmission Control Protocol (ACC TCP) [47, 49] and Forward Active Congestion Control Algorithm [168] are two AN-based unicast congestion control protocols which are found in the literature, based on the ns-2 network simulator. They are built on top of the existing TCP congestion control mechanism. ACC was designed to work well in a high bandwidth-delay product regime.

3.2.2.2 Single Rate Multicast

The use of ANs paradigm where routers could contribute to improvements in the network services by customised functionalities have been proposed by the multicast research community. Active Reliable Multicast (ARM) [104] is one of the first AN-based multicast congestion control protocols. In ARM, upon experiencing a packet loss a receiver sends immediately a NACK to the source. Active Error Recovery/Nominee Congestion Algorithm (AER/NCA) [88] uses a strategy based on local timers at the receivers in which prior to sending a NACK packet, a receiver initiates a timer and waits for a random amount of time. The active router in AER multicasts the repair packet to every receiver associated with it, whereas ARM sends the repair packet only to the set of receivers that have sent a NACK packet.

The Active Multicast Protocol (AMP) [11, 174] is a single rate AN-based multicast protocol to help the congestion control. AMP has been implemented using ANTS EE and has inspired further work on this problem domain.

Reliable Multicast Active Networks Protocol (RMANP) [33], a sender-driven rate-based active multicast congestion control protocol has been introduced in [8, 143]. Congestion control is the responsibility of both senders and receivers. In this work, the urgency to study the definition of TCP compatibility (TCP-friendliness) and the combination of the flows in multicast flow has been underlined. However, the mechanism of TCP compatibility in RMANP has not been established.

Dynamic Replier Active Reliable Multicast Protocol (DyRAM) [109] was designed to provide a reliable multicast service without any facility in the multicast tree. This receiver-based (replier) local recovery multicast protocol would use a dynamic replier elected on a per-packet basis. In DyRAM, routers play an active role, i.e. they suppress duplicate NACKs to minimise the NACK implosion problem, subcast the repair packets to the relevant receivers only, and elect a replier in which a link is chosen to perform local recovery from the receiver instead of caching data

packets at the router. DyRAM is simulated using Parsec [126] and implemented with the Tamanoir execution environment [108].

Active Hierarchical Congestion Control for Large-scale Multicast (AHCCM) design [9] was based on the AER/NCA framework. AHCCM is a router-assisted, window-based hierarchical congestion control in which multicast flows have an adjustable share of the link's bandwidth with TCP flows. Upon congestion, active routers use a congestion smooth buffer to smooth congestion by reducing the forward rate, and then increase its forwarding rate to utilise the available bandwidth. In other words, TCP-friendliness is supported by adjusting the amount of resources by properly using a congestion smooth buffer to reduce the forwarding rate when congestion occurs, and increase the rate when the congestion resolves.

3.2.2.3 Multicast Multirate Congestion Control Protocol (Layered Multicast Protocol)

Layered multicast protocols provide a good solution to the problem encountered in single rate reliable multicast protocols in which all receivers have to receive service at the same time. Single rate techniques can not accommodate the heterogeneous nature of receivers and differences in the bandwidth capacity of different networks links such as T1, T3, Integrated Services Data Network (ISDN) and Asynchronous Digital Subscriber Line (ADSL).

A layered multicast protocol enables multiple multicast groups to transmit data at different rates for a diverse set of receivers. Active Traffic Control Mechanism for Layered Multicast (ATML) [85] and Active Layered Multicast Adaptation (ALMA) [187] protocols are the AN-based multicast multirate protocols which can be found in the literature.

3.3 Review of AN-based Protocols

The main theme of ANs, congestion control, and the selection of the ns-2 simulator as a performance evaluation framework lead us to a coherent research programme. In order to evaluate the AN-based congestion control protocols, we have chosen one protocol for each mode of operations to be investigated in detail. ACC TCP, AER/NCA, and ALMA have been chosen as the AN-based protocol representatives for unicast, multicast single rate, and multicast multirate protocols respectively. The following subsections outline these protocols in detail. These protocols will be

extended and elaborated further in the next chapters to cover specific congestion control schemes.

3.3.1 Active Congestion Control TCP (ACC TCP)

The ACC TCP protocol was introduced by Faber in 1998 [47, 49], aiming to use Active Applications (AA) to detect congestion, modify packet flows in the network to relieve congestion, and notify endpoints of the traffic modification. In ACC TCP congestion relief is initiated at the congestion point to shorten the response time. ACC TCP is one of the first unicast protocols introduced to perform feedback congestion control using ANs.

ACC TCP was designed to reduce the high bandwidth-delay product effect by making congestion response a distributed activity, rather than endpoint activity only. Once the ACC TCP active application is inserted into the programmable element of the network, it requires access to the queue occupancy information of the router and the header information of packets to be discarded by congestion. It must also have the ability to set filters to remove specific packets from the data stream.

ACC TCP applies the ANs technology to feedback congestion control specifically to the existing TCP. It follows the TCP congestion control rules in deciding to delay sending packets and update endpoints' states as well as in performing fast retransmission. ACC TCP has been designed to reduce the effect of high bandwidth-delay product on feedback control by making congestion response be performed by the network instead of the endpoints.

Under ACC TCP active routers generate a congestion signal in the form of a TCP acknowledgement (ACK) for the TCP source. This will shorten the time required for congestion notification to reach the sender. The TCP acknowledgement (congestion signal) advertises a TCP window that is half the size of the current window.

When congestion is experienced by the system, the ACC TCP notifies the end system by forwarding a packet with the new window size and then filters one window of traffic. On the occurrence of packet loss the window message will be closed and one window should be discarded. Thus, ACC TCP uses ANs services to reduce the control delay exhibited by the feedback congestion control system by means of generating ACC TCP packets and starting traffic modification from the congested router. Each packet contains a program or data for a router that enable it to react to network congestion, hence avoiding the delay in communicating congestion information to endpoints. ACC TCP uses the router intelligence to reduce congestion,

in which the router calculates endpoint responses and nearby routers implement local responses.

ACC[8] has been implemented by Faber [48] in a simple transport protocol called Reliable Data Protocol (RDP). The results of that implementation and further work on ACC TCP have been published in [49]. It stated the implementation overhead of ACC on RDP is low. Both simulation and implementation results have shown that ACC congestion control is effective in the regime for which it was designed (high bandwidth-delay product). However, in a low bandwidth-delay region, the standard feedback control system works better.

3.3.2 Active Error Recovery (AER) / Nominee Congestion Algorithm (NCA)

AER/NCA is an approach to multicast congestion control which aims to overcome the problem of congestion and to achieve better transmission performance. AER/NCA is a reliable multicast architecture that invokes active services at strategic locations inside the network to address the problem of feedback implosion, retransmission scoping, distributed loss recovery, and congestion control. It consists of two protocols, i.e. Active Error Recovery (AER) for packet loss recovery, and Nominee Congestion Avoidance (NCA) for Congestion Control. The objective of the protocol [88] is that the active services architecture improves resource usage, latency for loss recovery, and provides 'TCP-friendly' congestion control. ANs contribution in AER is a best-effort cache of data packets to permit local recovery. Upon a packet loss, prior to sending a NACK packet, a receiver initiates a timer and waits for a random amount of time. The active router multicasts the repair packet to all its associated receivers.

NCA uses a single-rate, source-based adjustment algorithm that regulates the transmission rate according to packet loss indications from a single receiver - the 'nominee'. A multicast session is required not to receive more bandwidth than competing TCP sessions on any of the sources to destination paths in the multicast tree, in order to achieve TCP-friendliness. The weakest path is the path on which a TCP session will receive the least bandwidth. Ignoring the time-outs effect, the average throughput (bandwidth) of a TCP session with fixed size packets is given by Equation 3.4:

[8] ACC refers to both the TCP and Reliable Data Protocol's (RDP) implementation of the Active Congestion Control protocol.

$$B(p,T) = \frac{C * MTU}{T * \sqrt{p}} \tag{3.4}$$

where p is the Loss Probability Estimate (LPE), T is the round trip time estimate, C is a constant (the common value is 1.22) and Maximum Transfer Unit (MTU) is the packet size used [88]. The most bandwidth-constrained path is the one with the highest value of the function $g(p,T)=T*\sqrt{p}$. This throughput modelling is similar to the TCP throughput modelling of Padhye et al. in [122].

NCA simulations in [88] and [92] demonstrate that the NCA protocol enables multiple multicast sessions to share the available network bandwidth fairly among themselves and with competing TCP sessions. Bandwidth sharing aspects of small multicast groups in this simulation illustrate the TCP-friendliness considerations.

In AER/NCA, active services are used for the purpose of performance enhancements only. The required router support is also minimal. The nature of ANs that allows new services to be deployed in an evolutionary way makes it possible for the current Internet architecture's routing and forwarding semantics to be preserved. The AER/NCA simulator [88] introduced new active packets for signalling mechanisms, i.e. Negative Acknowledgement (NACK), Repair, Congestion Status Message (CSM), Source Path Message (SPM), and Congestion Control Message (CCM) packets.

Moreover, in achieving its fairness goal, AER/NCA uses a rate adjustment algorithm to regulate the source to adjust its rate in response to the ACK received from the nominee. The algorithm uses the congestion window (CWND), and a slow-start threshold variable (Thresh). The adjustment of CWND is similar to the TCP New Reno's mechanism as follows [88] :

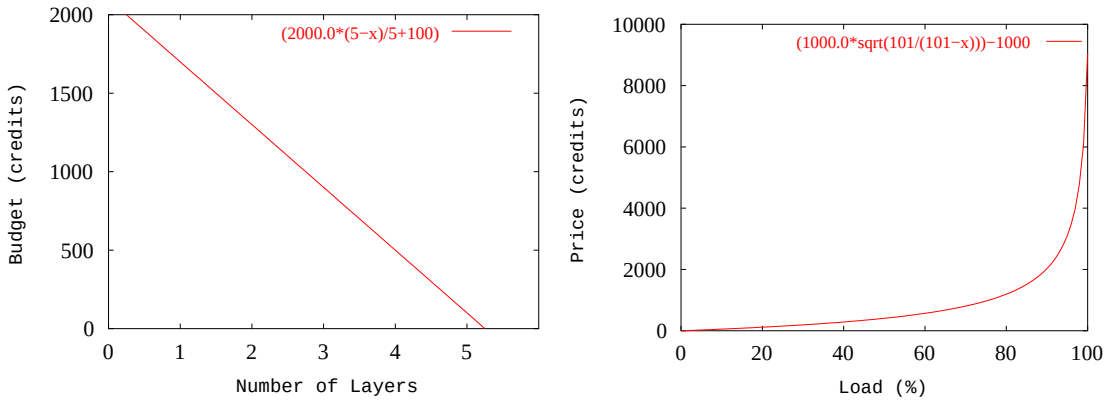
On receiving ACK, if (CWND < Thresh) CWND-> CWND+1; else CWND->CWND+ 1/CWND
On detecting loss, Thresh -> CWND/2; CWND->CWND/2 (congestion avoidance)
On timeout, Thresh -> CWND/2; CWND->1 (slow start)

3.3.3 Active Layered Multicast Adaptation Protocol (ALMA)

ALMA is an adaptive application which is designed to treat bandwidth fluctuations and minimise congestion. ALMA is designed for a layered multicast protocol in which the data stream is divided into a number of layers after being encoded hierarchically [187, 186]. Several flows are generated and the receiver gets a subset. This protocol has a layered scheme for competing flows with different priorities in

performing bandwidth adaptation. In ALMA, TCP-friendliness is not a focus of the protocols, particularly when the delay to join/drop from a group is high.

ANs are used in ALMA to overcome the problem encountered in classical adaptation approaches, i.e. the lack of mechanisms to obtain the required information about resource availability. The capsules are used to provide the information on congestion, which is the queue occupancy. The average queue length is used to calculate the price using a designated price function (see Equation 4.2), and then will be compared with the budget, which is proportional to the relative marginal utility of adding more layers. This filtering mechanism is simple, and only requires the link to export the function of the load in terms of price to provide flexibility in offering different satisfaction requirements to the users. Figure 3.4 shows the budget and initial price function used in ALMA.



a) Budget b) Price

Figure 3.4: Budget function for each layer and price function for intermediate nodes [187]

The price function used in ALMA simulation experiments can be seen in Equation 4.2 [188, 161], in which the load is the average buffer occupancy. This is a convex increasing function which forces the sharp rise of price on high loads to prevent the use of bandwidth.

$$price = 1000 * \sqrt{\frac{1.01}{1.01 - load}} - 1000 \tag{3.5}$$

This protocol uses some assumptions: namely all the nodes are AN capable; all packets are active capsules; the protocol uses the source-based sparse-mode[9] routing

[9]Refer to Internet Engineering Task Force (IETF) Request For Comment (RFC) 2117 and RFC 2362 for further information on sparse-mode routing

mechanism, but not group address. The layered multicast application is composed of a single source and several different receivers. The source is hierarchically encoded, in which higher layers carry more detailed information in addition to that available in the lower layers. The data are carried by capsules which also contain the instructions to process them inside the active nodes. The relationships among layers are built in.

ALMA employs subscribe capsules to be used by session receivers for probing bandwidth and indicating the subscription level. The capsules are also used by the active router for pruning to perform congestion control. The subscribe capsules consist of the source address and the subscription level. The congestion control mechanism can be summarised as follows: outgoing interfaces export a price as a function of link characteristics and load, data capsules filter themselves by comparing price and budget, data capsules prune multicast branches that have persistent congestion, and then the receiver spawns the subscribe capsules to probe for additional bandwidth.

A data-filtering mechanism to overcome short-term congestion is performed by comparing the price with the budget, whereas long-term congestion is anticipated by pruning decisions. The behaviour of the protocol can be evaluated by analysing the bandwidth subscribed by each receiver, the fast convergence to optimal rate, the number of layers subscribed, and the loss rate.

3.4 Summary

The first part of this chapter reviews the principles of current Internet congestion control and some advanced topics in networking such as AQM, TCP-friendliness, and pricing. It also classifies the congestion control protocols into unicast and multicast protocols and addresses the congestion control schemes in detail.

From the literature reviewed, we can see that there have been extensive studies on the congestion control problem and the AN-based solution to it. However, little attention has been given to the specific issue of unicast and multicast congestion control protocols, as well as assessing the impact on the use of ANs on the overall network performance. It can be stated that the 'red thread' which brings all the issues into context in this thesis is the ANs, congestion control, and the ns-2 network simulations.

The last part of this chapter describes the protocols investigated in this work: ACC TCP, AER/NCA, and ALMA. In the next chapters, we will expose our work on

the performance evaluation of each unicast and multicast protocol in order to assess the behaviour of each AN-based protocol. The reason behind it is that we would like to improve and integrate new enhancements into the currently available protocols as well as evaluating the impact of different network variations. It is our aim to gather an understanding of the advantages of having ANs to assist in performing congestion control mechanisms.

Chapters 4 to 6 present our work in the evaluation of AN-based unicast and multicast protocols. We perform experiments U1 (Unicast 1), U2, and U3 to evaluate the unicast protocols. Subsequently, for single rate multicast protocols, we perform experiments S1 (single rate multicast 1), S2 and S3. Next, Experiments M1 (multirate multicast 1), M2, M3 and M4 are the conducted for the multirate multicast protocols. The framework unifies the view of multi modes of communications in the Internet and congestion control, under a single performance evaluation theme.

Chapter 4

Performance Evaluation of AN-based Unicast Congestion Control Protocols

In the previous chapter, we introduced the problem domain of Active Networks (ANs) and congestion control. In this chapter, we present the core of the work which will cover the AN-based unicast congestion control protocols. The main focus of the work is to compare the performance of REM and RED active queue management mechanisms. In Section 4.1, a brief background on the AN-based congestion control protocols and active queue management is presented. Although we have introduced the unicast AN-based congestion control protocol researched in this study (ACC TCP) [47] in Chapter 3, we will revisit its congestion control mechanism in Section 4.2. Section 4.3 presents the general approach to simulation experiments used throughout this thesis. Next, we present the evaluation of ACC TCP in three experiments. The first experiment in Section 4.4 shows the microscopic behaviour of ACC TCP. The second experiment in Section 4.5 presents the experimental design and discussion of the results of the simulation on a more complex network topology, in which several parameters have been varied. Section 4.6 presents the third experiment and its result, in which bursty cross-traffic has been added to the network. For each experiment, we present the description of the simulation, the topologies, the parameters and their rationale, and discuss the results of the experiments. In Section 4.6.6, the limitations of the AN-based scheme are evaluated. Section 4.7 discusses the overall performance evaluation conducted for the AN-based unicast protocol. Finally, Section 4.8 gives a summary of the chapter.

4.1 Introduction

Improving the performance of the Internet TCP (Transmission Control Protocol) is one of the major research areas in networking, focusing on router queue management and congestion avoidance. Unicast TCP currently uses dynamic window flow control to implement congestion avoidance in which packet loss and round trip time (RTT) information are used to convey implicit feedback to the sources.

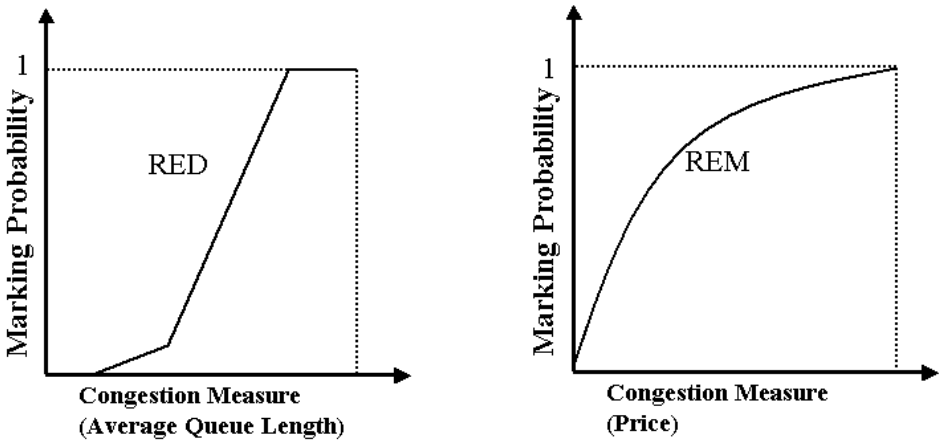
In an AN-based unicast congestion control protocol - ACC TCP, active applications are used to detect congestion, modify packet flows for congestion relief, and notify endpoints of the modification to their traffic. In order to shorten the response time, the congestion relief is initiated at the congestion point.

Active Queue Management (AQM) provides queuing and dropping strategies to reduce the time to relieve congestion by inferring congestion before the conditions become worse. Recall that RED and REM AQM algorithms and parameters have been discussed previously in Chapter 3. The RED AQM algorithm has been designed to overcome the problem with Droptail queuing, and is now widely implemented in commercially available routers. The RED mechanism reduces unfairness by dropping random packets before the queue becomes full. The results in [47, 49] show that ACC TCP is complementary with RED.

The REM AQM algorithm has been proposed in [6, 99] and generally only requires minimal changes to the existing RED AQM routers. The philosophy behind REM is to bring the network to an operating point that will not require the changing of standard TCP[1] algorithms in order to improve the throughput and link utilisation. Figure 4.1 illustrates the marking probability of RED and REM, in which the difference between their congestion measure functions can be noted. This in turn yields differences in how they handle the congestion. The unit of congestion measure in RED is the average queue length, whereas in REM it is price. Price is updated periodically or synchronously based on *rate mismatch* (difference between input rate and link capacity) and *queue mismatch* (difference between queue length and target queue). If the weighted sum of these mismatches is positive, the price is incremented. Otherwise, the price is decremented. Some literature on congestion control has acknowledged the advantages of REM [79, 119]. However, in relation to ANs, no evaluation of REM AQM has been attempted.

In this chapter, we conduct performance evaluation of the integration of two novel approaches: a) network architecture called active networks, and b) the active queue

[1]In this work, we deal with TCP-based congestion control protocols; consequently, all of the AQMs used are associated with TCP. All literature found so far also applied REM to TCP.



a) RED b) REM

Figure 4.1: Marking probability of RED and REM

management algorithm called Random Early Marking [6, 99]. Subsequently, we compare the RED and REM performance on ACC TCP. RED or REM are required beyond the ACC TCP, because routers need a buffer scheduling policy to handle the packets they receive. For example, RED buffer scheduling algorithm reduces unfairness by dropping random packets before the queue becomes full. The AQM will be an additional mechanism to the AN-based congestion control protocols, in which the active router shortens the time for congestion relief.

We have integrated REM into the ACC TCP ns-2 simulator and evaluated its dynamic performance. We evaluate the microscopic behaviour of REM in ACC TCP for a simple network, and then compare the performance to that of RED on ACC TCP. Subsequently, we study the behaviour of ACC TCP across a range of bandwidth-delay product networks, in which we use more complex, 'stable' and 'bursty cross-traffic' network scenarios (by varying the delays in one of the links and keeping the other parameters constant), in order to see the impact of a range of congestion conditions on the network performance.

4.2 ACC TCP and REM

The work of Faber [47] applies the ANs technology to feedback congestion control, specifically to TCP. Active Congestion Control (ACC) exploits state and programmability to reduce the delay when congestion is signalled to the sending systems. Recall that we have discussed the ACC protocol in Chapter 3. ACC packets contain 4 to 8 bytes characterisations of the state of the endpoint's congestion feed-

back. When congestion occurs at a router, particularly when a packet is dropped, the router determines the congestion window size, deletes packets from the sender that violate the new window size, and informs the sender of the new window size.

Under ACC TCP, active routers generate a congestion signal in the form of a TCP acknowledgement for the TCP source. This will shorten the time required for congestion notification to reach the sender. The TCP acknowledgement (congestion signal) advertises a TCP window that is half the size of the current window.

When congestion is experienced by the network, the ACC TCP notifies the end system, and then filters one window of traffic. On the occurrence of packet loss, the window message will be closed and one window should be discarded. Thus, ACC uses AN technology to reduce the control delay exhibited by the feedback congestion control system by means of generating ACC packets and starting traffic modification from the congested router. Each packet contains a program or data for a router that enables it to react to network congestion, hence avoiding the delay in communicating congestion information to endpoints. ACC uses the router intelligence to reduce congestion, in which the router calculates endpoint responses and nearby routers implement local responses.

ACC TCP is aimed at networks with high bandwidth-delay products. In addition, the performance evaluation of TCP/IP for networks with high bandwidth-delay products and random loss is also important [97]. Hence in our work, we also take this into account. We evaluate the ACC TCP with different queue management schemes across a range of bandwidth-delay product networks - using 50 to 250 ms delays of uncongested link.

The problems addressed in this chapter are illustrated in Figure 4.2, in which the AN architecture has been complemented by queue management schemes to achieve better performance. The two approaches could be built together in a router to achieve desirable network performance with high throughput and low packet delay and loss.

Application of ANs in ACC TCP

The ACC TCP ns-2 extension of Faber [47] has been used and modified in this work to be able to employ a new queue management mechanism (REM). Therefore, in addition to Droptail active and RED active flows, we have REM active flows that can be run to take credit from the existence of both the AN-based unicast TCP congestion control and REM. We have implemented this by creating Queue/REM/Active mechanism by modifying the Tcl and C code available from ns 2.1b5 with ANs

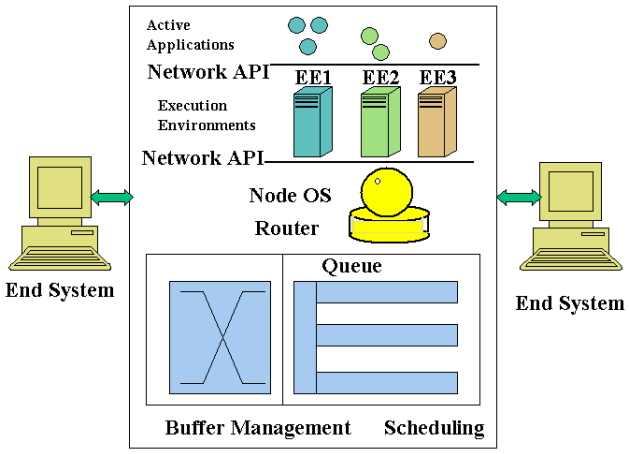


Figure 4.2: Active networks and queue management architecture

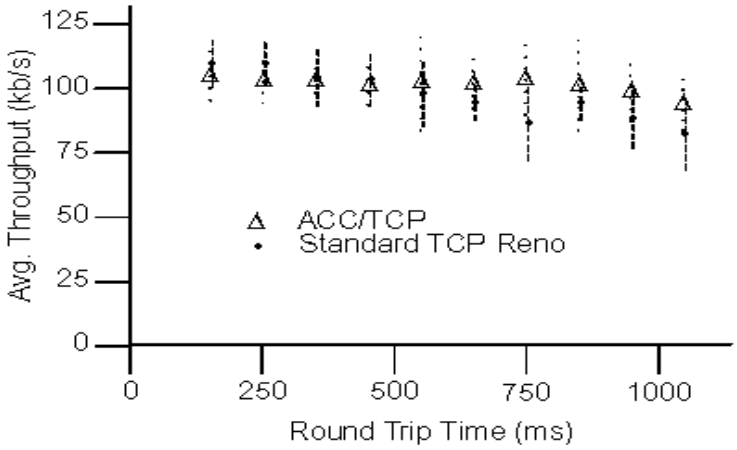
extension and REM C codes of Athuraliya et al. [6].

AN capsules have been used in ACC TCP, in which the active services are embedded in the Agent/TCP/Active and Agent/TCPSink/Active. The active packet's size is similar to the TCP Reno data (1000 bytes), and the acknowledgement packet size is 40 bytes. Result generation can be obtained by evaluating active agents used on TCP sources and sinks. ACC packets are formed as a 4-8 bytes characterisation. It can be emphasised that the congestion control performed can be attributed to the active routers' actions to generate acknowledgements, and filter windows to inform end points of congestion.

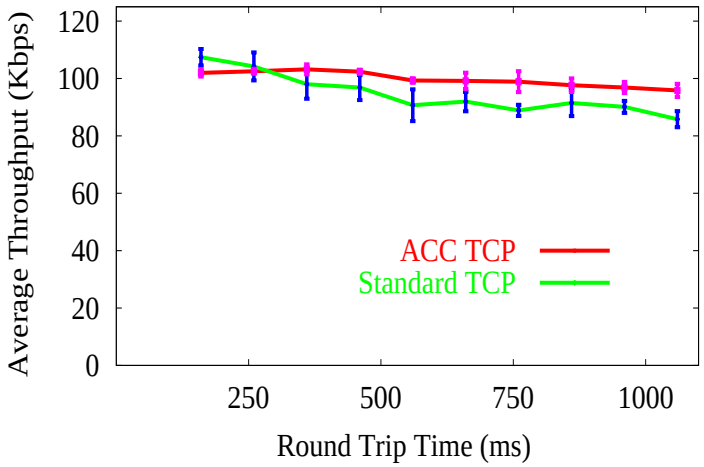
Validation

Validation is an important step in performing network simulation as described in Section 2.3.3. It is worth mentioning that after installing ns-2, the validation tests which come with the software distribution were successful. Moreover we ensured that the built ACC TCP program runs properly and tested it in a simple scenario in which there is no congestion situation.

Furthermore, in order to ensure the validation of the ACC TCP simulation in this research, we have chosen the following results from a specific case in the literature, the work of Faber in [49] (Figure 4.3(a)). The simulation is based on the same topology with Experiment U3 (topology in Figure 4.16). The objective of the simulation is to compare the throughput of ACC TCP and TCP Reno, in the case where the uncongested link's delay is varied: 50, 100, 150, 200, ..., 500 ms. The rest of the links in the path have 10 ms delay each. Identical parameters have been



a) ACC TCP simulation result obtained from [49] (This figure has been scanned from the work of Faber in [49]). The graph shows the average throughput versus round trip time generated from the simulation



b) Our ACC TCP simulation results for the same scenario

Figure 4.3: Validation of ACC TCP ns-2 package

used for ACC TCP and non ACC TCP simulations, considering that RED queuing algorithm is implemented in routers. Figure 4.3(a) shows the round trip time versus throughput plot, in which the round trip time is twice the total of delays in the path. We have performed a similar simulation using ns-2 (Figure 4.3(b)).

RTT (ms)	**TCP throughput (obtained from [49]) (Kbps)**	**TCP simulation results average throughput (Kbps)**	**TCP simulation results standard deviation (Kbps)**	**Difference wrt standard deviation**	**ACC/TCP throughput (obtained from [49]) (Kbps)**	**ACC/TCP simulation results average throughput (Kbps)**	**ACC/TCP simulation results standard deviation (Kbps)**	**Difference wrt standard deviation**
160	109	102.42	4.63	0.34	105	101.96	2.11	1.43
260	109	104.21	7.88	0.60	102.5	102.52	1.4	-0.01
360	103	97.97	8.16	0.61	102.5	103.18	2.88	-0.23
460	102	96.82	7.01	0.73	101	102.39	1.2	-1.23
560	96	90.69	7.32	0.72	100	99.31	1.3	0.52
660	94	91.96	5.45	0.37	100	99.17	4.59	0.18
760	86	88.86	9.16	-0.90	102.5	98.89	5.9	0.61
860	92	91.45	7.37	0.07	99	97.63	3.82	0.35
960	87	90.1	3.38	-0.91	97	96.84	3.6	0.05
1060	82	85.81	4.53	-0.84	95	95.83	3.72	-0.22

Table 4.1: ACC TCP Validation: Difference between the results obtained from [49] with our simulation results with reference to (wrt) standard deviation =((Reference value - average simulation result)/(standard deviation)).

In Table 4.1 we show the throughput results from our simulations. The reference data shown is obtained by observation of Figure 4.3(a) performed by different individuals independently[2], because we had no access to the raw data.

This table shows the calculation results of the difference between the throughput results obtained from [49] and the results from our simulations with reference to the standard deviation[3]. The average data gained from [49] lies within the confidence limit of the data we gathered from the simulation. In other words, the average TCP and ACC TCP throughput values from [49] can be considered as a sample case of our simulation. These values are consistent with our simulation result.

Further statistical tests such as Kolmogorov-Smirnov test (KS-test)[4] could have been used if more raw data from [49] had been available. KS-test determines if two

[2]Three members of staff at the School of Computing assisted with this task.

[3]Standard deviation = $\sqrt{\frac{n\sum X^2-(\sum X)^2}{n(n-1)}}$

[4]http://alvarez.physics.csbsju.edu/stats/KS-test.html

data sets differ significantly. This goodness of fit test is used to formally determine if two sets of data are significantly different from each other. KS-test produces a D value which is the largest absolute difference between the cumulative distributions of two sets of data. First, the D value of two data sets is computed and then the result is compared to the critical value of D. There is enough evidence not to reject the null hypothesis (that the two groups are the same) if the P value is not 'small', and D (Distance) is smaller than the critical value which depends on the number of sample space and significance level[5]. The work of Heidemann et al. [98] has used this test to justify if two sets of traffic data are significantly different from each other, in addition to visually examining their plots. However, for the above case we could not draw any further statistical conclusion because of the limitation of the sample space.

4.3 Simulation Experiments General Approach

The following general approach is applied to the work throughout this thesis, unless specifically stated:

1. Different network topologies and simulation scenarios with varying numbers of sources and receivers are used in our simulations to enable us to view different characteristics of unicast and multicast congestion control mechanisms.

2. There are always some sources, some receivers and a bottleneck link in a network topology in which congestion occurs.

3. Many simulations to assess AQM algorithms implemented in the routers and congestion control mechanisms in this work are using a dumbbell topology (using one bottleneck link such as in Figure 4.6) to analyse the protocols. This topology is commonly used in network research for the reason that the purpose of the simulation is to evaluate the protocol's behaviour by tracing every event in the network. The dumbbell topology is used not only for its simplicity, but also to ensure that our aim to evaluate a congestion control protocol's mechanism in the bottleneck link can be achieved. The dumbbell topology in ns-2 has been extensively used to evaluate many congestion control protocol proposals [179].

[5] http://nedwww.ipac.caltech.edu/level5/Wall2/Wal4_2.html

4. Our simulation experiments use two traffic models to represent the Internet traffic requirements. The traffic models can be classified into responsive and unresponsive flows. The first is the TCP-based File Transfer Protocol (FTP) flows which are used to represent a bulk data transfer model that has data to transmit infinitely. The second traffic type is the User Datagram Protocol (UDP)-based Constant Bit Rate (CBR) flows to represent unresponsive flows such as a real-time multimedia traffic. Unlike TCP, UDP has no congestion avoidance mechanism and does not rely on the receiver's feedback to continue the transmission. Some experiments in this research use CBR traffic for different purposes, i.e. presenting an exponentially bursty (on and off) CBR cross-traffic, and presenting new AN-based multicast protocols flows.

5. The reason for choosing the propagation delays in our simulations is to consider a value below the limit of the 80-100 ms latency. This is due to the fact that these values are close to the high end of what has been perceived as the common Round Trip Time (RTT) in the Internet [47].

6. Each simulation is constructed by defining the desired topology and defining the repeat executions of independent simulations (using the main and child processes) to run using a random number generator. We then run the ns-2 simulator for the topology, capturing experiment data in trace files, parsing the data file for meaningful data using awk[6] or Perl[7] scripts, and calculating the performance results statistically. We ensured that the simulation is run up to a steady state condition before results are taken. Generally the results are stable after 3 seconds of simulation time.

7. In our experiments, we use repetitive/replication methods in which we run the simulation n times (typically 5 or 10). A substantial number of experiments have been performed for each AN-based protocol under various simulation scenarios. In doing this, for throughput measurement for example, for each unicast and multicast protocols' flow, we calculate their average throughput after repeating the experiment n times. We perform statistical evaluation using the mean, standard deviation, and a confidence interval of 95% to analyse the data. The confidence interval (CI) is gained by using the t value of 1.96 for

[6]awk is a scripting language to extract and perform calculation on data from the trace files. It is used for data file manipulation and named after its three developers (A. **A**ho, P. **W**einberger, and B. **K**ernighan).

[7]Perl is Larry Wall's 'Practice Extraction and Report Language' which is good for text processing, and support more complicated data structures.

95% confidence interval and is generated by Equations 4.1 and 4.2 [83]. The confidence intervals in all the graphs in this thesis have been plotted using gnuplot, which is a utility to generate 2D and 3D data plots. Having this in the graph means that we are 95% confident that the population mean is inside the interval. The Tcl script to generate the mean, standard deviation, and confidence interval can be found in the Appendix. This statistical method is used throughout the thesis and is indicated in all of the graphs. In the case where the confidence intervals are not clearly shown in the graph, it simply means that the widths are too small to be discerned. Throughout this thesis, we represent all of our simulation result with its 95% confidence interval. For example, 85.81 $\pm$ 2.81 represent 95% confidence limits of 83 and 88.62.

$$CI = mean \pm standarderror * t_{value} \tag{4.1}$$

$$Standarderror = \frac{Standarddev}{\sqrt{number - of - experiments}} \tag{4.2}$$

In the following subsections, we will describe some experiments with ACC TCP which can be divided into 2 parts. The first is to perform a microscopic analysis of the queue length on a simple network topology (Experiment U1). The second part (Experiments U2 and U3) deals with a more complex network topology in which the parameters are varied to note their impact on a feedback congestion control mechanism of ACC TCP.

4.4 Experiment U1: Microscopic Behaviour

4.4.1 Experiment Construction and Description

The main purpose of this work is to evaluate the use of the new AQM scheme called REM and compare it with RED in an AN-based congestion control protocol. We show the comparison of RED and REM and study their behaviour under various network scenarios. This section describes the network configuration and parameters to evaluate the AQM in an AN-based congestion control protocol. The flavour of TCP used in the ACC TCP ns-2 extension is TCP Reno. In these experiments and discussions, we refer to RED and REM in ACC TCP as RED active and REM active, respectively.

In this experiment, we perform a microscopic analysis of the queue length in a simple single bottleneck topology. The objective of this experiment is to evaluate the

queue length in the router, in which RED and REM queue management algorithms are applied on ACC TCP.

In order to evaluate the impact of staggering the starting time of the flows for each active-based protocol and queue management scheme, the number of ACC TCP flows are scaled from 10 to 50. In other words, the goal of this experiment is to see how the queues built in RED active and REM active nodes are affected by starting 10 more sources every 50 seconds. Each simulation is executed 5 times, in which ACC TCP flows are started at random times between 0 and 10 seconds. This is repeated for RED and REM simulations on the AN-based TCP environments.

Figure 4.4 presents a flow-chart of the simulation procedure. Before all of the

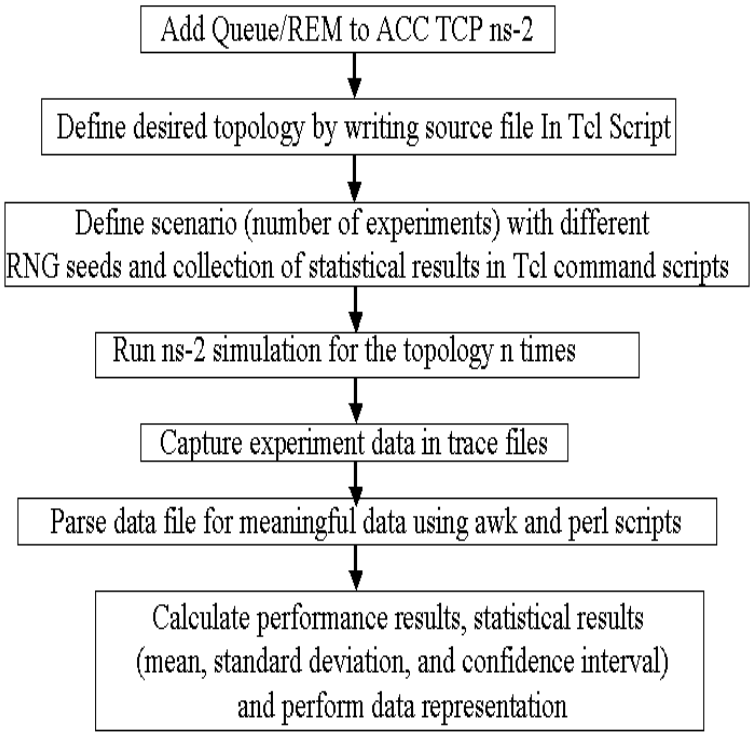


Figure 4.4: Experiment U1 - Flowchart of simulation procedure

performance evaluation experiments, we add the REM queue management to the ACC TCP. After the use of REM on ACC TCP is validated, we write Tcl files to define the desired topology. We construct the repeated simulations with different random number generator seeds, run the simulation[8], capture the experimental data in a trace file, and then parse the data file using awk scripts, which in turn is used

[8]ns simulation is run by executing the command 'ns tclfilename.tcl' into the environment in which a particular protocol has been installed.

to calculate performance results and generate graphs. The flow chart in Figure 4.5[9] shows the detail of the event list initialisation and procedure inside the simulator. This figure explains what happens inside the 'run ns-simulation' boxes in all of the flow diagrams in this thesis. The figure that shows and explain further what happens inside the simulator can be found in Chapter 2, Figure 2.3, which shows the routing between nodes, classifiers, multiplexers, and the variables which enable the packet routing mechanism such as queue, dequeue, and time to live (TTL).

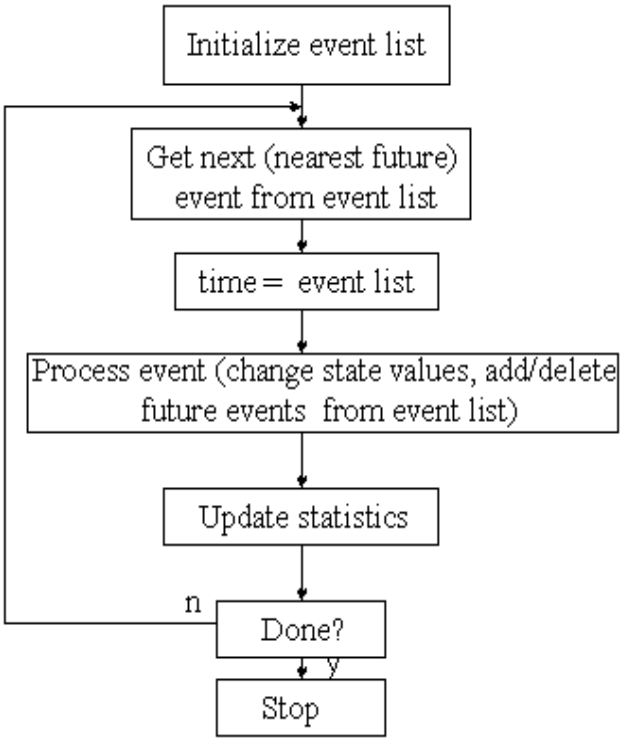


Figure 4.5: Flowchart of the event list initialisation and procedure inside ns-2 simulation

4.4.2 Topologies

The topology to analyse the queue length for each queue management algorithm is shown as a single link network with 10 to 50 greedy sources in the form of File Transfer Protocol (FTP) applications (Figure 4.6). Each FTP connection is a long-lived data transfer which takes place during the entire simulation time.

[9]This figure applies to all 'ns-2 simulation' boxes in all of the simulation flowcharts throughout this thesis.

