ABSTRACT

SHRIVASTAV, NITIN. A Network Simulator model of the DOCSIS protocol and a solution to the bandwidth-hog problem in Cable Networks. (Under the direction of Dr. Arne Nilsson and Dr. David Thuente.)

A number of broadband access solutions have been developed in the recent years to provide high speed Internet access to the residential users or the 'last-mile'. Some of the prominent technologies include Cable access, Digital Subscriber Loop, Integrated Services Digital Network, Satellite and Wireless. The cable broadband access is gaining widespread popularity. While several approaches to broadband access over cable have been proposed (e.g., DVB/DAVIC and IEEE 802.14), the industry is converging on the architecture developed by the Multimedia Cable Network System (MCNS) group (referred to as the Data-Over-Cable Service Interface Specification or DOCSIS standard). The DOCSIS Radio Frequency Interface specification defines the Media Access Control (MAC) layer as well as the physical communications layer.

The objectives of this thesis were: Design, Implementation and Study of a Network Simulator-2 (NS-2) [1] model of the DOCSIS 2.0 MAC protocol and Design and analysis of an algorithm to solve the bandwidth-hog problem in Cable Networks. It was realized that there is a need for the support of DOCSIS protocol in a publicly available popular simulator to do research on DOCSIS. Network Simulator was chosen as it is widely used by the Internet research community for research of networking protocols and it currently lacks the

support for DOCSIS. The Network Simulator implementation is conformant to the DOCSIS RF-interface specifications with support for Quality of Service features.

The second part of this thesis presents a solution for congestion control in cable networks. It has been observed in cable networks that few heavy bandwidth users can cause severe congestion affecting the performance of the other users. Typically, the end user applications incorporate end-to-end congestion control for best-effort traffic (e.g. TCP congestion control). However, with the growth of the Internet, it might no longer be practical to rely on all end-nodes to use end-to-end congestion control. There is a need for the participation of the network itself in controlling the resource utilization. A solution is proposed where the central entity (known as Cable Modem Termination System) in the cable network identifies and restricts the bandwidth of selected best-effort flows in times of congestion. The flows using disproportionate bandwidth during times of congestion are restricted. As outlined in [2], this approach promotes the use of end-to-end congestion control by the end-nodes as it provides an incentive to flows responsive to congestion by not restricting them.

A NETWORK SIMULATOR MODEL OF THE DOCSIS PROTOCOL AND A SOLUTION TO THE BANDWIDTH-HOG PROBLEM IN THE CABLE NETWORKS

by
NITIN SHRIVASTAV

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

COMPUTER NETWORKING

Raleigh

2003

APPROVED BY:

_____	_____
Dr. Jim Martin	Dr. Andy Rindos
_____	_____
Dr. Arne Nilsson Chair of Advisory Committee	Dr. David Thuente, Co-Chair

BIOGRAPHY

Nitin Shrivastav was born in Raipur, India in the state of Madhya Pradesh. He received his Bachelor's degree in Computer Science from National Institute of Technology (formerly Regional Engineering College), Warangal, India in 2000. He worked for a year in the capacity of Software Engineer at Nortel Networks India Technology Pvt. Limited. He then joined the North Carolina State University to pursue his Masters degree in Computer Networking, and has been working as a Teaching Assistant in the Computer Science department for past 2 years.

Acknowledgements

I would like to thank Dr. Jim Martin for his invaluable guidance, motivation, and encouragement and for providing a great opportunity to work with him. I would like to thank Dr. Arne Nilsson, Dr. David Thuente and Dr. Andy Rindos for accepting to be on my committee and reviewing my thesis. I would like to thank Dr. M. Vouk for allowing me to use the EGRC office space and machines.

I would like to thank my parents and my sister for their incredible support and encouragement.

My thanks are due to all my friends at NC State for their help in past few months. And, finally I would like to thank Farida for all the encouragement and support she gave me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

This chapter presents the motivational background, contributions and the layout of this thesis.

1.1 Last mile delivery

Traditionally, delivery of Internet access to the residential area, or the "last mile" as it is more commonly referred to, has relied upon the Plain Old Telephony System (POTS) network. However, residential users and small businesses now demand fast connectivity technologies to access the advanced web services like rich multimedia content and video on demand. This has fueled the demand for broadband access. It is predicted that the number of broadband cable users in North America will rise from about 12 million (in 2002) to about 26 million by 2006 [3].

Consumers are moving away from dialing-up using their phone lines towards new faster and improved connectivity technologies. To meet this demand a number of rival technologies like DSL, cable, satellite, wireless, powerline, and ISDN technologies have been developed. These technologies provide speeds up to 50 times faster than the traditional dial-up access. These broadband connections are always on, making Internet an integral part of our lives.

1.2 Community Antenna TeleVision (CATV) Networks

Community Antenna TeleVision (CATV) networks, often referred to as subscriber networks were initially designed and developed for television and radio broadcasts. Due to their wide availability and large amount of unused frequencies, they were promoted for delivering broadband high speed digital services. CATV networks are based on a tree cabling structure.

The source or the "Headend" (HE) is located at the root and the subscriber at the "leaves". Since CATV were initially designed for one-way broadcast, from HE to the subscriber, they required physical modifications such as bi-directional Radio Frequency (RF) amplifiers, as well as development of protocols to allow point-to-point communications over the shared link. A discussion of the physical characteristics and structure of modern CATV networks is provided in the next section. Then basic cable data network operation is discussed.

1.2.1 Network Topology

Majority of the CATV networks are based on the "tree and branch" architecture [4] as shown in Figure 1.1. There exist many other alternative cable topologies, like switched star, not very widely adopted [5]. There are three main tree and branch architectures: Hybrid Fiber Coaxial (HFC), Fiber To The Curb (FTTC) and Fiber To The Home (FTTH).

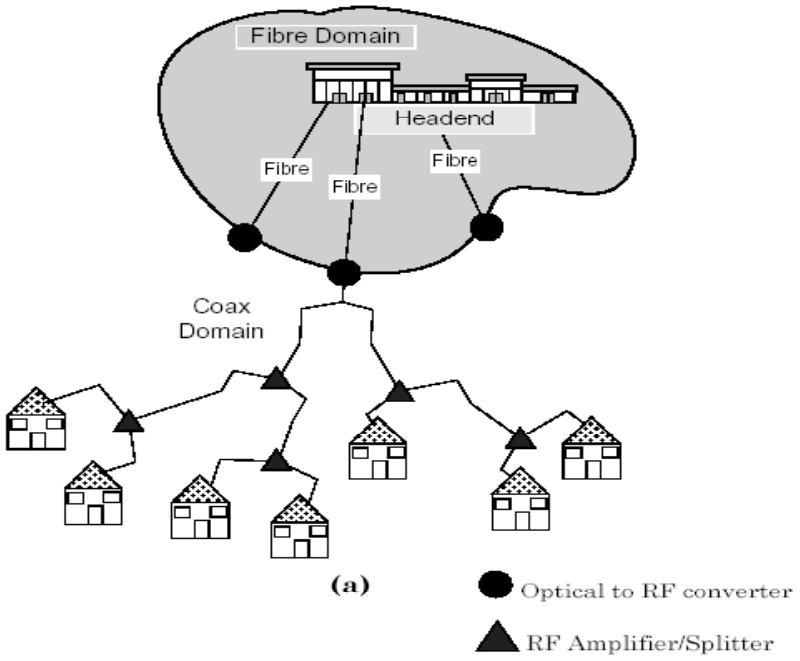


Figure 1.1: Tree and Brach Architecture

These three architectures differ in how close to the subscriber the fiber terminates. CATV networks transitioned from an all coaxial to Hybrid Fiber Coaxial (HFC) networks due to an increased requirements for more channels, lesser noise and reduced signal attenuation. Use of optical fiber gives the advantage of lower maintenance costs, higher bandwidth and no electromagnetic noise.