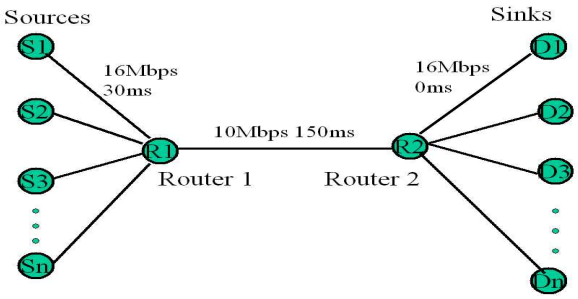


Figure 4.6: Experiment U1 - Topology for experiment on microscopic behaviour of ACC TCP

4.4.3 Key Parameters and Rationale

In these simulation experiments, in every router, RED parameters were set to the following values: RED buffer's maximum threshold (`maxth`) = 80 packets, minimum threshold (`minth`) = 20 packets, maximum probability (`maxp`) = 0.1, and queue weight (`wq`) = 0.002. In the case where REM queue management was used, the following parameters were set as follows: δ (weight in aggregate input rate estimate) = 0.02, β (weight in RTT estimation) = 0.02, N (sample size for price estimation) = 50, γ (step size in price adjustment) = 0.001, and α_1 (weight of buffer in price adjustment) = 0.1. These values and weightings are commonly used RED and REM parameter settings in ns-2, and have been recommended in [56]. The investigation of RED active and REM active parameter tuning[10] is beyond the scope of this investigation.

The simulation parameters are shown in Table 4.2. The number of sources of

Simulation Parameters	**Value**
Sources - Router 1links	16 Mbps, 30 ms delay
Router 1- Router 2 link	10 Mbps, 150 ms delay
Router 2 - Receivers links	16 Mbps, 0 ms delay
TCP Packet Size	1000 bytes
Simulation Duration	120 seconds
All routers maximum buffer size	120 packets

Table 4.2: Experiment U1 - Simulation parameters for topology in Figure 4.6

10, 20, 30, 40, 50 are chosen for ease of traffic evaluation. The size of each TCP

[10]References such as [99, 160] indicate that the RED and REM parameter sensitivity leads to different network behaviour.

packet is 1000 bytes. All traffic in these experiments is unidirectional. All incoming links to router 1 have 16 Mbps capacity and a delay of 30 ms. The bandwidth of the bottleneck link is 10 Mbps and the delay is 150 ms. All outgoing links to the sinks have 16 Mbps bandwidth and 0 ms delay. The purpose of the parameters' choice is to create a simple model in which the queue occupancy can be evaluated distinctively.

4.4.4 Performance Metrics

The performance metrics of the experiments consist of the queue length and the packet loss ratio. We observe the performance at the intermediate nodes in order to be able to explain the behaviour of the protocol in relation to the use of different AQM algorithms in detail.

4.4.5 Results and Evaluation

This section presents the results of the experiments and highlights some differences of RED and REM in AN-based TCP. The graphs are generated by tracing data packets entering and departing from the routers in the network. We discuss the results of the simulation experiment using RED and REM queuing policies to provide us with insight into the performance of the algorithms under various conditions.

Implication of Different Sources' Starting Time

In this section, we show the dynamics of the queue length of RED and REM AQM using ACC TCP. The aim of the experiment is to compare the queue build-up on RED active with the one using REM active scheme, taking into account the initiation of more traffic on the link. We simulated the cases explained in [7] (which compare RED and REM in standard TCP Reno) in our simulation environment, and compared the achieved results for RED and REM queue management mechanisms in ACC TCP. The simulation lasts for 250 seconds. At 0 second, 10 sources are activated, and subsequently every 50 seconds 10 more flows are activated, until the total number of sources is equal to 50 at 200 seconds. Figure 4.7 shows the life time of the sources in this experiment.

The purpose of the experiment is to highlight the impact of the increasing number of sources at regular intervals on the queue built up using the RED active and REM active queue management mechanisms. The results shown in REM active scheme are relatively robust when the traffic load is varied. REM's feature of low buffer

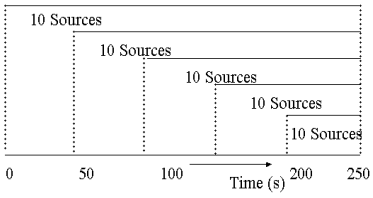
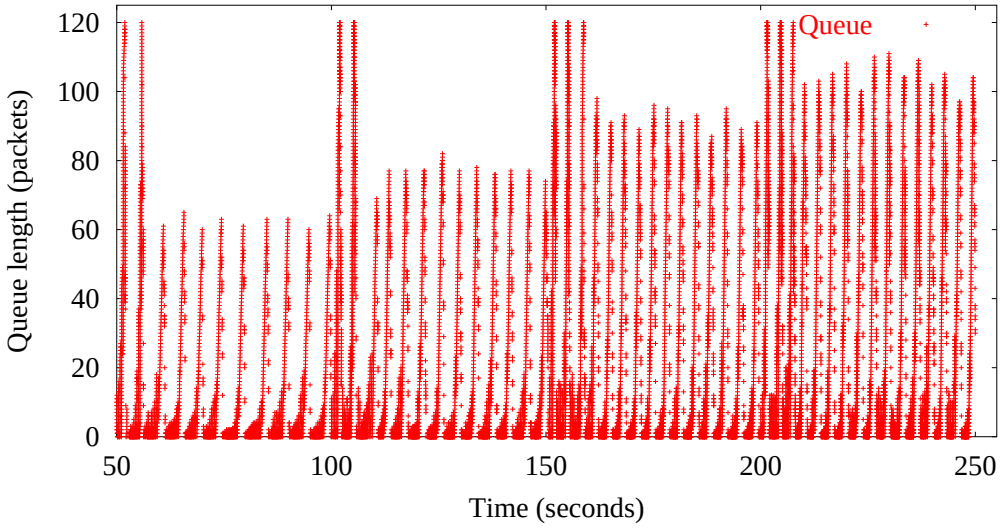


Figure 4.7: Experiment U1 - Life time of sources

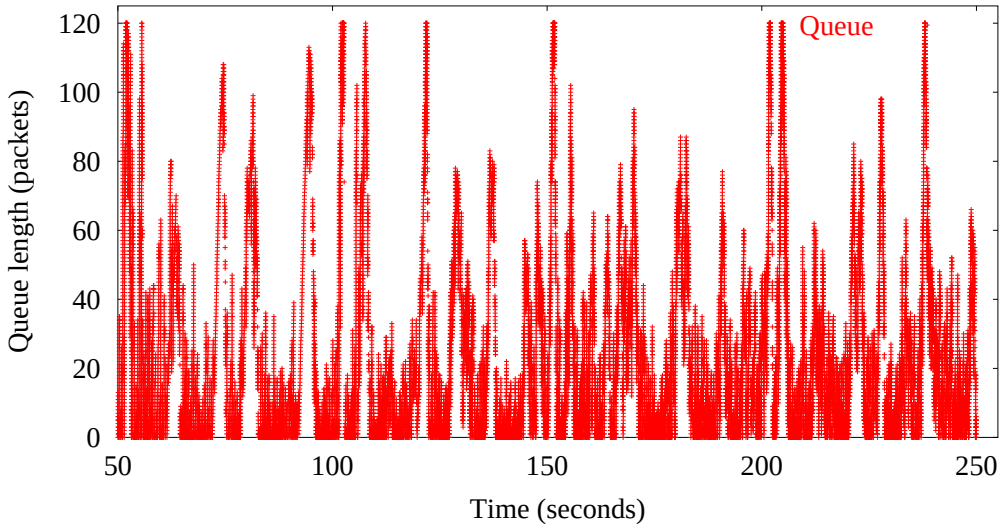
occupancy at steady state while achieving high utilisation at the same time as the number of connections changes can be seen here.

Figure 4.8 shows the differences between RED and REM when the ACC TCP is implemented, and their characteristics can be identified. The queue length built up when REM is used did not correspondingly expand with the increase in the number of sources. The behaviour of REM when the system entered and left congestion can be observed in this graph, in which the source starting times have been staggered. Using REM active, as the average load on the network exceeds the bottleneck link capacity, the buffer requirement does not increase correspondingly. In the experiment with REM on ACC, the routers allocate high rates to sources when the network is lightly loaded. The sources dump bursty packets into the network. Figure 4.8(b) shows that REM active is more stable than the RED active queuing management as the load of the network increases.

To analyse the queue, we divided the graph in Figure 4.8 into 4 regions, i.e. from 50 to 100 seconds of simulation time (x-axis), we observe the first phase of simulation (region I). As some part of the 0 to 50 seconds of simulation time are transient state, we do not consider them in this analysis. Region II represents the packet queue length during 100 to 150 seconds of simulation, etc.. Table 4.3 compares the average and standard deviation of each region in Figure 4.8 (generated by averaging the results of 5 experiments), which shows that the average queue length for the REM active discipline is more stable compared with that of RED active. Although in low congestion the average queue length of REM is higher than RED active, when the sources are added to the link, the RED active average queue length keeps increasing. REM active has demonstrated its stable queue occupancy at steady state, as shown by the value of the average queue length (the average of 24.7) in each region. REM active experienced more packet loss than RED active. The number of total packets dropped over the total packets transmitted for each time scale is also shown in this table. Overall, the packet loss ratios are low in both RED active and REM active schemes (less than or equal to 10%).



a) RED active



b) REM active

Figure 4.8: Experiment U1 - Simulation of packet queue affected by different sources' starting time (queue length vs time)

Region	RED active (average queue length)	Packet Loss Ratio (%)	REM active (average queue length)	Packet Loss Ratio (%)
I	18.34 ± 0.54	8	24.93 ± 4.43	8
II	20.43 ± 2.07	6	24.54 ± 1.47	7
III	26.53 ± 2.52	5	25.89 ± 0.86	9
IV	34.83 ± 0.95	7	23.64 ± 1.21	10

Table 4.3: Experiment U1 - Statistical comparison of RED active and REM active behaviour for Figure 4.8. The average queue length is gained from averaging the results of 5 runs

4.5 Experiment U2: Stable Network

4.5.1 Experiment Construction and Description

The second part of the experiments is the comparison of the AN and non-AN-based systems in a more complex but 'stable' topology. This was achieved by using a stable network topology in which no cross-traffic was involved. The simulation experiments are run for each queuing management policy, i.e. RED and REM, in order to do performance comparison using different performance metrics. Within each scenario, simulations were run 1) for a constant number of sources and by varying the length of delay at the uncongested links (which means varying the RTT), 2) for a constant length of delay for the uncongested link and by varying the number of sources. The objectives of the experiments are 1) to observe the behaviour of the ACC TCP relative to TCP under different circumstances in terms of the AQM used in the node (RED and REM), and 2) to analyse the set of experiments when running ACC TCP and REM.

Figure 4.9 presents a flow-chart of the simulation procedure. Tcl files are used

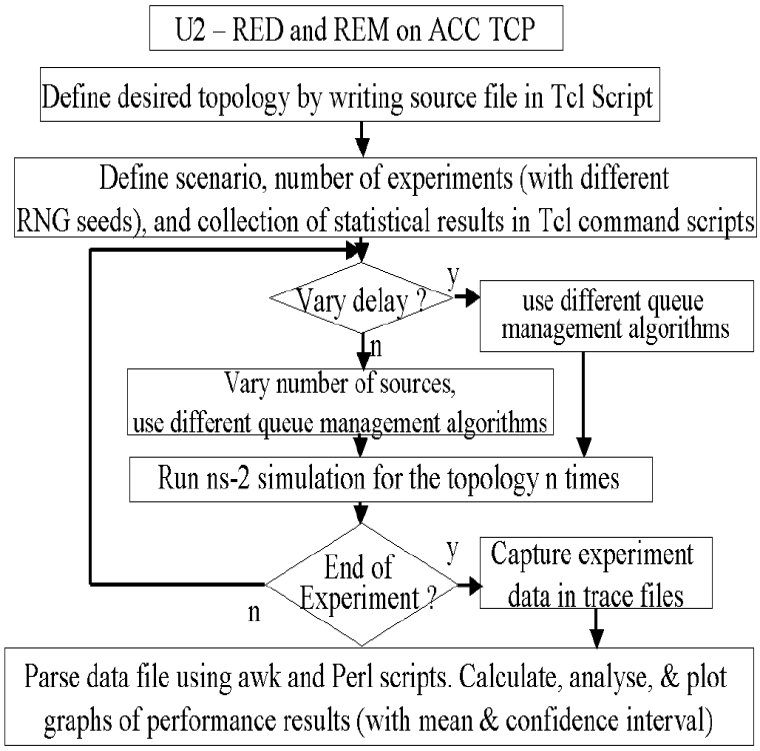


Figure 4.9: Experiment U2 - Flowchart of simulation procedure

to define the desired topology and construct the repeated simulations with different

random number generator seeds. The simulation is run $5 * 4 * 5$ times (varying the delay, using a different queue management algorithm, and executing each simulation 5 times), in a total of 100 executions. Then we vary the number of sources, use different queue management algorithms, and execute them 5 times (another 100 times of executions in total). Furthermore, the experiment results are captured in trace files, which are then parsed using awk or Perl scripts to get useful data. This then is used to calculate the throughput, queue length, packet retransmission ratio, normalised delay, and normalised throughput results. We then use gnuplot to to draw some graphs out of the results.

4.5.2 Topologies

The topology in Figure 4.10 shows the simulation performed on the N sources con-

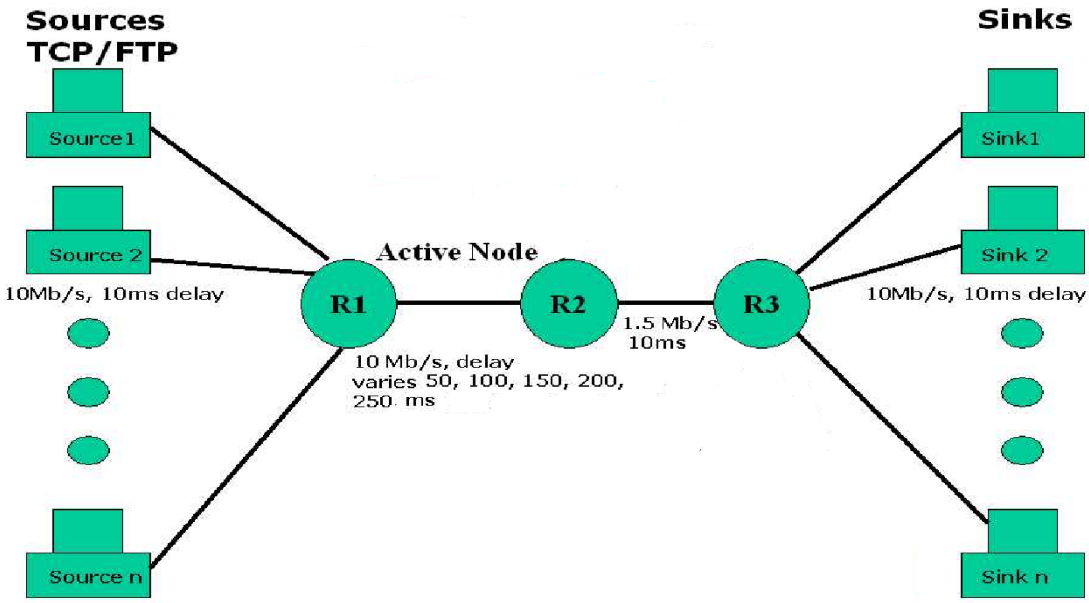


Figure 4.10: Experiment U2 - Topology for experiment of ACC TCP on a stable network (without cross-traffic)

figuration consisting of n identical TCP sources. n TCP sources and n TCP sinks are connected to the three intermediate routers. The link between routers R1 and R2 is the uncongested link with variable delay. The traffic is uni-directional. The simulation is performed in a network environment in which FTP applications are attached to the TCP sources and flow towards the TCP sinks. Similarly, we used

this procedure for connecting ACC TCP sources and sinks.

4.5.3 Key Parameters and Rationale

The parameter values applied in the simulation were chosen to induce congestion in the network. The delay at the uncongested link between R1 and R2 is varied from 50 to 250 ms to acquire variable bandwidth-delay product regimes. The performance of RED and REM on TCP and ACC TCP is expected to be revealed when the links are shared by a large number of sources. The bandwidth of the links is 10 Mbps, except for the link between R2 and R3, which is 1.5 Mbps. The link delays are 10 ms, except for the uncongested link between R1 and R2, in which we apply different link delays. The configuration parameters are detailed in Table 4.4, which shows

Simulation Parameters	**Value**
Sources	10, 20, 30, 40 and 50 TCP sources, linked to sinks through router
Sources - Router 1 and Router 3 - Sink links	10 Mbps, 10 ms delay
Router 1 - Router 2 link (Uncongested link)	10 Mbps, variable delay (50, 100, 150, 200, 250 ms)
Router 2 - Router 3 link	1.5 Mbps, 10 ms delay
All routers maximum buffer size	25 packets
Packet size	1000 bytes
Simulation duration	200 seconds

Table 4.4: Experiment U2 - Simulation parameters for topology in Figure 4.10

the variable number of sources and variable delays in the uncongested link used in these experiments.

The RED and REM parameters are set to the following values: RED buffer's maximum threshold (`maxth`) = 18 packets, minimum threshold (`minth`) = 12 packets, maximum probability (`maxp`) = 0.1, and queue weight (`wq`) = 0.002. In the case where REM queue management is used, the parameters are as follows: δ (weight in aggregate input rate estimate) = 0.02, β (weight in RTT estimation) = 0.02, N (sample size for price estimation) = 50, γ (step size in price adjustment) = 0.001, and α_1 (weight of buffer in price adjustment) = 0.1.

4.5.4 Performance Metrics

The performance metrics of the experiments reported consist of throughput, average queue length, and Packet Retransmission Ratio (PRR). The performance is observed

at both the intermediate nodes and end systems. The throughput is obtained by accumulating the bytes received by each sink endpoint during the simulation time. Retransmitted packets are not taken into consideration in calculating the throughput. These metrics were chosen to represent the behaviour of the systems during the transmission phase, the processing in the intermediate nodes, and the situation at the receiving ends. The normalised throughput - normalised delay graph is produced and used to evaluate the AN and non-AN-based schemes.

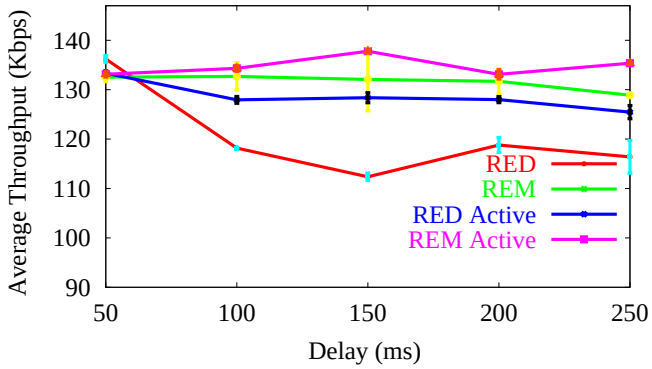
4.5.5 Results and Evaluation

4.5.5.1 Varying the Delay of the Uncongested Link

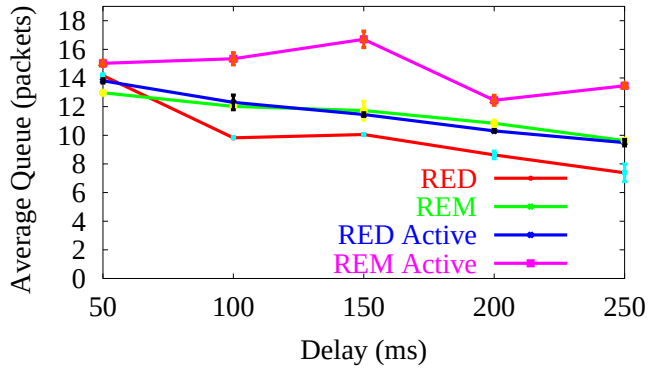
The following subsection considers the throughput, average queue length, and packet retransmission ratio for simulation, in which the delay of the uncongested link between R1 and R2 is varied to generate different bandwidth-delay product network conditions (Figure 4.11). Note that the number of sources is constant (10 sources). For throughput results, we plot the average throughput obtained by each of the queue managements algorithms deployed.

Figure 4.11(a) shows that the throughput of AN-based flows using REM is relatively stable when the delay of the uncongested link is increased. RED active throughputs are relatively constant across different round trip time experiments. REM active can improve the throughput up to 7.9% (the average throughput of 125.46 ± 1.34 Kbps for RED active and 135.36 ± 0.17 Kbps for REM active, when variable delay is 250 ms) compared with RED active. REM and RED act proactively upon detecting incipient congestion. The result in [106] shows that the throughput share under REM is independent of propagation delay. Our results confirm this, as we can see that the throughput obtained with various delays of the uncongested link is relatively constant which is 132 Kbps in average. Table 4.5 shows the average throughputs and their confidence intervals achieved from using RED active and REM active queuing algorithms. The percentage of improvement achieved from using REM active instead of RED active are also shown in the table. Furthermore, Table 4.6 shows the performance improvement of having ACC TCP for RED and REM, in which we compared average throughputs gathered from using RED active instead of RED. Then we also calculated the percentage of improvement from using REM active instead of REM.

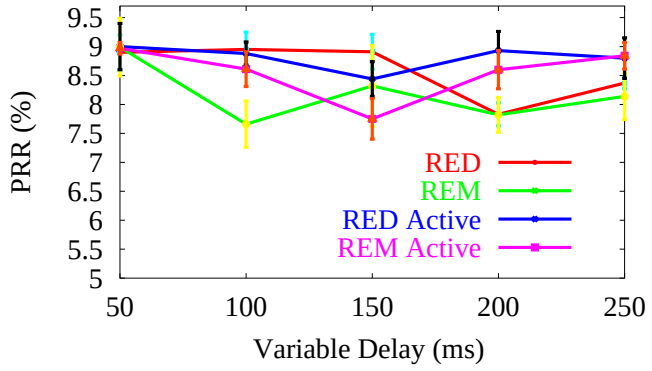
Figure 4.11(b) shows the average queue length for different queue managements. The results show that the average queue lengths for the AN-based flows are higher.



a) Throughput



b) Queue Length



c) PRR

Figure 4.11: Experiment U2 - Average throughput, queue length, and PRR for different queue management schemes (varying the delay)

Variable Delay (seconds)	RED Active (Kbps)	REM Active (Kbps)	Improvement (%)
50	133.33 ± 0.49	133.13 ± 0.59	-0.15
100	127.94 ± 0.70	134.32 ± 0.61	4.9
150	128.38 ± 1.02	137.79 ± 0.39	7.3
200	127.99 ± 0.62	133.11 ± 0.95	4
250	125.46 ± 1.34	135.36 ± 0.17	7.9

Table 4.5: Experiment U2 - Comparison of RED Active and REM Active total throughput statistics for Figure 4.11a. Improvement = ((REM active-RED active)/RED active)*100%.

Variable Delay (seconds)	RED Throughput (Kbps)	RED Active Throughput (Kbps)	Improvement (%)	REM Throughput (Kbps)	REM Active Throughput (Kbps)	Improvement (%)
50	136.21 ± 0.73	133.33 ± 0.49	-2.1	132.54 ± 0.92	133.13 ± 0.59	0.4
100	118.17 ± 0.14	127.94 ± 0.70	8.2	132.69 ± 2.80	134.32 ± 0.61	1.2
150	112.37 ± 0.76	128.38 ± 1.02	14.2	132.07 ± 6.36	137,79 ± 0.39	4.3
200	118.8 ± 1.51	127.99 ± 0.62	7.7	131.69 ± 2.56	133.11 ± 0.95	1.0
250	116.41 ± 3.33	125.46 ± 1.34	7.7	128.89 ± 0.34	135.36 ± 0.17	5.0

Table 4.6: Experiment U2 - Improvement when using RED active and REM active compared with non-AN for Figure 4.11a. RED Improvement = ((RED active-RED)/RED)*100%, REM Improvement = ((REM active-REM)/REM)*100%.

However, they are better with regard to capacity in absorbing the packet queue. In ACC TCP, REM packet queue lengths are consistently higher than those of RED. ACC TCP was designed for high bandwidth-delay product networks which allow the results for the network topology with different active and non-active queue algorithms in Figure 4.11(b) to be considerably maintained. REM active has a higher queue length due to its attempt to stabilise its queue using the queue mismatch and rate mismatch mechanisms. This characteristic leads to REM's high throughput feature. These results also show that the RED queue is decreasing and causes the link to be underutilised. The experiment results show that the low throughput and low queue length condition could happen when RED queueing scheme is used in the stable topology in which no cross traffic exists. This is due to the packet dropping mechanism in RED, which is influenced by RED parameters.

The packet retransmission ratio graph in Figure 4.11(c) shows that generally RED and REM show similar behaviour when the delay of the uncongested link increased. The packet retransmission ratio value for each variation in the delay is between 7.5 to 9.5 %. In the stable network, REM active does not exhibit significant performance benefit considering the whole spectrum of possible scenarios.

The normalised throughput - normalised delay characteristics of the schemes are shown in Figure 4.12. The normalised throughput is the total of throughput gained

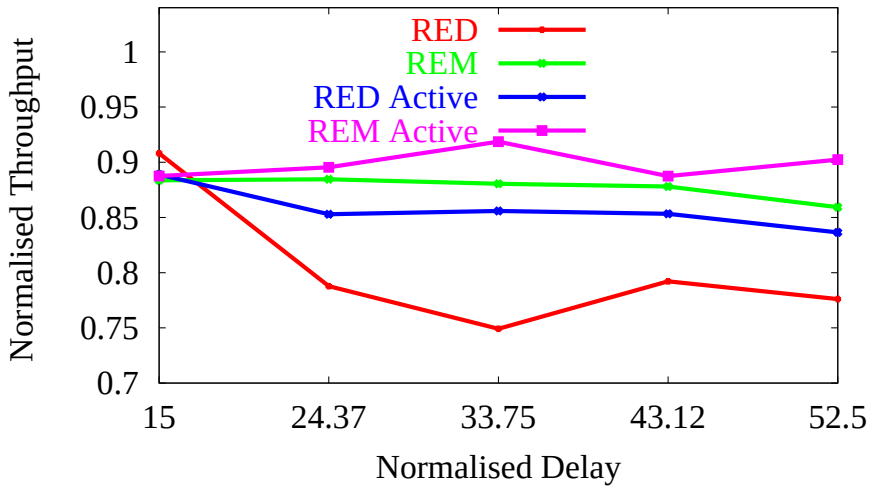


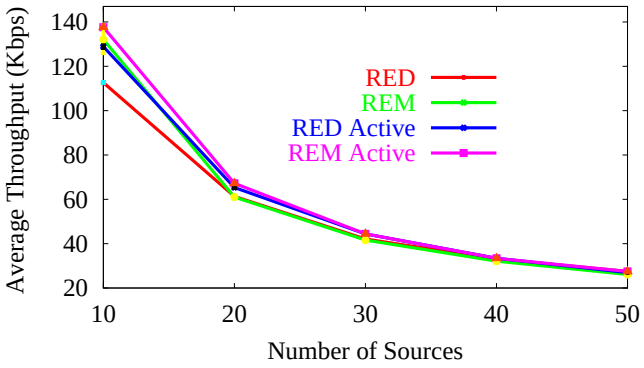
Figure 4.12: Experiment U2 - Normalised throughput - normalised delay plot for different queue management schemes (varying the delay)

divided by the maximum possible throughput (link capacity). The normalised delay is the sum of the total delay of each link divided by the mean packet transmission time (delay). The packet transmission time is the function of packet length over the available bottleneck bandwidth. The difference between the AN-based and non-AN-

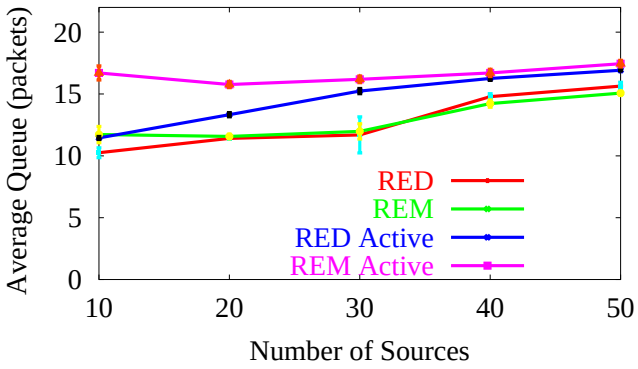
based schemes are clearly shown in this graph. The graph represents the average data shown in Figure 4.11(a), but the throughput and delay are normalised. The graph shows that REM active outperforms other schemes in terms of bandwidth utilisation. The bandwidth utilisation of the different queue management schemes is between 0.75 and 0.9.

4.5.5.2 Varying the Number of Sources

In the following subsection, we address the results of the experiment in which we increase the number of sources from 10 to 50 using the topology in Figure 4.10. Note that the delay of the uncongested link is kept constant at 150 ms. Figure 4.13(a) shows that when the load of the network is increased, the average throughput of



a) Throughput



b) Queue length

Figure 4.13: Experiment U2 - Average throughput and queue length with different queuing management schemes (varying the number of sources)

REM active is better than that of RED active (REM active average throughput (137.63 ± 0.13 Kbps) is 6.7% higher than RED active (128.87 ± 0.8 Kbps) for 10 sources). The throughput decreases correspondingly as the number of flows on the

link increases. As the total number of sources reaches 50, the average throughput for all the queue management mechanisms is 27 Kbps.

Figure 4.13(b) shows the average packet queue length for different schemes when the number of sources is increased. The REM active average queue length is higher than RED active in all cases (45.9% for 10 sources, 18.3% for 20 sources, 6.2% for 30 sources, 2.7% for 40 sources, and 3.1% form 50 sources). Furthermore, the AN-based schemes have longer average queue length than their non-AN-based counterpart (see Figure 4.13(b)). It can be noted that as the number of sources increases, all AQM schemes show a similar trend, in which the average queue length increases. This is due to the increasing number of packets flooding the link.

The normalised delay - normalised throughput characteristics of the schemes are shown in Figure 4.14. The normalised throughput is the total of throughput gained

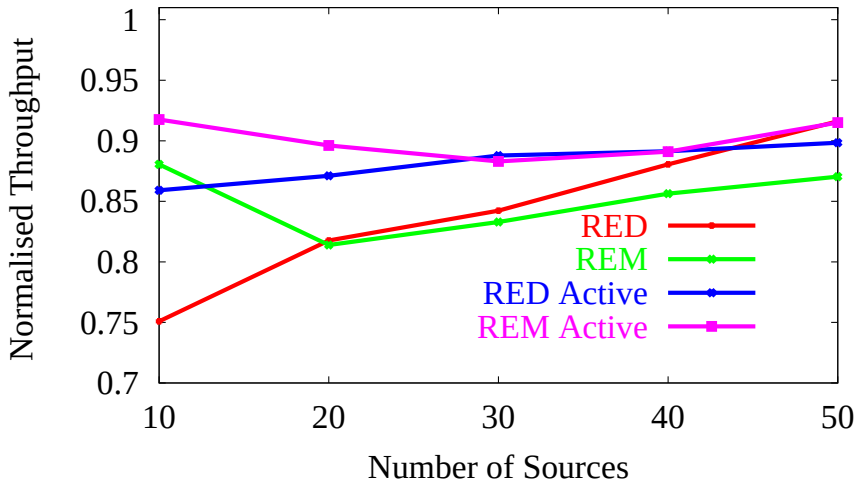


Figure 4.14: Experiment U2 - Normalised throughput plot for different queue management schemes (varying the number of sources)

(average throughput times the number of sources) divided by the maximum possible throughput (link capacity). The graph represents the average data shown in Figure 4.13(a), but the throughput is normalised to the bottleneck link bandwidth. The graph shows that in most cases REM active outperforms other schemes in terms of bandwidth utilisation.

4.6 Experiment U3: Bursty Network

The following subsections discuss the results of the simulation parameters' variation and their effects on congestion control mechanisms in the case where UDP

cross-traffic (to represent unresponsive traffic flows such as multimedia streams) is introduced to the network using the topology in Figure 4.16.

4.6.1 Experiment Construction and Description

This part of the experiments is the comparison of the AN and non-AN-based systems in a more complex topology. This is achieved by imposing bursty cross-traffic on the network. A set of UDP-based CBR cross-traffic sources is attached to the second router to represent real-time multimedia traffic, for example streaming. Similar to Experiment U2, the simulation experiments are run for each queuing policy, i.e. RED and REM, in order to compare their performance using different performance metrics. Within each scenario, simulations are run 1) for a constant number of sources and by varying the length of delay at the uncongested links, and 2) for a constant length of delay for the uncongested link and by varying the number of sources. The objectives of the experiments are 1) to observe the behaviour of the ACC TCP relative to TCP under different conditions in terms of the AQM used in the node (RED and REM) and, 2) to analyse the set of experiments when running ACC TCP REM.

The simulation is performed in a network environment in which FTP applications are attached to the TCP sources and flow towards the TCP sinks. Subsequently, constant bit rate cross-traffic sources are added to the network to evaluate the impact of bursty cross-traffic on the throughput of the network.

Figure 4.15 presents a flow-chart of the simulation procedure. Similar to Experiment U2, some Tcl files are used to define the desired topology, but in this experiment we use a set of cross traffic. The simulation is repeated with different random number generator seeds. Subsequently the simulation is run $5 * 4 * 4$ times (varying the delay, use different queue management algorithms). Each execution is repeated 5 times, adding another 100 executions in total. Then we vary the number of sources, use different queue management algorithms, and execute them 5 times each (another 100 executions in total). Furthermore, the experiment data is captured in trace files which are then subsequently parsed using awk or Perl scripts to get useful data. They in turn are used to calculate performance results and to plot graphs using gnuplot.

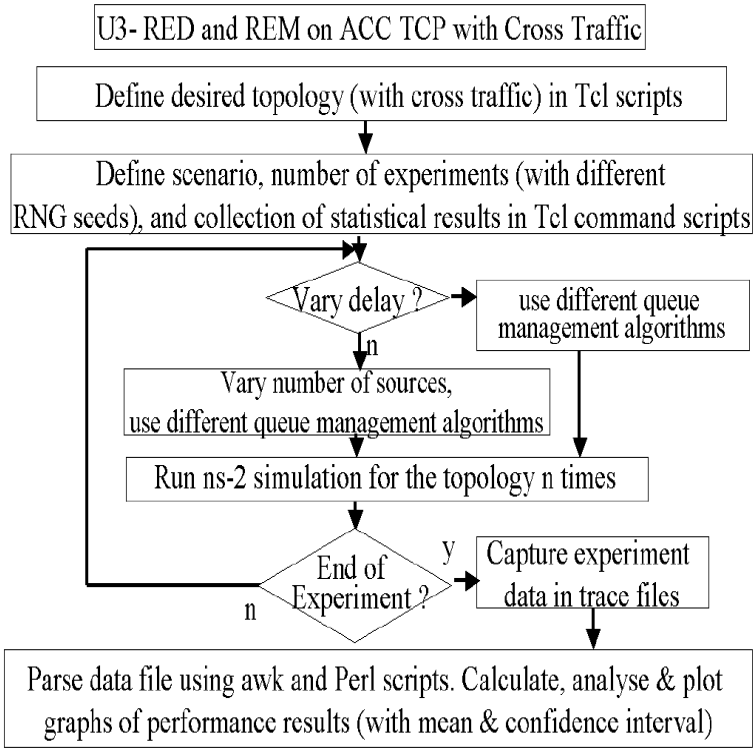


Figure 4.15: Experiment U3 - Flowchart of simulation procedure

4.6.2 Topologies

The simulation experiments performed in this study use the network configuration depicted in Figure 4.16. Similar to Experiment U2, N sources transmit data over the link between routers 1, 2 and 3. The traffic is uni-directional. The link between routers R1 and R2 is the uncongested link with variable delay. Unlike Experiment U2, in this experiment, more congestion is generated when a set of UDP cross-traffic sources is added to router 2. The traffic is generated from 10 UDP sources sending bursty traffic at a rate of 100 Kbps. The configuration parameters are detailed in Table 4.7, which shows the variable number of sources and variable delays in the uncongested link used in these experiments. We use the same topology and network parameters for both TCP and ACC TCP experiments.

4.6.3 Key Parameters and Rationale

The parameter settings applied in the simulation were chosen to induce congestion in the network. The delay at the uncongested link between R1 and R2 is varied from 50 to 250 ms. We evaluate the performance of RED and REM in ACC TCP when

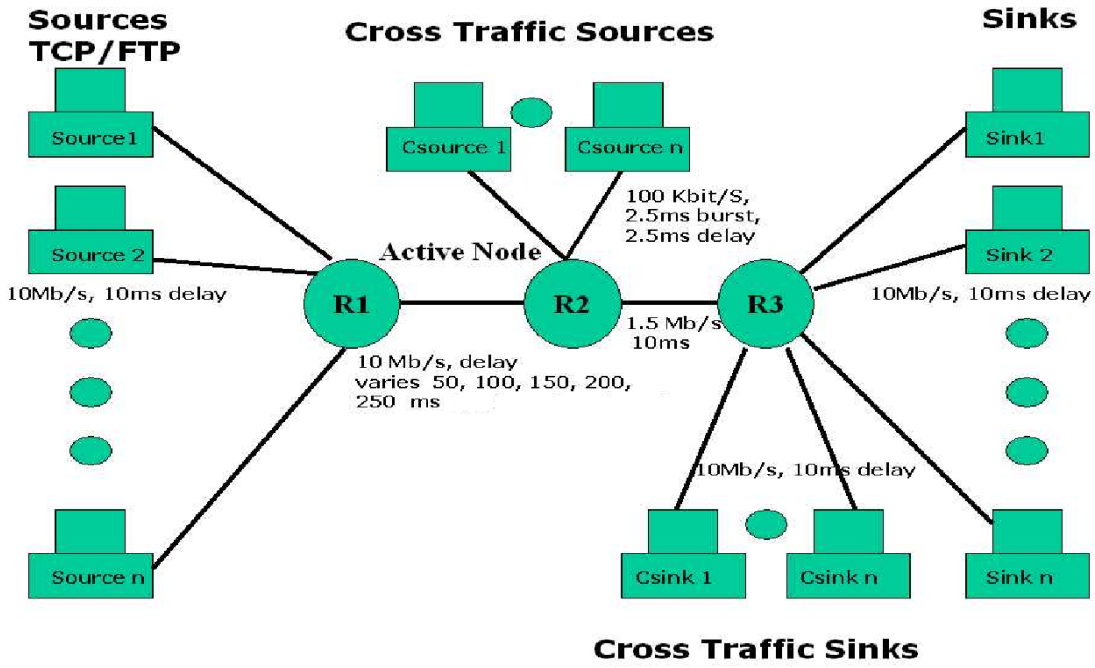


Figure 4.16: Experiment U3 - Topology for experiment imposing cross-traffic on the network (bursty traffic)

large numbers of sources and cross-traffic sources share the links. The cross-traffic is bursty UDP flows which are not responsive to congestion. The bandwidth of the links is 10 Mbps and the link delays are 10 ms, except for the link between R2 and R3, which is 1.5 Mbps. The simulation parameters can be seen in Table 4.7.

In these simulation experiments, RED and REM parameters were also set to the same common values as in the previous experiment (Experiment U2). The parameters are set as follows: RED buffer's maximum threshold (`maxth`) = 18 packets, minimum threshold (`minth`) = 12 packets, maximum probability (`maxp`) = 0.1, and queue weight (`wq`) = 0.002. In the case where REM queue management was used, the following parameters were set according to REM parameter's value as follows, δ (weight in aggregate input rate estimate) = 0.02, β (weight in RTT estimation) = 0.02, N (sample size for price estimation) = 50, γ (step size in price adjustment) = 0.001, and α_1 (weight of buffer in price adjustment) = 0.1.

4.6.4 Performance Metrics

The performance metrics of the experiments reported consist of throughput, average queue length, and packet retransmission ratio. The performance metrics used and

Simulation Parameters	Value
Sources	10, 20, 30, 40 and 50 TCP sources, linked to sinks through routers
Sources - Router 1 and Router 3 - Sink links	10 Mbps, 10 ms delay
Router 1 - Router 2 link (Uncongested link)	10 Mbps, variable delay (50, 100, 150, 200, 250 ms)
Router 2 - Router 3 link	1.5 Mbps, 10 ms delay
Cross-traffic	10 UDP exponential on-off sources
Cross-traffic - Router 2 link	100 Kbps, 2.5 ms burst, 2.5 ms delay
All routers maximum buffer size	25 packets
Simulation duration	200 seconds
Packet Size	1000 bytes

Table 4.7: Experiment U3 - Simulation parameters for topology in Figure 4.16

the objectives are the same as in the previous experiment (Experiment U2). The normalised throughput - normalised delay graphs have also been produced and used to evaluate the schemes.

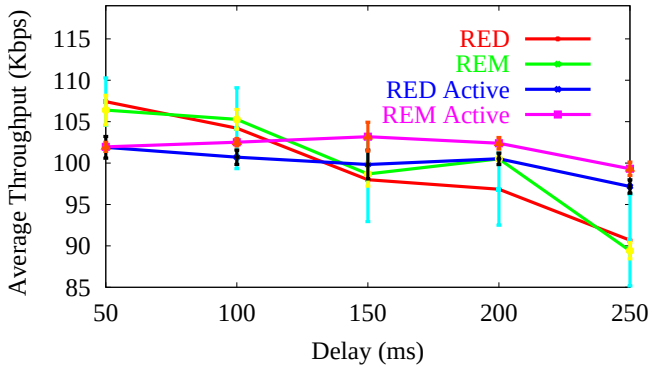
4.6.5 Results and Evaluation

4.6.5.1 Varying the Delay of the Uncongested Link

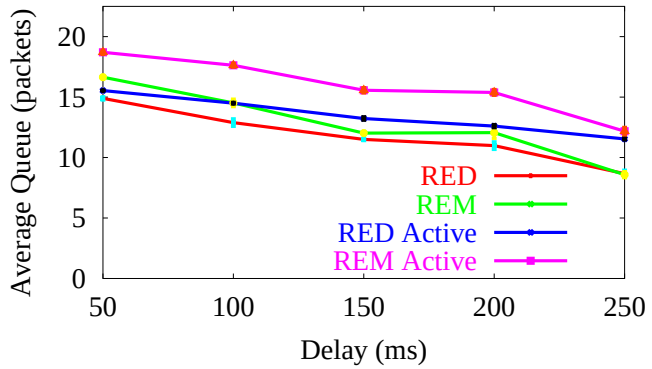
This section discusses the experiments in which different delays (50, 100, 150, 200, 250 ms) in the uncongested link (between Router 1 and Router 2 in Figure 4.16) are considered. Figure 4.17(a) shows that when the delay at the uncongested link is set to 150, 200 and 250 ms, all active-based congestion control schemes perform better than the non-active ones. RED active performed 7.1% better than RED (the average throughput of 90.69 $\pm$ 5.54 Kbps for RED, and 97.17 $\pm$ 0.7 Kbps for RED Active). On the other hand, REM active is 11% better than REM (the average throughput of 89.41 $\pm$ 0.98 Kbps for REM, and 99.41 $\pm$ 0.81 Kbps for REM active).

It can be seen in Figure 4.17(b) that the average queue length of different queue mechanisms shows a decreasing trend. The buffer occupancy of REM active is consistently the highest among all queuing mechanisms. In the high bandwidth-delay product link which must be shared with bursty cross-traffic, REM AQM in ACC TCP tried to utilise the buffer to ensure the stability of the flows. In some points, the confidence interval is very small to show up clearly in the graph.

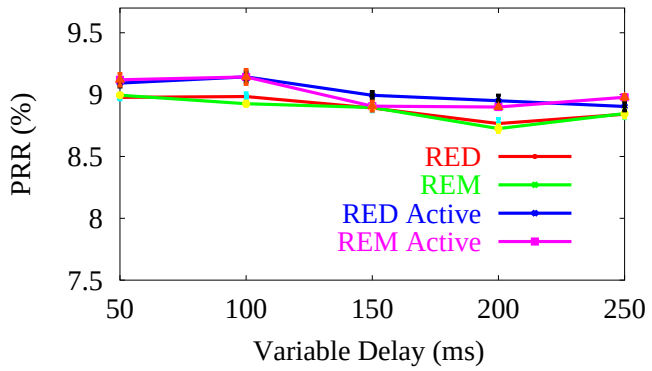
Figure 4.17(c) shows a slight reduction trend in the packet retransmission ratio versus variable delay graphs. The PRR for all schemes is almost the same, approximately 8.75% - 9.25%, which means that the number of packets retransmitted is



a) Throughput



b) Queue Length



c) PRR

Figure 4.17: Experiment U3 - Average throughput, queue length, and PRR with different queuing management (varying the delay with UDP cross-traffic)

low for all schemes.

Generally, ACC TCP-based schemes react better than the TCP-based ones to bursty cross-traffic. It can be seen that RED and REM queuing mechanisms with ACC have higher throughput, especially in high bandwidth delay product areas. ACC does not only depend on router communication but has a congestion relief mechanism which yields better performance.

Table 4.8 shows the comparison of the throughput of REM active to the RED

Variable Delay (seconds)	RED Active (Kbps)	REM Active (Kbps)	Improvement (%)
50	101.90 ± 1.31	101.96 ± 0.55	0.1
100	100.71 ± 0.87	102.52 ± 0.45	1.7
150	99.83 ± 1.75	103.18 ± 1.75	3.3
200	100.52 ± 0.7	102.39 ± 0.70	1.8
250	97.17 ± 0.81	99.31 ± 0.81	2.2

Table 4.8: Experiment U3 - Comparison of RED and REM total throughput on ACC TCP for Figure 4.17a. Improvement = ((REM active-RED active)/RED active)*100%.

active queuing mechanism, in which REM active performed up to 3.3% better than the RED active. In order to evaluate the improvement generated by using ACC TCP we have calculated the improvement of using RED active instead of RED, and also REM active instead of REM. Table 4.9 shows the results of the calculation, in

Variable Delay (seconds)	RED Throughput (Kbps)	RED Active Throughput (Kbps)	Improvement (%)	REM Throughput (Kbps)	REM Active Throughput (Kbps)	Improvement (%)
50	107.42 ± 2.87	101.9 ± 1.25	-5.1	106.4 ± 1.79	101.96 ± 0.55	-4.1
100	104.21 ± 4.89	100.71 ± 0.35	-3.3	105.28 ± 1.19	102.52 ± 0.45	-2.6
150	97.99 ± 5.06	99.83 ± 1.33	1.8	132.07 ± 6.36	137.79 ± 0.39	4.5
200	96.84 ± 4.35	100.52 ± 0.68	3.8	100.51 ± 0.08	102.39 ± 0.7	1.8
250	90.69 ± 5.54	97.17 ± 0.7	7.1	89.41 ± 0.98	99.31 ± 0.81	11.1

Table 4.9: Experiment U3 - Improvement when using RED active and REM active compared with non-AN for Figure 4.17a. RED Improvement = ((RED active-RED)/RED)*100%, REM Improvement = ((REM active-REM)/REM)*100%.

addition to the average throughput data and the confidence interval for each schemes when the delay of the uncongested link is varied.

Figure 4.18 illustrates the normalised throughput - normalised delay characteristics of the schemes. The normalised throughput is the total of throughput gained

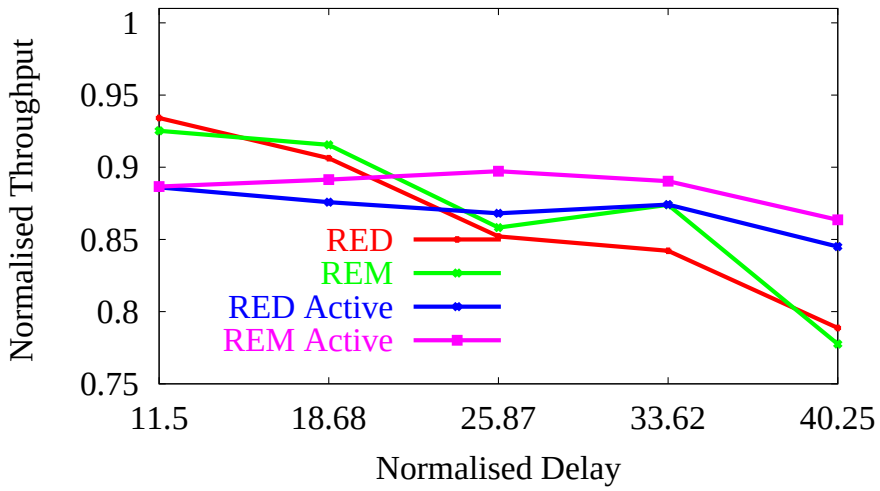
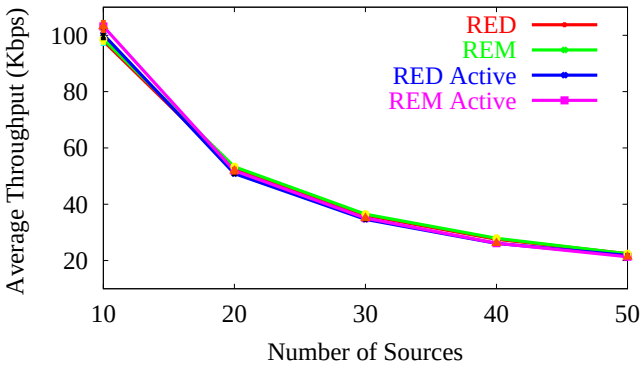


Figure 4.18: Experiment U3 - Normalised throughput - normalised delay for different queue management schemes (varying the delay)

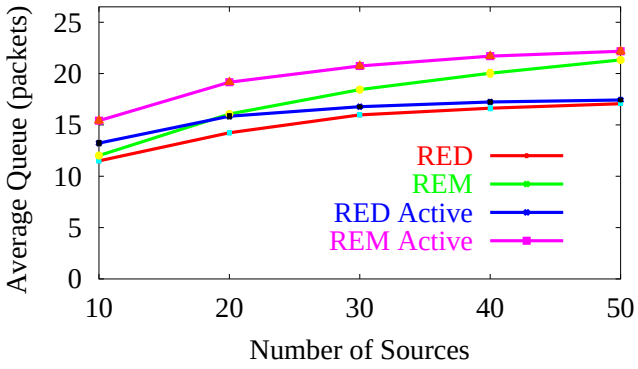
divided by the maximum possible throughput (link capacity). The throughputs gained by the sources are affected by the bandwidth used by the cross traffic. Normalised delay is the sum of the total delay of each link divided by the packet transmission time (delay). The packet transmission time is the function of packet length divided by the available bandwidth at the bottleneck link. This graph shows that for all schemes, the utilisation of the bandwidth is 0.77-0.93. This graph represents the average data of the results in Figure 4.17(a), but the representation is based on normalised throughput and delay calculation to compare the schemes.

4.6.5.2 Varying the Number of Sources

The following result shows the study in which we introduce more sources to the network with bursty cross-traffic. We keep the uncongested link delay constant at 150 ms. Figure 4.19(a) shows that the throughput for all active and non-AN-based flows are decreasing, from the average of 100.46 Kbps when 10 sources are activated, to 21.84 Kbps when 50 sources are enabled at the same time. Most of the AQM schemes show that the throughput at the uncongested link decreases proportionally to the number of connections using the link. The reason is that with a larger number of connections, it takes a larger number of packet drops to be sufficient to signal the host to back off the sending rate. Figure 4.19(b) shows an increasing trend in queue occupancy. REM average queue length for the uncongested link is higher than that of RED. Furthermore, REM-based schemes also have a higher queue length than those of other schemes, for the reason that in this region REM tries to stabilise the



a) Throughput



b) Queue length

Figure 4.19: Experiment U3 - Average throughput and queue length with different queuing management schemes (varying the number of sources with UDP cross-traffic)

network.

In general, for multiple sources and queuing techniques at the router with basic TCP, the graphs show a similar trend to the results given by ACC TCP. Overall, the results show that using active congestion control and AQM will not always improve the performance of the system in every case. In some cases, an AN-based scheme is better at providing high throughput, but the algorithm manages the queue length in order to achieve desired features such as stability.

The normalised delay - normalised throughput characteristics of the schemes are shown in Figure 4.20. The graph represents the total throughput gained (the

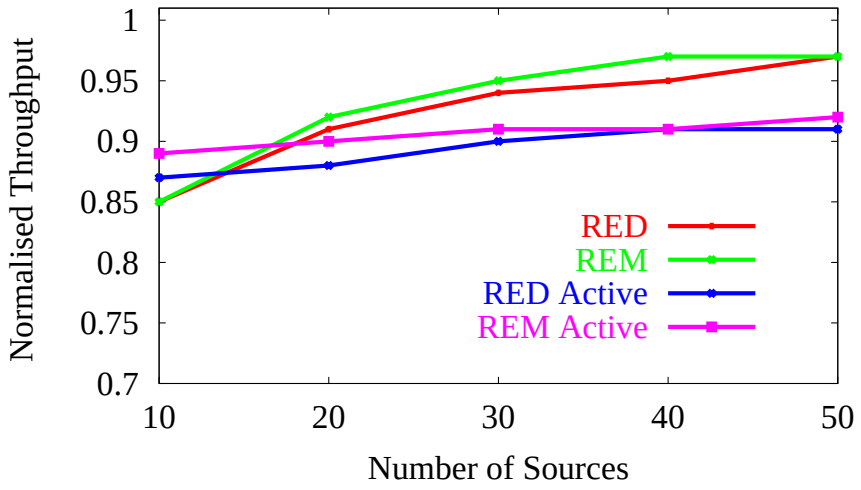


Figure 4.20: Experiment U3 - Normalised throughput plot for different queue management schemes (varying the number of sources)

average throughput times the number of sources) of the data shown in Figure 4.19(a), but the throughput has been normalised to the maximum available bottleneck link bandwidth which is affected by the bursty cross traffic. We take into account the 7*100 Kbps flow of the cross traffic, and for the on-off character of bursty traffic it has been assumed that they consumed 350 Kbps. We consider the maximum available bandwidth is 1150 Kbps out of the 1500 Kbps bandwidth. The graph shows that the bandwidth utilisation of the non-AN based schemes outperforms RED active and REM Active schemes when the number of sources increases.

4.6.6 Limitation of the Scheme

As one of the early active networks application proposals, ACC TCP follows the 'active packets' approach, which is characterised by the fact that the code is carried

by packets. The active code can either be executed on the data of the same packet that carries the code or in order to change the state or the behavior of the node.

The ACC TCP uses acknowledgements and active packets to perform congestion control. In order to rectify the message passing in ACC TCP, we have conducted some simulations using a dumbbell topology with 2 TCP sources. All the links have 10 Mbps bandwidth and 10 ms delay, unless otherwise stated. The bottleneck link bandwidth is 500 Kbps and 50 ms delay. Each packet is 1000 bytes long. The simulation lasts for 50 seconds. Simulation of each scheme is run 10 times with different random number generator seeds. Table 4.10 illustrates the results

Messages\Schemes	**RED (Kbps)**	**REM (Kbps)**	**RED active (Kbps)**	**REM active (Kbps)**
TCP1 data	258.7 ± 10.5	259 ± 38.5	280.8 ± 40	253.4 ± 59
TCP2 data	257.76 ± 10.4	244.48 ± 45.1	256.3 ± 37.9	259.2 ± 62.5
Acknowledgements	59.32 ± 0.01	56.74 ± 0.90	57.5 ± 0.40	51.6 ± 0.56
Active packets	0	0	1.92 ± 0.1	6.2 ± 0.3
PLR	0.29%	0.39%	2.9%	3.3%

Table 4.10: Throughput, Acknowledgement, active packets, and PLR of AN and non-AN based schemes

of each simulation which include the total number of bits normalised into time for acknowledgements packets, TCP data packets, and active packets obtained from different AN and non-AN based schemes.

The average throughput of acknowledgements packets of all AN-based schemes and non-AN based schemes calculated from Table 4.10 is 56.29 Kbps. However, for the AN-based scheme, we have another type of message called 'active packets'. During the simulation an average of 39 active packets of length 1000 bytes are generated for REM active (6.2 ± 0.3 Kbps) and 12 such packets for RED active (1.92 ± 0.1 Kbps). This is the associated cost of having the AN-based packets.

We reveal that the ACC TCP algorithm was not tuned. One of the limitations of ACC TCP is that the current available implementation was written for functionality rather than for performance comparison to non-AN-based protocol. This can be seen from the higher number of packets dropped experienced by the active-based schemes. This limitation is also acknowledged by the author in his paper [49].

The ACC is in a similar situation to other AN-based protocols. It is still far from the deployment stage to be implemented on TCP, due to its complexity and small performance improvement. We found that ACC TCP is not a complete set of protocols. Results published in [121] on High Performance Active Node (PAN)

showed that AN could provide significant flexibility relative to traditional network with only a small performance overhead (13% for 1500 bytes packets). The fixed message passing overhead reported for this full set of protocols is 20 microseconds per capsule. This is for an experiment in which capsules execute native object code. For capsules executing Java code, the communication overhead is 3 times higher than that with native code for 128 byte packets. The message passing overhead of an AN-based system will largely be determined by the language used and the framework.

At the application level, if we assume that ACC TCP can be implemented using the ANEP encapsulation protocol or Active IP options such as in Figure 4.21, the

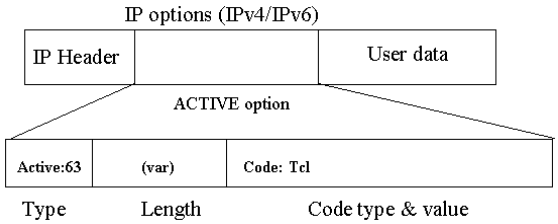


Figure 4.21: Active IP packet header [128]

message-passing overhead of ACC TCP can be calculated as follows. Recall that ACC TCP needs a 4 bytes characterisation per packet. Assume that we need to download an *openoffice*[11] package (72 Mb) from a site at the University of Indiana, USA. The distance between Leeds to Indiana is approximately 4064 miles. The delay between the two places is approximately 100 ms[12]. If it is assumed that we need to send the package over a 128 Kbps link, and each packet is 1000 bytes long, then we will need to send 9000 packets. The time to download the package is the total of the time to pump the packet in to the network, the round trip time delay, and the time to send the ACC TCP active packets. The 9000 packets will need 562.5 seconds (9.375 minutes) to be downloaded. With the consideration that each packet will have another extra 4 bytes, it will take 2.25 seconds to send the active packets. This is the overhead of having extra active options in the packet header. Thus, the total time to download the *openoffice* package will be 9 minutes 24.85 seconds. In addition we have to include the computation time in the active node (router) to perform the congestion control and routing mechanism, as well as queuing delays.

In the higher level, a protocol such as ACC/TCP is also limited by the execution

[11] www.openoffice.org
[12] www.internetrafficreport.com

environment used and how the protocol should be expressed in it. The active code must be compact (less than 16 KB) [173]. Moreover, the protocol implementation must also take into account that not all nodes will be active. Overall, although ACC/TCP provides a good approach in applying AN to congestion control, in practice its deployment to the Internet is difficult, due to the limitations of this scheme and its overheads.

4.7 Discussion

The following is a brief summary of the results of the experiments on the AN-based unicast congestion control protocol:

- Experiment U1 presents some results on queue length and implication of different sources' starting time of RED and REM in ACC TCP. The statistical analysis shows that in a topology in which the initiation of sources has been staggered, the average queue length of REM is more stable than RED. Using RED active, the average queue length keeps increasing as the number of sources increases.

- Experiment U2 on a three router-based network topology shows that when *the delay of the uncongested link is varied*, the average throughput of REM active is better than RED active, whereas the queue length is higher and the packet retransmission ratio is lower. On *varying the number of sources*, the throughput tends to decrease and the queue length increases. REM active throughput is better than RED active from 6.8% (the average throughput of 128.87 $\pm$ 0.8 Kbps for RED active and 137.63 $\pm$ 0.13 Kbps for REM active at 10 sources), drops to 0.05% at 30 sources (and then the difference becomes negligible).

- Experiment U3 shows that when cross-traffic is added to the network, the throughput and the packet retransmission ratio for each queuing management scheme are similar, although the *RTT has been increased*. The average queue length of REM active is constantly the highest, despite the fact that all queuing schemes tend to increase. On *varying the number of sources*, the throughput is decreasing, and the average throughput value is very similar for all queuing schemes (the overall average of 100.03 $\pm$ 2.3 Kbps for 10 sources, 52.18 $\pm$ 1.04 Kbps for 20 sources, 35.57 $\pm$ 0.84 Kbps for 30 sources, 26.94 $\pm$ 0.94 Kbps

for 40 sources, and 21.84 $\pm$ 0.70). On the other hand, the queue length is increasing, and REM active queue length is the highest.

In the following points we discuss the use of ANs and AQM in these experiments and the reason behind ACC TCP's behaviour:

- Throughout our investigation on ACC TCP, we assume that the *AN mechanisms* to perform feedback congestion control is working concurrently with the AQM schemes to tackle the congestion simultaneously. In other words, we consider that RED and REM perform the normal dropping and marking operations in handling congestion. The RED router discards packets before the queue becomes full. The REM router determines the marking probability using the 'price' (rate mismatch and queue mismatch information). In REM, on the increase of the load, the mismatches in rate and queue increase make the price and the marking probability increase too, and enforce sources to reduce their rates.

- In employing REM as the AQM scheme, simulations with standard TCP demonstrated that the overall link utilisation was better or at least as good as the link utilisation of RED. In most cases, REM made better use of available resources, due to its congestion measure mechanism, in which congestion means that demand exceeds supply, but the performance remains in control. This is different from RED, in which congestion means deteriorated performance. These results can be attributed to the features of REM itself, in which high utilisation and negligible loss and delay are the objective. For the AN-based schemes, the REM active performs better than RED active. This shows that the *AN mechanisms perform feedback congestion control* and play their role together with the AQM algorithms in improving the performance.

- Our experimental results suggest that *a direct feedback congestion control* mechanism using *the AN approach* contributes to the slight improvement in the throughput. ACC TCP does not depend solely on end-point to route communication. At small bandwidth delay product regime, the routers in the ACC system are editing traffic on behalf of the end point for only a short interval. On the other hand, the similar results (in which standard TCP throughput are higher compared with ACC TCP for 50, 100, and 150 ms link delays) in U3 can be attributed to the fact that the standard TCP is effective enough in a small bandwidth delay products regime because the feedback loop is short.

- The simulation results imply that the ANs and AQM mechanisms in the high bandwidth-delay product regime do provide performance gains, although insignificant. This can be attributed to the effect of using *AN functionalities* in the unicast protocol, in which the delay in communicating congestion information to the endpoints can be avoided.

- The ACC/TCP has some limitations in terms of the cost of having extra active packets, the use of a higher level execution environment and their overhead, and the number of nodes which must be activated and capable of computing as well as forwarding active packets.

It can be summarised that AN support in performing unicast congestion control has proved to be working well. Although this performance gain is not very significant, it shows that ANs can improve the performance of the network. The power of the feedback congestion control protocol using AN and the use of AQM contribute to the slight improvement in the network performance.

4.8 Summary

In this chapter we have presented the performance evaluation of the TCP congestion control protocol using ANs. In essence, the chapter embraces two visions in networking, namely ANs and AQM. The impact of two different AQM algorithms, i.e. RED and REM, have been evaluated to see their effect on the network topologies with and without using the AN approach. We have compared the performance of RED, RED active, REM, and REM active with standard parameter settings of RED and REM. Performance metrics are throughput, queue length, and packet retransmission ratio. Evaluation was performed on a traffic mix and variable uncongested link delay, leading to variable bandwidth delay product link and network load. The microscopic behaviour analysis of RED active and REM active demonstrated that REM is better in terms of stability in steady state condition, in which the average queue length occupancy is lower.

In general, we have shown that REM active schemes can improve the performance of the network up to 7.9% compared with RED active (stable network at 250 ms delay, the average throughput of 125.46 ± 1.34 Kbps for RED active and 135.36 ± 0.17 Kbps for REM active). This is mainly due to the fact that REM leads to low buffer occupancy in the steady state. In this case, REM active enables more packets to be transmitted compared with RED active. However, the throughput, average

queue length, and packet retransmission ratio comparison really shows that the performance of REM, as well as the performance of RED in ACC TCP, is susceptible to traffic variability in terms of network parameters and topology. This work shows that augmenting an existing feedback system with an AN-based algorithm, and employing RED and REM queuing mechanisms, are beneficial to the overall performance of the network, although the improvement is somewhat marginal. This aggregate assessment contributes to the literature on network programmability in adapting to changes such as ANs and queuing management.

One of the goals of deploying an AN-based congestion control protocol is to ensure that a mechanism to overcome the congestion can be applied closer to the source of the congestion itself, instead of having to wait until the end system reacts to it. Within the context of ANs and congestion control, the motivation behind it is to gain a better understanding of the impact of ANs on unicast traffic management. The results show that when both ACC TCP and AQM approaches are implemented, the performance results would not always be superior. All queue management mechanisms perform well in certain scenarios, but sub-optimally in others. We argue that this is due to the fact that the algorithm was designed for functionality rather than performance improvement. Hence, there is a need to tune the algorithms to co-operate in achieving better network performance.

In the next chapter, we address the performance evaluation of AN-based single rate multicast protocols. It will focus on the AER/NCA protocol and the desirable TCP-friendly feature of a multicast protocol.

Chapter 5

Performance Evaluation of AN-based Multicast Single Rate Congestion Control Protocols

In Chapter 4 we presented our performance evaluation results on a unicast Active Networks (ANs)- based congestion control protocol, particularly on the effect of complementary active queue management schemes. This chapter discusses experimental work conducted to evaluate the performance of AN-based single rate multicast congestion control protocols. Section 5.1 introduces the chapter by reviewing the AN-based congestion control protocols and TCP-friendliness. Section 5.2 reviews the AN and non-AN-based single rate multicast congestion control protocols addressed in this work, i.e. Active Error Recovery/Nominee Congestion Avoidance (AER/NCA) [88] and Pragmatic General Multicast Congestion Control (PGMCC) [135]. Section 5.3 shows the experimental design, presents the simulation environment, and discusses the results of the experiments on AER/NCA and its dynamic behaviour. Section 5.4 presents the comparison of a traditional multicast single rate protocol (PGMCC) with AER/NCA in terms of inter-protocol fairness (TCP-friendliness). Section 5.5 addresses the simulation experiments to gain the fairness index values for AER/NCA and TCP flows which share a bottleneck link using RED or Droptail routers. In Section 5.6, the limitations of the AN-based scheme are evaluated. Section 5.7 discusses the overall performance evaluation conducted for the single rate multicast protocols. Finally, Section 5.8 summarises our study on AN-based single rate multicast congestion control protocols.

5.1 Introduction

Reliable multicast has been proposed in the last decade to support the transmission of data from a sender to many receivers. It generally includes two mechanisms, error recovery and congestion control. The nature of multicast communication brings about some problems, such as the large volume of feedback traffic (*feedback implosion*), the large demand for retransmission to ensure data delivery to many receivers (*retransmission scoping problem*), the complexity of recovering from packet loss (*distributed error recovery*), and *congestion control mechanisms* to achieve inter-protocol fairness. Recall that in this work we use the word TCP-friendliness and inter-protocol fairness interchangeably when the evaluated protocol shares the link with TCP flows.

PGMCC and AER/NCA are two window-based protocols which have a similar congestion control approach to improve the reliable multicast. The PGMCC way to select the *acker* is similar to the way AER/NCA selects the *nominee* (the *weakest receiver*). Table 5.1 compares PGMCC and AER/NCA congestion control based on

Consideration	**PGMCC**	**AER/NCA**
NACK implosion	Aggregation, limited suppression	Aggregation, suppression
Loss recovery burden	Sender	Sender, repair server
Retransmission scoping	Router controlled	Router-supported subcast, retransmission interception
Congestion control	Reacts to every NACK at source	Reacts to losses at worst receiver

Table 5.1: Comparison of PGMCC and AER/NCA based on reliable multicast protocols' consideration [88]

reliable multicast protocols' consideration in solving the NACK implosion problem, loss recovery, and retransmission scoping.

Reliable multicast has been relying on end-to-end support in which the inter-receiver fairness criterion is hard to achieve. In order to overcome the problem and the limitation of reliable multicast, the use of the ANs approach has been proposed. ANs permit the application to achieve optimal data rates by specifying the network to take into account factors such as data rates, data formats, network access speed, and end-system performance [163]. Active services can be performed at strategic locations inside the network to help reliable multicast transport protocols recover from loss at the point of loss.

In this chapter, we address the AER/NCA protocol from the Panama project of TASC[1] and the University of Massachusetts, which presented packet loss recovery and congestion control protocols. We have chosen AER/NCA to be evaluated, as it is a complete instance of an active service. Furthermore, AER/NCA has been simulated using ns-2 [51], implemented as an active application in ANTS EE [171], and tested over ABone (Active Network Backbone) - the DARPA-funded testbed for AN research. Performance evaluation of this protocol to reveal its characteristics will contribute to the literature of simulation-based AN-based protocols.

As TCP-based traffic forms a dominant portion of the existing global Internet traffic, the TCP-friendly characteristics of a newly introduced protocol are an important issue. A new protocol's peaceful coexistence with TCP and their fairness characteristics ensure smooth deployment of the new protocol into the Internet. AER/NCA is a window-based reliable multicast protocol which has been designed to be able to share a link with predominant TCP-based traffic.

In this chapter, we compare the AN-based single rate multicast congestion control protocol - AER/NCA, with a representative of a non-AN-based one - PGMCC. We evaluate and compare the performance of these protocols in terms of inter-protocol fairness (TCP-friendliness). One of the goals of the simulation experiments is to assess the contribution of the active application, considering that TCP-friendliness is one of the objectives of both the AN and non-AN-based protocols. The aim was to confirm that single rate window-based multicasting is feasible with ANs and could bring some benefits, such as TCP-friendliness.

5.2 AER/NCA, PGMCC and TCP-friendliness

AER/NCA is a multicast congestion control protocol which aims to overcome the problem of congestion and to achieve better performance. It is a reliable multicast architecture that invokes active services at strategic locations inside the network to address the problem of feedback implosion, retransmission scoping, distributed loss recovery, and congestion control. It consists of two protocols: Active Error Recovery (AER) - a kind of flow control for packet loss recovery, and Nominee Congestion Avoidance (NCA) for congestion control [88]. The objective of using the active services architecture is to improve resource usage, latency for loss recovery, and provide 'TCP-friendly' congestion control. Indeed we found these properties to be attained during our experiments with AER/NCA.

[1]TASC is a research and development company. This AER/NCA project is funded by DARPA.

The problem of multicasting covers a broad area, such as routing, flow control, error recovery and congestion control. In this work, we will focus only on its congestion control mechanism and TCP-friendliness. In addition, AN help in the repair server mechanism, scalability of receivers, and the effect of varying the RTT are also considered.

Reliability of the protocol is a major concern of this 'hop-by-hop' feedback-based multicast protocol [88]. Error correction is performed by the active error recovery algorithm, in which feedback implosion avoidance, packet retransmission scoping, distribution of loss recovery burden, and fair congestion control are the advantages. The AER/NCA signalling mechanism uses additional active packets with predefined meaning, such as Source Path Message (SPM), Repair, Negative Acknowledgement (NACK), Congestion Status Message (CSM), Congestion Control Message (CCM), Data, Enable, Disable, and Round Trip Time (RTT). Congestion avoidance is based on the average throughput calculated from the weakest (nominee) link's RTT and Loss Path Estimates (LPE). Rate adjustment is performed by the adjustment of the congestion window, based on the slow-start threshold (similar to fast recovery of TCP New Reno).

AER/NCA is designated for single rate multicast flows to ensure reliability that a packet sent by a multicast session arrives in all subscribing receivers. One of the uses of ANs is to help in providing router support for hop-by-hop feedback. The ANs capsules help in error recovery by having a buffer in the intermediate nodes, and also function in congestion relief.

Rizzo's PGMCC multicast protocol [135] is another approach to congestion control using the same fundamental idea to select the worst receiver as the group representative. This protocol is a 'traditional'[2] multicast congestion control protocol, and does not use active services to perform multicasting. PGMCC is a congestion control scheme which can use router support but does not rely on it. It is based on Pragmatic General Multicast (PGM) [152], a multicast protocol in which Negative Acknowledgements (NACK)-based retransmission requests have been used to improve the basic IP multicast. PGM-enabled routers perform hop-by-hop NACK forwarding and support replicated NACKs coming from the same subtree.

Both PGMCC and AER/NCA use a special control message called Source Path Message (SPM) to make feedback go through the same path as the forward data traffic. PGMCC is an extension of PGM where some options to PGM packets are added: one new packet type, and some modifications to the sender and receiver

[2]non-AN-based protocol

procedures. The selection of the representative[3] is based on receiver reports in which the NACK options with three fields have been added to the packet format, i.e. the identity of the receiver, the highest known sequence number, and the loss rate measured locally. PGMCC measures RTT in terms of packets, in which the sender computes the difference between the most recent sequence number sent and the highest known sequence number from the receiver. The RTT measurement in the receiver is only used to compare the receivers, which in turn will select the Acker. The transmission rate will be the one of the Acker's. The procedure to select the group representative and track changes is important to the correct operation of PGMCC. In PGMCC, network elements can be used to implement feedback aggregation to improve scalability. This is because in PGM, NACKs coming from the receiver are filtered by network elements, so that only the first instance is forwarded. PGMCC considers the switchover between different ackers not as a change in the acker, but instead as a move of the acker to a different location.

Both AER/NCA and PGMCC are TCP-friendly protocols in which the flow must adapt its throughput T according to the TCP equilibrium equation [75, 88]:

$$T = \frac{C * MTU}{RTT * \sqrt{p}} \tag{5.1}$$

where C is a constant (usually equal to 1.22), Maximum Transfer Unit (MTU) is the packet size used, RTT is the round trip time estimate, and p is the loss probability estimate experienced by that flow.

Floyd et al. derived Equation 5.1 in [60] as follows. First, a TCP connection sending packets of B bytes (MTU) with a constant round trip time RTT is considered. The TCP sender has a congestion window of W packets, each time a packet is dropped. The window is increased by half for each packet drop and decreased by at most one per round-trip time. Therefore, the TCP sender transmits a minimum of $\frac{3W^2}{8}$ packets for each packet dropped (Equation 5.2).

$$\frac{W}{2} + (\frac{W}{2} + 1) + ... + W \approx \frac{3W^2}{8} \tag{5.2}$$

The fraction of the sender's packets dropped (p) is bounded by the reciprocal of that value

$$p \leq \frac{8}{3W^2} \tag{5.3}$$

[3] so called 'nominee' in AER/NCA or 'acker' in PGMCC

Then, the congestion window is bounded by

$$W \leq \sqrt{\frac{8}{3p}} \tag{5.4}$$

In the steady-state model, the congestion window is increased by one packet per round-trip-time. The average congestion window over 1 cycle of steady state mode is 0.75 W. The maximum throughput T in bytes/s is then

$$T \leq \frac{0.75 * W * B}{RTT} \tag{5.5}$$

from 5.4

$$T \leq \frac{0.75 * \sqrt{\frac{8}{3p}} * B}{RTT} \tag{5.6}$$

and therefore

$$T \leq \frac{1.22 * B}{RTT * \sqrt{p}} \tag{5.7}$$

This average sending rate applies for any conformant TCP and explains the relationship between T and p (the loss probability estimate).

In PGMCC, an acker switch happens from receiver j to receiver i if $T(i) < c * T(j)$, where $T(n)$ is the throughput computed by the sender for receiver n [141]. In AER/NCA, p (end-to-end packet loss probability) is estimated by each receiver using a fixed size sliding window. The p and RTT values are transmitted by each receiver in a Congestion Status Message (CSM) to its closest upstream repair server.

The IETF recommended some criteria for evaluating reliable multicast transport and applications which are scalability, congestion control, error recovery, and robustness [110]. Fairness to TCP is an important criterion required by IETF in order to accept any multicast congestion control mechanism. Table 5.2 compares PGMCC and AER/NCA in terms of approaches, congestion control mechanisms, characteristics, advantages and disadvantages.

In the next section, we present three experiments, in order 1) to show the behaviour of AER/NCA, 2) to compare the TCP-friendliness of AER/NCA and PGMCC, and 3) to gain the fairness index of competing AER/NCA and PGMCC flows using RED and Droptail queuing algorithms. Note that PGMCC simulation is not the main concern of our experiment, considering that PGMCC is just another single rate multicast protocol (non-AN-based protocol).

	PGMCC	AER/NCA
Network and approach	End-to-end multicast, window-based rate adjustment	ANs, window-based rate adjustment
Congestion control mechanism	Dynamically chooses representative - slowest receiver (acker). Sender adjusts rate according to ACKS sent by acker.	Dynamically chooses the slowest receiver (nominee). Sender adjusts Cwnd by using algorithm similar to TCP Reno
Characteristics	Intra-protocol fairness, TCP-friendliness	Intra-protocol fairness, TCP-friendliness
Advantages	Widely used in Multicast backbone (Mbone)	ANs help in deploying congestion control mechanisms.
Disadvantages	Only single rate multicast congestion control, does not consider heterogeneous nature of receivers	ANs are still far from deployment stage

Table 5.2: Comparison of PGMCC and AER/NCA

Application of ANs in AER/NCA

We use ns-2.1b5 [51] with an AER/NCA extension code provided by the PANAMA project [88]. The AER/NCA ns-2 extension from the project assumes that the code has already been distributed into the nodes (repair servers). The code is ready to process the active packets it receives. In ns-2, routers are created by generating classifiers and agents. The AN extension in ns-2 has a modified classifier which provides the entry point to the node. This is a best effort NodeOS layer which is able to derive an EE from an AN agent.

The ns-2 simulator extension of AER/NCA enables us to perform multicasting by using AN capsules to get information across end systems and active nodes. The multicast routing used in AER/NCA is sparse mode[4].

AER/NCA uses several new active packets for signalling mechanisms. The packet size is 76 bytes, i.e. for NACK, CSM, SPM and CCM packets. In AER/NCA, the active services are embedded into the AN objects (Agent/AN/Active, Agent/AN/Rcvr and Agent/AN/CBR).

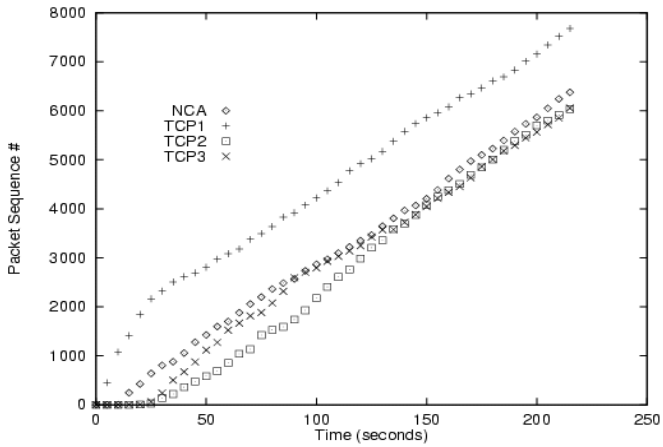
Result generation is obtained by simulation experiments by varying the parameters. New classes are derived from their AN Agent and then installed at each active node. In this case, we assume that the code belonging to the cache protocol has

[4]This is a routing algorithm used in AER/NCA ns-2 extension. The multicast routing protocols used can be dense mode or sparse mode determined by available bandwidth and distribution of end user. Sparse mode algorithm is used if the bandwidth is limited and end users are thinly distributed.

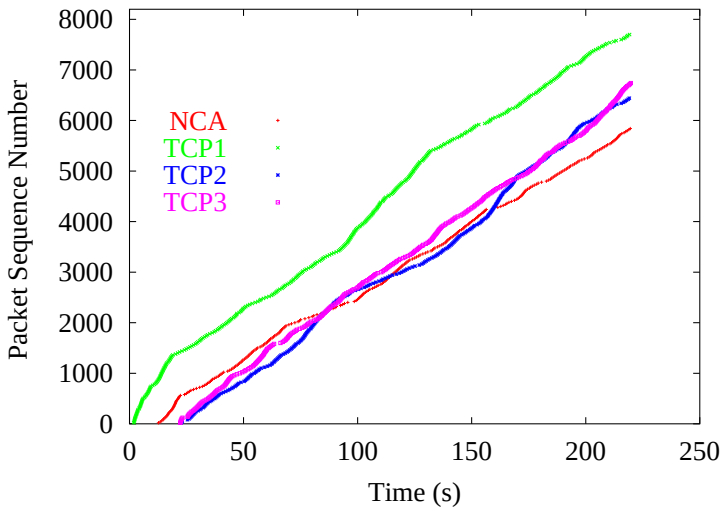
already been distributed and is available at nodes. The classifier.cc file in ns-2 had been modified to allow the node entry point to make forwarding decisions to the active agent or the next hop. The AER/NCA traffic is embedded as CBR flows.

Validation

In order to ensure the validation of the AER/NCA simulation in this research, we chose the following results from a specific case in the literature, the work of Kasera et al. in [88]. Figure 5.1(a) shows the fairness of the AER/NCA session (starting at



a) AER/NCA simulation results obtained from [88] (This figure has been scanned from the work of Kasera et al. in [88]). This graph shows the evolution of packet sequence number versus time during their simulation



b) Our AER/NCA simulation results for the same scenario

Figure 5.1: Validation of AER/NCA ns-2 package

10 seconds of simulation time) when competing with 3 TCP flows (starting at 1.15, 21.75 and 21.43 seconds respectively). The simulation is based on the same topology as the one in Figure 5.3[5](Experiment S1). The AER/NCA and TCP sessions are sharing a bottleneck link in which the bandwidth is 1 Mbps and the propagation delay is 60 ms. The routers have a buffer size of 30 packets. All of the sender and receivers links have 3 Mbps bandwidth and 10 ms delay. We performed similar simulation using ns-2 and the results are depicted in Figure 5.1(b)). Table 5.3 shows the packets sequence number of the NCA flows obtained from [88] and our simulation results. The reference data is obtained by observation of Figure 5.1(a), because we had no access to the raw data. We compare both sets of results using Kolmogorov-Smirnov test (KS-test) (explained in Chapter 4) to determine if these datasets differ significantly.

The KS-test shows the following results. For data set 1 and 2, KS-test finds the data consistent with a normal distribution (D=0.1 (the maximum difference between the cumulative distributions) with corresponding P=1). Since the observed value of D is less than the critical value of 0.410 for 0.05 level of significance[6], there is no real evidence against the null hypothesis that the two populations are the same. In conclusion, this test proved that the results of our simulation and the results in [88] are almost the same.