HFC

The HFC network architecture consists of fiber links connecting the HE with Service Access Points (SAP). A number of homes are then connected to a SAP using coaxial cable. In the downstream direction, HFC networks use 450 to 860 Mhz for digital services and 50 to 450 Mhz for analogue broadcast services. The Upstream spectrum is 5-42 Mhz. Quadrature Amplitude Modulation (QAM) and Quaternary Phase Shift keying (QPSK) modulation is used to support digital services. Over a 6Mhz channel, 64 QAM carries 27Mbps of user data taking forward error correction into account. Figure 1.2 represents a typical HFC architecture.

FTTC

In FTTC networks, fiber runs all the way from HE to curb-side of the houses they support. Fiber links terminate at Optical Network Units (ONUs). Twisted-pair or coaxial can be used to connect ONUs to the home. Time Division Multiplexing is used for signal multiplexing. Over the FTTC networks, bandwidth for downstream is as high as 51 Mbps.

FTTH

Fiber to the home architecture is solely based on fiber links. Fiber links run down to the last mile into each home. Implementation costs of FTTH networks are higher as compared to previously mentioned topologies. Bandwidths up to 155Mbps can be achieved due to all fiber architecture.

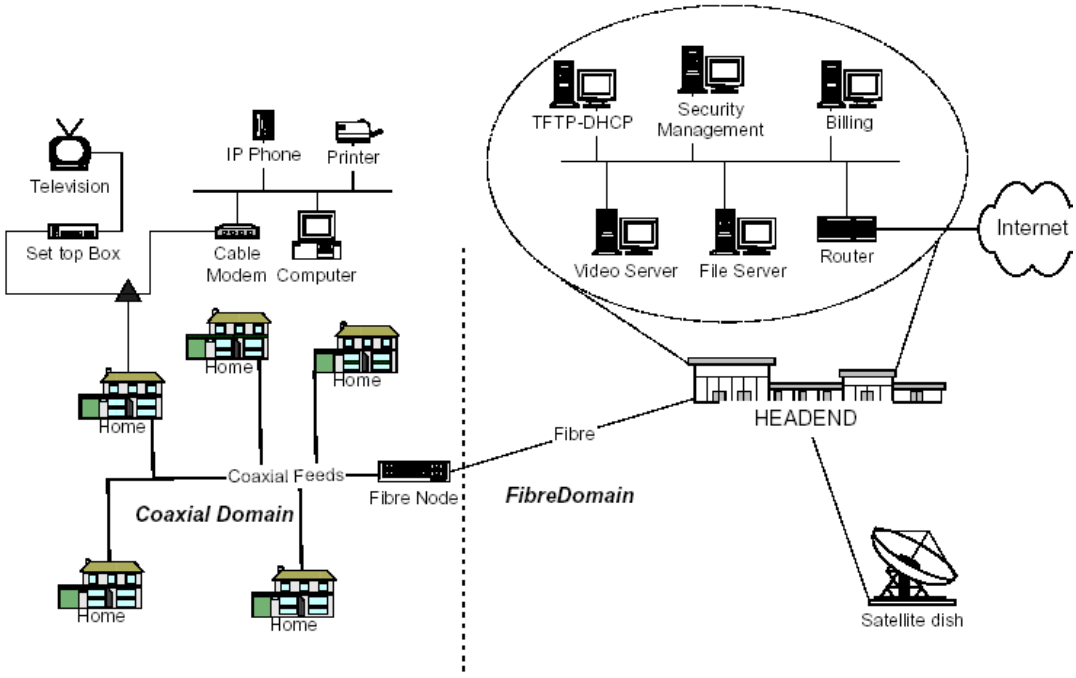


Figure 1.2: HFC Architecture

1.2.2 Cable Data Network Operation

Using a cable modem, data network subscribers connect to the system. The modems may be internal to the PC or attach to it via some interfaces. Once the users are connected, they obtain a continuous connection to the Internet.

All the cable modems communicate with the Cable Modem Termination System (CMTS) located at the central office of a cable service provider as shown in Figure 1.3. Typically, a CMTS serves hundreds of cable modems over a network that can stretch to over 100 kms.

The CMTS connects to the Internet and other networks, sending and receiving user packets. When a packet arrives from the Internet at the CMTS, a processor module converts the IP packets into MPEG packets and performs error checks. Subsequently, the packet is modulated onto a carrier wave using QAM/FEC or QPSK modulation. The output is then forwarded to the subscriber. The cable modem at the subscriber's location converts the radio frequency information back to IP packets and sends them to the end device.

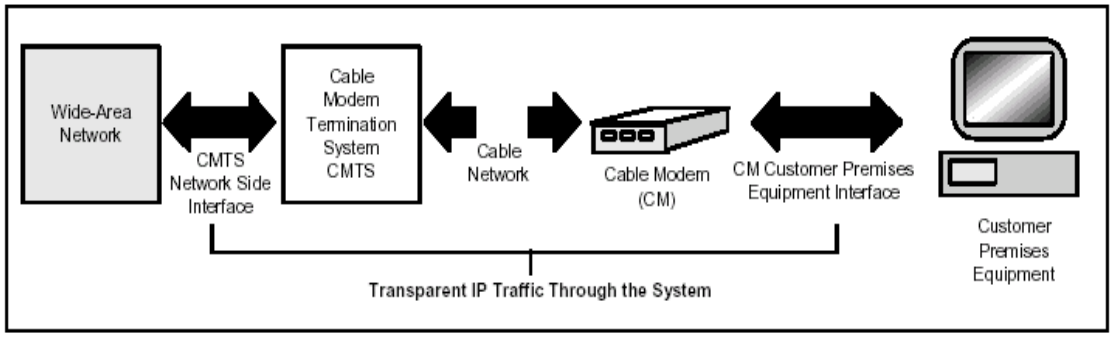


Figure 1.3: Data Over Cable System

Frequency Division Multiplexing (FDM) is used in cable networks. One or more TV channels, which are 6Mhz wide, are used to carry data. The upstream channel typically occupies lower parts of the bandwidth not occupied by TV channels. Downstream rates are typically in the range of 30 Mbps or less while upstream rates are in the range of 300 Kbps to 10 Mbps.

Users will observe significant variation in the downstream data rate as the downstream channel is shared by all active users. There may be brief periods where all the bandwidth is available to just a few users, which would allow huge download speeds. Cable operators can also increase the bandwidth allocation by making additional channels available for data. Since there is only one sender in the downstream direction (CMTS), so no channel access mechanism is needed.

The upstream channels typically occupy lower frequencies that are subject to noise. Since there can be multiple senders in the upstream direction, so a channel access control mechanism is needed to govern control of the common channel. As the number of users increase, there will be a performance drop.

Noise on the upstream connection is caused due to electrical interference from home appliances, old cable, improperly shielded cable etc. The noise problem puts a severe limit on the upstream bandwidth. Synchronous Code Division Multiplexing (S-CDMA) modulation technique is being promoted to increase the available bandwidth in the Upstream.

The shared nature of cable network makes it vulnerable to security problems. Using network snoopers, users can watch traffic and attempt to capture valuable information. The cable industry has worked out encryption techniques and other security measures that support privacy [6].

1.3 Standards For CATV Networks

Cable standards are being designed to provide interoperability between cable modems and the head-end gear. This will give flexibility to the subscriber to buy off-the-shelf cable modems conforming to the standards. Standards benefit both the subscribers and the operators by making connection easier and promoting new applications. The most important standards are outlined below.

1.3.1 Data over Cable Interface Specification (DOCSIS)

DOCSIS is the result of work done by MCNS (Multimedia Cable Network System Partners Ltd.). The group was formed by North American cable industry in response to the slow progress of IEEE 802.14 group. The DOCSIS Radio Frequency Interface specification defines the Media Access Control (MAC) layer as well as the physical communications layer. DOCSIS 1.0 specification focused on the best-effort traffic without much support for Quality of Service (QoS) features. DOCSIS 1.0 was approved and publicized as International Telecommunications Union – Telecommunication Sector (ITU-T) recommendation J.112 on March 1998. DOCSIS 1.1 was the first revision of initial specification and introduced new features for QoS support over IP. DOCSIS 2.0 enhances the physical layer communication methods with higher upstream data rates and improved performance tolerance to bursts of noise. DOCSIS 2.0 has also been approved as the international standard for data over cable systems by ITU-T in 2002 and publicized as ITU-T recommendation number J.122. A detailed description of the protocol features and the operations is discussed in chapter 2. EuroDOCSIS is based on the DOCSIS MAC specification and DVB/DAVIC (section 1.3.3) physical frequency plans.

1.3.2 IEEE 802.14 Working Group

This working group was disbanded in March 2000. The standardization process was slow within IEEE, which failed to observe the time to market constraint and hence lost the industry support. A brief description of the proposed specification is given.

This group has defined the physical layer and a Medium Access Control (MAC) layer protocol for the HFC networks. The architecture specifies an HFC cable plant with a radius of 80 kilometers from the head end. The group's goal was to develop a specification for delivering Ethernet traffic over the network. Asynchronous Transfer Mode (ATM) networking was also considered for the delivery of multimedia traffic.

There are three physical layer specifications reflecting the European, the United States and Japanese requirements (types A, B and C). The upstream channel bandwidth is 8 Mhz for type A and 6 Mhz for types B and C. The Physical layer is composed of the Transmission Convergence (TC) and the Physical medium Dependent (PMD) sublayers. The MAC sublayer consists of the MAC Convergence Subprocess (MACCS), the ATM and the MAC Access Arbitration (MAA) sublayers. This sublayering helps in multiplexing the Logical Link Control (LLC) and the ATM traffic streams effectively.

For upstream transmission, MAC PDUs are segmented into ATM cells using the AAL5 protocol and passed to the MAA process. The MAA process adds 1-octet to each ATM cell to form an ATM PDU (APDU). The APDU frames are encapsulated into MPEG2 packets of 188-octet for downstream transmission.

The specification supports ATM services like Constant Bit Rate (CBR), Variable Bit Rate (VBR) and Available Bit Rate (ABR). The data transmission is based on a grant/request scheme. The CMTS controls access to the upstream channel by allocating transmission opportunities to the users for contention and reservation-based transfer [7].

1.3.3 Digital Audio Visual Council (DVB/DAVIC)

DAVIC was a non-profit group that promoted digital audio-visual applications and services based on the specifications that maximized interoperability across countries and applications/services [8]. DAVIC developed a digital video broadcast reference model that is popular in Europe and preferred by the European Cable Communications Association (ECCA), a European cable industry organization. DAVIC completed its work and closed in 2000. The DVB/DAVIC reference model is shown in Figure 1.4.

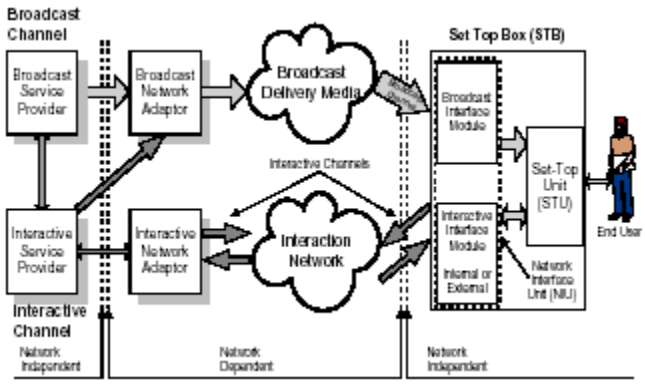


Figure 1.4: DVB/DAVIC Reference Model

Between the service provider and the user, two channels are established – Broadcast channel (BC) and the Interaction channel (IC). BC is unidirectional broadband channel to transmit audio, video and data. IC is a bi-directional channel and is composed of – *Return Interaction* path (Upstream channel), from the user to the service provider and *Forward Interaction* path,

from the service provider to the user. The user terminal interfaces with the broadcast channel as well as interaction channel. The interface between the user terminal and the broadcast network is via the Broadcast Interface Module (BIM). Interactive Interface Module (IIM) provides interface between the user terminal and the interaction channels. A 64-octet slot format is used for transmission of data in the upstream. In the downstream direction, downstream In Band (IB) and downstream Out of Band (OOB) are used. For IB, MAC PDU frames are encapsulated into MPEG2 frames.

DVB/DAVIC supports Internet telephony, VoIP and videoconferencing. At the MAC layer, four access modes are supported. Contention Access mode allows users to send information any time with the risk to experience collisions. The Fixed-rate and Reservation access mode are contentionless. The Interactive Network Adapter (INA) provides a fixed amount of slots to a specific Network Interface Unit (NIU) or a requested bit-rate until NIU demands to stop the connection. In the Ranging Slot Access mode, slots are used for adjusting the time delay and power.

The modulation scheme is QPSK for upstream with data rates of 6176,3088,1544 or 256 kbps. QPSK is also used for downstream OOB and QAM is used for downstream IB [7].

1.3.4 IETF IP over Cable Data Network (IPCDN) Working Group

The IPCDN is defining how Internet Protocol (IP) can be delivered over the cable network. Most of its work is centered on DOCSIS. IPCDN is defining a specification to map both IPv4 and IPv6 into the HFC access networks. The group is interested in multicast, broadcast, address mapping and resolution (for IPv4), and neighbor discovery (for IPv6). IPCDN is also

working on bandwidth management and guarantees using RSVP, security using IPSec, and management using SNMP [9].

1.4 Contributions

The contributions of this thesis are: Design, Implementation and Study of a NS-2 model of the DOCSIS 2.0 MAC protocol and Design and analysis of an algorithm to solve the bandwidth-hog problem in cable networks. The NS-2 model developed supports most of the important features of the protocol and can be extended easily to incorporate new features. The model will be submitted to 'NS-2' maintainers to make it available to the Internet research community. The model can then be used by the researchers to explore various research issues in Cable Networks.

Congestion caused by heavy bandwidth users pose a great concern to the cable service providers. These heavy bandwidth users can cause severe degraded performance for other users. The service providers are considering several options like Service-tiered approach or Bandwidth usage fees to solve this problem. The algorithm presented in this thesis provides an alternative to these pricing solutions and can be effectively used by the Cable Service Providers to do congestion control in the cable network.

1.5 Thesis Layout

Chapter 2 presents an overview of the DOCSIS protocol. Chapter 3 describes the design of the NS-2 model of the protocol. Chapter 4 presents a verification and analysis of the scheduling algorithm. It also presents a study of the performance impact of various system parameters. Chapter 5 describes the bandwidth-hog problem and an algorithm to solve the

problem. Chapter 6 concludes the thesis with a summary of achievements and suggestions on future work.

2. DOCSIS OVERVIEW

This chapter presents an overview of the DOCSIS features and operations [10].

2.1 DOCSIS Evolution

The MCNS group has produced a series of specifications for DOCSIS. The first specifications, known as DOCSIS 1.0 was mainly for high speed Internet access. The standard allowed multiple vendors to build interoperable products resulting in large number of deployment. 23 million DOCSIS 1.0 equipments had been shipped worldwide as of 2002. DOCSIS 1.0 was targeted primarily at the residential market. It lacked enhanced QoS and security features to support multi-media applications like videoconferencing in the Small office/Home office (SOHO) and the small business market.