Time (second)	**NCA Results obtained from [88] (packets) (data set 1)**	**NCA simulation results (packets) (data set 2)**
5	0	0
25	2200	1500
50	2800	2200
75	3600	3000
100	4100	3900
125	5000	5000
150	6000	6000
175	6400	6300
200	7100	7100
220	7800	7800

Table 5.3: ACC TCP Validation: Comparison of results obtained from [88] with our simulation results

[5]Note that Figure 5.3 shows 2 TCP sessions only competing with 1 AER/NCA session.

[6]The critical value is obtained from the Kolmogorov-Smirnov test table (http://nedwww.ipac.caltech.edu/level5/Wall2/Wal4_2.html)

5.3 Experiment S1: Behaviour of AER/NCA

5.3.1 Experiment Construction and Description

The aim of this first experiment is to assess AER/NCA capabilities and performance. We evaluate the use of ANs in AER/NCA through simulation experiments, by considering that this protocol uses equations to calculate the average throughput bandwidth which is affected by the round trip time and the packet loss probability.

This work serves two purposes: 1) to evaluate the performance of the protocol in terms of its throughput, loss rate and bandwidth usage ratio, and 2) to show by simulation the TCP-friendliness behaviour of AER/NCA. In the simulation experiments, we study how the rate adjustment process performed by AER/NCA affects the throughput of flows. The ratio of TCP to AN packets received by the intermediate node shows how the bottleneck link bandwidth has been shared by this AN-based multicast congestion control protocol and its competing unicast TCP flows. The TCP-friendliness, throughput, bandwidth share, and packets dropped of the topology that employed the AER/NCA protocol are studied. In addition, the scalability, i.e. how large the receiver group can be before the performance of the protocol starts to degrade, has also been considered.

Figure 5.2 presents a flow-chart of the simulation procedure. We write the Tcl file on to the ns-2 extension of AER/NCA to define the desired topology, construct and repeat simulations with different RNG seeds. we set the AN packets such as data, nca, node, repair, spm, get_rtt, csm packets. The multicast session settings such as start_rtt, send_spm, send_init, disable, and initialise are also set. The AER/NCA packets are initialised by creating an active CBR source with rate adjustment by NCA (using Agent/AN/CBR). Subsequently, 2 TCP sessions are created to share the bottleneck link with AER/NCA flows.

Next, we run the simulation, capture the experiment data in a trace file and then parse the data file using awk scripts, which in turn will be used to calculate performance results such as throughput, jitter, packet loss ratio, bandwidth usage and inter-protocol fairness index.

Subsequently, we repeat the simulation by scaling the number of receivers and apply different loss probability of 1%, 5%, and 25%. The experiments are also repeated with enabling and disabling the AN Repair Servers (RS). Note that we still use AN packets to enable AER/NCA multicast, when the repair server is off.

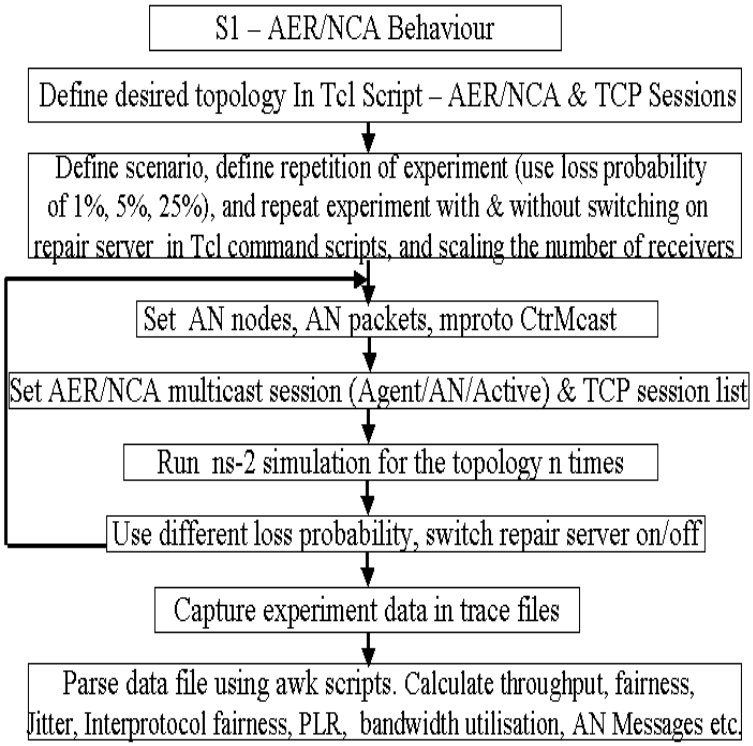


Figure 5.2: Experiment S1 - Flowchart of Simulation Procedure

5.3.2 Topologies

In this first experiment, we use an AER/NCA multicast source that diverges at node 2 towards two multicast receivers, as shown in Figure 5.3. In addition, a number of TCP New Reno sources are connected to node 1, and linked to the receivers attached to node 2. All TCP flows used throughout these AER/NCA experiments are TCP New Reno. The topology in Figure 5.3 shows how the AER/NCA and TCP flows share a bottleneck link.

5.3.3 Key Parameters and Rationale

The following are the key parameters used in the simulation, and the rationale for their choice. These simulation parameters can be seen in Table 5.4. In order to create a congestion situation, the bottleneck bandwidth used between nodes 1 and 2 is 1 Mbps, whereas the propagation delay is 60 ms. We set a small bandwidth in the bottleneck link to ensure the occurrence of congestion. The reason we chose 60

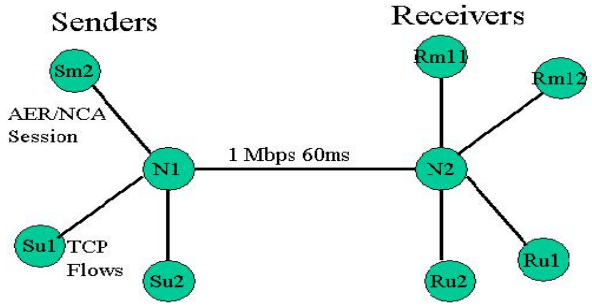


Figure 5.3: Experiment S1 - Topology of AER/NCA and TCP flows sharing bottleneck link

Simulations Parameter	Value
All links	1 Mbps, 60 ms delay
Maximum buffer size	30 packets
AN packet size	76 bytes
AER/NCA and TCP packets	1000 bytes
Simulation duration	120 seconds

Table 5.4: Experiment S1 - Simulation parameters for topology in Figure 5.3

ms propagation delay is because it is below the limit of the 80-100 ms latency, and it is close to the high end of what has been perceived as the common RTT in the Internet [47]. This propagation delay setting is also applied to other experiments in this work. The buffer size in the repair server is 30 packets. The TCP New Reno and AER/NCA data packet size is 1000 bytes. The capsule size for the ANs packets is 76 bytes which is the initial value assigned by the AER/NCA's authors. Each simulation lasts for 120 seconds.

The simulation's initiation of actions can be seen in Table 5.5. The acknowledgement packet, AN capsules for RTT, CSM, CCM, enabling and disabling of repair server, etc. can be monitored by evaluating the change in the packets' colour through the network simulation animator (nam).

AER/NCA sessions start sending Source Path Message (SPM) packets at 0.6 seconds after the simulation started to provide the multicast path router address information. At 0.8 seconds, the NCA starts sending constant bit rate data. The TCP1 session starts sending packets at 1.15 seconds, whereas TCP2 sends packets at 21.75 seconds. These simulation events and parameter settings are inspired by a scenario in [88].

We also varied the loss rate setting in the AER/NCA protocol into 1%, 5% and 25%, to evaluate their effect on the bandwidth usage of AER/NCA with and without

Time (second)	Action
0	Multicast session initiated
0.2	Node(3) and node(4) join the multicast group (receivers)
0.4	Node (2) and Node(3) becomes active (repair server)
0.6	Send Source Path Message capsule (cbrsndr)
0.8	Start sending CBR data from NCA source - node(0)
1.5	Link 0-1 and 1-2 become lossy (set loss rate to 1%)
1.15	Start FTP sessions for TCP1 connection
21.75	Start FTP sessions for TCP2 connection
120	Finish simulation

Table 5.5: Experiment S1 - Sequence of simulation events and actions

enabling the active services in the repair server (node). Furthermore, we also scaled the total number of AER/NCA and TCP receivers into 4, 8, 16 and 32 (split equally between AER/NCA and TCP receivers).

5.3.4 Performance Metrics

The performance metrics reported in these experiments consist of throughput, jitter, TCP-friendliness, Jain fairness index and bandwidth usage (ratio of AN data packets and AN messages over the total transmitted packets).

5.3.5 Results and Evaluation

This section presents the experimental results and highlights the TCP-friendliness in the AN-based single rate multicast protocol. The graphs were generated by tracing data packets entering and leaving the routers in the network. We discuss the results of the simulation experiments to provide us with insight into the performance of AER/NCA under various conditions.

First, we observe the situation in which the AER/NCA session with two receivers competes with two TCP flows. Experiment results show that the multicast session receives an almost equal share of bandwidth as any of the TCP sessions. The Jain inter-protocol fairness index gained from the calculation of the throughputs for the flows in the bottleneck link is 0.99, which means that the TCP and AER/NCA sessions have a fair share of bandwidth.

Table 5.6 shows the TCP and AN ratio of packets received. The number of TCP packets received at Node 2 is more than the AER/NCA packets, even though the

Performance Results	Value
Average TCP throughput (bps)	54122 ± 3351
Average AN throughput (bps)	19097 ± 1514
Total bytes received in Node 2	8786422 ± 406572
TCP ratio	0.74 ± 0.02
AN ratio	0.26 ± 0.02
TCP Packet loss ratio (%)	0.73 ± 0.01

Table 5.6: Experiment S1 - Results of experiment with 4 receivers (1 multicast session with 2 receivers and 2 TCP flows) with non-lossy link

AER/NCA session started earlier than the TCP flows. The simulations performed also verified that AER/NCA does not clutter the network when it shares the resource with other protocols. In every source to destination path in the multicast tree, the AER/NCA protocol shows its friendliness behaviour.

Figure 5.4 shows AER/NCA packet jitter which happens at the repair server

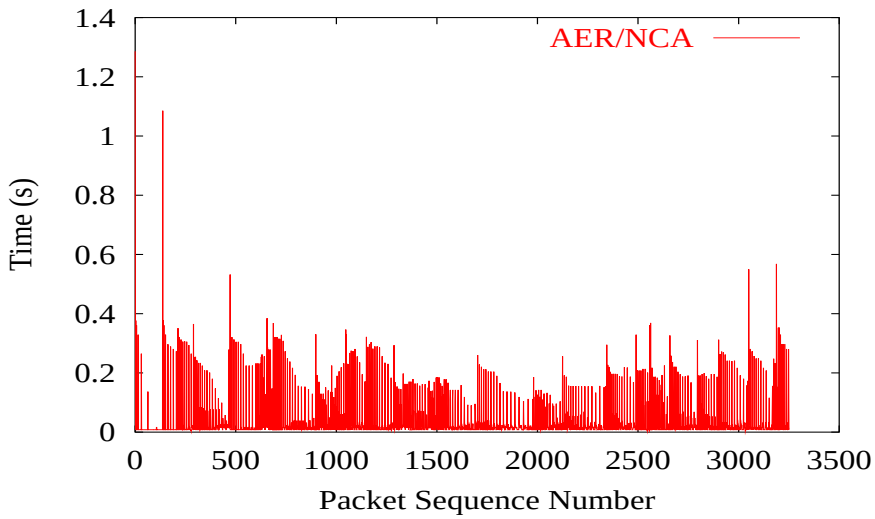


Figure 5.4: Experiment S1 - AER/NCA jitter at node 2

(node 2) during the simulation. The loss of packets causes a delay in data delivery to receivers. The delay increases when the queue built up, and this is caused by rate reduction at the repair server when congestion occurs. In AER/NCA, the queuing delay for a packet can be reduced by the caching and buffering mechanism at repair server. The jitter is calculated by measuring the time since the last packet received divided by the difference in sequence number (for loss packets) for each sequence number.

Next, we will evaluate the bandwidth share used by AN data and AN message packets in relation to the increasing number of receivers. Figure 5.5 shows the

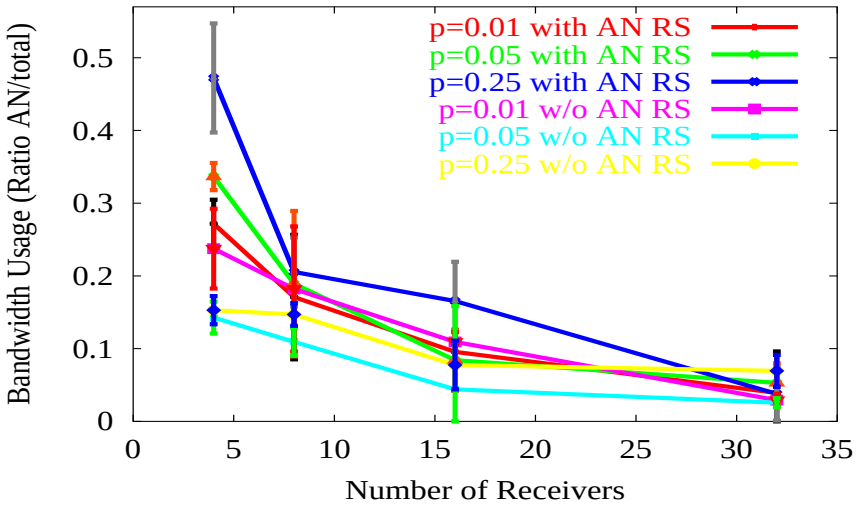


Figure 5.5: Experiment S1 - Comparison of bandwidth usage with and without AN Repair Server (scaling the number of receivers). Bandwidth usage is defined as the total AN data packets and AN messages over the total transmitted packets. The number of TCP flows is 2, 4, 8 and 16 respectively, whereas the AER/NCA flow remains one (1 session with 2, 4, 8, and 16 receivers respectively)

bandwidth usage at the bottleneck link when the total number of AER/NCA and TCP receivers is scaled to 4, 8, 16 and 32 receivers, and the repair servers are active/non-active to perform repairs to the packet loss. The number of receivers is split equally between AER/NCA and TCP, e.g. for 8 receivers (4 AER/NCA and 4 TCP receivers). Note that because ANs are used to perform multicasting, the AN packets are always present. Hence, the number of AN packets is always non-zero. In the case of a repair server being disabled, the congestion control mechanism using AN packets might have to compensate with more AN message packets (76 bytes each) sent to handle the communication between the sender and receiver that experienced packet loss (global recovery). This will be evaluated further in the packet loss and message passing discussion in Section 5.6.

In the simulations with AN repair server on, the higher the loss probability of an AER/NCA session, the more bandwidth is used by the AN data and AN message packets than the TCP packets (Table 5.7). This is due to the high number of active packets required to perform active error recovery. In addition, more AN packets are successfully transmitted through the links. The fraction of bandwidth used by the

Number of Receivers	Without AN RS (1% loss rate)	Without AN RS (5% loss rate)	Without AN RS (25% loss rate)	With AN RS (1% loss rate)	With AN RS (5% loss rate)	With AN RS (25% loss rate)
4	0.23 ± 0.05	0.27 ± 0.03	0.32 ± 0.01	0.27 ± 0.03	0.33 ± 0.02	0.47 ± 0.07
8	0.18 ± 0.01	0.20 ± 0.01	0.22 ± 0.01	0.19 ± 0.02	0.20 ± 0.01	0.23 ± 0.04
16	0.13 ± 0.01	0.13 ± 0.01	0.17 ± 0.03	0.11 ± 0.11	0.14 ± 0.04	0.17 ± 0.05
32	0.02 ± 0.01	0.04 ± 0.01	0.06 ± 0.03	0.03 ± 0.05	0.05 ± 0.01	0.07 ± 0.03

Table 5.7: Experiment S1 - AN Bandwidth usage (the ratio of AN data packets and messages over the total transmitted packets)

AN data packets and AN messages decreases, because the number of multicast flows does not increase when the number of multicast receivers is scaled. On the other hand, more TCP flows will share the available bandwidth. In general, the use of AN repair server prompts more bandwidth usage by AN data packets and messages than the condition without AN repair server, because more AN data packets can be transferred across the network.

This situation is also the same when the AER/NCA multicast flows are activated, but the repair server is turned off. When the link loss rate is 25% and 4 receivers are used, the ratio of AN data packets and AN messages over the total transmitted packets is the highest (47%). The total number of AN data packets and AN messages crossing the bottleneck link decreases for the same reason. In addition, some AER/NCA packet losses are expected for the absence of local recovery. The total number of AN bytes passing through the links decreases, due to the fact that the AN data packet size is much bigger than the AN messages'. Note that AN packets are still used for the purpose of AN-based multicasting, although the AN repair server function is not available at the intermediate active node.

Further analysis of Table 5.7 shows that when we consider the link loss rate probability (varied to 1%, 5%, and 25%), it can be observed that the reductions in AN data and messages bandwidth usage (when the repair server is on/of) are between 1.6% to 31.4% when the repair server is disabled (calculated using Equation 5.8). The reduction for experiments with 4 receivers is the highest. This is due to the fact that the ideal AN bandwidth share is 33% (1/3 of the available bandwidth for 1 AER/NCA and 2 TCP flows), but AN repair server mechanism attempted to use more bandwidth (an average of 47%). The reduction becomes significant because each flow gets a larger share of bandwidth than in the situation in which

the number of receivers is high (32 for example).

$$Reduction = \frac{bwusage_{(withANRS)} - bwusage_{(withoutANRS)}}{bwusage_{(withANRS)}} * 100\% \qquad (5.8)$$

In summary, this experiment shows the effect of scaling the number of receivers, different link loss rates, and the use of an AN repair server to the AN bandwidth usage. The higher the loss probability, the more AN data and messages passing through the bottleneck links. This is due to the AN repair server attempts to repair the packet loss. In addition, the sender is also actively sending the AN data packets and messages.

The more receivers connected to the active node, the less bandwidth is shared by the AN messages. This is because AER/NCA is based on the TCP friendly equation (Equation 5.1). Ideally, in a non-lossy link AER/NCA will have 1/17[7] (5.8%) bandwidth share when the number of receivers is 32. The simulation results show that the bandwidth share is 7% (when the AN RS is on) and 6% (without AN RS).

A high link loss rate combined with the use of AN repair server prompts AER/NCA to send more AN packets than expected. For example, for experiments with 1 AER/NCA and 2 TCP flows (4 receivers), ideally AN flow should have 1/3 (33%) of bandwidth share. However, the results of experiment show that it has an average of 47% (with AN RS) and 32% (without AN RS) of bandwidth share.

In sharing the bandwidth with the AER/NCA flow, there are some factors that affect TCP. First, TCP has a back-off mechanism to prevent the occurrence of packet loss. In addition to this, the non-linear nature of TCP also contributes to variation in the bandwidth distribution between AER/NCA and TCP flows.

Throughout this experiment, it can be observed that the average AN bandwidth usage is smaller than the bandwidth used by TCP, which shows that AER/NCA is a TCP-friendly protocol.

[7]For 32 receivers, we have 1 AER/NCA and 16 TCP flows in the bottleneck link. Hence, the ideal share of AER/NCA is 1/17 of the available bandwidth.

5.4 Experiment S2: Comparison of PGMCC and AER/NCA

5.4.1 Experiment Construction and Description

The aim of this experiment is to assess the AER/NCA's inter-protocol fairness. We compare the AER/NCA with the non-AN-based protocol - PGMCC [135], in terms of their behaviour toward sharing the same bottleneck link with TCP. One of the goals of this experiment is to reveal advantages and drawbacks of the use of the ANs paradigm in AER/NCA when compared with the non-AN-based single rate congestion control protocols. We use the available results reported by Seada et al. in [142] and its extended version in [141] without resimulating the PGMCC experiments, to avoid possible errors caused by wrong algorithm implementation and interpretation.

Figure 5.6 presents a flow-chart of the simulation procedure. We write the Tcl

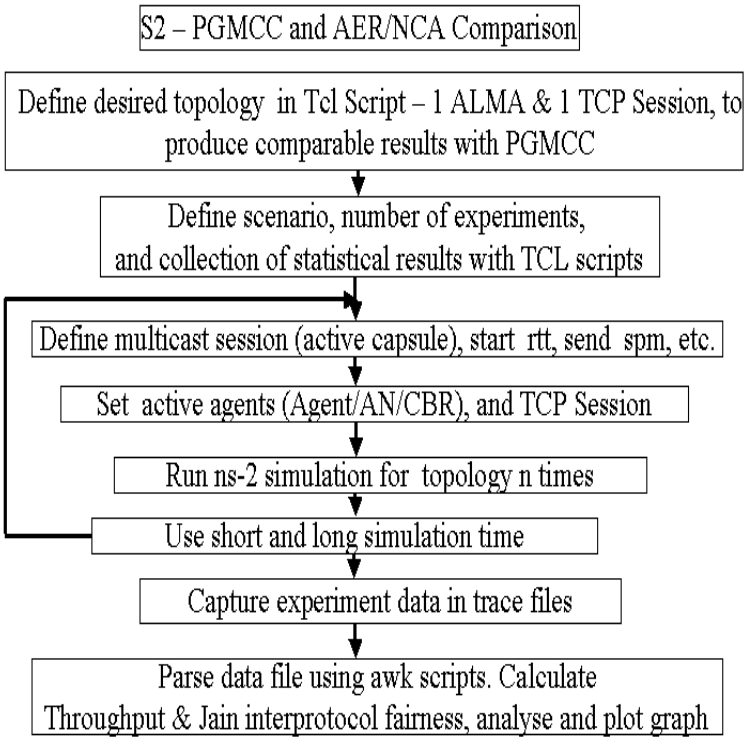


Figure 5.6: Experiment S1- Flowchart of simulation procedure

file on to the ns-2 extension of AER/NCA to define the desired topology. We define the multicast sessions and corresponding AER/NCA capsules such as start_rtt,

send_spm, etc. Next, the active agents and the competing TCP flows are also set. We run the simulation 10 times with different RNG seeds. The experiment data is captured in a trace file. Then, the data file is parsed into meaningful data using awk scripts. This result is then used to calculate Jain fairness index and to plot the graph of sequence number versus time.

5.4.2 Topologies

In this experiment, we use a topology for the purpose of evaluating the inter-protocol fairness (a mix of multicast and TCP flows). Figure 5.7 shows the topology for the

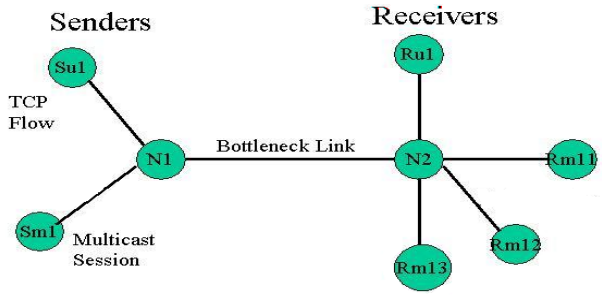


Figure 5.7: Experiment S2 - Topology for PGMCC and AER/NCA inter-protocol fairness

inter-protocol fairness experiment which also has been used in [141] for PGMCC evaluation. This topology consists of 1 multicast session with 3 receivers sharing the link with a TCP flow.

5.4.3 Key Parameters and Rationale

For the purpose of comparing the schemes, some representative test cases as described in [135, 141] are simulated in the experiment in this section using the ns-2 network simulator. First, for the inter-protocol fairness experiment in Figure 5.7 using a non-lossy link, we apply the same parameters as in [141], for ease of comparison with PGMCC.

We run the AER/NCA multicast session concurrently with a TCP flow. The key parameters of this simulation can be seen in Table 5.8. The bottleneck bandwidth used between nodes 1 and 2 is 500 Kbps, whereas the propagation delay is 50 ms in order to create a congestion situation. The buffer size in the repair server is 30 packets. The packet size for the TCP New Reno and AER/NCA transmission

Simulation Parameters	Value
Sources	1 Multicast session with 3 receivers, 1 TCP flow
Sources - Router 1	10 Mbps, 1 ms delay
Router 1 - Router 2 link (bottleneck link)	500 Kbps, 50 ms delay
Router 2 - Receivers	10 Mbps, 1 ms delay
Router 1 & Router 2 maximum buffer size (Droptail queue management algorithm)	30 packets
Simulation time	300 seconds & 3000 seconds
Packet Size	1400 bytes

Table 5.8: Experiment S2 - Simulation parameters for AER/NCA experiment (topology in Figure 5.7 over short and longer periods of simulation)

is 1400 bytes. Each simulation lasts for 300 seconds of simulation time. In this experiment, we started the TCP flow at the same time with AER/NCA flow (0.1 seconds).

The second part of the experiment deals with the same topology (in Figure 5.7), but the simulation execution lasts over a longer period. We use the parameters as described before, and run the simulation for 3000 seconds. The TCP flow is also started together at the same time as the multicast flow.

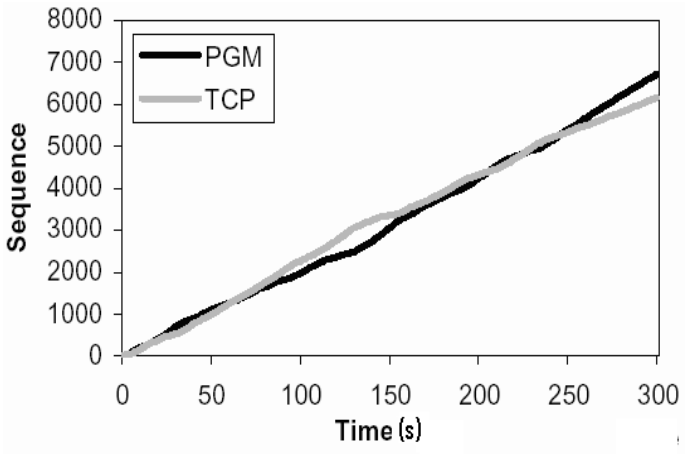
5.4.4 Performance Metrics

The performance metrics reported in these experiments consist of inter-protocol fairness shown by means of plotting the packet sequence number increment in relation to time and calculating the Jain fairness index.

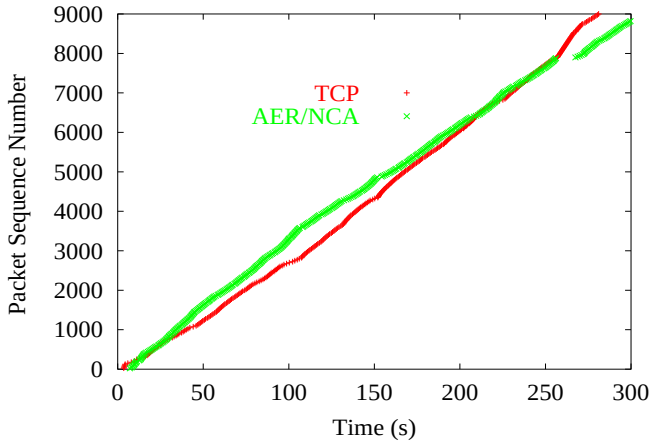
5.4.5 Results and Evaluation

In this subsection, we address the situation in which these single rate multicast protocols compete with another protocol, particularly in the presence of competing TCP flows. The objective of this experiment is to compare the inter-protocol fairness (TCP-friendliness) of PGMCC and AER/NCA. We evaluate the inter-protocol fairness by running an AER/NCA session and TCP flows through the same bottleneck link.

Figure 5.8(a) shows the simulation results for a PGMCC multicast session competing with TCP flow, provided in [141]. In Figure 5.8(b) we presents the result of the same simulation using AER/NCA. The time versus sequence number graphs



a) PGMCC (fairness index = 0.94) (This figure has been scanned from the work of Seada et al. in [141]). In this graph PGMCC shows its TCP-friendliness



b) AER/NCA (fairness index = 0.99). AER/NCA is a TCP-friendly protocol

Figure 5.8: Experiment S2 - Inter-protocol fairness of PGMCC and AER/NCA over short period (1 multicast session and 1 TCP session)

are presented to show TCP-friendliness of the multicast protocol. It has the same effect as showing the throughput of each flow. This is a common approach to assess multicast protocols, as can be found in [88, 135].

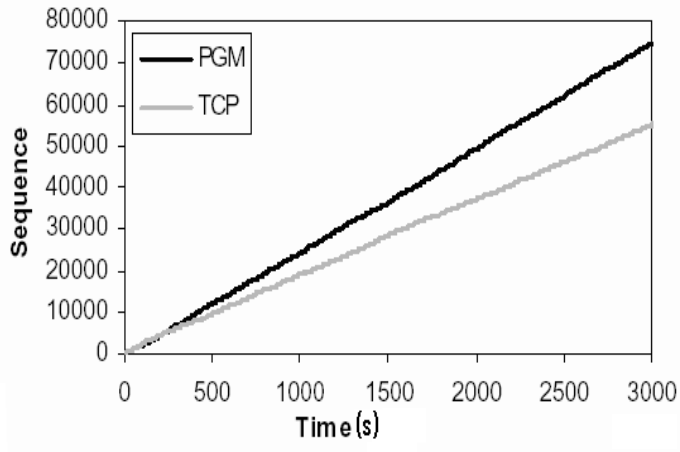
Figure 5.8(a) shows the PGMCC behaviour when one multicast session with 3 receivers competes to use the bottleneck link with one TCP flow. On this non-lossy link, the TCP flow and multicast flows start and progress almost at the same rate, although during 75-150 seconds TCP has a little more share of bandwidth. The existence of many receivers does not affect the data transfer, because all receivers experience the same losses and measure the same loss rate. The same experiment with AER/NCA in Figure 5.8(b) shows a similar increasing trend. The AER/NCA session generally has a fair share of bandwidth, as the AER/NCA and TCP plots are both progressing at almost the same time. During the 50 to 225 seconds of simulation time, the TCP session has a slightly larger share of bandwidth. In general, the AER/NCA flow can share the bandwidth almost evenly with the TCP flow. The inter-protocol fairness between the multicast session and TCP flows sharing the bottleneck link is clearly demonstrated by the high value of Jain fairness index (0.99). This fairness index has been calculated using Equation 3.3 previously described in Chapter 3.

The simulation results in Figure 5.9 present the outcome of the second part of this inter-protocol fairness experiment. This figure shows the AER/NCA behaviour over a longer period of time. In this condition, TCP is shown to consume more bandwidth than AER/NCA. However the differences are negligible, as shown by the fairness index which is 0.98. Over a longer period, TCP gets larger share of bandwidth compared with AER/NCA. Overall, TCP obtains a larger share of bandwidth than AER/NCA (Figure 5.8(b)). Table 5.9 summarises the calculated Jain fairness indices gained from this experiments.

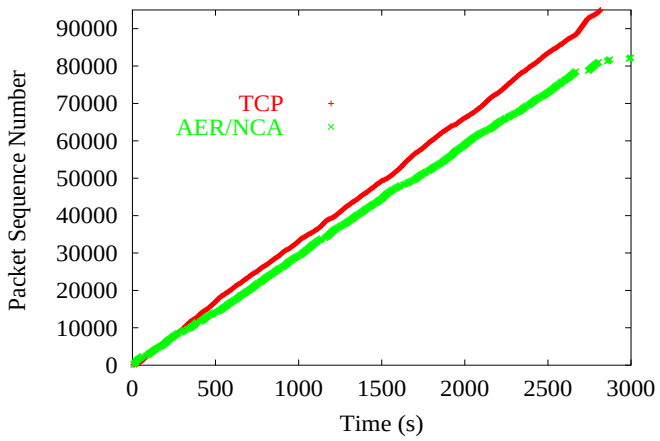
Experiment/Jain Fairness Index	PGMCC	AER/NCA
Short simulation	0.94	0.99
Long simulation	0.74	0.98

Table 5.9: Experiment S2 - PGMCC and AER/NCA Jain Fairness Index

The PGMCC results in Figure 5.8(a) show that over a longer period of experiment, PGMCC gets a larger share of bandwidth and becomes TCP unfriendly. This is shown by the low Jain fairness index value which is 0.74. PGMCC is not fair to TCP for the reason that PGMCC times out after a stall when the ACKs stop coming



a) PGMCC (fairness index = 0.74) (This figure has been scanned from the work of Seada et al. in [141, 142]). This result shows that in long simulation PGMCC is not fair to TCP



b) AER/NCA (fairness index = 0.98). AER/NCA shows its friendliness towards TCP.

Figure 5.9: Experiment S2 - Inter-protocol fairness of PGMCC and AER/NCA over a longer period (1 multicast session and 1 TCP session)

in, and a long timeout occurs [141]. This is different from the AER/NCA results in which the multicast protocol is submissive to TCP. In this situation, AER/NCA acts differently from PGMCC, due to the differences in handling timeouts and responding to ACKs and losses.

Experiments with AER/NCA for the same topology and network parameters over a longer period show a different characteristic between AER/NCA and PGMCC in terms of inter-protocol fairness. TCP shares more bandwidth than AER/NCA, whereas PGMCC dominates the use of bandwidth. The congestion control algorithm in AER/NCA can regulate the rate so that it is not more aggressive than TCP flows on sharing the same link. AER/NCA uses a NACK-based approach for congestion control, and the receiver which has the lowest throughput in a multicast group is selected as a nominee. This mechanism ensures that fairness in sharing the link with other competing flows can be achieved. PGMCC is suited for applications that can cope with larger variations in the sending rate. The close mimicking of TCP's window behaviour by PGMCC enables this feature.

These experiments show that both PGMCC and AER/NCA are both TCP-friendly multicast single rate congestion control protocols to some extent. However, over a longer period, due to its congestion control mechanism, PGMCC is not as fair as AER/NCA. The TCP-friendliness in these two protocols is achieved through different mechanisms. PGMCC uses the basic multicast NACK-based retransmission request, adds some options to the multicast packet format, i.e. one new packet type, and modifies the sender and receiver procedures for congestion control purposes. On the other hand, AER/NCA uses an active application to perform the selection of a nominee (the weakest receiver) to adjust the sender rate.

The visualisation of packet sequence number in relation to time has been used widely by the networking research community to show TCP-friendliness, such as in [88, 135]. This method can show TCP-friendliness in a simple way, although it is not always very accurate. Further quantitative analysis using different methods such as inter-protocol fairness index [177, 179] and Jain fairness index [76] is necessary to evaluate the intra-protocol fairness of different protocols. This quantitative methods have been used concurrently with packet sequence number versus time analysis in all respective experiments.

5.5 Experiment S3: Fairness Index

5.5.1 Experiment Construction and Description

The aim of Experiment S3 is to evaluate the fairness index of AER/NCA for different queue management schemes at the intermediate routers. The purpose is to assess the inter-protocol fairness of AER/NCA with a competing TCP flow. We set up the simulations using a network topology with different round trip times which are adjusted by using different delays in the side links. We vary the propagation delay of the receiver side links to evaluate the throughput and fairness index of the link. For calculation, we use Jain's fairness index defined in [76]. Recall that we have introduced the equation in Chapter 3 (Equation 3.3). The fairness index lies between $1/n$ and 1. Recall that the fairness index of 1 means that the bandwidth is shared fairly between the flows.

We run the simulation using a variable propagation delay in the receiver side's link, and RED and Droptail as queue management mechanisms in the intermediate routers. We use the average throughput results to calculate the fairness index for experiments with RED and Droptail queue management algorithms at the routers.

Figure 5.10 presents a flow-chart of the simulation procedure. We write the Tcl file on to the ns-2 extension of AER/NCA to define the topology. The AN node, AN packets multicast session and TCP list are set. We repeat simulations with different random number generator seeds. The round trip time of the simulation is varied by changing the delay at the exterior sink side link. We run the simulation, capture the experiment data in a trace file, and then parse the data file using awk scripts, which are used to calculate the throughput and Jain fairness index.

5.5.2 Topologies

The topology in Figure 5.11 is used in this experiment to examine the fairness of an AER/NCA flow towards TCP. It consists of 1 multicast session and 1 TCP flow. The multicast session consists of one sender and one receiver which is always the representative (nominee). We do not consider the changes of the representative, due to the fact that the aim of the experiment is to evaluate the fairness index in the congested link (shared between 1 multicast flow and 1 TCP flow). In scaling the number of multicast receivers, the number of flows in the link remains 2 and the multicast flow diverges in Router 2. Therefore it is enough to take account of only one receiver in a multicast session. The round trip time of the simulation is varied

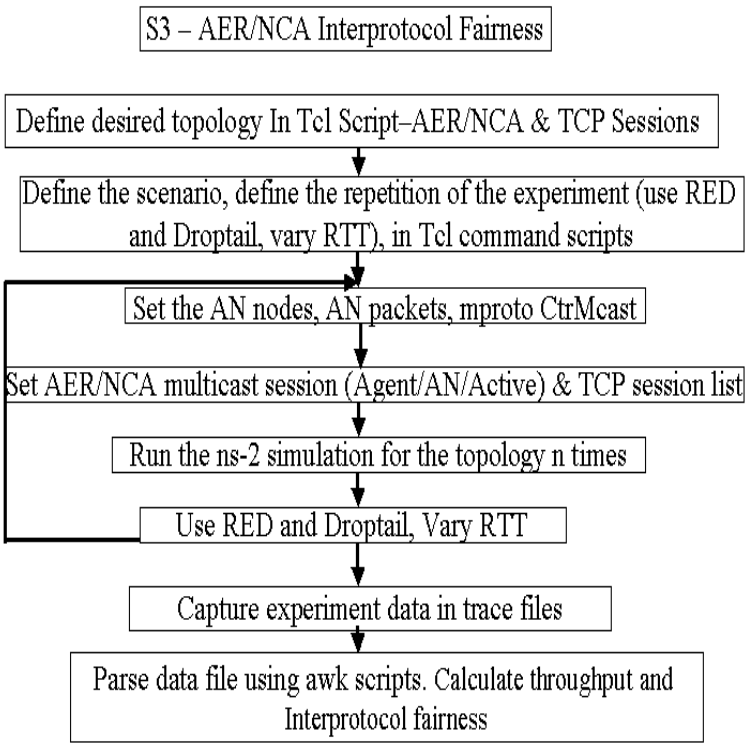


Figure 5.10: Experiment S3- Flowchart of simulation procedure

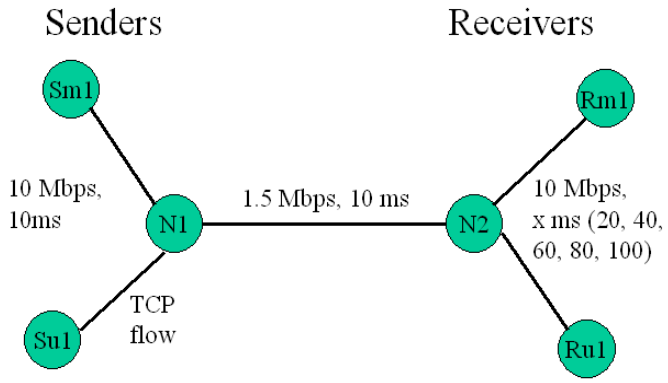


Figure 5.11: Experiment S3 - Topology for AER/NCA inter-protocol fairness

by changing the receiver's link delay (uncongested link).

5.5.3 Key Parameters and Rationale

For the inter-protocol fairness experiment in Figure 5.11 we used the simulation parameters listed in Table 5.10. These parameters have been used in [184] to discuss

Simulation Parameters	Value
Sources	1 Multicast session and 1 TCP flow
Sources - Router 1	10 Mbps, 10 ms delay
Router 1 - Router 2 link (bottleneck link)	1.5 Mbps, 10 ms delay
Router 2 - Receivers	10 Mbps, x ms delay (x = 20, 40, 60, 80, 100)
Router 1 & Router 2 maximum buffer size	30 packets
Simulation time	20 seconds
Packet Size	1000 bytes

Table 5.10: Experiment S3 - Simulation parameters for topology in Figure 5.11

multicast protocol fairness. The bottleneck bandwidth used between nodes 1 and 2 is 1.5 Mbps, whereas the propagation delay is 10 ms. Each multicast session has one sender and one receiver. We use RED and Droptail queue management algorithms in the routers. Each simulation lasts for 20 seconds.

5.5.4 Performance Metrics

The performance metrics of the experiments reported in this work consist of throughput and fairness index. The fairness index is calculated with Jain's Equation (Equation 3.3).

5.5.5 Results and Evaluation

Table 5.11 shows the fairness index of the AER/NCA with TCP when we use RED or

Queue Management Algorithm\Receiver side link's delay(ms)	20	40	60	80	100
RED	0.95	0.98	0.97	0.94	0.92
Droptail	0.87	0.89	0.84	0.83	0.84

Table 5.11: Experiment S3 - Fairness index with Red and Droptail queue management algorithms

Droptail queue management schemes on the routers. We can see that this window-based protocol can achieve TCP fairness with a RED and Droptail algorithm at the router, in which the fairness indices are between 0.83 and 0.97 (close to 1 - all flows almost have the same throughput). The performance with a Droptail algorithm is not as good as the one with RED. This is due to the fact that, when using the Droptail algorithm at the router, AER/NCA and TCP experience different loss rates. The Droptail algorithm is based on a first-in first-out mechanism. It would drop the last packet in the queue when the queue becomes full. Packet dropping is more likely to happen compared with RED queuing algorithm. RED prevents packet drop before the queue becomes full. The experiment results show that AER/NCA will lead to a better fairness performance when it is complemented with RED queuing algorithm. The fairness index results show that AER/NCA is fair towards TCP.