DOCIS 1.1, the next iteration of the standard, introduced the QoS features required for telephony, videoconferencing, and other services beyond best-effort data service delivery. It is interoperable and backwards-compatible with DOCSIS 1.0. To complement its QoS capabilities, DOCSIS 1.1 also adds additional security features. DOCSIS 1.1 was in the field-trial phase as of January 2003. DOCSIS 2.0, the last in the series of specifications, introduces higher upstream channel bandwidth. In DOCSIS 1.1, the upstream channel bandwidth is much less than the available downstream bandwidth. This greatly limits the scalability of the symmetrical services. DOCSIS 2.0 provides up to 3 times better upstream performance than DOCSIS 1.1. It does this by enhanced modulation and improved error-correction techniques. Table 2.1 presents a comparison of DOCSIS 1.0, 1.1 and 2.0 upstream channel data rate. DOCSIS 2.0 is also backwards compatible with the previous versions.

Table 2.1: Upstream channel rates for DOCSIS 1.0, 1.1 and 2.0[11]

Modulation	Bandwidth (Mhz)	Raw data rate (kbps)	Version
QPSK	1.6	2,560	DOCSIS 1.0/1.1
16QAM	1.6	5,120	DOCSIS 1.0/1.1
QPSK	3.2	5,120	DOCSIS 1.0/1.1
16QAM	3.2	10,240	DOCSIS 1.0/1.1
32QAM	3.2	12,800	DOCSIS 2.0
64QAM	3.2	15,360	DOCSIS 2.0
16QAM	6.4	20,480	DOCSIS 2.0
32QAM	6.4	25,600	DOCSIS 2.0
64QAM	6.4	30,720	DOCSIS 2.0

Cablelabs, a non-profit research and development consortium, has established a certification program for vendors to get official DOCSIS certification if there equipments pass a series of interoperability test. Till Jan 2003, 228 vendors have been certified for DOCSIS 1.0 CM and 29 for CMTS. 64 vendors have been certified for DOCSIS 1.1 CMs and 22 for CMTS. For DOCSIS 2.0, 5 CM vendors have been certified and 1 for CMTS [3].

In the rest of this chapter, we discuss DOCSIS 2.0 protocol features and operation.

2.2 Protocol Features

2.2.1 Interfaces

The DOCSIS specification defines the data communication standard for the cable networks. The service allows transparent bi-directional transfer of Internet Protocol (IP) traffic, between the cable system head-end and the customer locations. The cable network can be all coaxial or hybrid-fiber/coaxial network. Figure 2.1 shows the reference architecture for the data-over-cable services and interfaces.

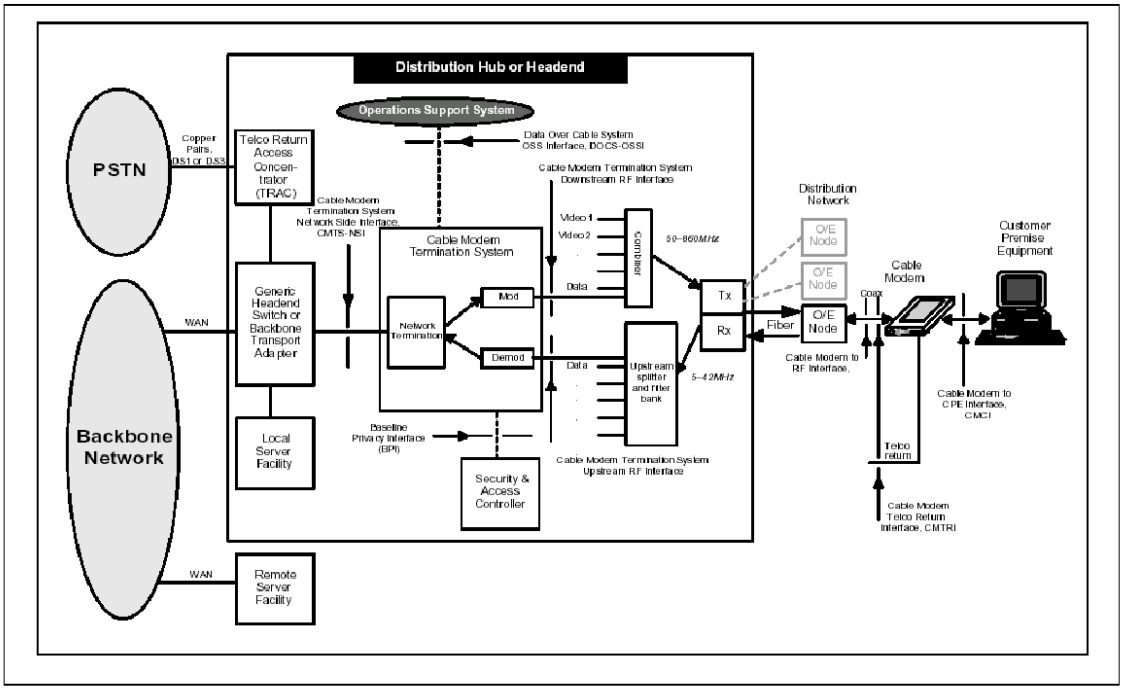


Figure 2.1: Data-Over-Cable Reference Architecture

The following interfaces are defined:

Data Interfaces: There are two data interfaces, namely CMCI [12] and CMTS-NSI [13]. CMCI corresponds to the interface between the CM and the customer premises equipment (CPE). CMTS-NSI is the interface between the CMTS and the data network.

Operations Support Systems Interfaces: This interface defines the service provisioning model and the network management MIBS [14]. The interface is between the network elements and the Operation Support Systems (OSS).

RF Interfaces: RFI is the core interface that defines communication between the CMs and the CMTS. Three RF interfaces are specified:

- Between the CM and the cable network
- Between the CMTS and the cable network, in the downstream direction

- Between the CMTS and the cable network, in the upstream direction

Security Interfaces (BPI): BPI specifies the mechanisms and algorithms used for secure communication between the CMTS and the CMs.

CMTRI: DOCSIS protocol may be deployed over CATV networks that have not been modified to support bi-directional communications. CMTRI defines the communication between the CM and the CMTS when the upstream path is provided by POTS line.

In the rest of this chapter, we focus on the RF interface specifications.

2.2.2 Communication Protocols

In this section an overview of the communication protocols used in data over cable systems is presented.

The CM and the CMTS acts as a forwarding agent and end-systems (hosts). The protocol stack when the CM and the CMTS act as hosts is shown in Figure 2.3. They operate as IP and LLC hosts conforming to IEEE standard 802 for communications. Data forwarding through the CMTS may be transparent bridging or network layer forwarding (routing). If bridging is used by the CMTS, then it must conform to the IEEE 802.1d guidelines for link-layer forwarding. If network layer forwarding is used, then IETF router requirements standard is used by the CMTS. Data forwarding at CMTS happens at two interfaces: between the CMTS-RFI and CMTS-NSI and between the upstream and the downstream channels. The standard does not specifies any address-learning and aging mechanism for the CMTS and is vendor dependent.

Data forwarding through CM is transparent bridging. ISO/IEC10038 forwarding rules are used with some modifications. The protocol stacks at the CMTS and the CM for data forwarding is shown in Figure 2.2.

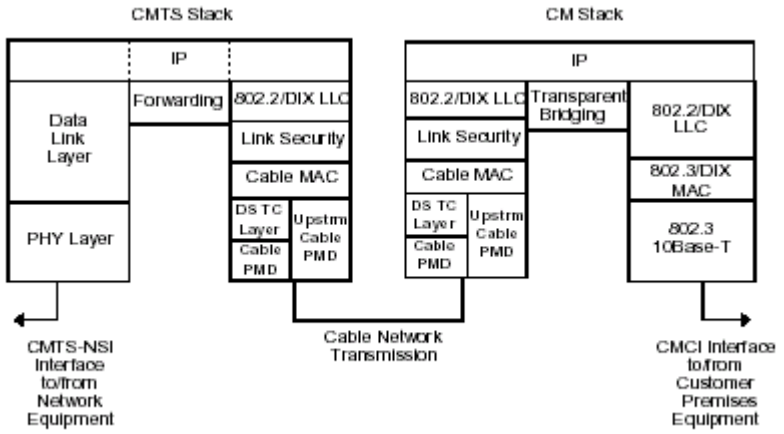


Figure 2.2: Data Forwarding through the CM and the CMTS

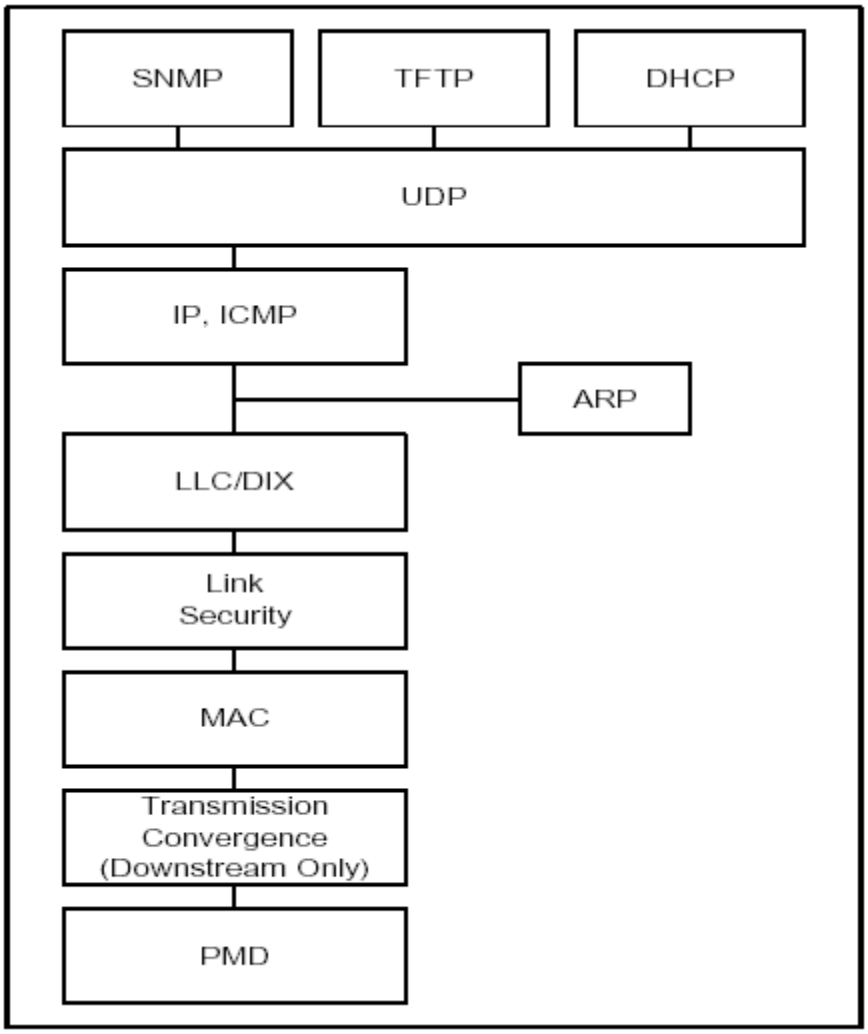


Figure 2.3: Protocol Stack on the RF Interface

The CMTS must either forward ARP packets or provide proxy-ARP service to the CMs.

The network layer protocol is IP version 4 and migrating to IP version 6. Higher layer services can be used transparently over IP.

In accordance with [15], the data-link layer is divided into sublayers with addition of Link-Layer security. Link-layer security is defined in [16]. The MAC sublayer defines a single transmitter for each downstream channel – the CMTS. Each CM is registered on one downstream channel. The CMs listen to all frames transmitted on this downstream channel

and accept those that are destined for the CM or the CPE. CMs communicate with other CMs only through the CMTS. The upstream channel has many transmitters (CMs) and one receiver (CMTS). Time is slotted on the upstream channel and the CMTS provides the time reference as well as controls the usage of each slot.

The Physical layer is comprised of - Transmission convergence sublayer (In downstream direction only) and Physical media dependent sublayer.

In the next section we look at the Physical layer features of DOCSIS 2.0.

2.2.3 Physical Layer

Physical media dependent (PMD) sublayer: The upstream PMD sublayer uses a FDMA/TDMA (TDMA mode) or FDMA/TDMA/S-CDMA (S-CDMA mode) burst type format. Using MAC messages, CMTS configures the mode to be used by all the CMs. FDMA indicates that multiple RF channels are used in the upstream band. A CM is configured to transmit over a single RF channel. This RF channel is shared by a number of CMs via dynamic assignment of time-slots (TDMA). If S-CDMA is used, then multiple CMs can transmit simultaneously on the same RF channel during the same TDMA time-slot. For upstream channels, following modulations are provided for TDMA and S-CDMA : QPSK, 8QAM, 16QAM, 32QAM, 64QAM.

The downstream PMD must conform to ITU-T Recommendations J-83. 64 QAM and 256 QAM modulation is supported on the downstream channel.

Downstream transmission convergence sublayer (DTCS): DTCS is interposed between the downstream PMD sublayer and the data-over-cable MAC sublayer. This is helpful in facilitating common receiving hardware for both video and data and provides an opportunity

for future multiplexing of video and data over the PMD sublayer. The downstream is defined as a continuous series of 188-byte MPEG [10] packets. The packet consists of 4-byte header and 184 bytes payload. The header identifies the type of the payload (Data-over-cable MAC or video). MAC frames may span several MPEG packets and several MAC frames may exist within an MPEG packet.

2.2.4 Media Access Control (MAC) Specifications

"Some of the main MAC protocol features are:

- Bandwidth allocation controlled by the CMTS
- A stream of mini-slots in the upstream
- Dynamic mix of contention- and reservation-based upstream transmit opportunities
- Bandwidth efficiency through support of variable-length packets
- Extensions provided for future support of ATM or other Data PDU
- Quality-of-service features
- Support for Bandwidth and Latency Guarantees
- Packet Classification
- Dynamic Service Establishment
- Extensions provided for security at the data link layer
- Support for a wide range of data rates

A MAC-sublayer domain is a collection of upstream and downstream channels for which a single MAC Allocation and Management protocol operates." There will be one CMTS and

some number of CMs in a mac-domain. Each CM accesses one logical upstream and one logical downstream channel at a time.

Service flows are established between the CMs and the CMTS for communication. A service flow ID defines a unidirectional mapping between a CM and the CMTS. A CM may request one or more service flows to be established during the CM registration or use dynamic service establishment. A CM must establish atleast one upstream and one downstream service flow for basic communication. Active upstream service-flow Ids have associated Service Ids or SIDs. SIDs are used by the CMTS for Quality of service management and bandwidth allocation. The length of Service Flow ID is 32 bits and the length of the SID is 14 bits.

A mini-slot is the unit of granularity for upstream transmission opportunities. The upstream time is divided into intervals by upstream bandwidth allocation algorithm. Each interval is an integral number of mini-slots. For each interval, a usage code describes the type of traffic that can be transmitted during that interval and the physical-layer modulation encoding. For TDMA-mode, mini-slots is a power-of-two multiple of 6.25 microsecond increment. There is no such restriction for S-CDMA mode. The CMTS numbers each mini-slot relative to a master reference. CMTS distributes this master reference to the CMs by means of management messages.