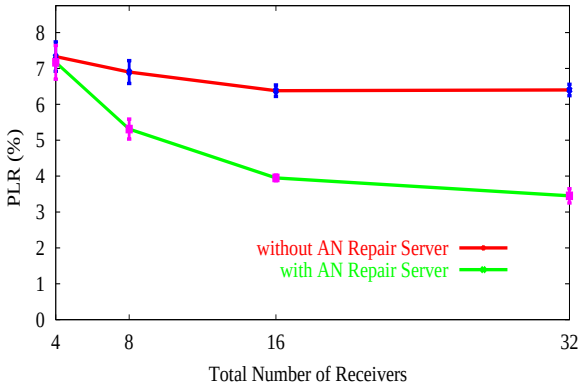
5.6 Limitation of the Scheme

Previous experiments with AER/NCA show its characteristics in terms of throughput, fairness, scalability, and the effect of different link loss rate. In this section, we discuss the trade-off between packet loss ratio and the message passing overhead. This will lead us to focus on one of the limitations of the AER/NCA, which is the requirement to perform caching and buffering at the intermediate nodes.

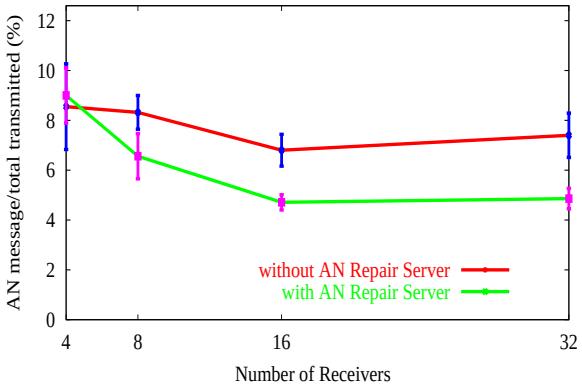
The following analysis on packet loss ratio and the ratio of AN messages over the total number of transmitted packets is based on the same scenarios and network topology as Experiment S1. The Packet Loss Ratio (PLR) is defined as the number of packets dropped over the total number of transmitted packets. The AN message ratio is calculated based on the total number of AN messages in bits over the total number of transmitted bits. The simulation is executed 10 times with different seeds of the random number generator. The PLR and AN messages ratio gained are the average value.

Figure 5.12 shows the relationship between the packet loss and the use of AN message capsules in AER/NCA. Figure 5.12(a) shows the results of the packet loss ratio of the experiments with 4, 8, 16, and 32 receivers. We run the simulation with and without enabling the repair server. The average PLR without repair server is stable around 6.5%. However, when the repair server is enabled, it can help to repair the loss. The PLR is reduced to 3.45% when the number of receivers is 32. The results of the experiment show that not having the AN repair servers does not correspondingly reduce the number of AN messages. Table 5.12 presents the

simulation results in terms of PLR and the ratio of AN messages over the total number of transmitted packets.



a) PLR



b) Percentage of AN messages over total transmitted packets

Figure 5.12: Experiment S1 - Trade-off between Packet Loss Ratio and number of AN message packets percentage over total transmitted packets

On the other hand, the message-passing overhead (all of the AN messages with a size of 76 bytes each over the total number of transmitted packets) is higher when the AN repair server is disabled (as can be seen in Figure 5.12(b)). This is due to the fact that the absence of a repair server means that the sender has to resend lost packets, which involves a lot of communication overhead across the link. The error recovery must be initiated from the sender's side, instead of the local repair server. Both PLR and AN messages graphs show a clear trend, in which PLR and ratio of AN messages over the total number of transmitted packets are significantly higher without AN repair server than with the AN repair server. The variation is significant for the AN messages on 8 receivers with AN repair server, which can be noted from the large confidence interval. However, it is still below the value of the one with AN repair server. The experiment results vary due to the randomness in

the discrete event simulator.

Number of Receivers	PLR without AN RS (%)	PLR with AN RS (%)	Ratio of AN messages without AN RS (%)	Ratio of AN messages with AN RS (%)
4	7.3 ± 0.4	7.1 ± 0.4	8.5 ± 1.7	9 ± 2.1
8	6.9 ± 0.3	5.3 ± 0.3	10.3 ± 1.9	6.5 ± 0.9
16	6.3 ± 0.1	3.95 ± 0.1	6.8 ± 0.6	4.7 ± 0.3
32	6.4 ± 0.1	3.4 ± 0.2	7.4 ± 0.8	4.8 ± 0.4

Table 5.12: PLR and ratio of AN messages over total transmitted packets

Considering that the AN packet size in AER/NCA is only small (76 bytes) compared to the data packets (1000 bytes), the simulation results show that having an AN functionality such as a repair server will not necessarily create a larger communication overhead. This is when we take into account the overall data transmission and the capacity to avoid a large number of retransmitted packets. Local recovery proves to be better than global recovery. However, the trade-off of this is that some cache/buffer space needs to be allocated to perform the task (local recovery), as the active services cache packets at the designated servers provide the ability to recover from loss quickly and efficiently.

Our experiments with AER/NCA also show other limitations of this scheme. AER/NCA sometimes shows an instability in managing the multicast membership. This is due to the limitation of the currently available ns-2 extension. Despite having demonstrated its stability for a single multicast session, it is less stable on managing more than 1 multicast sender. On a higher level, the use of execution environment, the decision on how many active packets each flow will take, and the initiation of active nodes will be the trade-off of having a solution to the multicast congestion control protocol. The message passing overhead also exists in an Execution Environment such as ANTS. Results in [173] show that message passing in a reliable multicast implementation on ANTS adds 30% overhead.

Overall, AER/NCA has many limitations for its deployment to the Internet. First, it is limited by its single rate multicast nature, which can not cater for the heterogeneity of receivers. Second, AER/NCA depends entirely on the ANs support on the routers. ANs in general are still quite far from their deployment stage. Consequently, AER/NCA could only achieve the ABone prototyping stage. Third, the active services in AER/NCA require space for caching at the intermediate AN nodes, which must be selected and prepared carefully. Currently, it is difficult to

allocate the distribution of active servers, choose the best location to deploy them, and determine the size of the buffer due to the variation of requirements.

5.7 Discussion

The following is a brief summary of the results of the experiments on the AN-based multicast single rate congestion control protocol:

- Experiment S1 shows how AER/NCA can share the network with another protocol (TCP). The performance when the total number of AER/NCA and TCP receivers is scaled up shows that the higher the loss probability induced to the AER/NCA sessions, the higher the bandwidth used by the multicast sessions (as expected). The existence of AN repair server will decrease the number of packet drops as well as the total number of AN messages over the total transmitted packets.

- Experiment S2 demonstrates the AER/NCA's inter-protocol fairness with TCP. The comparison of results with those of PGMCC [141] shows that both of these protocols compete fairly with TCP, but over a longer simulation period, PGMCC becomes less fair than AER/NCA.

- Experiment S3 calculated Jain's fairness index for AER/NCA when either RED or Droptail queuing algorithms were used in the intermediate node, and the AER/NCA shared the link with TCP flows. The Jain fairness index of RED is higher than that of Droptail because of the packet dropping mechanism of Droptail.

In the following points, we discuss the use of ANs in these experiments and the reason behind the AER/NCA's behaviour:

- The *AN-based active capsules* have been used in AER/NCA to perform the single rate multicast mechanism which includes congestion control and local error recovery. The mechanism in which a multicast sender regulates its rate in response to congestion feedback from its receiver is the most widely used approach, and therefore has been used both in AER/NCA and PGMCC (the non-AN based protocol). The AER/NCA congestion control mechanism has used the *AN-based capsules to select the weakest receiver's rate* to be the rate of the sender. This algorithm ensures that the AER/NCA multicast session is fair to the other competing flows.

- In addition to this, AER/NCA also applies a rate adjustment algorithm to regulate its rate in response to per-packet acknowledgement (ACK) received from the nominee. The NACK information is in the form of *AN capsules,* which has proved to be useful in the event of congestion. Recall that the algorithm used the congestion window (CWND) information, and slow-start threshold variable (Thresh), in which CWND is adjusted in a similar way to TCP New Reno. On receiving an ACK, detecting loss, or time-out, the congestion window is adjusted. This promotes fair bandwidth sharing among AER/NCA and its competing flows.

- Over a longer period, AER/NCA is fairer towards TCP compared with PGMCC. This is the result of the different approach in handling time-outs and responding to Ack and losses used in these two protocols.

- Further advantages of the AER/NCA protocol were generated by its buffering and caching schemes in the active node using an *AN-based mechanism.* This resulted in significantly lower packet loss ratio for all AER/NCA session experiments conducted in the course of our research. However, the space required to perform caching is one of the AER/NCA limitations, taking into account that it is not trivial to forecast and prepare their allocation on the active nodes.

- The low packet loss in AER/NCA can be attributed to the *active network functionality* to perform local recovery. The AN capsules help to perform active error recovery closer to the source of congestion. The packet loss can be avoided by the use of the other capsules, such as congestion control message and congestion status message. Empirically, AER/NCA creates communication overhead when transmitting active packets. However, the size of this active packet has been set to be very small (76 bytes) compared with the size of data packets. For this reason, the total number of active messages is negligible compared to its functionality to scope the retransmission.

- The *active network capsules* help in communicating the congestion information and in regulating the use of bandwidth. The multicast source rate has been regulated by the selection of a nominee from the packet loss probability and RTT received from the receivers through the congestion status message capsule. It can be noted that the filtering of the nominee started from the upstream active node, which means that the selection processes are distributed to the local repair server (which can also resend/repair packet losses).

In general, AN's services in performing single rate multicast congestion control have proved to be working well. The AN-based mechanism to perform single rate multicast congestion control could perform the same functionality with the non-AN-based one, in which we have to ensure that all routers are capable of running the new multicast protocol. The use of ANs will shorten the time to deploy a new multicast protocol.

5.8 Summary

A number of experiments have been performed to reveal the behaviour of the AER/NCA protocol under different scenarios. In Experiment U1, in which a multicast session with 1 sender and 2 receivers shares the bottleneck link with 2 TCP flows, we observed the bandwidth share of the AER/NCA and TCP flows, and the inter-protocol fairness characteristic.

The impact of AN packets on bandwidth usage is observed by scaling the number of the TCP and AER/NCA receivers to 4, 8, 16, and 32. The loss probability of the multicast session has also been altered to 1%, 5% and 25%. In this lossy-link topology, the repair servers are turned on and off to evaluate the effect on the bandwidth usage. This experiment indicated that the bandwidth sharing capacity of the AER/NCA protocol is beneficial for the feedback aggregation.

We have also compared the inter-protocol fairness of PGMCC and AER/NCA in Experiment U2. It can be summarised that the behaviour of AER/NCA and PGMCC is similar in the case in which they have to compete with other TCP flows to use the bottleneck link. Both AER/NCA and PGMCC can fairly share the bandwidth over a short simulation time.

It is apparent that AER/NCA can improve the performance of a reliable multicast protocol, without the need for the intermediate node to be installed and rebooted with the new protocol. In fact, the use of AN services can be deployed on demand in an evolutionary way. The AN services can be used in the hop-by-hop feedback implosion avoidance, as well as in the distribution of loss recovery burden to repair server and receivers. Having the information on the congestion status of the network and the multicast sessions enables the sender to perform packet retransmission scoping.

The rate adjustment algorithm of the protocol has also been reviewed. The AER/NCA rate adjustment is similar to the fast recovery of the TCP New Reno session of the competing unicast session. The AER/NCA also performs the adjust-

ment of the congestion window, based on the slow-start threshold, in the situation of the arrival of the acknowledgement, detection of loss, or a time-out. The routing of the active packets in the simulation is based on the end-to-end default routing of the ns-2 simulator.

The congestion avoidance mechanism of the AER/NCA protocol has been shown to be robust, even though the idea of employing the rate of the weakest receiver as the new rate of the overall multicast session has also been implemented in other multicast protocols, with or without having an AN router-assisted scheme. This study demonstrates the TCP compatibility of the AER/NCA protocol under different topologies and combinations of flows. The simulation revealed that AER/NCA is a TCP-friendly congestion control protocol which uses active services when necessary. In other words, experimental results show that AER/NCA can coexist with TCP. The AER/NCA protocol achieves friendliness based on feedback signals from the receivers. The congestion control mechanism of AER/NCA has been able to adapt the multicast group throughput in achieving better overall performance in order to transmit information to all of the receivers.

We also have evaluated the fairness index of AER/NCA and TCP flows when sharing the bottleneck link using RED and Droptail queue management algorithms. The RED mechanism helps in promoting inter-protocol fairness/TCP-friendliness.

It can be summarised that the main goal of congestion control in the multicast sessions has been achieved by the AER/NCA protocol in ensuring the fairness towards TCP (inter-protocol fairness), a good coordination between the receivers, and fairness between the multicast sessions.

The discussion on the limitation of AER/NCA showed that the caching scheme required by the active error recovery is one of the trade-offs of AER/NCA. AER/NCA is also in a similar stage to other AN-based protocols, which we found to be far from reaching the stage of deployable to the Internet. The cost of having an AN architecture such as AN node initiation, execution environment settings, and the acceptance of the AN-based protocol in general will be the hindrance of deploying the algorithm widely.

In the next chapter, we will evaluate a multirate multicast congestion control protocol called ALMA. We also compare the characteristics of the layered multicast protocol with those of a non-AN-based protocol called PLM.

Chapter 6

Performance Evaluation of AN-based Multicast Multirate Congestion Control Protocols

The previous chapter deals with Active Networks (ANs)-based single rate multicast protocols. In this chapter, we conduct some performance evaluation on a multirate multicast congestion control protocol called Active Layered Multicast Adaptation (ALMA). We compare the ALMA protocol with a non-active multicast multirate protocol called Packet-pair Receiver-driven Layered Multicast protocol (PLM) [102]. Section 6.1 reviews the layered multicast protocols. Section 6.2 addresses the multirate multicast congestion control protocols evaluated in the research - ALMA and PLM, and includes a comparison study. Section 6.3 shows a first experiment and simulation results in evaluating the layered subscription of the multicast sessions. Section 6.4 presents a second experiment in which we evaluate the ALMA protocol's behaviour when competing with CBR flows. Section 6.5 presents a third experiment in which the multicast sessions share the link with TCP flows. Section 6.6 addresses simulation experiments to gain the fairness index values of ALMA and TCP flows which share a bottleneck link. In Section 6.7 the limitations of the AN-based scheme are evaluated. Section 6.8 discusses the overall performance evaluation conducted for the AN-based multicast multirate protocols. Finally, Section 6.9 summarises this chapter.

6.1 Introduction

Numerous multicast congestion control protocol proposals are found in the literature, aiming to provide an efficient method to transmit data from a sender to many receivers. These multicast congestion control protocols can be classified into single rate (discussed in the previous chapter) and multirate multicast protocols. The issue encountered in single rate reliable multicast protocols is that all receivers should receive service at the same rate, which can not accommodate the heterogeneous nature of the receivers and the differences in the bandwidth capacity of different networks such as T1, T3, Integrated Services Digital Network (ISDN), and Asynchronous Digital Subscriber Line (ADSL).

The *layered multicast congestion control* technique enables multiple multicast groups to transmit data at *different rates* for a diverse set of receivers. PLM [102] is an extension of the classical layered multicast approach such as the Receiver-driven Layered Multicast (RLM) [114] and the Receiver-driven Layered Congestion Control (RLC) [166], in which *fast convergence* to the optimum layer[1] is its main objective. Convergence time is the time taken by a receiver to adjust its reception rate to a stable state from any starting state [129], which is expected to be short. A *fair scheduler* has been used to ensure *inter-protocol fairness* in this protocol. Fair scheduling is a way to guarantee application performance by explicitly allocating shares of system resources among competing workloads [22]. For this purpose, some scheduling algorithms such as weighted round robin, deficit round robin, Generalised Processor Sharing (GPS), weighted fair queuing, class-based queuing, earliest due date, and rate controlled scheduling are used to perform the allocation of resources. A fair scheduler could protect one user's traffic from another, which is becoming more important with the increasing demand for differentiated services in the Internet.

Most classical layered multicast protocols require a receiver-consensus which implies all of the network's users (receivers) cooperate. In contrast to the previous classical layered multicast congestion control protocols which assume the existence of a multicast routing mechanism in the router, the AN-based ALMA protocol [187] uses *microeconomic theory* which envisions bandwidth as a *scarce resource.* We recall that historically the congestion control problem has been approached in three different ways: queuing theory, control theory, and recently game theory (pricing). ALMA is based on the recent approach introduced to the theory of computing which is based on an economic supply and demand model for non-collaborative players

[1] highest number of layers possible to use available bandwidth.

[187]. A reliable resource management method to support adaptive applications to share multiple resources with multiple nodes (complex network) dynamically is the goal of the market-based-driven approach.

ANs offer a mechanism to shorten the time to implement new protocols. This approach has been used in ALMA by using *capsules* (active packets) to supply congestion information to the link manager [186]. In this chapter, we conduct a performance evaluation of the ALMA protocol by comparing it with the PLM protocol. We have chosen ALMA for its layered multicast simulation approach on ns-2, the use of active packets for congestion control, and its concern on non-cooperative heterogeneous receivers in which the pricing function can be used to shape the distribution of bandwidth. On the other hand, we have selected PLM for its clarity in representing the solution to the layered multicast problem.

We perform some analysis of the *properties* of congestion control protocols which can be gained from ALMA. An *ideal congestion control* protocol would have some properties such as *stability* (existence and uniqueness of Nash equilibrium[2]), *efficiency* (fast convergence towards Pareto optimality[3]), *fairness* (max-min fairness), *robustness* (against misbehaving greedy users), *scalability* (with bandwidth heterogeneity, number of receivers, network paths, etc.), and *feasibility* (technical requirement) [102]. The scenarios to evaluate the congestion control properties of a layered multicast scheme used here are based on the PLM scenario stated in [102].

6.2 ALMA, PLM and Pricing

ALMA was the only adaptation protocol found in the literature which supports layered multicast for heterogeneous receivers and uses active services, until Congestion-Aware Layered Multicast (CAML) [169] was published two years later in 2001. CAML uses a very limited and lightweight form of AN to identify specific locations in the network where application-specific functionality needs to be placed to provide layered multicast systems. Recall that we have introduced ALMA in Chapter

[2]Nash equilibrium is a concept of game theory originated by John Nash. It serves to define a kind of 'optimum' strategy for games where no such optimum was previously defined. If there is a set of strategies for a game with the property that no player can benefit by changing his strategy while the other players keep the strategy unchanged then that set of strategies and the corresponding payoff constituted a 'Nash equilibrium' [www.wikipedia.org]

[3]Pareto optimality is defined as the best that could be achieved without disadvantaging at least one group. It is an optimality criterion for optimisation problems with multi-criteria objectives. Pareto optimal describes an equilibrium state in which no further trades among agents can occur that benefit both agents, with respect to a set of objective functions [23, 185].

3. It consists of a *layered multicast transmission* mechanism with *selective filtering* and *pruning* of layers within the active node. The ALMA source generates traffic encoded hierarchically, such that lower layers are more important than higher layers. Active packets (capsules) carry data and instructions to process the data inside the active node en route to the receivers. ALMA uses a source-based sparse-mode multicast routing scheme, but does not use a group address for simplification. The receivers subscribe directly to the address of the source they wish to receive data from, using a capsule called `subscribe(source address, subscription level)`. A receiver spawns a `subscribe(source address, 0)` capsule to abandon layers.

End-to-end congestion control in ALMA is achieved by exporting a *price* (as a function of the current load/queue occupancy), comparing the link price with the *budget* (proportional to marginal utility of adding more layers), pruning congested multicast tree branches (at the intermediate node), and probing for additional bandwidth using subscribe capsules (by the receivers).

On the other hand, PLM is a representative of a classical receiver-driven layered multicast congestion control protocol selected to be compared with ALMA in this thesis. However, it has the following assumptions: 1) the source sends cumulative layers (in sequence) and emits the packets as packet-pairs on each of the layers, 2) a fair scheduler exists and considers the set of layers of the same session as a single flow, and 3) the receivers who want to join the multicast session know the bandwidth distribution of the layers. The layer dropping/adding mechanism in PLM is conducted by comparing a sample bandwidth inferred from the packet-pair received at time $t(\mathrm{PP}_t)$ with the current bandwidth obtained with n cumulative layers (B_n). Due to the use of a packet-pair as the bandwidth estimation method, PLM does not induce any loss to discover the available bandwidth and infer which layer to join. PLM is a receiver-driven protocol in which the join and leave actions of the receivers lead to a congestion control mechanism.

We can summarise that layered multicasting with ALMA is feasible, as it uses ANs capsules to manage non-collaborative users. However, the ALMA protocol and ANs in general are still far from the deployment stage. The PLM approach to congestion control using packet-pair and fair queuing is another alternative, although the implementation is hard to achieve due to the need to implement fair schedulers in all routers. Table 6.1 summarises the differences between PLM and ALMA.

Congestion control in ALMA is seen as a special case of resource management, in which the congestion measure is performed using a pricing function. The pricing terminology represents a certain global measure of performance. The role of 'prices'

	PLM	ALMA
Proposed Network	Fair scheduler networks	ANs
Congestion Information	Packet-pair probing mechanism	Queue occupancy (buffer size) interpreted as Price
Properties	Fast convergence, stability, efficiency, scalability	Use of a price function to model the relationship between users and the network manager
Advantages	Packet probing mechanism is simple and does not take a lot of network resources	AN helps in deploying congestion control mechanisms
Disadvantages	Many assumptions, fair scheduler is difficult to implement	ANs are still far from deployment stage

Table 6.1: Comparison of PLM and ALMA

is to coordinate the optimisation of each source for its own benefit. Prices may have different meanings in different protocols, i.e. loss probability in TCP Reno, queuing delay in TCP Vegas, and queue length in RED [106]. The price in ALMA is calculated as a function of the link characteristics and the current total link load, thus it acts as an indication of the level of congestion for the filtering mechanism. A pricing function will enable the network manager to shape the distribution of resources to heterogeneous non-cooperative receivers.

In the following section, we evaluate ALMA's performance in terms of its speed and stability of layered convergence. Its behaviour in sharing the link with TCP and CBR flows is also investigated. In order to compare PLM with ALMA, we conduct four experiments. The first experiment (M1) consists of a single multicast session in which we evaluate the speed of the convergence to higher layers in the context of heterogeneous link bandwidths and link delays. The second experiment (M2) consists of a mix of multicast sessions and Constant Bit Rate (CBR) flows. In this experiment, we evaluate the scalability of PLM and ALMA with an increasing number of CBR flows. The third experiment (M3) deals with a mix of PLM or ALMA multicast sessions and TCP flows. The fourth experiment (M4) obtains the inter-protocol fairness index of ALMA.

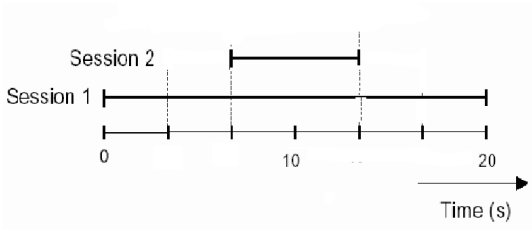
Application of ANs in ALMA

We use ns-2.1b6 [51] with an ALMA extension using Tcl-based code provided by Yamamoto [186]. In the ns-2 implementation of ALMA, active capsules are included as a message 500 bytes long. The protocol also modified the agent procedure with Agent/AN.

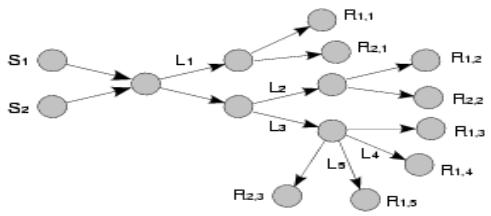
The execution environment is implemented as a subclass of Agent/Message. The active packets (capsules) are simply Agent/Message packets in which the 'message' is the code to be executed. In other words, the code loading and caching is not simulated in ALMA, as the evaluation of active packets is conducted by sending a string identifying the Tcl procedure to be run at each node. The multicast routing used in this protocol is dense mode. The ALMA traffic is embedded as CBR flows. All experiments in this work use Droptail queuing algorithm.

Validation

Validation has been performed in all steps of the experiment with ALMA. In a personal communication with the author of ALMA we ensured that the package we used is valid. The results of our experiment were acknowledged as an independent performance evaluation of ALMA in [185]. In order to ensure the validation of the ALMA simulation in this thesis, we chose the following results from a specific case in the literature, the work of Yamamoto et al. in [186], in which 2 ALMA sources transmit packets to a number of receivers (Figure 6.2(a)). The ALMA multicast sessions S1 and S2 lifetime and simulation topology are illustrated in Figures 6.1(a) and 6.1(b). All links have a capacity of 4 Mbps, except for links L1, L2, and L3



a) Life time of the sessions

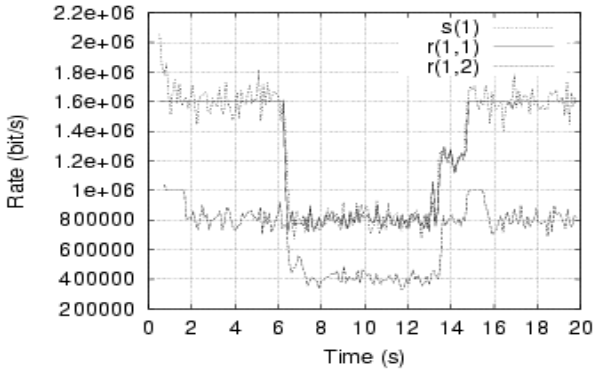


b) Simulation topology

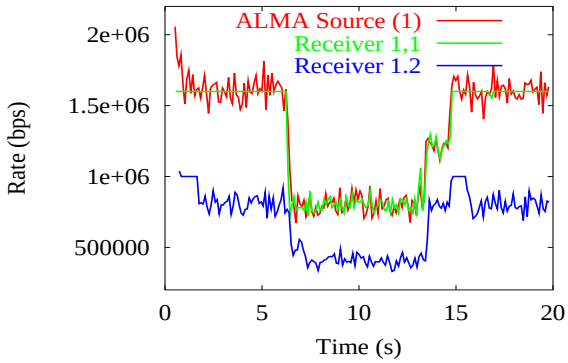
Figure 6.1: Simulation topology and life time of the sessions

which have a capacity of 1.6 Mbps, 1 Mbps, and 0,8 Mbps respectively. The delay of all the links is 10 ms, except for L4 and L5 which have propagation delays of 100

ms and 500 ms respectively. The maximum queue length of the nodes is 20 packets. Sources S1 and S2 generate streams with an average rate of 2 Mbps, divided into 5 layers. The evolution of the rates in time for S1 (source), receiver (1,1) and receiver (1,2) are depicted in Figure 6.2(a). We have performed a similar simulation and



a) ALMA simulation results obtained from [186] (This figure has been scanned from the work of Yamamoto et al. in [186]). This graphs shows the evolution of the rates versus time during the simulation time



b) Our ALMA simulation results for the same scenario

Figure 6.2: Validation of ALMA ns-2 package

as a result Figure 6.2(b) presents our simulation outcome when running the same topology and parameter settings. The simulation results produced are the same as the existing results as can be seen in Table 6.2. We also have run the simulation with different random generator seeds to generate the average value of 10 simulation runs. The error bars cover the plots presented in Figure 6.2(a) and (b).

Time (s) / Results (bps)	Alma source (S1) rate (obtained from [186])	Alma source rate from simulation	R11 (obtained from [186])	R11 rate from simulation	R12 rate (obtained from [186])	R12 obtained from simulation
2	1711715	1711715	1600484	1600484	860930	860930
4	1658211	1658211	1600000	1600000	727015	727015
6	1622749	1622749	1600000	1600000	919439	919439
8	789359	789359	759999	759999	396067	396067
10	897385	897385	819512	819512	450966	450966
12	7899001	7899001	744186	744186	368419	368419
14	1186932	1186932	1173984	1173984	836470	836470
16	1574319	1574319	1599999	1599999	735523	735523
18	1557803	1557803	1600000	1600000	781127	781127
20	1632815	1632815	1600000	1600000	818721	818721

Table 6.2: ALMA validation: Comparison of results obtained from [186] with our simulation results

6.3 Experiment M1: Speed and Stability of ALMA Convergence

In the following subsections we describe the simulation experiments, topologies, and key parameters of the first multicast multirate experiment. We discuss the results of the speed and stability of ALMA compared with PLM, and analyse the behaviour of these protocols towards layer subscription.

6.3.1 Experiment Construction and Description

For the purpose of comparing the protocols, some representative test cases as described in [102] are simulated. In other words, we compare the ALMA results with those achieved by PLM and reported by its author Legout et al. in [102]. We use the available results without resimulating the PLM experiments, to avoid possible errors caused by wrong algorithm implementation and interpretation.

The first simulation deals with a single ALMA session, in which one sender (S) transmits data to four receivers (R1, R2, R3, R4) through three routers (N1, N2, N3). The goal of the simulation is to gather information on the speed and stability of the ALMA convergence. The parameters are chosen to be close to the audio broadcast current classification, despite the fact the we do not purposely plan to simulate audio broadcast. The reason is just to observe whether in the real environment layer granularity will not be a problem for layered multicast. In this

experiment, we emphasise the importance of the bottleneck links for the layered multicast subscription of the receivers.

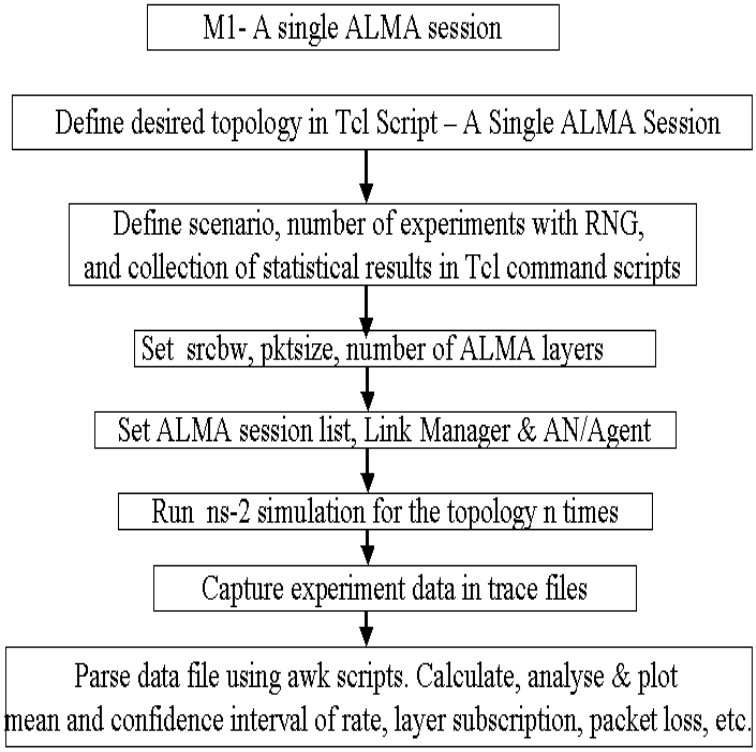
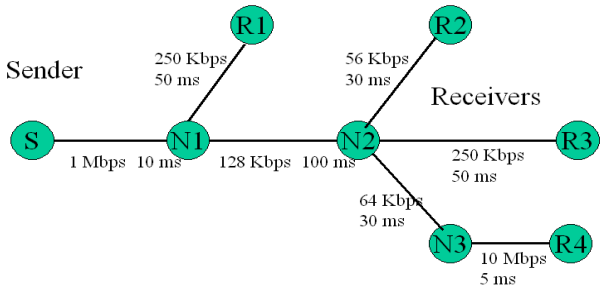


Figure 6.3: Experiment M1- Flowchart of simulation procedure

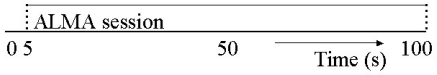
Figure 6.3 presents a flow-chart of the simulation procedure. We write the Tcl file on to the ns-2 extension of ALMA to define the desired topology, a single ALMA session with 5 receivers. The source bandwidth, packet size, and number of ALMA layers are set. Subsequently the ALMA session list, link manager, and AN agents need to be defined. We repeat the simulation with different random number generator seeds, run the simulation, capture the experiment data in a trace file, and then parse the data file using awk scripts. The results are used to calculate the average receiver rates (throughput) and their confidence interval, packet loss, as well as the layered subscriptions.

6.3.2 Topologies

The topologies used can be seen in Figure 6.4(a). In this topology, we have a single ALMA multicast sender which transmits packets to many receivers with heterogeneous link capacity and delay. The life time of each flow and session is shown in



a) Topology



b) Lifetime of session

Figure 6.4: Experiment M1 - Topology for single ALMA session

Figure 6.4(b).

6.3.3 Key Parameters and Rationale

In order to compare the results of ALMA and PLM on the same topology and parameter setting, we set the ALMA parameters to be the same as the PLM simulation experiments reported in [102]. The simulation parameters can be found in Table 6.3. The links bandwidths and delays are set to create a congestion situation and

Simulation Parameters	Value
Link S-N1	1 Mbps, 10 ms
Link N1-R1	256 Kbps, 50 ms
Link N1-N2	128 Kbps, 100 ms
Link N2-R2	56 Kbps, 30 ms
Link N2-R3	250 Kbps, 50 ms
Link N2-N3	64 Kbps, 30 ms
Link N3-R4	10 Mbps, 5 ms
ALMA Packet size	500 bytes
All routers maximum buffer size (Droptail queuing)	20 packets
Simulation time	100 seconds
Maximum number of layers	30 layers

Table 6.3: Experiment M1 - Simulation parameters for topology in Figure 6.4

to show the behaviour of ALMA towards heterogeneous receivers.

Some of the bandwidth values chosen in the links represent the current classification for audio broadcast: quality 1 (GSM (Global System for Mobile communication) quality): 10 Kbps; quality 2 (LW (Long Wave) radio quality): 32 Kbps; quality 3 (quality 2 stereo): 64 Kbps; quality 4 (quality FM (Frequency Modulated) radio): 128 Kbps; quality 5 (quality 4 stereo): 256 Kbps. This classification implies that for the purpose of real layered audio transmission, there is no need to create an intermediate layer, as it will not modify the perceived quality.

6.3.4 Performance Metrics

The performance metrics chosen in the experiment consist of throughput/rates[4] and layer subscriptions. The receiving rate was obtained from accumulating the bytes received by each sink endpoint during the simulation time. The layer subscription level shows the number of layers subscribed by a receiver during the simulation time. These metrics were chosen to represent the behaviour of the systems during the layered data transmission phase, the processing in the intermediate nodes, and the situation at the receiving ends.

6.3.5 Results and Evaluation

This section presents the experimental results of the speed and stability of the ALMA convergence. The graphs were generated by tracing data packets entering and leaving the routers in the network. We discuss the results of the simulation experiments to provide us with an insight into the performance of the ALMA protocol under various conditions.

First we will evaluate the layer subscription level of PLM and ALMA sessions. Table 6.4 shows the layer subscription level of PLM multicast sessions to 4 receivers. The simulation results for a single PLM session, provided in [102] shows that PLM receivers can subscribe to an appropriate number of layer. They also reported that all receivers converge to the highest number of layers possible in order to use the available bandwidth before 7 seconds. These are the maximum number of layers to be subscribed based on the granularity of 10 Kbps per layer. On the other hand, the same experiment with ALMA (Figure 6.5) shows similar results, except for Receiver-4 (R4) which subscribes to 5 instead of 6 layers. Soon after the ALMA session initiation time (5 seconds), all receivers converge into optimal layers between 5.5 to

[4]In this multicast congestion control work, we use the terms throughput and rates interchangeably.

Receiver	PLM Layer Subscription
R1	25
R2	5
R3	12
R4	6

Table 6.4: Experiment M1 - Total number of packets transmitted and received during simulation (1 PLM session and 4 receivers)

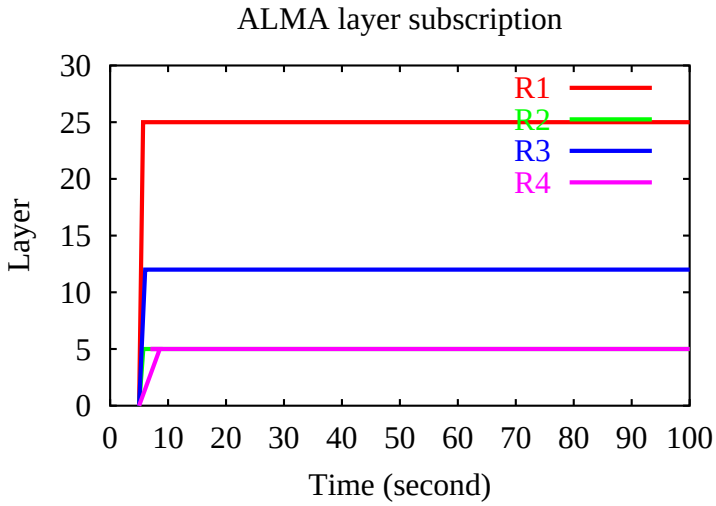


Figure 6.5: Experiment M1 - Speed and stability of ALMA convergence for single multicast session

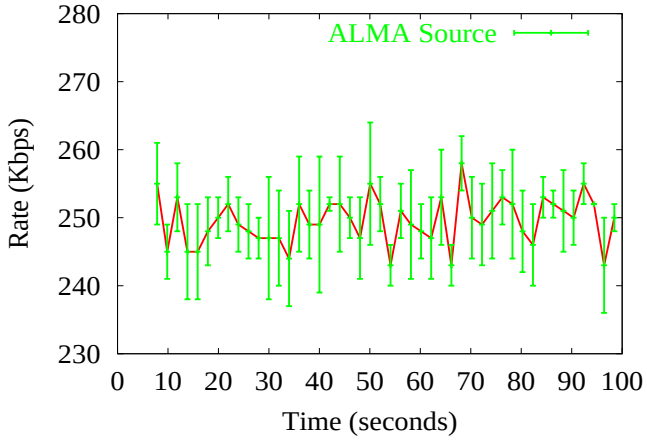
8.6 seconds. The receivers stay at the optimal layer during the entire simulation, and this shows stability of the *optimal layer*. There is no loss detected during the simulation, as the links can accommodate the transfer rate. In addition, this is also due to the ALMA mechanism to control the congestion. This experiment shows that ALMA can provide fast convergence to the optimal link utilisation (less that 3.6 seconds to reach a stable state). Both R2 and R4 in ALMA subscribed to 5 layers, whereas in PLM R4 subscribed to 6 layers. This is because R4 only received an average rate of 57 $\pm$ 0.5 Kbps. The average rates of ALMA session's packets transmitted and received are presented in Table 6.5.

Sender	**Packets transmitted average rate (Kbps)**	**ALMA Receiver**	**Packets received average rate (Kbps)**
$\mathbf{S}_{ALMA}$	200.1 $\pm$ 0.8	**R**1	250 $\pm$ 3.2
		R2	46 $\pm$ 8.2
		R3	112 $\pm$ 0.5
		R4	57 $\pm$ 0.5

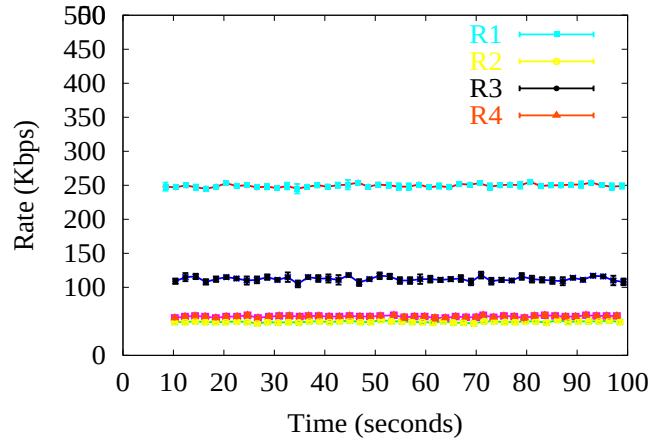
Table 6.5: Experiment M1 - Total number of packets transmitted and received during simulation (1 ALMA session and 4 receivers)

Figure 6.6(a) shows the average sending rate and the confidence interval of the ALMA source during 100 seconds of simulation time, whilst Figure 6.6(b) shows how the receivers' rate evolves in distributing the data. Note that the confidence intervals of the average receiver rates are small. This figure shows that the ALMA receiver subscribes to the optimal layer it can accommodate and uses the maximum available bandwidth. Receiver 1 can easily receive all layers of the multicast session, whereas Receiver 2 can only subscribe according to its maximum 56 Kbps link bandwidth. On the other hand, Receiver 3 is limited by the 128 Kbps link, whereas Receiver 4 is bounded by the 64 Kbps link. The ALMA congestion mechanism allows the receiver to subscribe according to their available bandwidth which causes no packet loss experienced by any of the receiver.

The objective of the experiment was to compare the performance of PLM and ALMA, taking into account that both protocols are multirate multicast congestion control protocols, even though they use different network technology, i.e. fair scheduler network and ANs, respectively. We found the ANs mechanism helps the layered multicast congestion control in ALMA to be effective, i.e. fast convergence (subscribe to the maximum possible number of layers) and stability (maintain the rates). This protocol works well, which indicates that the link manager can manage



a) Sender



b) Receivers

Figure 6.6: Experiment M1 - Bandwidth distribution of a single ALMA session

the resources as expected.

6.4 Experiment M2: Competing with CBR Flows

6.4.1 Experiment Construction and Description

The second simulation experiment consists of a mix of ALMA sessions and CBR flows, i.e. 3 ALMA sessions and 3 CBR flows. The aim is to study the effect of high bandwidth-delay unresponsive CBR flows on ALMA flows. CBR is used to represent a UDP-based multimedia traffic which is sensitive to packet loss. Each ALMA sender corresponds to one receiver and starting time is staggered from 10, 20, and 30 seconds.

The aim of this experiment is to evaluate the behaviour of ALMA with an increasing number of ALMA sessions (before starting the CBR sources), and then after the transmission of the CBR sources which represent heavy traffic in the network. It is the goal of this experiment to evaluate the responsiveness and limitation of ALMA towards a mix of traffic, particularly when competing with CBR flows.

Figure 6.7 presents a flow-chart of the simulation procedure. First, we write the Tcl file on to the ns-2 extension of ALMA to define the desired topology, an ALMA session and 3 CBR flows. Next, we define the source bandwidth, packets size, and the number of ALMA layers. Then, we define the ALMA and CBR sessions list, the link manager, and active agents. We repeat the simulation with different random number generator seeds 10 times. The experiment data is then captured in a trace file, and is then parsed into meaningful data using awk scripts, which in turn will be used to calculate the average rates (throughputs) and their confidence interval, layer subscription, and packet loss.

6.4.2 Topologies

The topology used and the life time of each flow and session are shown in Figure 6.8(a) and (b), respectively.

6.4.3 Key Parameters and Rationale

In order to compare the results of ALMA and PLM on the same topology and parameter settings, the ALMA parameter values were set the same as in the PLM

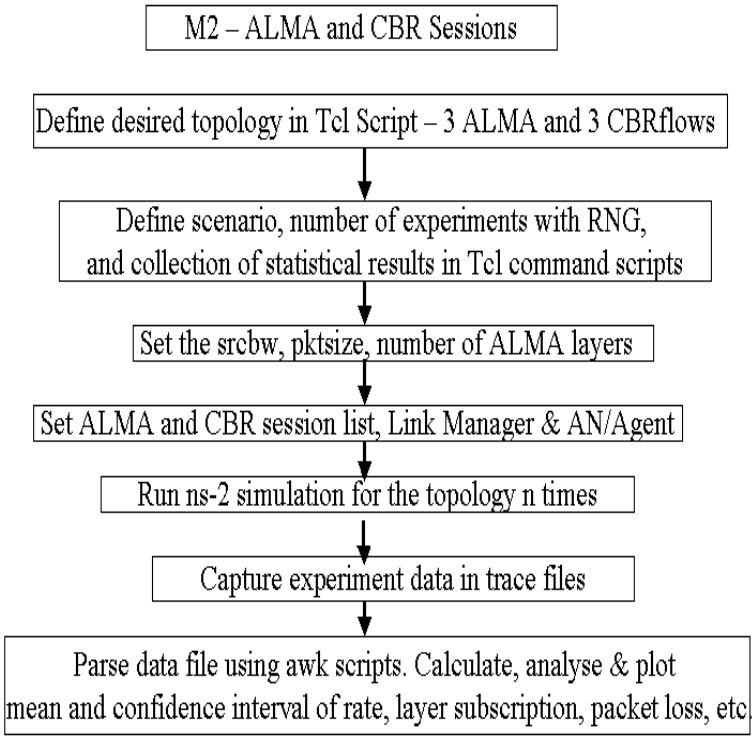
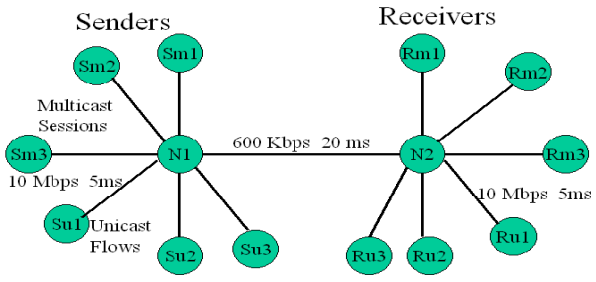
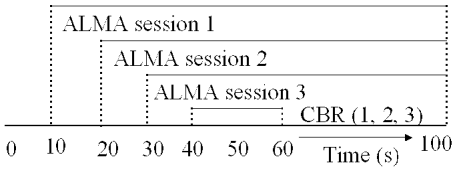


Figure 6.7: Experiment M2 - Flowchart of simulation procedure



a) Topology



b) Life time of sessions

Figure 6.8: Experiment M2 - Topology for 3 ALMA sessions and 3 CBR flows

simulation experiment reported in [102]. The simulation parameters are shown in Table 6.6.

Simulation Parameters	Value
Link N1-N2	600 Kbps, 20 ms
Links Sm-N1, Su-N1, N2-Rm, and N2-Ru	10 Mbps, 5 ms
ALMA packet size	500 bytes
All routers maximum queue size (Droptail queuing)	20 packets
Simulation time	100 seconds
Bandwidth increments (for PLM and ALMA)	2 seconds
Maximum number of layers	17
ALMA source rate	340 Kbps

Table 6.6: Experiment M2 - Simulation parameters for topology in Figure 6.8(a)

The links between the sources and router N1 have 10 Mbps bandwidths and 5 ms delays. Similarly, the links from router N2 to the sinks also use the same bandwidth and delay values. The bottleneck link is 600 Kbps and 20 ms delay. The ALMA sessions were started at 10, 20, and 30 seconds, respectively, whereas all the CBR flows were started at 40 seconds and terminated at 60 seconds of simulation time (Figure 6.8(b)). These parameter settings create a severe congestion situation.

6.4.4 Performance Metrics

The performance metrics reported in this experiment consist of throughput/rate, packet loss percentages, and layer subscriptions. The throughput/rate has been obtained by accumulating the total bytes received (excluding retransmitted packets) by each sink endpoint during the simulation time. This rate is the same as the throughput in PLM results. The layer subscription level shows the number of layers subscribed by a receiver during the simulation time.

6.4.5 Results and Evaluation

This section presents the experimental results and highlights the dynamic behaviour of the AN-based multirate multicast protocol when sharing the bottleneck link with CBR flows which represent real-time unresponsive traffic.

The dynamic scenario is used with the topology in Figure 6.8(a). The objective of the scenario is to study the behaviour of ALMA in the first part (before starting the CBR sources), with an increasing number of ALMA sessions. Then, after starting

the CBR sources (second part), the behaviour of ALMA in a congestion situation can also be evaluated.

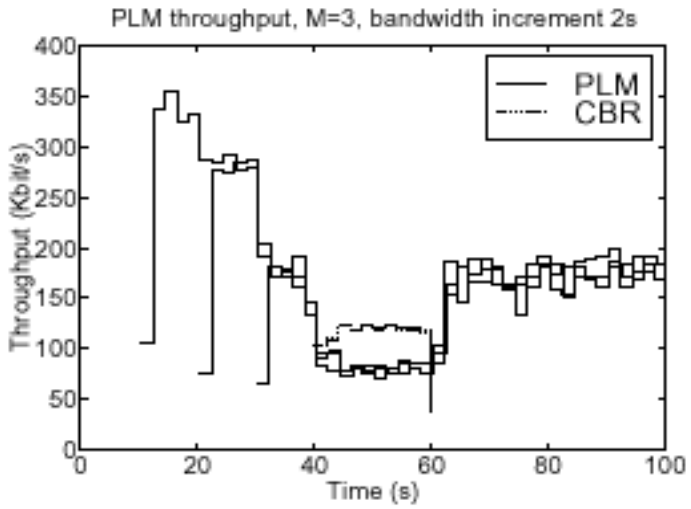
In Figure 6.9(a), we can see the bandwidth distribution with PLM, which shows that the throughput seen by the PLM receivers decreases as the number of PLM sessions increases. The situation is aggravated when the bandwidth is shared with 3 new CBR flows initiated at 40 seconds. Figure 6.9(b) shows ALMA and CBR average receivers' rates and their confidence interval during the entire simulation. The rate variations are small and the confidence intervals of the flows are negligible. The figure shows that ALMA sessions adjust their rate when the CBR flows start. However, the CBR flows are competing among themselves, and one of the flows has a higher share of bandwidth than the other CBR flows. In addition, the first CBR flow has a higher bandwidth share than the ALMA flows. ALMA infers an average available bandwidth of 110 Kbps and joins 10 layers. We observe that ALMA is reactive and its speed to grab the available bandwidth is quick. This is shown by the fact that the aggregate ALMA rate increases as soon as the CBR flows release the bandwidth after 60 seconds of the simulation time. Each ALMA session receives almost an equal share of bandwidth (the average of 200 Kbps). The ALMA flows rates after the completion of the CBR sessions are stable and utilise the available bandwidth fairly among themselves. This fast convergence has also been achieved by PLM flows with an average rate of 180 Kbps. The non-AN and AN-based protocols have shown their reactiveness in grabbing the available bandwidth released by the CBR flows.

Table 6.7 presents the number of transmitted bits normalised to time (in Kbps)

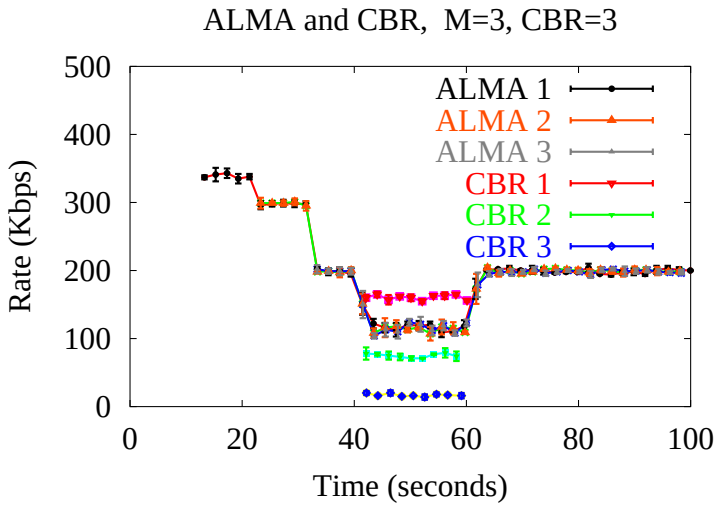
Sender	**Packets transmitted average rate (Kbps)**	**Receiver**	**Packets received average rate (Kbps)**	**Packet lost ratio (%)**
$\mathbf{S}_{ALMA1}$	226.9 ± 1.3	$\mathbf{R}_{ALMA1}$	207.9 ± 0.6	8.3 ± 0.2
$\mathbf{S}_{ALMA2}$	212.4 ± 0.3	$\mathbf{R}_{ALMA2}$	190.5 ± 0.5	10.3 ± 0.2
$\mathbf{S}_{ALMA3}$	202.4 ± 5	$\mathbf{R}_{ALMA3}$	174.76 ± 0.9	12.5 ± 0.1
Scbr1	200 ± 0	**Rcbr1**	161 ± 2.2	19.2 ± 1.2
Scbr2	201 ± 2.7	**Rcbr2**	75.2 ± 1.9	62 ± 1
Scbr2	192 ± 8.6	**Rcbr2**	17.3 ± 0.7	91 ± 0.4

Table 6.7: Experiment M2 - Total number of packets transmitted and received during simulation (1 ALMA session and 2 TCP flows)

and the average throughput (in Kbps). The table shows that the CBR flows experience high packet loss. This is due to the limited available bandwidth and the



a) PLM and CBR Receivers (This figure has been scanned from the work of Legout et al. in[102]). The fair scheduler mechanism in PLM enforced the fair sharing of bandwidth between PLM and CBR flows



b) ALMA and CBR receivers

Figure 6.9: Experiment M2 - Bandwidth distribution with PLM and ALMA (3 multicast sessions and 3 CBR flows)

fact that these CBR flows must compete among themselves to use it. Bandwidth is allocated among sessions on a first-come-first-served basis. Once a session has achieved a dominant allocation, other CBR sources may starve. This characteristic is inherited from the first layered multicast protocol, RLM, which is the predecessor of ALMA. Results in [71] show that RLM is also arbitrarily unfair to CBR traffic.

The evolution of the layered multicast subscription level of ALMA can be seen in Figure 6.10. The layer subscription of the ALMA receivers changes as the number

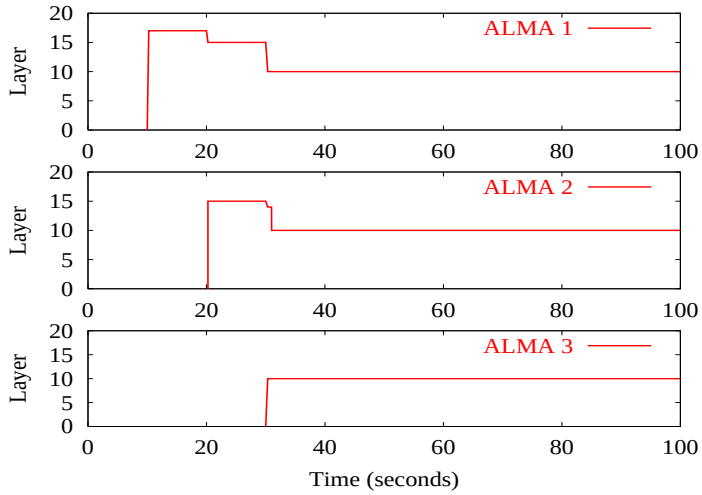


Figure 6.10: Experiment M2 - Layer subscription with ALMA (3 multicast sessions and 3 CBR flows)

of ALMA flows increases. This figure shows that the CBR sources do not affect the ALMA sessions layer subscription. All of the ALMA flows keep subscribing to 10 layers. However, the existence of CBR caused the ALMA receivers to experience packets loss, which have been subscribed but not received by the receivers.

ALMA's performance in sharing the bandwidth with CBR sources is different from PLM, because there is no mechanism used to ensure fairness such as fair scheduling in PLM. The ALMA experiment results show that the bottleneck bandwidth is not fairly shared between the ALMA and the CBR flows. ALMA allows one of the CBR flows to take more bandwidth, but the rest of the CBR flows get a very small amount of bandwidth. It can be noted that although based on different mechanisms, both ALMA and PLM flows grab an appropriate amount of bandwidth and stay stable during the lifetime of the CBR sessions.

Although ALMA was not designed to achieve fast convergence, the experiment shows that it can achieve convergence relatively fast, due to its capsule-based congestion indication mechanism. This is done by using the queue occupancy information

to decide how many layers to subscribe when the subscribe capsule is sent. ALMA is a cumulative multicast protocol in which receivers subscribe to the higher layers in subsequent order. The study on the mix of ALMA and CBR flows shows the layered multicast protocol's reaction on dynamic scenarios.

6.5 Experiment M3: Competing with TCP Flows

6.5.1 Experiment Construction and Description

For the purpose of comparing the schemes, some representative test cases as described in [102] were simulated. The aim of the simulation is to evaluate ALMA's performance in sharing a bottleneck link with other TCP flows. This first part of the simulation experiment deals with the mix of a single ALMA session and two TCP sources. The second part illustrates how an ALMA session shares a link with 1 TCP flow.

Figure 6.11 presents a flow-chart of the simulation procedure. We write the

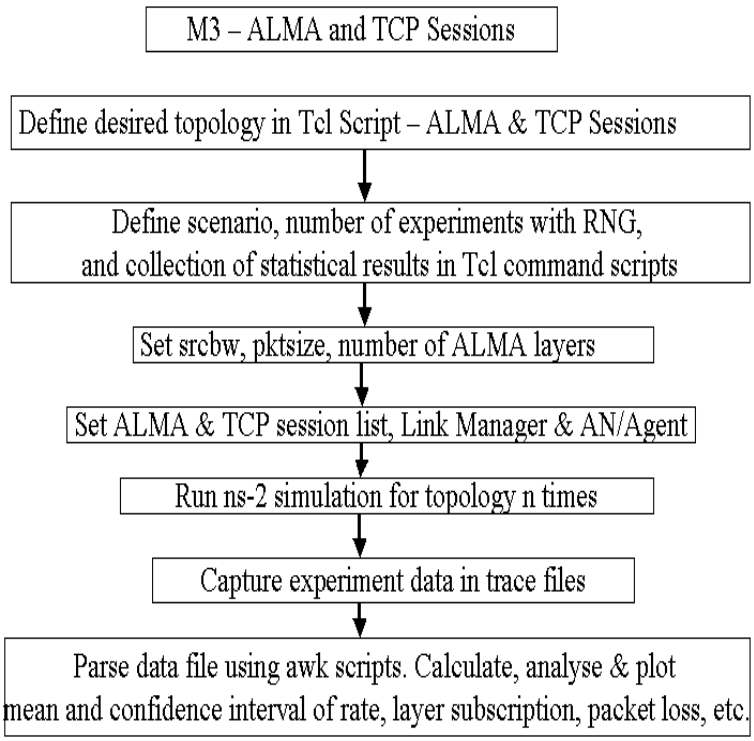


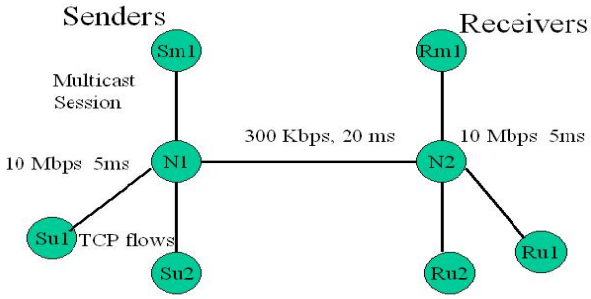
Figure 6.11: Experiment M3 - Flowchart of simulation procedure

Tcl file on to the ns-2 extension of ALMA to define the desired topology, an ALMA

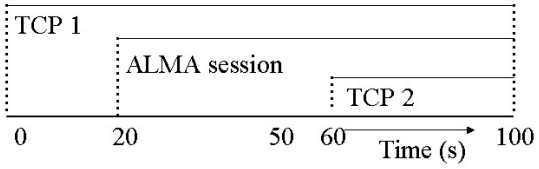
multicast session and TCP sessions. The source bandwidth, packet size, and number of ALMA layers are defined. The ALMA and TCP session list, link manager, and active agents are set. We construct and define the repetition of the simulations with different random number generator seeds. Next, we run the simulation 10 times. The experiment data are captured in a trace file. Then we parse the data file into meaningful results using awk scripts, which in turn will be used to calculate the average rates and their confidence interval, layer subscription, and packet loss ratio.

6.5.2 Topologies

The topology used to evaluate the performance of a single ALMA session competing with 2 TCP flows is depicted in Figure 6.12(a). The lifetime of each flow and session



a) Topology



b) Life time of sessions

Figure 6.12: Experiment M3 - Topology for a single ALMA session and 2 competing TCP flows

is shown in Figure 6.12(b). The topology to evaluate the ALMA effect on a TCP flow (the second part of the experiment) can be found in Figure 6.13, in which we use one ALMA session and one TCP flow.

6.5.3 Key Parameters and Rationale

The objective of this experiment was to study ALMA with TCP flows in a simple experiment. In order to compare the results of ALMA and PLM using the same topology and parameter settings, the ALMA parameter values were set to be the

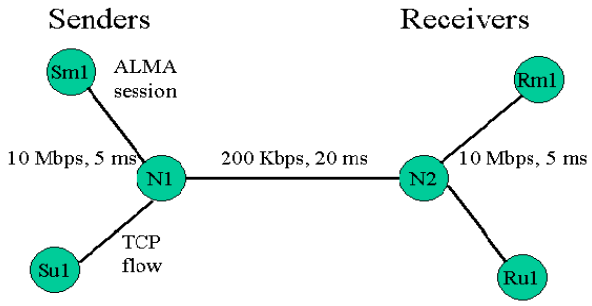


Figure 6.13: Experiment M3 - Topology for a single ALMA session and 1 competing TCP flow

same as in the PLM simulation experiment reported in [102]. The bottleneck link's bandwidth is 300 Kbps, and the delay is 20 ms (Figure 6.12). The initial transmission rate of ALMA flows was set to 100 Kbps. At 0 seconds of the simulation time, we started the first TCP connection; at 20 seconds, the ALMA session was initiated, and the second TCP flow was initiated at 60 seconds simulation time. Unlike PLM which specifies the layer granularity as one of the parameters, in ALMA the layer granularity is defined by specifying the source bandwidth and the number of layers. In order to have the same layer granularity as in the PLM experiments, we define the maximum subscription level of 15 and 20 layers in the following experiments. Therefore, the layer granularity used in these experiments is 20 Kbps. The ALMA packet size used in all experiments was 500 bytes. The simulation parameters can be found in Table 6.8.

Simulation Parameters	**Value**
Link N1-N2	300 Kbps, 20 ms (200 Kbps for Fig 6.13)
Links Sm-N1, Su-N1, N2-Rm, and N2-Ru	10 Mbps, 5 ms
ALMA packet size	500 bytes
All Routers maximum queue size (Droptail queuing)	20 packets
Simulation duration	100 seconds
Maximum number of layers	15 layers (20 layers for Fig 6.13)
ALMA source rate	400 Kbps (300 Kbps for Fig 6.13)

Table 6.8: Experiment M3 - Simulation parameters for topologies in Figures 6.12 and 6.13

6.5.4 Performance Metrics

The performance metrics reported in this experiment are similar to those of Experiments M1 and M2 which consist of throughput/rate, packet loss ratio, and layer subscriptions.

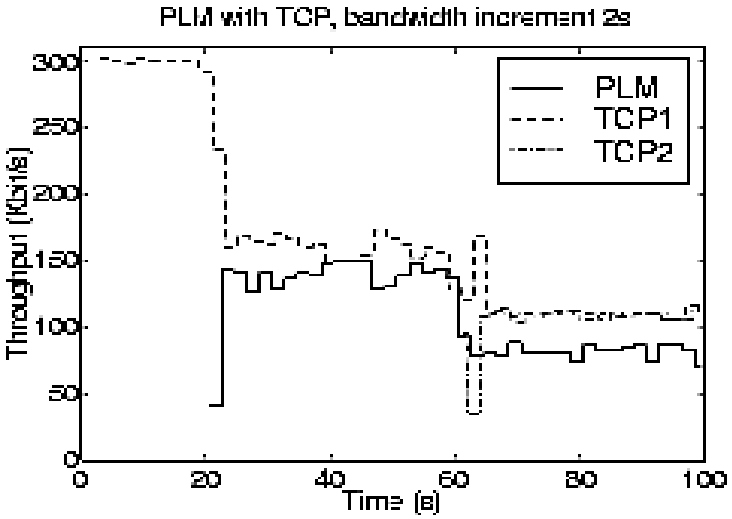
6.5.5 Results and Evaluation

This section presents the experimental results and highlights the behaviour of the AN-based multirate multicast protocol towards the TCP flows. A 'peaceful' coexistence with TCP is a desirable property of a protocol in which it must be able to share the bandwidth fairly with TCP flows.

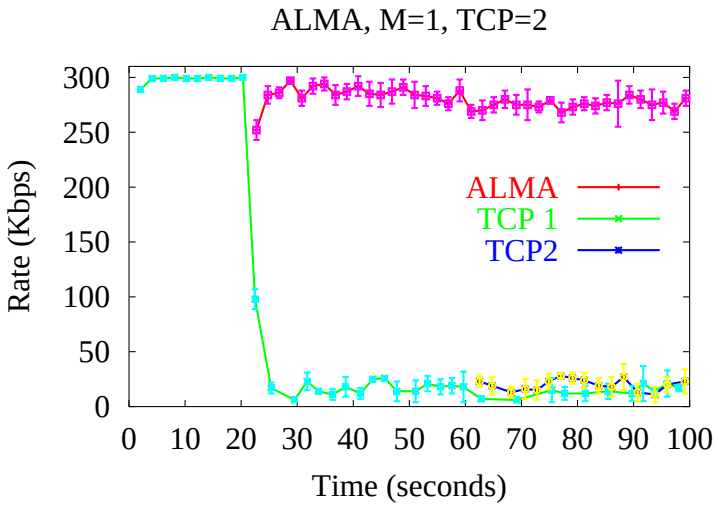
The bandwidth distribution of PLM and ALMA in conjunction with the topology in Figure 6.12 is shown in Figure 6.14. The ALMA session is initiated 20 seconds after the transmission of the first TCP flow (at the beginning of the simulation), and subsequently another TCP flow is induced to the network at 60 seconds. The ALMA results show that the bandwidth share of the ALMA session is a lot higher than PLM. Note that in PLM, the inter-protocol fairness is enforced by the use of a fair scheduler in every router.

In terms of compatibility with TCP flows, Figure 6.14(a) shows that with PLM, both TCP flows hardly oscillate and the bandwidth share of PLM is relatively constant, which is an expected behaviour of PLM which uses a fair scheduler. Recall that this is due to the packet-pair bandwidth estimation and fair queuing principles deployed in PLM. The single PLM session adapts to the available bandwidth in the presence of TCP flows. In contrast, simulation with ALMA shows that ALMA is not TCP-friendly (Figure 6.14(b)) as it grabs the available bandwidth as soon as it starts and then starves the TCP flows. The TCP flows can only use the rest of the bandwidth. The average packet received rates of the first and second TCP flows are only 75.6 and 17.1 Kbps respectively. This is due to the unfriendliness of ALMA towards TCP. This result shows that ALMA maintains its stability and high bandwidth share at the expense of the rate fluctuations and low bandwidth share of the TCP flows.

Table 6.9 exhibits the average total number of packets transmitted and received normalised for time (in Kbps) by the multicast and unicast nodes in the experiment. It shows that the ALMA session experienced 7.14% packet loss, whereas the TCP sessions encountered 5.8% and 26.8% of packet loss, respectively. The high loss rate experienced by TCP2 which started at 60 seconds of simulation time is due to



a) PLM and TCP receivers (This figure has been scanned from the work of legout et al. in[102]). This figure shows PLM's TCP-friendliness as TCP flows received a fair share of bandwidth, due to the fair scheduling mechanism in PLM.



b) ALMA and TCP receivers

Figure 6.14: Experiment M3 - Bandwidth distribution with PLM and ALMA (1 PLM sessions and 2 TCP flows)

Sender	Packets transmitted average rate (Kbps)	Receiver	Packets received average rate (Kbps)	Packet lost ratio (%)
$\mathbf{S}_{ALMA}$	299.9 ± 3.7	$\mathbf{R}_{ALMA}$	280.3 ± 3.1	7.14 ± 1
Stcp1	79.8 ± 3.7	**Rtcp1**	75.6 ± 3.4	5.8 ± 0.3
Stcp2	23.3 ± 4.7	**Rtcp2**	17.1 ± 4	26.8 ± 2.8

Table 6.9: Experiment M3 - Total number of packets transmitted and received during simulation (1 ALMA session and 2 TCP flows)

the unfriendly behaviour of ALMA, as it did not adjust its rate when the new flow started. Therefore the TCP flows have been forced to share the rest of the bandwidth among themselves and experience oscillation in attempting to send packets.

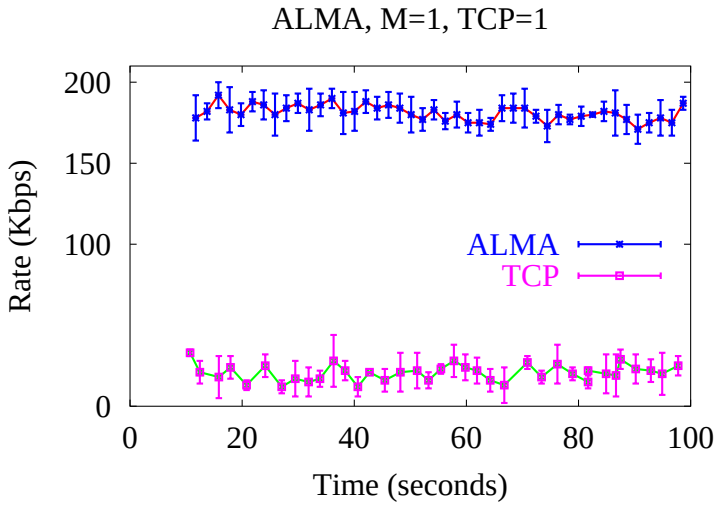


Figure 6.15: Experiment M3 - Bandwidth distribution with ALMA (1 ALMA session and 1 TCP flow)

In order to observe the ALMA's compatibility with a TCP flow further, we conducted the second part of the experiment, which is a simple simulation in which the link is shared only by 1 TCP flow and 1 ALMA session (topology shown in Figure 6.13). Figure 6.15 shows the ALMA and TCP bandwidth distribution. Experiment results show that the average rate of ALMA is approximately 180.9 ± 3.2 Kbps. On average, the TCP flow gained only a small portion of bandwidth compared with ALMA session (32.8 ± 3.5 Kbps). The packet loss ratio of TCP is high 18.23%. This shows the unfriendly behaviour of ALMA. Table 6.10 exhibits the average rate of packets transmitted and received (in Kbps) and the packet loss ratio for ALMA and TCP flows. From these experiments, it can be noted that ALMA flow grabs

Sender	Packets transmitted average rate (Kbps)	Receiver	Packets received average rate (Kbps)	Packet lost ratio (%)
$\mathbf{S}_{ALMA}$	200.1 ± 0.8	$\mathbf{R}_{ALMA}$	180.9 ± 3.2	9.95 ± 1.81
Stcp	32.8 ± 3.5	**Rtcp**	26.7 ± 3.06	18.23 ± 0.8

Table 6.10: Experiment M3 - Total number of packets transmitted and received during simulation (1 ALMA session and 1 TCP flow)

the maximum bandwidth causing TCP flow to starve.

6.6 Experiment M4: Fairness Index

6.6.1 Experiment Construction and Description

The aim of this experiment is to evaluate the fairness index of ALMA in relation to the use of the Droptail queue management scheme at the intermediate routers. We set up the simulations using a network topology with different delays of the receiver side links. In other words, we vary the round trip times to evaluate the throughput and fairness index of the link. We use the same Jain's fairness index defined in [76] and used previously in Chapter 5 (Experiment S3).

We run the simulation using various propagation delays in the receiver side's link and Droptail as a router queue management mechanism in the intermediate nodes. RED queue management is not used , as it is not available in ALMA's environment due to its function as a method to calculate the price. The average throughput results are used to calculate the fairness index for Droptail routers.

Figure 6.16 presents a flow-chart of the simulation procedure. We define the desired topology on a Tcl file, one ALMA and n TCP sessions. Then the source bandwidth, packet size, and the number of ALMA layers are set. The multicast and TCP sessions are defined, as well as the link manager and AN agents. We define the repetition of the simulation with different random number generator seeds and then run the ns-2 simulations. The experiment results are captured in a trace file. The data are parsed using awk scripts. The results are used to calculate the throughput and inter-protocol fairness of ALMA.

6.6.2 Topologies

The topology in Figure 6.17 is used in this experiment to see the fairness of an ALMA

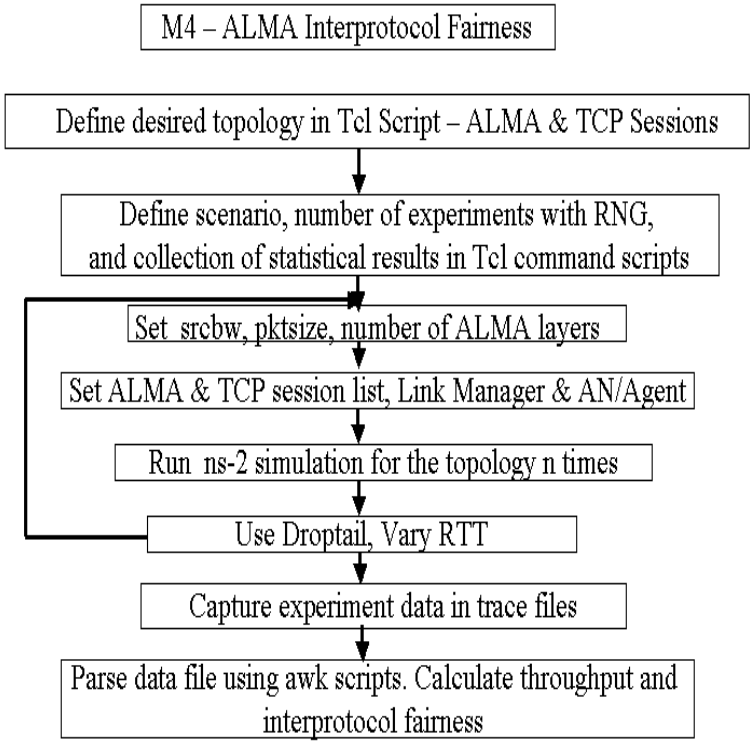


Figure 6.16: Experiment M4 - Flowchart of simulation procedure

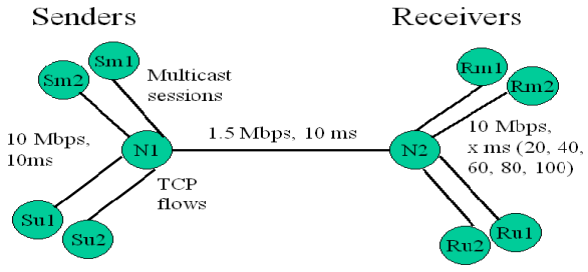


Figure 6.17: Experiment M4 - Topology for ALMA inter-protocol fairness

flow towards TCP. Firstly, we use only 1 ALMA session and 1 TCP flow (K1), and then 2 multicast sessions and 2 TCP flows (K2). The aim of these experiments is to obtain the fairness index and evaluate the effect of increasing the number of flows.

6.6.3 Key Parameters and Rationale

For the inter-protocol fairness experiment in Figure 6.17 we use the simulation parameters listed in Table 6.11. We use the following parameters as described in [184] in order to have a comparative reference with other multicast protocols' fairness evaluation. The bottleneck bandwidth used between nodes 1 and 2 is 1.5 Mbps, whereas the propagation delay is 10 ms. Each multicast session has one sender and one receiver. We use Droptail queue management algorithm at the routers. Each simulation lasts for 100 seconds.

Simulation Parameters	**Value**
Sources	1 Multicast session and 1 TCP flow
Sources - Router 1 links	10 Mbps, 10 ms delay
Router 1 - Router 2 link (bottleneck link)	1.5 Mbps, 10 ms delay
Router 2 - Receivers links	10 Mbps, x ms delay (x = 20, 40, 60, 80, 100)
All routers maximum buffer size (Droptail queuing)	30 packets
Simulation time	100 seconds
ALMA Packet Size	500 bytes
ALMA source rate	2 Mbps
Number of layer	17

Table 6.11: Experiment M4 - Simulation parameters for topology in Figure 6.17

6.6.4 Performance Metrics

The performance metrics reported in these experiments consist of throughput and fairness index.

6.6.5 Results and Evaluation

The following results describe the inter-protocol fairness of ALMA quantitatively using Jain's fairness index equation. Table 6.12 shows the fairness index of ALMA with TCP and a Droptail queue management algorithm. If all sessions have the same throughput, the fairness index is 1. The fairness index value is bounded between *1/n* and 1 (n is the number of flows in the bottleneck link). In all experiments the

Receiver side link's delay (ms)	20	40	60	80	100
Droptail (K1= Multicast & 1 TCP flows)	0.57	0.58	0.60	0.63	0.65
Droptail (K2 =Multicast & 2 TCP flows)	0.66	0.66	0.65	0.64	0.63

Table 6.12: Experiment M4 - Fairness index with routers with Droptail queue management algorithm

ALMA flows always get more bandwidth than the TCP flows. This is due to the ALMA's congestion control mechanism using the AN-based pricing functionalities, in which competing TCP sessions have not been considered.

The fairness index performance of the network with Droptail queue management algorithm and 2 multicast sessions and 2 TCP flows (K2) is slightly higher than the topology with one multicast session and one TCP flow. This can be seen from the results of the experiment with varying the round trip time, in which the average fairness indices of the experiment with 2 multicast sessions and TCP flows (K2) is 0.65. This is higher compared with the experiment with K1 (1 multicast session and 1 TCP flow) in which the average of the fairness indices is 0.61. Notice that this is due to the fact that the experiment with K2 is employing more flows (4 flows) than the experiment with K1. Overall, it can be noted that the fairness index of ALMA is low. This experiment concludes that ALMA is not a TCP-friendly protocol.

6.7 Limitation of the Scheme

In the previous experiments, we have evaluated ALMA's capability to perform congestion control by means of layer subscription, filtering and pruning, as well as its behaviour in sharing the bottleneck link with CBR and TCP flows. In this section, we analyse the trade-off between the flexibility ALMA can provide to the network manager and its limitations.

ALMA was based on a case study for sharing multiple resource types. As the focus was on employing a market-based resource sharing scheme using equations, the problem of message passing and communication overhead was not considered in ALMA. Code caching and forwarding are not performed in ALMA ns-2 simulation. Considering that it is not yet a complete set of protocols, ALMA's implementation in terms of how the distribution of active packets can be defined and the relationship with other components of the AN architecture is left for future work.

The message passing overhead of ALMA will largely depend on whether every

capsule or just the first capsule in the flow will carry the code. All AN-based congestion control protocols are built on the idea that message passing and computational power needed is negligible compared with the functionality of the protocol. For example, retransmission scoping in multicasting can be performed to save the effort of retransmitting the whole data packets.

For the case of ALMA, if we define that all capsules can carry code, the message passing overhead can be quite large. We will encounter some overhead in the execution environment in which the new protocol needs to be executed. In addition some user level send/receive of packets are the bulk of extra overhead. The overhead will include longer transfer time, longer packets, and increasing bandwidth demand.

If only the first flow carries capsules, then the amount of message passing across the network can be made smaller. Most protocols using this approach would have some ready program available on the node, which can be initiated by an active packet sent in the first flow. All the packets of a flow will have a common source and destination address. The computation of the code can be done once at the beginning of the flow instead of being done each time a packet arrives.

Our experiments show that ALMA has many limitations and uses many assumptions. The work was built on top of the end-to-end classical layered multicast such as RLM, which has been reported to be unfair to any type of competing flow. This protocol uses source-based reverse path routing and does not need native IP multicast groups. The number of multicast groups which can be simulated is limited, and probing for additional bandwidth is a potential cause of intensifying congestion. There is a trade-off between the flexible market-based approach to the layered multicast used in ALMA and the practicality of its implementation. Moreover, ALMA is not TCP friendly and may potentially harm the Internet. The amount of information circulating in the network can be high because users can send data as well as code to be executed. This protocol gives better flexibility to perform layered multicast congestion control. However, the message passing used to perform the AN services and the cost of a higher level enabling architecture is high. Therefore, this approach has yet to prove its practical usefulness.

6.8 Discussion

The following is a brief summary of the results of the experiments with the AN-based multicast multirate protocol:

- Experiment M1 evaluates the speed and accuracy of the layered multicast

protocol's convergence in subscribing the multicast layers. The comparison of PLM (non-AN-based) with ALMA (AN-based) shows that both protocols have a fast convergence mechanism. ALMA can converge to the optimal link utilisation in similar ways as PLM does.

- Experiment M2 exhibits ALMA's behaviour towards sharing the bottleneck link with unresponsive CBR flows. ALMA is capable of adjusting itself when sharing the bottleneck bandwidth with CBR flows. ALMA is a reactive protocol as it grabs the available bandwidth released by the CBR flows quickly.

- Experiment M3 shows that on sharing the link with TCP flows, the ALMA session is not as friendly as PLM. A TCP flow has a smaller share of bandwidth when sharing the link with an ALMA session than when it shares the same network environment with a PLM flow. It can be concluded that ALMA is not a TCP friendly protocol.

- In Experiment M4, the fairness index of the ALMA sessions which share the link with TCP flows using the Droptail queue management algorithm is low. ALMA shows its aggressiveness by always having more bandwidth than the TCP session.

In the following points, we discuss the use of ANs in these experiments and the reason behind ALMA's behaviour:

- Our experimental results suggest that ALMA has proved to be relatively simple in showing how *AN can help in performing pricing mechanisms.* The pricing mechanism has been used to determine layer subscription in this layered multicast congestion control protocol. This protocol treats bandwidth fluctuations and minimises congestion by having an AN-based mechanism to obtain resource availability information.

- The *AN-based active packets* (capsules) have been used in ALMA to regulate the layer subscription of each receiver. The average queue occupancy information has been used to calculate the price and to assign the layer subscription level. This is a simple filtering mechanism and only requires the link to export the function of the load in terms of price to provide flexibility in offering different satisfaction requirements to the users. It can be noted that the result of ALMA's subscription level and its stability is similar to PLM. The mechanism allows the receiver to quickly subscribe the multicast layers and maintain its

subscription stability. In PLM, the decision to drop a layer is made if the packet-pair estimate is less than the available bandwidth, whereas in ALMA the multicast tree will be pruned if the budget is smaller than the average price.

- *AN helps in communicating the queue occupancy* and bandwidth availability to enable multicast protocols to work and share the link with unresponsive flows such as CBR. The layered multicast adaptation achieved similar behaviour compared with the non-AN-based protocol, PLM. Both PLM and ALMA are based on the receiver-driven approach, in which congestion control is performed by the join and leave actions of the receiver, adding and dropping layers. To this extent, PLM uses the packet pair technique to discover the available bandwidth and to give information on which layers to join. PLM assumes that a fair scheduler is employed in every router to enforce fairness. Therefore ALMA's response to different traffic type such as CBR and TCP is different from PLM. However, both PLM and ALMA achieve the desired functionality of layered multicast protocols such as fast convergence and stability, as shown in our experiment results.