A MAC frame is the basic unit of transfer between MAC sublayers at the CM and the CMTS. The generic frame format is shown in Figure 2.4.

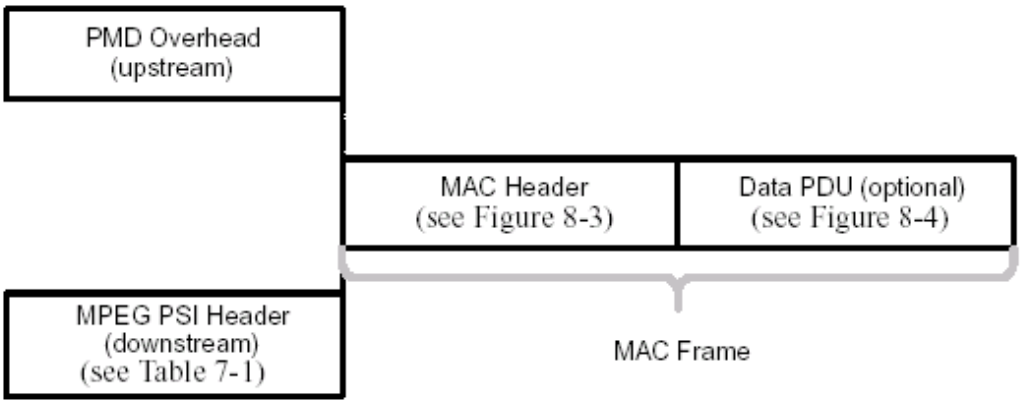


Figure 2.4: Generic Frame Format

The MAC header is preceded by either PMD overhead (for upstream transmission) or MPEG header (for downstream transmission). The MAC header identifies the content of the MAC frame. Following the header is an optional data PDU region. The format of the MAC header is shown in Figure 2.5.

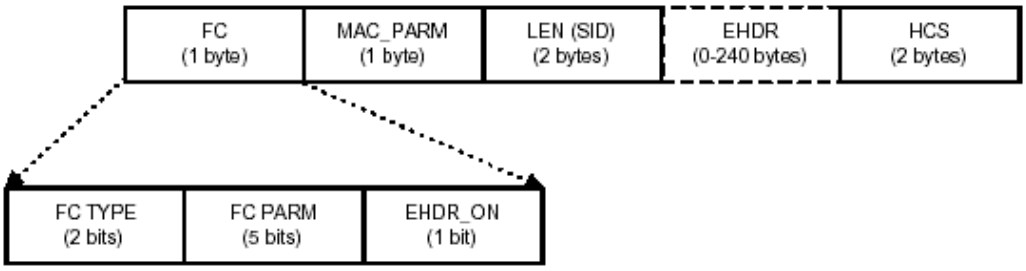


Figure 2.5: MAC Header Format

The FC byte uniquely identifies the rest of the contents within the MAC header. 3 bytes of MAC control follows the FC byte, then an optional extended header field and header check

sequence to ensure the integrity of the MAC header. The FC type defines whether the header is followed by a packet data PDU, an ATM PDU or MAC management message.

The MAC sublayer supports a variable length Ethernet-type packet data PDU. The format of the data PDU is shown in Figure 2.6.

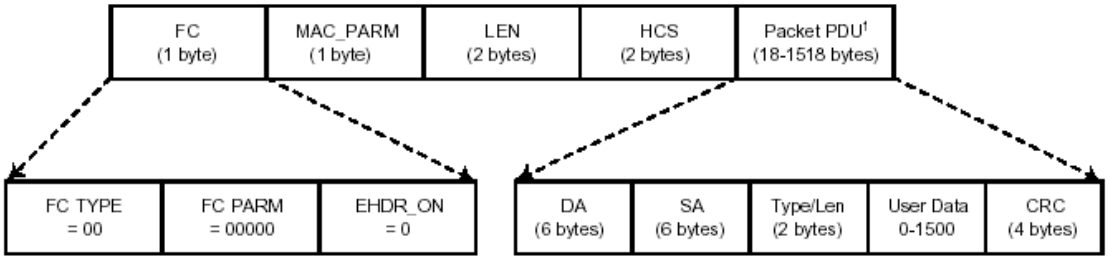


Figure 2.6: Data PDU Format

There are several MAC headers used for very specific functions like support for downstream timing and upstream ranging/power adjust, bandwidth request, fragmentation and concatenating multiple MAC-frames.

2.2.5 Media Access Control Protocol Operation

2.2.5.1 Upstream Bandwidth Allocation

The CMTS controls access to the upstream channel by transmitting MAP messages on the downstream channel which describes, for some interval, the uses to which the upstream mini-slots must be put (shown in Figure 2.7). Depending on the upstream bandwidth allocation algorithm, a given MAP may describe some slots as grants for particular stations to transmit data in, other slots as available for contention transmission, and other slots as an opportunity

for new stations to join the link. The standard does not specifies any particular bandwidth allocation algorithm. CMs may issue requests to the CMTS for upstream bandwidth.

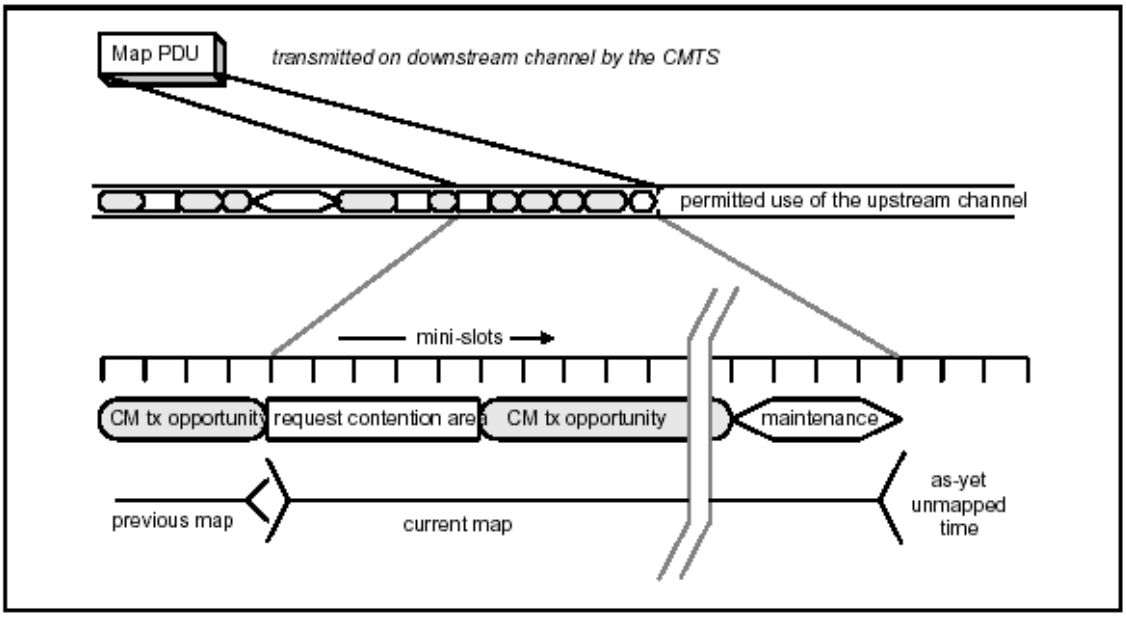


Figure 2.7: Allocation MAP

The allocation MAP message consists of a fixed length header followed by variable number of Information Elements (IEs). Each IE describes allowed usage for a range of mini-slots. Each IE consists of 14 bit SID, 4-bit type code and 14-bit starting offset. SID identifies the service flow that can use the mini-slots.