- In general, the *AN-based capsules* help in ensuring the *bandwidth probing* and *layer pruning* of the multicast session. The layered multicast adaptation to the available bandwidth using ANs is the main objective of this protocol, and its effect on another protocol's oscillation (such as the TCP flow in this experiment) is not the main concern. Our simulation results using different network topologies and parameters show that ALMA is not a TCP friendly protocol. This is all due to the fact that by design, TCP-friendliness is not a focus of the protocol, particularly when the delay to join/drop from a group is high.

- The ALMA congestion control mechanism uses *AN capsules* to regulate the multicast session. On having a source bandwidth exceeding the bottleneck link capacity, ALMA starve its TCP counterparts. ALMA has been found to have a *higher share of bandwidth* compared with its TCP counterpart. The data-filtering mechanism to overcome *short-term congestion* (by comparing the price with the budget) and *long-term congestion (*pruning decisions) in ALMA, do not take into account other competing traffic such as TCP or CBR flows.

In summary, ANs help in performing multirate multicast congestion control has proved to be useful in distributing the bandwidth. The AN-based mechanism is capable of performing layered multicast protocol functionalities. These multicast functionalities are equivalent with the non-AN-based one in which we have to ensure that all routers are capable of running the new multicast protocol. The advantages of the use of ANs is its flexibility and its capacity in shortening the time to deploy a new multicast protocol.

6.9 Summary

In this work, we conducted a simulation-based performance evaluation of ALMA and compared the AN-based protocol with a classical layered multicast protocol, PLM. The comparison of the two protocols is necessary in order to reveal and analyse them, although the framework of the two algorithms is different, i.e. PLM is based on a fair scheduler mechanism to enforce fairness and a packet-pair probing to get a bandwidth estimate, whereas ALMA is based on an economic supply and demand model using ANs as an enabling technology. The experiments show that ALMA can achieve fast convergence. On sharing with constant bit rate unicast flows, ALMA shows its capacity to share the bandwidth and quickly grab the available bandwidth released by the CBR flows. ALMA is found to have an unfriendly behaviour towards unicast TCP flows in terms of leaving only a small bandwidth share to the competing TCP flows.

Although ALMA was not designed to achieve fast convergence, the experiments show that ALMA can achieve convergence relatively fast, due to its capsule-based congestion indication mechanism. ALMA is a cumulative multicast protocol in which receivers subscribe to the higher layers in subsequent order. The simulation experiments show that ALMA achieves the desired speed for single session data transmission. The study on the mix of ALMA and CBR flows shows the layered multicast protocol's reaction in a dynamic scenario.

The behaviour of the mixture of TCP and ALMA flows sharing the bottleneck link has been studied in this work, in which the TCP-friendliness property of ALMA has been looked into. Some limitations were found in the existing ALMA ns-2 simulator extension, i.e. the scalability, the price function initiation, and the use of different buffer queue management mechanisms in the router. We have to note that ALMA is not a complete protocol and is proposed to use ANs in the price function approach to manage the bandwidth demand and supply. The price function has

been embedded in the ALMA algorithm which has been used throughout these experiments. The protocol works well in performing bandwidth adaptation to the layered multicast sessions.

The experiments show that the layered multicast congestion control protocol can be applied using AN capsules which distribute the subscribe capsules, bandwidth probing, and layer pruning information. The protocol is a precise and simple approach to congestion control. The limitations and many assumptions used in this protocol are a common situation found in newly proposed protocols.

Chapter 7

Conclusion, Contributions, and Future Work

Next we give a brief summary of our research. This will be followed by contributions and suggestions for further work.

7.1 Summary

Throughout this thesis we have made a study of AN-based congestion control protocols, paying special attention to their dynamic behaviour and features. Our reasons for undertaking this study are:

1. The need to evaluate the impact of using AN services to provide a congestion control mechanism.

2. The need to consider the modes of operations in the Internet, i.e. unicast, single rate multicast, and multirate multicast.

3. The need to use AN simulation tools based on the ns-2 network simulator to evaluate the protocols and compare their features with other non-AN based protocols which have an equivalent functionality.

4. The need to consider different kinds of active queue management algorithms at the routers, as well as the inter-protocol fairness of the protocols.

5. The need to discover the impact of the use of ANs services in performing congestion control by comparing the AN-based and non-AN-based protocols.

In the following section, we summarise the key areas (K) and contributions (C) performed of our study. We have studied the advances in ANs deployment, and identified the problem domain of congestion control as the main area of study (K1). We have identified the performance evaluation framework which involved ANs, congestion control, and network simulation as the research methodology (K2).

We reviewed the congestion control, AN-based unicast and multicast congestion control protocols, active queue management, TCP-friendliness, and layered multicast topics to introduce some novelty into the performance evaluation of the unicast and multicast active network-based congestion control protocols (K3). We have presented the evaluation of three different protocols: ACC TCP, AER/NCA, and ALMA.

We conducted a simulation-based performance study to compare the performance of RED and REM on the AER/NCA unicast AN-based protocol (C1). In these experiments, we have extended the ns-2 network simulator with the REM active queue management scheme. ACC TCP has been analysed and tested first to study its microscopic behaviour, to analyse its general characteristics, and then to evaluate its behaviour when using complex network topologies and imposing cross-traffic.

We studied the performance of a single rate AN-based multicast protocol - AER/NCA and its TCP-friendliness (inter-protocol fairness) (C2). We performed a simulation-based performance comparison of an AN-based and non-AN-based layered multicast congestion control protocol (C3). We compared the ALMA protocol with a non-AN-based protocol, obtained the results of ALMA's characteristics, and evaluated the impact of using AN as a layered multicast congestion control mechanism. Table 7.1 summarises the experiment information.

As can be seen from the AN-based protocols presented in Chapters 4 to 6, congestion control mechanisms are proposed in different approaches, i.e. routers notify the endpoints of congestion on ACC TCP, the nominee congestion avoidance mechanism of AER/NCA, and the pricing function for the layered multicast adaptation of ALMA. Hence, ANs promote local adaptation to different congestion control approaches. ANs enable the node to perform computation locally when events happen. This is certainly of benefit to the congestion control mechanism, where it is important to isolate the congestion and notify the end-host of the congestion state. ANs help in performing caching and buffering in the intermediate nodes.

The solution presented by unicast and multicast protocols in previous chapters can be hard-coded into the routers, therefore facilitating immediate deployment of the proposed protocols. However, the AN-based solution is more appealing for long-

Experiment #	**Demonstrates**
ACC TCP	
U1 (Section 4.4)	Microscopic behaviour of ACC TCP
U2 (Section 4.5)	Effect of varying multiple parameters on a stable network
U3 (Section 4.6)	Effect of varying multiple parameters on a network with bursty cross-traffic
AER/NCA	
S1 (Section 5.3)	Behaviour of AER/NCA
S2 (Section 5.4)	Comparison of AN-based with non-AN-based single rate multicast protocols
S3 (Section 5.5)	Effect of different RTT and queue algorithms used in the router to obtain inter-protocol fairness index
ALMA	
M1 (Section 6.3)	Speed and stability of layered multicast convergence (compared with PLM - a non-active network-based protocol)
M2 (Section 6.4)	Effect of layered multicast competing with CBR flows (compared with PLM - a non-AN-based protocol)
M3 (Section 6.5)	Effect of layered multicast competing with TCP flows (compared with PLM - a non-AN-based protocol)
M4 (Section 6.6)	Effect of different RTT used to obtain inter-protocol fairness index

Table 7.1: Summary of experiment information

term deployment, as the extension and upgrade of the protocols can also be used to address other issues in networking.

In general, our simulation experiments have shown that the AN-based protocols can function better than or at least as well as the non-AN-based protocols. In some cases, the AN-based protocols outperformed the non-AN-based one. We have studied different approaches to use active services as congestion control mechanisms. In addition, the simulations to generate the results have been run a significant number of times as we have had to vary many network parameters and repeat the experiments.

Most of the congestion control mechanisms introduced can be implemented in a non-AN-based environment, by means of pre-programming the mechanism in the intermediate routers. However, it requires a lengthy procedure to set the intermediate routers in order to be able to execute the protocols.

From our experience with the protocols, we found that ANs capsules generate communication overhead in most cases. However, the overhead is negligible compared with their help in performing unicast, multicast single rate, and multicast multirate congestion control mechanisms. The unicast protocol uses the feedback congestion control mechanisms to relieve congestion closer to the congested links.

The single rate protocol employs an equation-based nominee congestion control algorithm to determine the rate of the multicast transmission. The multirate multicast protocol utilises the pricing function based on queue occupancy to determine the layered multicast congestion control mechanisms. We confirm that the protocols behave according to their design specifications.

The additional work and modification to ACC TCP, AER/NCA, and ALMA have improved the network efficiency in times of heavy congestion and provided a variety of predictable services to emerging applications. By using the ANs paradigm across a large range of modes of operations and network loads, we have attempted to prevent congestion collapse. A flexible environment and shorter time to deploy new protocols to solve network problems using ANs provide wide opportunities to ensure that robust services can be made available to the users of the Internet.

Our experiments show that TCP is an aggressive protocol compared with the AN-based multicast protocols studied. This is due to the internal mechanism embedded in the AN-based multicast protocols, which consists of nominee congestion algorithm and pricing functions. Our evaluation shows that the AN-based protocols evaluated can share the bandwidth relatively fairly with TCP and UDP flows.

During the course of our experimentation, areas were covered which provided useful insight into the issues surrounding ANs and congestion control. The simulation of many scenarios, topologies, and protocol mixes works well in the three AN-based protocols evaluated. We tested the protocols under a variety of environments. Table 7.2 gives a summary of the evaluation of AN-based protocols using ns-2.

The first part of the work proved that it is feasible to implement ANs to reduce the time to feedback congestion control information on unicast flows. The results of experiments with the REM (Random Early Marking) algorithm into ACC TCP show an improvement in the network compared with RED. In this thesis, we concentrate on the comparison of RED and REM only.

ANs capsules were used in single rate multicast congestion control to perform congestion control and error recovery. In AER/NCA, active error recovery has been performed closer to the source of congestion, demanding less than 1 Round Trip Time (RTT), because the recovery data can be distributed from the nearest node. A solution to the feedback implosion problem and signalling mechanisms for multicast groups are some capabilities of AER/NCA which have been proved to be scalable. We have analysed the decrease in bandwidth utilisation when the number of the multicast flows and competing TCP flows have been scaled up. The capsules help

Characteristics	**ACC TCP**	**AER/NCA**	**ALMA**
Modes of operations	Unicast	Single rate multicast	Multirate multicast
ns-2 extension	Available and extended with REM (C++ and OTcl)	C++ and OTcl	C++ and OTcl
Routing mechanism	Unicast	Sparse mode	Dense mode
Receiver/Sender driven	-	Receiver	Receiver
Extension/Result	Comparison of REM and RED on ACC TCP	Comparison of AER/NCA with PGMCC (only for the inter-protocol fairness)	Comparison of ALMA with PLM
Traffic	TCP, UDP-based (CBR)	TCP, UDP-based (AER/NCA, CBR)	TCP, UDP-based (ALMA, CBR)
Flexibility	Flexible	Relatively flexible, IP routing dependent	Flexible, generic, IP routing independent
Performance	Good, code loading and caching not simulated	Good, code loading and caching not simulated	Good, code loading and caching not simulated
Packet Sizes (Bytes)	1000 (data) and 40 bytes (ACK)	76 (capsules) and 1000 (data)	500 (message capsules)
Active services procedure	Agent/TCP/Active, Agent/TCPSink/Active	Agent/AN/Active, Agent/AN/Rcvr, Agent/AN/CBR	Agent/AN
Experiment and result generation	Integrating rem.cc and experiments	Many experiments by varying parameters	Many experiments
Evaluation of active packets	Evaluating active agents used on TCP source and TCP sinks	Deriving new classes from their ANAgent and install them at each active node. Assumes that code belonging to the cache protocol has already been distributed and available at nodes	Sends a string identifying the Tcl procedure to be run at each node
ANs mechanism	ACC packets: 4-8 bytes characterisation. Active routers generate acknowledgements and filter windows to inform end points of congestion	Modified classifier.cc to allow node entry point to make decision in forwarding to active agent or next hop	Define 'EE' agents (subclass of Agent/message) that send capsules among themselves. Capsule is Agent/message packet. Code is message being executed

Table 7.2: Evaluation of AN-based protocols using ns-2

in performing congestion control efficiently as an improvement to a TCP New Reno traffic for multicast sessions. We have also compared AER/NCA with a prominent example of a single rate multicast congestion control protocol called PGMCC in terms of their inter-protocol fairness. The fairness indices for AER/NCA have also been generated.

Multicast multirate congestion control and bandwidth adaptation mechanism is the third stage of the work in evaluating the unicast and multicast AN-based congestion control protocols. The evaluation of ALMA affirmed that the active application can adapt to available resource variation by comparing the average queue length (indicated as price) as information on congestion in the link (indicated as budget/credit). ALMA has introduced a price function to adapt the use of network bandwidth which is considered a scarce resource. The pricing functions have been used to impose a market-based mechanism. In this work, we have evaluated the behaviour of the AN-based layered multicast adaptation protocol. It has been revealed that ALMA has achieved stability and fast convergence to subscribe to the optimal layer. We have compared ALMA with an example of a traditional layered multicast protocol, PLM. We have also run some experiments to see the impact of increasing the RTT on the inter-protocol fairness index.

All of the experiments on ANs, congestion control and the ns-2 network simulator bring us to the conclusion that everything that can be done with ANs could also be done by having the given functionality built into every router. However, active code helps in assuring decremental deployment of ANs services. On the other hand, the issue of efficiency, resource allocation, safety and security in ANs should be solved, before ANs can be widely deployed in the next generation of the Internet. The advantages of AN are balanced by their disadvantages which can be summarised as trade-offs between cost and flexibility. Network devices must be able to support appropriate AN transport protocols, execution environments, device service interface and resources to run active code.

To summarise, this thesis has investigated an important aspect of computer networks, that of employing ANs in congestion control. The extensive study has demonstrated that ANs can in fact be deployed in congestion control protocols.

7.2 Contributions

The contributions of our research work can be summarised by the following points:

1. A review of the congestion control mechanisms for AN-based unicast, multicast

single rate, and multicast multirate streams.

2. A thorough review and evaluation of existing AN-based congestion control protocols available in ns-2.

3. Performance evaluation and improvement of the AN-based congestion control protocol in unicast, single rate multicast and layered multicast protocols, which can be summarised in the following subsections.

7.2.1 In Unicast Protocols

- Integration and implementation of REM in ACC TCP, as well as the evaluation and comparison with the results of RED queue management algorithms.

- Dynamic behaviour analysis of REM active queue management scheme which has been successfully integrated into ACC TCP. This new scheme has shown its constant desirable behaviour on ACC TCP, which promotes stability and low buffer occupancy by marking packets. The comparison of RED and REM mechanisms shows that REM exhibits an improved performance on a high bandwidth-delay product regime on ACC TCP.

7.2.2 In Multicast Single Rate Protocols

- Evaluation of AER/NCA, which is a single rate reliable multicast protocol with nominee congestion control approach presented as an active service.

- Evaluation of TCP-friendliness (inter-protocol fairness) as a desirable feature for proposed multicast congestion control protocols. We have shown the inter-protocol fairness of AER/NCA protocols and analysed the bandwidth usage in the situation of the existence of lossy links and the enabling of active repair routers. The AER/NCA protocol can share the bandwidth fairly with the TCP flows, perform the distributed local recovery, and limit the retransmission of packets.

- Performance comparison of inter-protocol fairness of AER/NCA with that of PGMCC (a traditional single rate multicast protocol).

- The characterisation that AER/NCA is fair towards TCP using fairness indices. The AER/NCA shares the bandwidth fairly with TCP in the situation in which RED or Droptail queue management algorithms is used in the router.

7.2.3 In Multicast Multirate Rate Protocols

- The layered multicast adaptation protocol based on active applications have been analysed in this work. Independent performance evaluation of the ALMA protocol has been performed. Its pricing-based congestion control mechanism for layered multicast has been looked into. We found that the AN-based approach to layered multicast is feasible. The market-based approach towards the use of bandwidth as a scarce resource has been proved to be a good approach.

- We have compared ALMA with PLM (an end-to-end fair scheduler-based layered multicast protocol) in terms of their stability, speed, and layer convergence. We infer that the ANs approach used in ALMA proved to be suitable for a multiservice interaction study and a heterogeneous receivers environment. The results of ALMA's behaviour when competing with CBR flows shows that ALMA reacts differently from PLM. ALMA adjusts itself when the CBR flows started, but the CBR flow do not share the bandwidth fairly as in PLM experiment results (which uses fair scheduler). ALMA is a reactive protocol which can grab the available bandwidth released by the CBR flows.

- ALMA sessions have been found to starve the competing TCP flows. Our result shows that TCP flows get only a small share of bandwidth. TCP flows oscillate more on sharing the link with an ALMA session than when they share the link with PLM sessions. All experiments shows ALMA unfriendliness to TCP.

- Layered multicast transmission is necessary to support the heterogeneous nature of receivers. Our independent evaluation of the ALMA protocol shows that this protocol's mechanism for the receiver to join multicast groups (subscription) and pruning of the multicast tree works well in order to control congestion. The protocol uses a price function which is a step further in controlling congestion in a non-cooperative environment. The price function will enable the network manager to shape the distribution of network resources.

- We have observed the characterisation of ALMA's fairness towards TCP using fairness indices. The fairness index of some ALMA sessions sharing a bottleneck link with some TCP flows using a Droptail queue management algorithm in the router is low and the link is not fairly shared. ALMA sessions always

get a larger share of bandwidth than the TCP flows. The experiment results show that ALMA is not a TCP friendly protocol.

7.3 Future Work

There are many ways to extend the work presented in this thesis. The following is a list of the most appealing areas in *unicast* congestion control protocol:

1. In this research, the parameter tuning in RED and REM has not been performed. It is desirable to understand the effect of active queue management parameter settings more thoroughly in the AN-based congestion control protocols. This can be performed by taking into account the complexity of RED and REM parameter settings and the nature of short and long-lived traffic (web-like TCP traffic).

2. We are also interested in considering other queue management schemes on ACC TCP, such as Droptail, Blue, and Weighted Fair Queuing (WFQ), in order to see their impact on different bandwidth-delay product AN-based networks. These mechanisms might have a different response when used in an AN-based environment. In addition, it will be useful to study more about their impact on the performance of the AN-based congestion control protocols.

For the *single rate multicast* congestion control protocol, the work can be extended as follows:

1. The effect of RED and REM active queue management on AER/NCA needs to be studied. The power of a new queue management algorithm such as REM can improve the overall performance of the network. Integrating and running REM on the AN-based single rate multicast protocol would generate performance benefits. Specifically, we could also perform RED and REM parameter tuning in relation to the single rate multicast AN-based environment.

2. The use of AN nodes to perform local recovery can be investigated further. It will be beneficial to establish how many AN nodes are necessary and where to place them efficiently in more detail to achieve good network performance.

3. Further work on identifying pathological configurations in which AER/NCA does not behave as it should do will also contribute to the literature on active

networks and congestion control. In this case, we will also be able to explore how AN functionalities in AER/NCA can be improved.

4. We found that prototyping using the common execution environment such as ANTS is also an interesting research avenue. The AER/NCA prototype in ANTS should be used to evaluate the AN-based single rate congestion control protocol further.

5. Further work on the fairness comparison of AER/NCA and PGMCC in relation to the selection of the representative, network with high packet loss rate, and uncorrelated losses would be an interesting subject of near future work.

For the *multirate multicast* congestion control suggested further work is as follows:

1. In this research, we use the ALMA standard pricing function. Establishing the way to improve ALMA to achieve global optimum and reactivity will be of interest. In attempting to do this, first we can experiment heuristically with different pricing and utility functions to find a better or optimum solution. This will require extensive computing power to try every possible path.

2. Pricing mechanism in ALMA is a fine example of the use of a market-based approach to manage bandwidth as a scarce resource in layered multicast. On the other hand, the use of the game theory approach in congestion control has attracted a considerable attention of the research community. Finding a better pricing function as a good congestion indication using a game theory approach in this AN-based layered multicast congestion control protocol will be very useful.

3. The packet-pair mechanism has been used in PLM to estimate the available bandwidth of layered multicast congestion control protocol. This mechanism can be implemented using ANs to explore other alternatives to perform layered multicast adaptation.

4. Devising a better congestion control mechanism is another interesting research avenue. This can be done by finding an optimum pricing and utility function in order to achieve more efficient, stable, simple, with less loss, and fair layered multicast protocols using ANs.

References

[1] D. S. Alexander. *Alien: A Generalized Computing Model for Active Networks.* PhD thesis, University of Pennsylvania, September 1998.

[2] S. Alexander, C. Gunter, and D. Wetherall. Active Network Encapsulation Protocol, August 1997. http://www/cis.upenn.edu/ switchware/ANEP.

[3] E. Altman, C. Barakat, E. Laborde, P. Brown, and D. Collange. Fairness Analysis of TCP/IP. In *Proceedings of 39th IEEE Conference on Decision and Control*, pages 61–66, Sydney, Australia, 12-15 December 2000.

[4] E. Amir, S. McCanne, and R. Katz. An Active Service Framework and its Application to Real-Time Multimedia Transcoding. In *Proceedings of ACM Symposium on Communications Architectures and Protocols (SIGCOMM 1998)*, Vancouver, Canada, September 1998.

[5] M. Arpaci and J. Copeland. An Adaptive Queue Management Method for Congestion Avoidance in TCP/IP Networks. In *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM'00)*, pages 309–315, San Francisco, USA, 27 November - 1 December 2000.

[6] S. Athuraliya, V. H. Li, S. Low, and Q. Yin. REM: Active Queue Management. *IEEE Network*, 15(3):48–53, May/June 2001. http://netlab.caltech.edu.

[7] S. Athuraliya and S. H. Low. Simulation Comparison of RED and REM. In *Proceedings of ICON 2000*, pages 68–72, Singapore, October 2000.

[8] A. Azcorra, M. Calderon, M. Sedano, and J. Moreno. Multicast Congestion Control for Active Network Services. *European Transaction on Telecommunications*, 10(3):1–8, May/June 1999.

[9] B. Bai, J. Harms, and Y. Li. Active Hierarchical Congestion Control for Large-scale Multicast. In M. Obaidat, F. Davoli, I. Onyuksel, and R. Bola, editors, *Proceedings of International Symposium on Performance Evaluation of Computer and Telecommunication Systems 2002 (SPECTS2002)*, pages 251–262, San Diego, USA, 14-18 July 2002. SCS.

[10] S. Bajaj, L. Breslau, D. Estrin, and K. Fall. Improving Simulation for Network Research. Technical Report 99-702, USC Computer Science Department, March 1999.

[11] A. Banchs, W. Effelsberg, C. Tschudin, and V. Turau. Multicasting Multimedia Streams with Active Networks. In *Proceedings of Local Computer Network '98*, pages 150–159, 1998.

[12] J. Banks, J. Carson, and D. Nicol. *Discrete-Event System Simulation.* Prentice Hall International, 2001.

[13] J.D. Bansemer and M. Y. Eltoweissy. On Performance Metrics of IP Multicast Routing. In *Proceedings of IEEE International Symposium on Parallel Architectures, Algorithms, and Networks 2000 (I-SPAN 2000)*, pages 282–291, Dallas, USA, 7-9 December 2000.

[14] C. Barakat. TCP/IP Modeling and Validation. *IEEE Network*, 15(3):38–47, May/June 2001.

[15] A. Barone, P. Chirco, G. Di Fatta, and G. Lo Re. A Management Architecture for Active Networks. In *Proceedings of Fourth Annual International Workshopon Active Middleware Services*, pages 41–48, 2002.

[16] B. S. Bennet. *Simulation Fundamentals.* Prentice Hall International, 1995.

[17] S. Benson, B. Braden, and L. Ricciulli. Introduction to the Active Networks Backbone. Technical Report of ISI, June 2000. http://www.isi.edu/abone.

[18] S. Bhattacharjee. *Active Networks: Architectures, Composition, and Applications.* PhD thesis, College of Computing, Georgia Institute of Technology, July 1999.

[19] S. Bhattacharjee, K. Calvert, and E. Zegura. On Active Networking and Congestion. Technical Report GIT-CC-96/02, College of Computing, Georgia Institute of Technology, 1996. http://www.cc.gatech.edu/projects/canes/publications.html.

[20] S. Bhattacharjee, K. Calvert, and E. Zegura. Active Networking and End-to-End Argument. In *Proceedings of IEEE International Conference on Network Protocols (ICNP 1997)*, pages 220–228, Atlanta, USA, 28-31 October 1997.

[21] S. Bhattacharjee, K. Calvert, and E. Zegura. Reasoning About Active Network Protocols. In *Proceedings of IEEE International Conference on Network Protocols (ICNP 1998)*, pages 31–40, Texas, October 1998. Georgia Institute of Technology.

[22] E. Bolker and Y. Ding. On the Performance Impact of Fair Share Scheduling. Technical report, BMC Software, Inc, 2000.

[23] T. Bonald and L. Massoulie. Impact of Fairness on Internet Performance. In *Proceedings of ACM Sigmetrics International Conference on Measurement and Modelling of Computer Systems 2001*, pages 82–91, Cambridge, MA, USA, June 2001.

[24] V. Bose, D. Wetherall, and J. Guttag. Next Century Challenges: RadioActive Networks. In *Proceedings of ACM/IEEE Conference on Mobile Communications '99*, Seattle, WA, August 1999.

[25] R. Boutaba and A. Polyrakis. Projecting Advanced Enterprise Network and Service Management to Active Networks. *IEEE Network*, 16(1):28–33, January/February 2002.

[26] L. Brakmo. *End-to-End Congestion Detection and Avoidance.* PhD thesis, University of Arizona, USA, 1994.

[27] L. Brakmo and L. Peterson. Experiences with Network Simulation. In *Proceedings of ACM International Conference on Measurement and Modeling Computer Systems (SIGMETRICS 1996)*, pages 80–90, Philadelphia, USA, 1996.

[28] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, and H. Yu Y. Xu. Advances in Network Simulation. *IEEE Computer*, 33(5):59–67, May 2000.

[29] I. Brown. *End to End Security in Active Networks.* PhD thesis, Computer Science, University of London, UK, 2001.

[30] M. Brunner and B. Plattner. Management of Active Networks. In *Proceedings of ICC Workshop on Active Networking and Programmable Networks*, pages 1–12, Atlanta, June 1998.

[31] M. Brunner and R. Stadler. The Impact of Active Networking Technology on Service Management in a Telecom Environment. In *6th International Symposium on Integrated Network Management*, pages 385–400, Boston, MA, 24-30 May 1999. IFIP/IEEE.

[32] J. Byers, M. Luby, and M. Mitzenmacher. Fine-grained layered multicast. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM 2001)*, volume 2, pages 1143–1151, Anchorage - Alaska, 22-26 July 2001. IEEE.

[33] M. Calderon, M. Sedano, A. Azcorra, and C. Alonso. Active Network Support for Multicast Applications. *IEEE Network*, 12(3):46–52, May/June 1998.

[34] K. Calvert. Architectural Framework for Active Networks Version 0.9. Technical report, Active Network Working Group - DARPA, August 1998.

[35] K. Calvert, S. Bhattacharjee, and E. Zegura. Directions in Active Networks. *IEEE Communications*, 36(10):72–78, October 1998. http://www.cc.gatech.edu/projects/canes/publications.html.

[36] A. Campbell and J. Vicente. A Survey of Programmable Networks. *ACM Computer Communications Review*, 29(2):7–23, April 1999.

[37] A. Capone and F. Martignon. Bandwidth Estimates in the TCP Congestion Control Scheme. In S. Palazo, editor, *Proceedings of IWDC 2001*, volume 2170 of *Lecture Notes in Computer Science*, pages 614–626. Springer Verlag, 2001.

[38] K. W. Chin. *An Investigation into the Application of Active Networks to Mobile Computing.* PhD thesis, School of Computing, Curtin University of Technology, Australia, March 2000.

[39] J. Chung and M. Claypool. NS by Example. http://nile.wpi.edu/NS, November 2000.

[40] R. Cocchi, S. Shenker, D. Estrin, and L. Zhang. Pricing in Computer Networks: Motivation, Formulation, and Example. *IEEE/ACM Transaction on Networking*, 1(6):614–627, December 1999.

[41] J. Cowie, D. Nicol, and A. Ogielski. Modeling the Global Internet. *Computing in Science and Engineering*, 1(1):42–50, January 1999.

[42] D. Decasper and B. Plattner. A Scalable High-Performance Active Network Node. *IEEE Network*, 13(1):8–9, January-February 1999. ETH Zurich.

[43] J. Van der Merwe, S. Rooney, I. Leslie, and S. Crosby. The Tempest - A Practival Framework for Network Programmability. *IEEE Network*, 12(3):1–14, May/June 1998.

[44] A. Durand. Deploying IPv6. *IEEE Internet Computing*, 5(1):79–81, January/February 2001.

[45] S. Eichert, O. Ertugay, D. Nesset, and S. Vobbilisetty. Commercially Viable Active Networking. *ACM Sigops Operating Sytems and Review*, 58(1):8–22, January 2002.

[46] D. Estrin, M. Handley, and J. Heidemann. Network Visualisation with the VINT Network Animator Nam. Technical Report 99-703, USC Computer Science Department, March 1999.

[47] T. Faber. ACC: Using Active Networking to Enhance Feedback Congestion Control Mechanism. *IEEE Network*, 12(3):61–65, May/June 1998. http://www.isi.edu/ faber/pubs.html/active.ps.

[48] T. Faber. An Implementation of the Reliable Data Protocol for Active Networking. Technical report, USC/ISI, 1999. http://www.isi.edu/active-signal/ACC/docs/html/rdp/.

[49] T. Faber. Experience with Active Congestion Control. In *Proceedings of the DARPA Active Networks Conference and Exposition (DANCE'02)*, pages 132–142, San Francisco, CA, USA, 29-30 May 2002.

[50] K. Fall and S. Floyd. Simulation-based Comparisons of Tahoe, Reno, and SACK TCP. *Computer Communication Review*, 26(3):5–21, July 1996.

[51] K. Fall, K. Varadhan, and S. Floyd. ns Notes and Documentation. UCB/LBNL/VINT, July 1999. Software and Documentation available at http://www.isi.edu/nsnam/ns.

[52] W. Feng. *Improving Internet Congestion Control and Queue Management Algorithms*. PhD thesis, Computer Science and Engineering, The University of Michigan, 1999.

[53] A. Fernando, B. Kummerfeld, A. Fekete, and M. Hitchens. A New Dynamic Architecture for an Active Network. In *Proceedings of IEEE Open Architecture (Openarch 2000)*, pages 121–127, Tel Aviv, Israel, 26-30 March 2000.

[54] V. Firoiu and M. Borden. A Study of Active Queue Management for Congestion Control. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM 2000)*, volume 3, pages 1435–1444, Tel Aviv, Israel, 26-30 March 2000.

[55] S. Floyd. TCP and Explicit Congestion Notification. *ACM Computer and Communications Review*, 24(5):10–23, October 1994.

[56] S. Floyd. Discussion on Setting RED Parameters. http://www.aciri.org/floyd/REDparameter.txt, November 1997.

[57] S. Floyd. Congestion Control Principles. Request for Comments 2914 Internet Society, September 2000.

[58] S. Floyd. A Report on Recent Developments in TCP Congestion Control. *IEEE Communication*, 39(4):84–90, April 2001.

[59] S. Floyd. Network Dynamics and Scalable Congestion Control. Workshop on UltraLarge Networks, November 2001.

[60] S. Floyd and K. Fall. Promoting the Use of End-to-End Congestion Control in the Internet. *IEEE/ACM Transaction on Networking*, 7(4):458–472, August 1999.

[61] S. Floyd, M. Handley, and J. Padhye. A Comparison of Equation-Based and AIMD Congestion Control. Technical report, ACIRI, May 2000. http://www.aciri.org/floyd.

[62] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-Based Congestion Control for Unicast Applications. In *Proceedings of ACM Symposium on Communications Architectures and Protocols (SIGCOMM 2000)*, pages 1–12, Stockholm, Sweden, May 2000.

[63] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[64] S. Floyd and E. Kohler. Internet Research Needs Better Models. In *Proceedings of First Workhop on Hot Topics in Network (Hotnet-I)*, pages 1–15, Princeton, USA, 28-29 October 2002.

[65] S. Floyd and V. Paxson. Difficulties in Simulating the Internet. *IEEE/ACM Transaction on Networking*, 9(4):392–403, August 2001.

[66] L. Garcia and I. Widjaja. *Communication Networks: Fundamental Concepts and Key Architectures*. McGraw-Hill, 2001.

[67] J. Gelas and L. Lefevre. Tamanoir: A High Performance Active Network Framework. In S. Hariri, C. A. Lee, and C. S. Raghavendra, editors, *Proceedings of Active Middleware Services, Ninth IEEE International Symposium on High Performance Distributed Computing*, pages 105–114, Pittsburgh, Pennsylvania, USA, August 2000. Kluwer Academic Publishers.

[68] P. Gevros, J. Crowcroft, P. Kirstein, and S. Bhatti. Congestion Control Mechanism and the Best Effort Service Model. *IEEE Network*, 15(3):16–26, May/June 2001.

[69] A. Ghonainy. New Generation Internet and Evolution Towards Active and Programmable Networks (Survey). In *Proceedings of IEEE 16th National Radio Science Conference, NRSC'99*, pages INV3/1–INV3/14, Cairo, Egypt, February 1999. IEEE.

[70] R.J. Gibbens and F. P. Kelly. Resource Pricing and the Evolution of Congestion Control. *Automatica*, 35:1969–1985, 1999.

[71] R. Gopalakrishnan, J. Griffioen, G. Hjalmtysson, and C. Sreenan. Stability and Fairness Issues in Layered Multicast. In *Proceedings of IEEE International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'99)*, pages 1–14, 1999.

[72] L. Guan, M. E. Woodward, and I. U. Awan. An Analytical Queuing Model for the Performance Evaluation using Threshold Congestion Control. In *Proceedings of PGNET '03*, pages 1–6, Liverpool, UK, 16-17 June 2003.

[73] T. Gyires. Using Active Networking for Congestion Control in High-Speed Networks with Self-similar Traffic. In *Proceedings of IEEE International Conference on System, Man, and Cybernetics*, volume 1, pages 405–410, 2000.

[74] M. Handley and S. Floyd. Strawman Specification for TCP Friendly (Reliable) Multicast Congestion Control (TFMCC). Technical report, ISI/LBNL, December 1998. http://www.aciri.org/floyd/papers.html.

[75] G. Hasegawa, M. Murata, and H. Miyahara. Fairness and Stability of Congestion Control Mechanisms of TCP. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM 1999)*, pages 1329–1336, New York, USA, 21-25 March 1999. IEEE Computer and Communication Society.

[76] M. Hassan and R. Jain. TCP Performance in Future Networking Environments. *IEEE Communications*, page 51, April 2001.

[77] J. Heidemann. Expanding Confidence in Network Simulations. *IEEE Network*, 15(5):58–63, September/October 2001.

[78] M. Hicks, J. Moore, D. S. Alexander, C. A. Gunther, and S. M. Netles. Environments for Active Networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM 1999)*, pages 1124–1133, March 1999.

[79] C. Hollot, V. Mishra, D. Towsley, and W. Bong. On Designing Improved Controllers for AQM Routers Supporting TCP Flow. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM 2001)*, volume 3, pages 1726–1734, Anchorage, Alaska, 22-26 July 2001.

[80] G. Iannaccone, C. Brandauer, T. Ziegler, C. Diot, S. Fdida, and M. May. Aggregrate Traffic Performance with Active Queue Management and Drop from Tail. *Computer Communication Review*, 31(3):1–20, July 2001.

[81] V. Jacobson. Congestion Avoidance and Control. In *Proceedings of Symposium on Communications Architecture and Protocols (SIGCOMM 1988)*, pages 314–329, Stanford, CA, 16-18 August 1988.

[82] R. Jain. Congestion Control in Computer Networks: Issues and Trends. *IEEE Network*, 4(3):24–30, May 1990.

[83] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling.* John Wiley and sons, 1991.

[84] R. Jain, A. Durresi, and G. Babic. Throughput Fairness Index: an Explanation. Technical Report 0045, ATM Forum, 1999.

[85] S. Kang, H. Youn, Y. Lee, D. Lee, and M. Kim. The Active Traffic Control Mechanism for Layered Multimedia Multicast in Active Network. In M. Obaidat, F. Davoli, and M. Marsan, editors, *Proceedings of 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (SPECTS 2000)*, pages 325–332, Vancouver, Canada, July 2000.

[86] M. Kara and M. Rahin. Towards a Framework for Performance Evaluation of TCP Behaviour over ATM Networks. In *Proceedings of the 13th IFIP International Conference on Computer Communication (ICCC'97)*, pages 49–60, Cannes, France, 19-21November 1997.

[87] R. Karp, C. Papadimitriou, E. Koutsospias, and S. Shenker. Optimization Problem in Congestion Control. In *Proceedings of 41st Annual Symposium on Foundations of Computer Science*, pages 66–74, California, USA, 2000. IEEE.

[88] S. Kasera, S. Bhattacharyya, M. Keaton, D. Kiwior, J. Kurose, and D. Towsley. Scalable Fair Reliable Multicast Using Active Services. *IEEE Network - Special Issue on Multicast*, 14(1):48–57, January 2000. PANAMA Project Software available at http://www.tascnets.com/panama.

[89] R. Keller, S. Choi, M. Dasen, D. Decasper, G. Fankhauser, and B. Plattner. An Active Router Architecture for Multicast Video Distribution. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM 2000)*, volume 3, pages 1137–1146, Tel Aviv, Israel, 2000.

[90] S. Keshav. REAL: A Network Simulator. Technical Report 88/472, University of California, Berkeley, 1988.

[91] I. Khayat and G. Leduc. A Stable and Flexible TCP-Friendly Congestion Control Protocol for Layered Multicast Transmission. In D. Shepherd, editor, *Proceedings of IDMS'01*, Springer-Verlag LNCS 2158, pages 154–167, Lancaster, UK, October 2001.

[92] D. Kiwior and S. Zabele. Active Resource Allocation in Active Networks. *IEEE Journal on Selected Areas in Communications*, 19(3):452–458, March 2001.

[93] H. Koubaa and E. Fleury. Active Multicast Core Migration. In *Proceedings of International Conference on Networks (ICON 2000)*, pages 346–350, Singapore, July 2000.

[94] A. Kulkarni, G. Minden, R. Hill, Y. Wijata, and A. Gopinath. Implementation of a Prototype Active Network. In *Proceedings of IEEE Conference on Open Architecture (Openarch 1998)*, pages 130–142, San Francisco, USA, 12-14 April 1998.

[95] A. Kumar. Comparative Performance Analysis of Versions of TCP in a Local Area Network with a Lossy Link. *IEEE/ACM Transactions on Networking*, 6(4):485–498, August 1998.

[96] L. Lafevre, C. Pham, P. Primet, B. Tourancheau, B. Gaidioz, J. Gelas, and M. Maimour. Active Networking Support for the GRID. In I.W. Marshall, editor, *Proceedings of the IWAN 2001*, LCNS 2207, pages 16–33, Pennsylvania, USA, October 2001. Springer Verlag.

[97] T. Lakhsman and U. Madhow. The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss. *IEEE/ACM Transactions on Networking*, 5(3):336–350, June 1997.

[98] K. Lan and J. Heidemann. Rapid Model Parameterization from Traffic Measurement. Technical Report ISI-TR-561, USC Information Sciences Institute, 1 August 2002.

[99] D. Lapsley and S. Low. Random Early Marking for Internet Congestion Control. In *Proceedings of Global Telecommunications Conference (GLOBECOM 1999)*, pages 1747–1752, Rio De Janeiro, Brazil, 5-9 December 1999.

[100] A. M. Law and W.D. Kelton. *Simulation Modeling and Analysis*. Mc Graw-Hill, 2000.

[101] U. Legedza, D. Wetherall, and J. Guttag. Improving the Performance of Distributed Applications Using Active Networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM 1998)*, volume 2, pages 590–599, San Francisco, USA, 29 March - 2 April 1998.

[102] A. Legout and E. W. Biersack. PLM: Fast Convergence for Cumulative Layered Multicast Transmission Schemes. In *Proceedings of the ACM International Conference on Modeling and Measurement of Computer Systems (SIGMETRICS 2000)*, pages 13–22, Santa Clara, California, USA, June 2000. ACM.

[103] A. Legout, J. Nonnenmacher, and E.W. Biersack. Bandwidth-allocation Policies for Unicast and Multicast Flows. *IEEE/ACM Transactions on Networking*, 9(4):464–478, August 2001.

[104] L. Lehman, S. Garland, and D. Tennenhouse. Active Reliable Multicast. In *Proceedings of the Conference on Computer Communications (INFOCOM 1998)*, pages 581–589. IEEE, March 1998. San Francisco, USA.

[105] N. Liu. Adaptive Congestion Control and Pricing in Future Active IP Networks. Technical Report CCSR-2000-0525, Hitachi Project Centre for Communication Systems Research, 25 May 2000.

[106] S. Low. A Duality Model of TCP Flow Controls. In *Proceedings of ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management*, pages 1–24. International Teletraffic Congress, September 2000.

[107] S. H. Low, F. Paganini, and J. C. Doyle. Internet Congestion Control. *IEEE Control Systems Magazine*, 22(1):28–43, February 2002.

[108] M. Maimour and C. Pham. An Active Reliable Multicast Framework for the Grid. In *Proceedings of the International Conference on Computational Science (ICCS 2002)*, pages 588–597, Amsterdam, The Netherlands, 21-24 April 2002. LCNS 2329/30/31.

[109] M. Maimour and C. Pham. Dynamic Replier Active Reliable Multicast (DyRAM). In *Proceedings of 7th IEEE Symposium on Computers and Communications (ISCC 2002)*, pages 275–282, Taormina, Italy, 1-4 July 2002.

[110] A. Mankin, A. Romanow, S. Bradner, and V. Paxson. IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols. Technical Report Request For Comment 2357, IETF, June 1998.

[111] R. Maresca, M. D'Arienzo, M. Esposito, S. Romano, and G. Ventre. An Active Network Approach to Virtual Private Networks. In *Proceedings of Seventh International Symposium on Computer and Communications (ISCC 2002)*, pages 502–507, Taormina, Italy, 2002.

[112] S. Mascolo. Smith's Predictor for Congestion Control in TCP Internet Protocol. In *Proceedings of the IEEE American Control Conference 1999*, volume 6, San Diego, USA, 2-4 June 1999.

[113] N. F. Maxemchuck. Active Networks in Telephony. In *Proceedings of the IEEE Conference on Open Architecture (Openarch 1999)*, pages 2–8, New York, USA, 26-27 March 1999.

[114] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven Layered Multicast. In *Proceedings of Symposium on Communications Architectures and Protocols (SIGCOMM 1996)*, pages 1–14, Palo Alto, California, Augustus 1996. ACM.

[115] S. Merugu, E. Zegura S. Bhattacharejee, and K. Calvert. Bowman: A Node OS for Active Networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM 2000)*, pages 1127–1136, Tel Aviv, Israel, 26-30 March 2000.

[116] S. Munir. Active Networks - A Survey. Technical report, CIS-Ohio State University, 1997. http://www.cis.ohio-state.edu/ jain/cis788-97/activenet.

[117] D. Murphy. Building an Active Node on the Internet. Master's thesis, Electrical Engineering and Computer Science, MIT, May 1997.

[118] S. Murphy, E. Lewin, R. Puga, R. Watson, , and R. Yee. Strong Security for Active Networks. In *Proceedings of the IEEE Conference on Open Architecture (Openarch 2001)*, pages 1–8, Anchorage, USA, 2001.

[119] M. Nabeshima. Improving the Performance of Active Buffer Management with Per-Flow Information. *IEEE Communications Letters*, 6(7):306–308, July 2002.

[120] K. Najafi. *Modeling, Routing, and Architecture in Active Networks.* PhD thesis, Department of Electrical and Computer Engineering, The University of Toronto, Canada, 2001.

[121] E. Nygren, S. Galard, and M. Kanshoek. PAN: A High Performance Active Networks Node Supporting Multiple Mobile Code System. In *Proceedings of IEEE Conference on Open Architecture (Openarch 1999)*, pages 78–79, New York, USA, March 1999.

[122] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. In *Proceedings of ACM Symposium on Communications Architectures and Protocols (SIGCOMM 1998)*, September 1998. Vancouver, Canada.

[123] C. H. Papadimitriou. Algorithms, games, and the internet. In *Proceedings of Symposium on Theory of Computing (STOC '01)*, pages 749–753, Crete - Greece, July 2001. ACM.

[124] S. Park and K. Miller. Random Number Generator: Good Ones Are Hard to Find. *Communications of the ACM*, 31(10):1192–1201, October 1998.

[125] C. Parsa and J. Garcia-Luna-Aceves. Improving TCP Congestion Control over Internet with Heterogenous Transmission Media. In *Proceedings of IEEE Conference on Network Protocols (ICNP 1999)*, pages 213–221. IEEE, October 1999.

[126] Parallel Simulation Environment for Complex System. http://www.pcl.cs.ucl.edu/projects/parsec, January 1998.

[127] K. Pawlikowski, H. J. Jeong, and J. R. Lee. On Credibility of Simulation Studies of Telecommunication Network. *IEEE Communications Magazine*, 40(1):132–139, January 2002.

[128] K. Psounis. Active Networks: Application, Security, Safety, and Architecture. *IEEE Communications Survey*, pages 1–12, March 1999.

[129] S. Puangpronpitag, R. Boyle, and K. Djemame. Performance Evaluation of Multi-rate Multicast Congestion Control Protocols: FLID-DL vs PLM. In *Proceedings of Symposium on Performance Evaluation of Communications and Telecommunications Systems (SPECTS 2003)*, pages 1–8, Montreal, Canada, July 2003.

[130] H. K. Pung and N. Bajrach. An Active Congestion Control in ATM Multicast. In *IEEE ATM Workshop 1999*, pages 397–402. National University of Singapore, May 1999.

[131] V. Ramachandran, R. Pandey, and S. Chan. Fair Resource Allocation in Active Network. In *Proceedings of IEEE International Conference on Communications and Networks (ICCN 2000)*, pages 468–475, Las Vegas, USA, 16-18 October 2000.

[132] K. Ramakrishan and S. Floyd. A Proposal to add Explicit Congestion Notification (ECN) to IP. RFC-2481, January 1999.

[133] D. M Rao and P. A. Wilsey. Modeling and Simulation of Active Network. In *Proceedings of 34th Annual Simulation Symposium*, pages 177–184, Seattle, USA, 22-26 April 2001.

[134] L. Rizzo. Dummynet: A Simple Approach to the Evaluation on Network Protocols. *ACM Computer Communication Review*, January 1997.

[135] L. Rizzo. PGMCC: A TCP-friendly Single Rate Multicast Congestion Control Scheme. In *Proceedings of ACM Symposium on Communications Architectures and Protocols (SIGCOMM 2000)*, ACM Computer Communication Review, pages 17–28, October 2000. Sweden.

[136] R. Rodriguez and P. Merino. Modeling and Simulation of Active Network Protocol. In *Proceedings of IEEE International Workshop on Internet (IWI 2000)*, Taiwan, April 2000. University of Malaga, Spain. http://www.lcc.uma.es/ gisum/active.html.

[137] D. Rubenstein, J. Kurose, and D. Towsley. The Impact of Multicast Layering on Network Fairness. *IEEE/ACM Transaction on Networking*, 10(2):169–182, April 2002.

[138] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson. TCP Congestion Control with a Misbehaving Receiver. *ACM Computer Communications Review*, 29(5):51–59, October 1999.

[139] S. Schmid, L. Finney, A. Scott, and W. Shepherd. Component-based Active Network Architecture. In *Proceedings of Sixth IEEE International Conference on Computer and Communications (ISCC 2001)*, pages 114–121, Taormina, Italy, 2001.

[140] B. Schwartz, A. Jackson, W. Stroyen, and C. Partridge. Smart Packet for Active Networks. In *Proceedings of the IEEE Conference on Open Architecture (Openarch 1999)*, pages 90–97, New York, USA, 26-27 March 1999.

[141] K. Seada and A. Helmy. Fairness Analysis of Multicast Congestion Control: A Case Study on PGMCC. Technical Report 01-743, Computer Science Department, University of Southern California, April 2001.

[142] K. Seada and A. Helmy. Fairness Analysis of Multicast Congestion Control: A Case Study on PGMCC. In *Proceedings of Globecom (Global Communications Conference) 2002*, volume 3, pages 2614–2618, 17-21 November 2002.

[143] M. Sedano, A. Azcorra, and M. Calderon. Performance of Active Multicast Congestion Control. In H. Yasuda, editor, *Proceedings of International Working Conference on Active Networks (IWAN 2000)*, volume 1942 of *LCNS*, pages 145–156, Tokyo, Japan, October 2000. Springer Verlag.

[144] R. Sharma, S. Keshav, M. Wu, and L. Wu. Environments for Active Networks. In *Proceedings of IEEE International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 97)*, pages 77–84, St. Louis, USA, 19-21 May 1997.

[145] S. Shenker. Making Greed Work in Networks: A Game-Theoretic Analysis of Switch Service Disciplines. *IEEE/ACM Transaction on Networking*, 3(6):819–831, December 1995.

[146] S. Shenker, D. Clark, D. Estrin, and S. Herzog. Pricing in Computer Networks: Reshaping the Research Agenda. *ACM Computer Communication Review*, 26(2):19–43, April 1996.

[147] S. Shyne, A. Hovak, and J. Riolo. Using Active Networking to Thwart Distributed Denial of Service Attacks. In *Proceedings of IEEE Aerospace Conference*, volume 3, pages 3/1103–3/1108, Big Sky, MT, USA, 10-17 March 2001 2001.

[148] S. Da Silva, Y. Yemini, and D. Florissi. The NetScript Active Network System. *IEEE Journal on Selected Areas in Communication*, 19(3):538–551, March 2001.

[149] D. Sisalem and A. Wolisz. MLDA: A TCP-friendly Congestion Control Framework for Heterogenous Multicast Environments. In *Proceedings of IEEE Eighth International Workshop on Quality of Service (IWQOS) 2000*, pages 65–74, 5-7 June 2000 2000.

[150] J. Smith, K. Calvert, S. Murphy, H. Orman, and L. Peterson. Activating Networks: A Progress Report. *IEEE Computer*, 32(4):32–41, April 1999.

[151] P. Smith and D. Hutchinson. New Telecommunication Services using Programmable and Active Networks. In *Proceedings of 1st Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking, and Broadcasting (PGNet2000)*, pages 69–74, Liverpool, UK, 19-20 June 2000.

[152] T. Speakman, J. Crowcroft, J. Gemmell, D. Farinacci, S. Lin, D. Leshchiner, M. Luby, T. Montgomery, L. Rizzo, A. Tweedly, N. Bhaskar, R. Edmonstone, R. Sumanasekera, and L. Vicisano. PGM Reliable Transport Protocol Specification. RFC 3208, December 2001.

[153] W. Stalling. *High-speed Networks: TCP/IP and ATM Design Principles.* Prentice-Hall Inc, 1998.

[154] J. P. Sterbenz. Intelligents in Future Broadband Networks: Challenges and Opportunities in High Speed Active Networking. In *Proceedings of 2002 International Zurich Seminar on Broadband Communications*, pages 2.1 – 2.7, ETH Zurich, Switzerland, 19-21 February 2002.

[155] B. Suter, T. Lakshman, D. Stiliadis, and A. Choudhury. Efficient Active Queue Management for Internet Routers. In *Proceedings of INTERROP 1998*, Las Vegas, USA, 1998. IEEE.

[156] A. Tanenbaum. *Computer Networks*, pages 374–296. Prentice Hall International, 3 edition, 1996.

[157] The Openetlab Team. Accelar Programmable Networking and Active Networks. Technical report, Nortel Technology Center, June 2000.

[158] D. Tennenhouse, J. Smith, W. Sincoskie, D. Wetherall, and G. Minden. A Survey of Active Network Research. *IEEE Communications*, 35(1):80–86, January 1997.

[159] D. Tennenhouse and D. Wetherall. Towards an Active Network Architecture. *Computer Communication Review*, 26(2):1–14, April 1996.

[160] B. L. Tierney. TCP Tuning Guide for Distributed Applications on Wide Area Networks. *Usenix ;login:*, pages 33–39, February 2001.

[161] C. Tschudin. Open Resource Allocation for Mobile Code. In *Proceedings of the Mobile Agent'97 Workshop*, pages 1–6, Berlin, Germany, April 1997.

[162] P. Tullman, M. Hibler, and J. Lepreau. Janos: A Java-oriented OS for Active Networks. *IEEE Journal on Selected Areas of Communication*, 19(3):501–505, March 2001.

[163] I. Umaru-Mohammed. Adding the RLC and RMDP Multicast Protocols to the Application Level Active Networking (ALAN) Environment. Technical Report RN/00/1, University College London, UCL, 15 January 2000.

[164] V. C. Van. A Defence againts Address Spoofing using Active Networks. Master's thesis, Department of Electrical Engineering and Computer Science, May 1997.

[165] P. Vasallo. Variable packet size equation-based congestion control. Technical report, International Computer Science Insitite (ICSI) Berkeley, CA, May 2000. end-to-end mailing list distribution.

[166] L. Vicisano, L. Rizzo, and J. Crowcroft. TCP-like Congestion Control for Layered Multicast Data Transfer. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM 1998)*, pages 1–8, San Francisco, USA, 29 March - 2 April 1998.

[167] I. Wakeman, A. Jeffrey, R. Graves, and T. Owen. Designing a Programming Language for Active Networks. Technical report, University of Sussex, June 1998. http://focus.gmd.de/research/cc/tip/employees/activenets.

[168] B. Wang, B. Zhang, Z. Liu, and H. Li. Forward Active Networks Congestion Control Algorithm and its Performance Analysis. In *Proceedings of IEEE APCCAS (Asia Pacific Conference on Circuits and Systems)*, pages 313–318, Tianjin, China, December 2000.

[169] S. Wen, J. Griffioen, and K. Calvert. CALM: Congestion-Aware Layered Multicast. In *Proceedings of 5th International Conference on Open Architectures and Network Programming (IEEE OPENARCH 2002)*, pages 179–190, New York, USA, June 2002.

[170] D. Wetheral. *Service Introduction in Active Networks.* PhD thesis, Massachusets Institute of Technology, February 1999.

[171] D. Wetherall. Active Network Transfer Service, 1997 11/2/97, ver 1.2. Software and Documentation available at http://www.sds.lcs.mit.edu/activeware/ants.

[172] D. Wetherall. Developing Network Protocols with ANTS Toolkit. Technical report, MIT, MIT Software Devices and Systems Group, Lab for Computer Science, August 1997.

[173] D. Wetherall. Active Network Vision and Reality: Lesson from a Capsule-based System. In D. Kotz, editor, *Proceedings of the 17th ACM Symposium on Operating Systems Principles (SOSP'99)*, pages 64–79, Santa Clara, USA, December 1999.

[174] D. Wetherall, J. Guttag, and D. Tennenhouse. ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols. In *Proceedings of IEEE Conference on Open Architecture (Openarch 1998)*, volume 4, pages 42–47, San Francisco, CA, April 1998.

[175] D. Wetherall, J. Guttag, and D. Tennenhouse. ANTS: Network Services Without the Red Tape. *IEEE Computer*, 32(4):42–47, April 1999.

[176] D. Wetherall, U. Legedza, and J. Guttag. Introducing New Internet Services: Why and How. *IEEE Network*, pages 12–19, July 1998. htt://www.sdc.lcs.mit.edu/publications/network98.ps.

[177] J. Widmer, R. Denda, and M. Mauve. A Survey of TCP-Friendly Congestion Control. *IEEE Network*, 15(3):28–37, May/June 2001.

[178] J. Widmer and M. Handley. Extending Equation-Based Congestion Control to Multicast Applications. In *Proceedings of ACM Symposium on Communications Architectures and Protocols (SIGCOMM 2001)*, pages 275–286, San Diego, USA, 27-31 August 2001.

[179] J. Widmer, M. Mauve, and J. P. Damm. Probabilistic Congestion Control for Non-adaptable Flows. In *Proceedings of IEEE International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2002)*, pages 13–21, Miami, Florida, 12-14 May 2002.

[180] B. Williamson and C. Farrel. Active Congestion Control. In *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM 1998*, volume 3, pages 1509–1514, Sydney, Australia, 8-11 November 1998.

[181] Lloyd Woods. *Internetworking with Satellite Constellation.* PhD thesis, Centre for Communication Systems Research, University of Surrey, UK, June 2001.

[182] Z. D. Wu and X. G. Wang. Design and Analysis of Code Distribution Systems with Active Networks. In *Proceedings of 15 Int. Conf. on Information Networking 2001*, pages 313–318. IEEE, January 2001.

[183] W. Wulinger and V. Paxson. Where Mathematics Meets the Internet. *Notices of the AMS (American Mathematical Society)*, 45(8):961–970, 1998.

[184] K. Yamamoto, M. Yamamoto, and H. Ikeda. Congestion Control for Reliable Multicast achieving TCP Fairness. *IEICE Transactions on Communications, Special Issue on Internet Technology Series (2) Traffic Control and Performance Evaluation in Internet*, E85-B(1):183–190, January 2002.

[185] L. Yamamoto. *Adaptive Group Communications over Active Networks.* PhD thesis, School of Informatics, Faculty of Applied Science, University of Liege, Belgium, August 2002.

[186] L. Yamamoto and G. Leduc. Adaptive Applications over Active Networks: Case Study on Layered Multicast. In P. Lauretz, editor, *Proceedings of 1st European Conference on Universal Multiservice Network (ECUMN 2000)*, pages 386–394, Colmar, France, 2-4 October 2000.

[187] L. Yamamoto and G. Leduc. An Active Layered Multicast Adaptation Protocol. In Yasuda, editor, *Proceedings of International Working Conference on Active Networks (IWAN 2000)*, LCNS 1942, pages 1–6, Tokyo, Japan, October 2000. Springer Verlag.

[188] L. Yamamoto and G. Leduc. An Agent-inspired Active Network Resource Trading Model Applied to Congestion Control. In *Proceedings of Mobile Agents for Telecommunications Applications (MATA 2000) Conference*, Springer-Verlag LNCS 1931, pages 151–169, Paris, France, July 2000.

[189] C. Yang and V. Reddy. A Taxonomy for Congestion Control Algorithms in Packet Switching Networks. *IEEE Network*, 9(4):34–45, July/August 1995.

[190] Q. Yin and S. H. Low. Convergence of REM Flow Control at a Single Link. *IEEE Communications Letters*, 5(3):119–201, March 2001.

[191] S. Zabir, A. Ashir, and N. Shiratori. Estimation of Network Performance: An Approach based on Network Experience. In *Proceedings of the IEEE 15th International Conference on Information Networking (ICOIN 2001)*, pages 657–662, Oita, Japan, 31 January - 2 February 2001.

[192] C. Zhu, O. Yang, J. Aweya, M. Ouellette, and D. Montuno. Comparison of Active Queue Management Algorithm using Opnet Modeller. *IEEE Communications Magazine*, 40(6):158–167, June 2002.

Appendix A

Scripts and ns-2 Traces from Simulation

In this Appendix, first we present some awk scripts used for statistical analysis. We also provide an example of Tcl codes for each protocol. Next, we present a selected portion of the raw ns-2 trace obtained from our simulations with ACC/TCP, AER/NCA and ALMA. The ns trace file is used for debugging purposes and to generate results. Recall that an ns-2 trace consists of the information of the experiment as follows: event, time, from_node, to_node, packet_type, flags, flow_id, source_address, destination_address, sequence_number, and packet_id.

A.1 awk scripts for statistical analysis

```
BEGIN {
   n=0; s=0; ss=0;
   }
   NF == 1
   {n++; s+=$1; ss += $1 * $1;}
   END{
   print n " data points"
   m = (s+0.0)/n; print m " average"
   sd = sqrt( (ss - n * m * m)/(n-1.0))
   print sd " standard deviation"
   stderr = (1.96 * (sd/sqrt(n)))
   print stderr " standard error"
   CImin = (m-stderr)
   CImax = (m+stderr)
   print "Confidence Interval " CImin " to " CImax
   }
```

A.2 ACC TCP

An Example of Tcl file for REM active simulation

```
############################
# Experiment U1
############################
#N is the number of source nodes
#T is the duration of the simulation
set ns [new Simulator]
$ns set-address-format expanded
set N 50
set T 250
# Predefine tracing
set f0 [open results.tr w]
# Define the topology
set switch_src [$ns node]
set switch_dst [$ns node]
$ns duplex-link $switch_src $switch_dst 10Mb 150ms REM/Active
$ns duplex-link-op $switch_src $switch_dst orient right
set remq [[$ns link $switch_src $switch_dst] queue]
$remq set reminw_ 0.02
$remq set remgamma_ 0.001
$remq set remphi_ 1.001
$remq set remupdtime_ 0.001
$remq set rempktsize_ 1000
$remq set rempbo_ 20
# set queue size #
$ns queue-limit $switch_src $switch_dst 120
for {set i 0} {$i < $N} {incr i} {
set src_node($i) [$ns node]
set sink_node($i) [$ns node]
$ns duplex-link $src_node($i) $switch_src 16Mb 30ms DropTail
$ns duplex-link $switch_dst $sink_node($i) 16Mb 0ms DropTail
# set source agent and application
set tcp($i) [new Agent/TCP/Active]
$tcp($i) set class_ 1
$ns attach-agent $src_node($i) $tcp($i)
set ftp($i) [new Application/FTP]
$ftp($i) attach-agent $tcp($i)
# set the maxium window size #
$tcp($i) set window_ 150
# set source pkt size #
$tcp($i) set packetSize_ 1000
$tcp($i) set fid_ $i
$ns color $i Red
# set sink
set sink($i) [new Agent/TCPSink/Active]
$ns attach-agent $sink_node($i) $sink($i)
$ns connect $tcp($i) $sink($i)
# $ns at 0.0 "$ftp($i) start"
}
```

```
for {set i 0} {$i < 10} {incr i} {
$ns at 0.0 "$ftp($i) start"
}
for {set i 10} {$i < 20} {incr i} {
$ns at 50.0 "$ftp($i) start"
}
for {set i 20} {$i < 30} {incr i} {
$ns at 100.0 "$ftp($i) start"
}
for {set i 30} {$i < 40} {incr i} {
$ns at 150.0 "$ftp($i) start"
}
#set re_ [open re.tcp w]
for {set i 0} {$i < $N} {incr i} {
$tcp($i) trace nrexmitpack_
$tcp($i) attach $re_
}
# Tracing a queue
set remq [[$ns link $switch_src $switch_dst] queue]
set tchan_ [open all.q w]
$remq trace curq_
$remq trace lost_
$remq trace prob_
$remq trace throughput_
$remq attach $tchan_
# Monitor the queue between source switch and destination switch
set queueMoni [$ns monitor-queue $switch_src $switch_dst $f0 0.1]
# Monitor the queue on NAM
$ns duplex-link-op $switch_src $switch_dst queuePos 0.5
set drop_ini 0
set throughput_ini 0
proc record {} {
global f0 N tcp
global queueMoni
global remq
#get an instance of the simulator
set ns [Simulator instance]
#set the time afterwhich the procedure should be called again
set time 0.01
set now [$ns now]
# puts stdout "time= $now"
set queue0 [$queueMoni set pkts_]
set parriv0 [$queueMoni set parrivals_]
set pdepart0 [$queueMoni set pdepartures_]
set pdrop0 [$queueMoni set pdrops_]
set mark_b [$remq set prob_]
puts $f0 "$now $queue0 $pdrop0 $parriv0 $pdepart0 $mark_b"
# Re-schedule the procedure
$ns at [expr $now+$time] "record"
}
$ns at 0.0 "record"
$ns at $T "finish"
proc finish {} {
global tchan_, re_
```

```
close $re_
global ns f0
$ns flush-trace
close $f0
exit 0
}
$ns run
```

Trace File for REM active simulation

```
.....
r 2.13051 1 5 tcp 1000 ——N 1 4.0 5.0 49 172 26
+ 2.13051 5 1 ack 40 ——- 1 5.0 4.0 49 330 -1
- 2.13051 5 1 ack 40 ——- 1 5.0 4.0 49 330 -1
r 2.13202 4 0 tcp 1000 ——N 1 4.0 5.0 89 328 46
+ 2.13202 0 1 tcp 1000 ——N 1 4.0 5.0 89 328 46
r 2.13282 4 0 tcp 1000 ——N 1 4.0 5.0 90 329 46
+ 2.13282 0 1 tcp 1000 ——N 1 4.0 5.0 90 329 46
+ 2.13282 0 4 Active 1000 ——- 1 5.0 4.0 -1 331 26
- 2.13282 0 4 Active 1000 ——- 1 5.0 4.0 -1 331 26
d 2.13282 0 1 tcp 1000 ——N 1 4.0 5.0 90 329 46
- 2.13371 0 1 tcp 1000 ——N 1 4.0 5.0 54 179 28
r 2.13571 0 1 tcp 1000 ——N 1 4.0 5.0 50 173 26
+ 2.13571 1 5 tcp 1000 ——N 1 4.0 5.0 50 173 26
- 2.13571 1 5 tcp 1000 ——N 1 4.0 5.0 50 173 26
r 2.13722 0 4 ack 40 ——- 1 5.0 4.0 45 318 -1
+ 2.13722 4 0 tcp 1000 ——N 1 4.0 5.0 91 332 47
- 2.13722 4 0 tcp 1000 ——N 1 4.0 5.0 91 332 47
+ 2.13722 4 0 tcp 1000 ——N 1 4.0 5.0 92 333 47
- 2.13802 4 0 tcp 1000 ——N 1 4.0 5.0 92 333 47
r 2.14054 5 1 ack 40 ——- 1 5.0 4.0 49 330 -1
+ 2.14054 1 0 ack 40 ——- 1 5.0 4.0 49 330 -1
- 2.14054 1 0 ack 40 ——- 1 5.0 4.0 49 330 -1
r 2.14318 1 0 ack 40 ——- 1 5.0 4.0 46 321 -1
+ 2.14318 0 4 ack 40 ——- 1 5.0 4.0 46 321 -1
- 2.14318 0 4 ack 40 ——- 1 5.0 4.0 46 321 -1
r 2.14362 0 4 Active 1000 ——- 1 5.0 4.0 -1 331 26
+ 2.14362 4 0 tcp 1000 ——N 1 4.0 5.0 90 334 500
- 2.14362 4 0 tcp 1000 ——N 1 4.0 5.0 90 334 500
.....
```

A.3 AER/NCA

An Example of Tcl file for REM active simulation

```
#####################################
# Experiment S1 - Uncorrelated Loss
# 1 AER/NCA mcast flow, 2 TCP flows
# #####################################
set ns [new Simulator -multicast on]
global numNodes
set NumNodes 9
for {set i 0} {$i < $NumNodes} {incr i} {
set node($i) [$ns node]
}
set f [open ANnca0201.tr w]
$ns trace-all $f
```

```
set nf [open ANnca0201.nam w]
$ns namtrace-all $nf
$ns color 1 blue
# data pkt
$ns color 2 yellow
# nca pkt
$ns color 3 white
$ns color 4 red
# nak pkt
$ns color 5 green
# repair
$ns color 6 goldenrod
# spm pkt
$ns color 7 black
# get-rtt pkt
$ns color 8 pink
# csm pkt
#distinction added for prune/join messages
$ns color 30 purple
$ns color 31 orange
proc makelinks { bw delay pairs } {
global ns node
foreach p $pairs {
set src $node([lindex $p 0])
set dst $node([lindex $p 1])
$ns duplex-link $src $dst $bw $delay DropTail
$ns duplex-link-op $src $dst orient [lindex $p 2]
}
}
makelinks 1Mb 60ms {
{ 0 1 right }
{ 1 2 right }
{ 7 1 right-up }
{ 8 1 up }
}
makelinks 1Mb 60ms {
{ 2 3 up }
{ 2 4 right-up}
{ 2 5 right-down}
{ 2 6 down}
}
$ns duplex-link-op $node(1) $node(2) queuePos 0.5
$ns duplex-link-op $node(2) $node(3) queuePos 1.5
$ns duplex-link-op $node(2) $node(4) queuePos 1.5
global group
set group 0x8002
set mproto CtrMcast
set mrthandle [$ns mrtproto $mproto {}]
if {$mrthandle != ""} {
$ns at 0.01 "$mrthandle switch-treetype 0x8002"
}
Agent/AN set class_ 3
Agent/AN set packetSize_ 1000
Agent/AN set dst_ $group
```

```
Agent/AN/Rcvr set group_ $group
Agent/AN/Rcvr instproc start_rtt {} {
$self startrtt
}
Agent/AN/CBR set group_ $group
Agent/AN/CBR set dst_ $group
Agent/AN/CBR instproc send_spm {} {
global group
$self sendspm
}
Agent/AN/Active set group_ $group
Agent/AN/Active set dst_ $group
Agent/AN/Active instproc send_init {} {
global group
$self set dst_ $self
$self send init
$self set dst_ $group
}
Agent/AN/Active instproc disable {} {
global group
$self set dst_ $self
$self send disable
$self set dst_ $group
}
# Create an Active CBR source with rate adjusted by nca
global cbrsndr
set cbrsndr [new Agent/AN/CBR]
$ns attach-agent $node(0) $cbrsndr
$cbrsndr set packetSize_ 1000
$cbrsndr set dst_ $group
$cbrsndr set maxpkts_ 30000
$cbrsndr set class_ 1
$node(0) color blue
# Must include sourceID info
set sourceID [$cbrsndr set addr_]
Agent/AN/Rcvr set sourceID_ $sourceID
Agent/AN/Active set sourceID_ $sourceID
Agent/AN/CBR set sourceID_ $sourceID
#
foreach value {3 4} {
set rcvr($value) [new Agent/AN/Rcvr]
$ns attach-agent $node($value) $rcvr($value)
set thisAddr [$rcvr($value) set addr_]
$rcvr($value) set thisAddr_ $thisAddr
}
# Active Receivers join the multicast group
$ns at 0.2 {
foreach value { 3 4 } {
$node($value) join-group $rcvr($value) $group
$node($value) color red
}
}
# Set up and initialize the Active Agents at nodes 2 and 4
$ns at 0.4 {
```

```
foreach value {1 2} {
set anagent($value) [new Agent/AN/Active]
$ns attach-agent $node($value) $anagent($value)
$node($value) color green
$anagent($value) send_init
set thisAddr [$anagent($value) set addr_]
$anagent($value) set thisAddr_ $thisAddr
}
}
$ns at 0.6 "$cbrsndr send_spm"
$ns at 0.8 "$cbrsndr start"
# Create lossy links
proc makelossylinks { links } {
set i 0
global ns node
foreach l $links {
incr i
set loss_module($i) [new ErrorModel]
$loss_module($i) unit pkt
$loss_module($i) set rate_ .01
$loss_module($i) ranvar [new RandomVariable/Uniform]
$loss_module($i) drop-target [new Agent/LossMonitor]
$loss_module($i) set enable_ 1
$ns lossmodel $loss_module($i) $node([lindex $l 0]) $node([lindex $l 1])
incr i
set loss_module($i) [new ErrorModel]
$loss_module($i) unit pkt
$loss_module($i) set rate_ .01
$loss_module($i) ranvar [new RandomVariable/Uniform]
$loss_module($i) drop-target [new Agent/LossMonitor]
$loss_module($i) set enable_ 1
$ns lossmodel $loss_module($i) $node([lindex $l 1]) $node([lindex $l 0])
}
}
$ns at 1.5 "makelossylinks {
{0 1}
{1 2} }"
# set up 2 tcp connections
set tcp1 [$ns create-connection TCP/Newreno $node(7) TCPSink $node(5) 0]
$tcp1 set window_ 50
$tcp1 set class_ 15
$ns color 15 brown
# set up FTP Source
set ftp1 [$tcp1 attach-app FTP]
$ns at 1.15 "$ftp1 start"
#$self tcpDump $tcp1 1.0
set tcp2 [$ns create-connection TCP/Newreno $node(8) TCPSink $node(6) 0]
$tcp2 set window_ 50
$tcp2 set class_ 16
$ns color 16 gold
# set up FTP Source
set ftp2 [$tcp2 attach-app FTP]
$ns at 21.75 "$ftp2 start"
#$self tcpDump $tcp2 1.0
```

```
$ns at 250 "finish"
proc finish {} {
global ns f
$ns flush-trace
close $f
puts "running nam..."
exec nam ANnca0201.nam &
exit 0
}
$ns run
$ns clear-mcast
```

Trace File for AER/NCA simulation

```
....
- 30.8452 2 1 AN 76 ——- 7 2.1 0.1 -1 392
r 30.8635 2 3 AN 76 ——- 6 0.1 128.2 -1 390
r 30.8635 2 4 AN 76 ——- 6 0.1 128.2 -1 390
r 30.8635 2 11 AN 76 ——- 6 0.1 128.2 -1 390
r 30.8635 2 12 AN 76 ——- 6 0.1 128.2 -1 390
r 30.9058 2 1 AN 76 ——- 7 2.1 0.1 -1 392
d 30.9058 1 2 AN 76 ——- 7 2.1 2.1 -1 392
+ 30.913 12 2 AN 76 ——- 7 128.2 0.1 -1 393
- 30.913 12 2 AN 76 ——- 7 128.2 0.1 -1 393
+ 30.9231 11 2 AN 76 ——- 8 128.2 0.1 -1 394
- 30.9231 11 2 AN 76 ——- 8 128.2 0.1 -1 394
+ 30.9257 3 2 AN 76 ——- 7 128.2 0.1 -1 395
- 30.9257 3 2 AN 76 ——- 7 128.2 0.1 -1 395
+ 30.9279 10 1 tcp 1000 —A— 16 10.1 14.1 3 396
- 30.9279 10 1 tcp 1000 —A— 16 10.1 14.1 3 396
+ 30.9294 4 2 AN 76 ——- 7 128.2 0.1 -1 397
- 30.9294 4 2 AN 76 ——- 7 128.2 0.1 -1 397
+ 30.9316 11 2 AN 76 ——- 7 128.2 0.1 -1 398
- 30.9316 11 2 AN 76 ——- 7 128.2 0.1 -1 398
r 30.9736 12 2 AN 76 ——- 7 128.2 0.1 -1 393
+ 30.9736 2 12 AN 76 ——- 7 128.2 12.1 -1 393
- 30.9736 2 12 AN 76 ——- 7 128.2 12.1 -1 393
r 30.9837 11 2 AN 76 ——- 8 128.2 0.1 -1 394
...
```

A.4 ALMA

An Example of Tcl file for REM active simulation

```
########################################
# Experiment M3a - ALMA competing with 2 TCP flows
# ########################################
proc start_time {} {return [expr ([eval ns-random] / 2147483647.0) * 0.1]}
ns-random 0
proc setglobalvar {var value} {
global glvarlist $var
set $var $value
lappend glvarlist $var
}
setglobalvar simfileid "topology3"
setglobalvar finishtime 100
```

```
setglobalvar scenario Mcast
setglobalvar gen_plot 1
setglobalvar run_plot 1
setglobalvar run_nam 1
setglobalvar multicast on
setglobalvar ideal_fair 1
setglobalvar srcbw 1e5
setglobalvar pktsize 500
setglobalvar qtype RED
setglobalvar bqsize 20
setglobalvar whenon -1
setglobalvar whenoff -1
setglobalvar nlayers 17
foreach a $argv {
set L [split $a =]
if {[llength $L] != 2} { continue }
set var [lindex $L 0]
set val [lindex $L 1]
set $var $val
puts "ARGV: set $var $val"
}
if {$whenon < 0} {
set whenon [expr $finishtime / 3]
}
if {$whenoff < 0} {
set whenoff [expr $finishtime * 2 / 3]
}
foreach var $glvarlist {
puts "GLVAR $var = [set $var]"
}
source "math.tcl"
source "topology.tcl"
source "resmng.tcl"
source "ANnode.tcl"
source "ANagent.tcl"
source "ANinterf.tcl"
source "stream.tcl"
source "mpeg.tcl"
Agent/AN set packetSize_ $pktsize
Class Scenario0 -superclass Topology
Scenario0 instproc mklink { a b delay bw {orient ""} \
{qtype "DropTail"} {qlen ""} {qpos ""} {debug_ab 0} {debug_ba 0} } {
$self next $a $b $delay $bw $orient $qtype $qlen $qpos
$self instvar ns node nodeos
foreach i [list $a $b] {
if {![info exists nodeos($i)]} {
set nodeos($i) [new NodeOS $ns $node($i)]
}
}
$nodeos($a) add_link $node($b)
$nodeos($b) add_link $node($a)
set lmng_ab [$nodeos($a) set lnkmng_([$node($b) id])]
set lmng_ba [$nodeos($b) set lnkmng_([$node($a) id])]
$lmng_ab set debug_ $debug_ab
```

```
$lmng_ba set debug_ $debug_ba
$ns at 0.01 "$lmng_ab start"
$ns at 0.01 "$lmng_ba start"
}
Scenario0 instproc mkeqrates {srcbw nlayers} {
set rates {}
for {set i 0} {$i < $nlayers} {incr i} {
lappend rates [expr $srcbw/$nlayers]
}
return $rates
}
Scenario0 instproc dimension_queue {speed delay} {
global pktsize
set maxdelay [min 0.1 [expr 2.0 * $delay]]
set Q [expr int($maxdelay * $speed / ($pktsize * 8))]
}
Scenario0 instproc init args {
eval $self next $args
}
Class ScenarioMcast -superclass Scenario0
ScenarioMcast instproc init args {
eval $self next $args
$self instvar ns node nodeos anee
global colors qtype bqsize pktsize \
whenon whenoff srcbw finishtime nlayers
set linkbw [expr 1*$srcbw]
set b1 10e6
set b2 300e3
set d1 5e-3
set d2 20e-3
set BQ1 [max $bqsize [$self dimension_queue $b1 $d1]]
puts "SCENARIO BQ1=$BQ1"
set LQ1 [max $bqsize [$self dimension_queue $b2 $d2]]
puts "SCENARIO LQ1=$LQ1"
# sources
$self mklink 0 1 $d2 $b2 right $qtype $LQ1
$self mklink 2 0 $d1 $b1 down $qtype $BQ1 1.25 1
$self mklink 3 0 $d1 $b1 right-up $qtype $BQ1 1.25 1
$self mklink 4 0 $d1 $b1 up $qtype $BQ1 1.25 1
$self mklink 1 5 $d1 $b1 up $qtype $BQ1 1.25 1
$self mklink 1 6 $d1 $b1 right-down $qtype $BQ1 1.25 1
$self mklink 1 7 $d1 $b1 down $qtype $BQ1 1.25 1
set anreg [new ANagentreg $ns]
foreach i {0 1 2 3 4 5 6 7} {
set anee($i) [new Agent/AN $ns $node($i) $anreg]
$anee($i) setnodeos $nodeos($i)
$node($i) color green
}
set session1 [list 2 {5} 20 100]
$self place_tcp_session 3 6 0 100 1 red
$self place_tcp_session 4 7 60 100 2 black
set sessions [list $session1]
set rates {}
for {set i 0} {$i < $nlayers} {incr i} {
```

```
lappend rates [expr $srcbw/$nlayers]
}
foreach session $sessions {
set src [lindex $session 0]
set rcvlist [lindex $session 1]
set won [lindex $session 2]
set woff [lindex $session 3]
set coder [new Agent/Coder $ns $node($src) $rates $pktsize \
$anee($src) {blue yellow red aquamarine pink brown}]
$self reggr $src message s
$ns at $won "$coder start"
#$ns at $won "[$coder set smon_] record"
set count 1
foreach rcv $rcvlist {
set decoder [new Agent/Decoder $ns $node($rcv) $anee($src) \
$anee($rcv) $nlayers]
$self reggr $rcv message r
$ns at [expr $won + 0.17685 * $count] "$decoder start"
$ns at $woff "$decoder stop"
incr count
}
$ns at $woff "$coder stop"
$ns at $finishtime "[$coder set smon_] end"
}
}
set ns [new Simulator -multicast $multicast]
set scn [new Scenario$scenario $ns]
$ns at $finishtime "$scn finish"
$ns run
```

Trace File for ALMA simulation

....

```
r 27.155636 1 8 message 500 ——- 0 1.0 8.0 -1 7457
+ 27.156376 3 0 message 500 ——- 10 3.0 0.0 -1 7459
- 27.156376 3 0 message 500 ——- 10 3.0 0.0 -1 7459
r 27.156903 0 1 message 500 ——- 13 0.0 1.0 -1 7371
+ 27.156903 1 8 message 500 ——- 13 1.0 8.0 -1 7460
- 27.156903 1 8 message 500 ——- 13 1.0 8.0 -1 7460
- 27.156903 0 1 message 500 ——- 12 0.0 1.0 -1 7390
+ 27.157669 2 0 message 500 ——- 1 2.0 0.0 -1 7462
- 27.157669 2 0 message 500 ——- 1 2.0 0.0 -1 7462
r 27.161776 3 0 message 500 ——- 10 3.0 0.0 -1 7459
+ 27.161776 0 1 message 500 ——- 10 0.0 1.0 -1 7463
r 27.162303 1 8 message 500 ——- 13 1.0 8.0 -1 7460
r 27.163069 2 0 message 500 ——- 1 2.0 0.0 -1 7462
+ 27.163069 0 1 message 500 ——- 1 0.0 1.0 -1 7464
r 27.163569 0 1 message 500 ——- 5 0.0 1.0 -1 7375
+ 27.163569 1 9 message 500 ——- 5 1.0 9.0 -1 7465
- 27.163569 1 9 message 500 ——- 5 1.0 9.0 -1 7465
- 27.163569 0 1 message 500 ——- 6 0.0 1.0 -1 7393
+ 27.165965 2 0 message 500 ——- 12 2.0 0.0 -1 7467
- 27.165965 2 0 message 500 ——- 12 2.0 0.0 -1 7467
r 27.168969 1 9 message 500 ——- 5 1.0 9.0 -1 7465
r 27.170236 0 1 message 500 ——- 7 0.0 1.0 -1 7377
+ 27.170236 1 9 message 500 ——- 7 1.0 9.0 -1 7468
```

....