A number of IEs such as Request IE, Initial maintenance IE, Station maintenance IE, Short and Long data grant IE have been defined. The 4-bit type field identifies the type of IE. Request IE provides an interval in which CMs can make requests for upstream bandwidth. The request IE may be for all the stations or just for one station. If the IE is for all the stations, then CMs contend for sending request. Request/Data IE provides an interval in which the CMs may contend for sending requests or short data packets. Station maintenance

IE provides an interval in which the CMs are expected to do ranging or power adjustments. Short/Long data grant IE provides a upstream transmission opportunity for a CM. These IEs may be issued in response to a CM request or because of an administrative policy. When these IEs are issued by a CMTS with a grant length of 0, this indicates that a request has been received by CMTS and is pending. Every MAP also contains an ACK time that indicates the latest time, from CMTS initialization, processed in upstream. This is used by the CMs for collision detection.

2.2.5.2 Sending Requests

CMs may request the CMTS for upstream bandwidth. This request can be sent either as a stand-alone request frame or it can be piggybacked in the Extended header of another frame. The request frame includes the SID making the request and the number of mini-slots requested. The number of mini-slots requested should be sufficient to transmit an entire frame or a fragment containing the entire remaining portion of a frame that a previous grant has caused to be fragmented. Physical layer overhead must also be considered while determining the request size. A CM can have only one request outstanding at a time per SID. The CM can unambiguously determine that its request is still pending because the CMTS must continue to issue a Data Grant pending in every MAP until the request is granted or denied. In MAPs, the CMTS must not make a data grant greater than 255 mini-slots to any SID.

2.2.5.3 Map Transmission and Timing

MAPs must be transmitted considerably earlier than its effective time. The MAP might be delayed due to propagation delays of downstream channel, Queuing delay within the CMTS or processing delay within the CMs. The number of mini-slots described per MAP

may vary. At a maximum, a MAP is bounded by a maximum limit of 240 Information Elements. Also, a MAP must not describe more than 4096 mini-slots in future. The set of all the MAPs must describe every mini-slot in the upstream channel.

2.2.5.4 Protocol Example

We look at an example to illustrate the interchange between the CM and the CMTS when the CM has data to transmit.

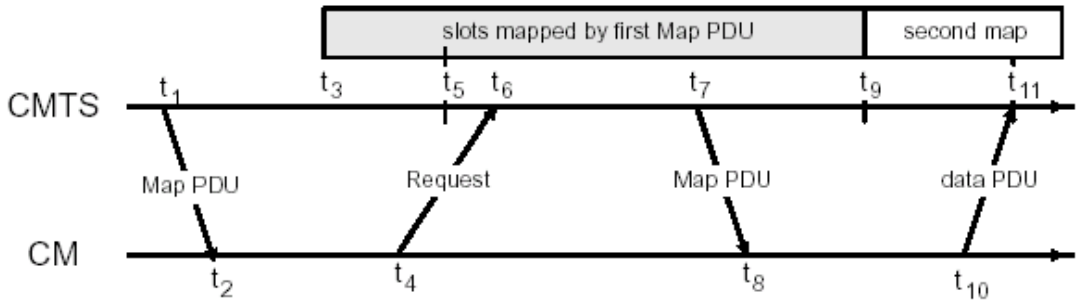


Figure 2.8: Protocol Example

"Description steps:

1. At time t1, the CMTS transmits a MAP whose effective starting time is t3. Within this MAP is a Request IE which will start at t5. The difference between t1 and t3 is needed to allow for all the delays discussed earlier.

2. At t2, the CM receives this MAP and scans it for request opportunities. In order to minimize request collisions, it calculates t6 as a random offset based on the Data Backoff Start value in the most recent MAP.

3. At t4, the CM transmits a request for as many mini-slots as needed to accommodate the PDU. Time t4 is chosen based on the ranging offset so that the request will arrive at the CMTS at t6.

4. At t6, the CMTS receives the request and schedules it for service in the next MAP. (The choice of which requests to grant will vary with the class of service requested, any competing requests, and the algorithm used by the CMTS.)

5. At t7, the CMTS transmits a MAP whose effective starting time is t9. Within this MAP, a data grant for the CM will start at t11.

6. At t8, the CM receives the MAP and scans for its data grant.

7. At t10, the CM transmits its data PDU so that it will arrive at the CMTS at t11. Time t10 is calculated from the ranging offset as in step 3.

At Step 3, the request may collide with requests from other CMs and be lost. The CMTS does not directly detect the collision. The CM determines that a collision (or other reception failure) occurred when the next MAP with an ACK time indicating that the request would have been received and processed fails to include an acknowledgment of the request. The CM then performs a back-off algorithm and retries.

At Step 4, the CMTS scheduler may fail to accommodate the request within the next MAP. If so, it will reply with a zero-length grant in that MAP or discard the request by giving no grant at all. It will continue to report this zero-length grant in all succeeding maps until the request can be granted or is discarded. This will signal to the CM that the request is still

pending. So long as the CM is receiving a zero-length grant, it will not issue new requests for that service queue."

2.2.5.5 Upstream Contention Resolution

The CMs may contend to send request or data PDUs. The contention resolution is based on a truncated binary exponential back-off, with the initial back-off window and maximum back-off window controlled by the CMTS. These values are specified as part of the MAP message. Every time a CM wants to transmit in a contention-region, it enters the contention resolution process by setting its internal back-off window to Data Backoff start. The CM selects a random number within this back-off window. CM will defer this random number of contention transmit opportunities. After the transmission, CM waits for a Data Grant or Data Grant pending or Data Acknowledge. Once this is received, the contention phase is complete. If the CM receives a MAP without a Data Grant or Data Grant pending or Data Acknowledge for it and with an ACK time more recent than its time of transmission, then it determines that its request has been lost. The CM increases the back-off window by a factor of two, as long as it is less than maximum back-off window. The CM again selects a random number within its back-off window and repeats the deferring process. The PDU is discarded after 16 retries.

If the CM receives a Data Grant or unicast Request while deferring, then it stops the contention resolution process and uses the explicit transmit opportunity.

2.2.6 Quality of Service(QoS)

The following QoS concepts were introduced in DOCSIS 1.1 and were not present in DOCSIS 1.0.

- Packet Classification & Flow Identification
- Service Flow QoS Scheduling
- Dynamic Service Establishment
- Fragmentation

The packets traversing the RF MAC interface are classified into a service flow. A Service Flow is a unidirectional flow of packets that is provided a particular Quality of Service.

The CM and the CMTS shape, police and prioritize traffic according to the QoS parameter set defined for the service flow. The QoS parameter set also includes the details of how the CM requests upstream mini-slots and the expected behavior of the upstream scheduler.

Atleast two service-flows are setup for every CM: one for upstream and one for downstream service. The first upstream service flow is also known as the Primary Upstream Service Flow, and is the default service flow for the unclassified upstream traffic.

A service flow is partially characterized by following attributes:

ProvisionedQoSParamSet: This defines the QoS parameters that appear in configuration file and is presented during registration.

AdmittedQoSParamSet: Defines the QoS parameters for which the CMTS and the CM are reserving resources.

ActiveQoSParamSet: The QoS parameters defining the service actually being provided to the service flow. Only an active service flow can forward the packets.

Every change to QoS parameter is approved or denied by Authorization module within the CMTS. Each packet entering the cable network is mapped to a service flow using Classifiers. Classifiers consists of some packet matching criteria (like destination IP addresses), priority

and reference to a service flow. CMTS applies downstream classifiers to the packet it is transmitting and CM applies upstream classifiers to the packets being transmitted on the upstream channel.

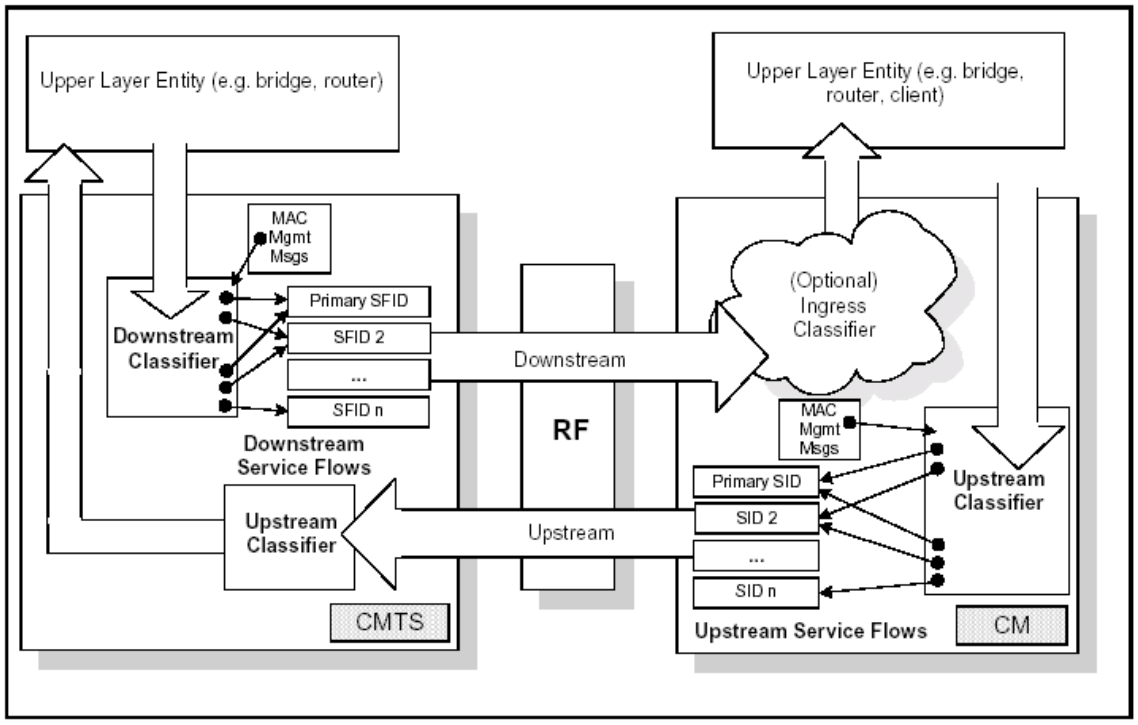


Figure 2.9: Classification within the MAC layer

Figure 2.9 shows the packet classification at the CM and the CMTS. The CM and the CMTS may have multiple packet classifiers. The priority field is used to determine the search order for the classifier. Classifiers can be added via registration or by dynamic operations. The classifier may also be associated with a Payload Header Suppression (PHS) rule. Using PHS, some header bytes of a packet PDU can be omitted, replaced with a Payload Header Suppression Index for transmission and subsequently regenerated at the receiving end.

Registration process is used for static configuration of Classifiers and Service Flows. The CM obtains the configuration file from a provisioning server. CM passes this information to the CMTS in a registration request message. CMTS responds with a registration response message, which is acked by the CM to complete the registration process.

Service flows might be created dynamically by either the CM or the CMTS. A three way handshake is used to create these dynamic service flows. Service flow parameters can also be modified dynamically.

2.2.6.1 Upstream Service Flow Scheduling Services

Upstream scheduling services improve the efficiency of the poll/grant process. The CMTS can provide polls and/or grants at appropriate time based on scheduling service and its associated QoS parameters. The basic services comprise:

Unsolicited Grant Service (UGS): This service supports real-time flows that generate fixed size data packets on a periodic basis, such as Voice over IP. CMTS offers fixed size data grant on a periodic basis to the service flow. This eliminates the overhead and latency of the CM requests. The flow can only use unsolicited data grant and is prohibited from using contention-request or unicast request opportunities. The mandatory service parameters are Grant size, Grants per interval, Nominal grant interval and Tolerated grant jitter.

Real-Time Polling Service (rtPS): This service supports real-time service-flows that generate variable size data packets on a periodic basis. The CMTS provides periodic unicast request opportunities that allows CMs to specify the size of desired grant. Although the request overhead is more than the UGS, but this service supports variable grant sizes for

optimum data transport efficiency. The flow is prohibited from using any contention request or request/data opportunities. The mandatory service parameter is Nominal polling interval.

Unsolicited Grant Service With Activity Detection(UGS/AD):

This service supports UGS flows that may become inactive for substantial period of time, such as Voice over IP with silence suppression. CMTS provides unsolicited grant when the flow is active and unicast requests when flow is inactive. The CMTS might detect inactivity by detecting unused grants. The algorithm used to detect inactivity depends on the CMTS implementation. The flow is prohibited from using any contention request or request/data opportunities. The mandatory service parameters are Grant size, Grants per interval, Nominal grant interval and Tolerated grant jitter.

Non-Real-Time Polling Service: This service supports non real-time service-flows that require variable size data grants on a regular basis, such as high bandwidth FTP. The CMTS might poll on a periodic or non-periodic basis. The flow can also use the contention request or request /data opportunities.

Best Effort Service: This service is for best-effort traffic. The flow can use contention, request/data opportunities and unsolicited requests opportunities.

2.2.6.2 Fragmentation

Fragmentation refers to the process of CM sending fragments of a frame on the upstream channel as sufficient data grant was not allocated by the CMTS to transmit the complete frame. Fragmentation is initiated by the CMTS when it allocates partial grant to a flow. The algorithm to initiate fragmentation is CMTS implementation dependent.

CM fragmentation support: Any time fragmentation is enabled and CMTS allocates partial grant to a flow, the CM fills up the partial grant with maximum amount of data taking fragmentation and physical overhead into account. A special fragment header precedes the data. If there is no grant or grant-pending for the flow, then CM sends a piggyback request. If there is no grant or grant-pending within the ACK time of sending a request, the CM back-offs and re-requests bandwidth for untransmitted portion. This process is continued until the CMTS grants request or the retry threshold is exceeded. The remaining portion of the packet is dropped if retry threshold is exceeded. Each fragment sent has a sequence number used by the CMTS to reassemble the frame.

CMTS fragmentation support: CMTS can perform fragmentation in two modes - Multiple Grant mode and Piggyback mode. In Multiple Grant mode, CMTS retains the state of fragmentation and allows multiple partial grants outstanding for any SID. In piggyback mode, the CM sends a piggyback request for the remaining frame. Once the CMTS reassembles the entire frame, it is processed like a normal packet.

2.2.6.3 Concatenation

This feature allows a flow to concatenate multiple MAC frames and send them as one frame. This allows a “burst” to be transferred across the network. A special concatenation header is used to transmit the burst. The CM may initiate Concatenation when the number of packets in the queue exceeds some threshold limit. The CM requests bandwidth for the entire concatenated frame. If the CMTS allocates partial grant, then fragmentation of concatenated frame takes place. Once the CMTS receives the concatenated frame, individual packets are extracted and processed normally.

3. ARCHITECTURE

This chapter presents the design of a high speed data over cable model implemented in Network Simulator - 2 as part of this thesis. The first section provides a brief overview of the NS-2 simulation package. The design of the model is discussed in the next section.

3.1 Network Simulator (NS) Overview

NS is an **object-oriented discrete-event** simulator used for networking research. NS version 1.0 was developed by the Network Research Group at Lawrence Berkeley National Laboratory. NS development is now a part of the VINT project.

NS is well suited for the simulation of packet-switched networks. There is support for various implementations of TCP, routing, multicast, link layer and MAC protocols.

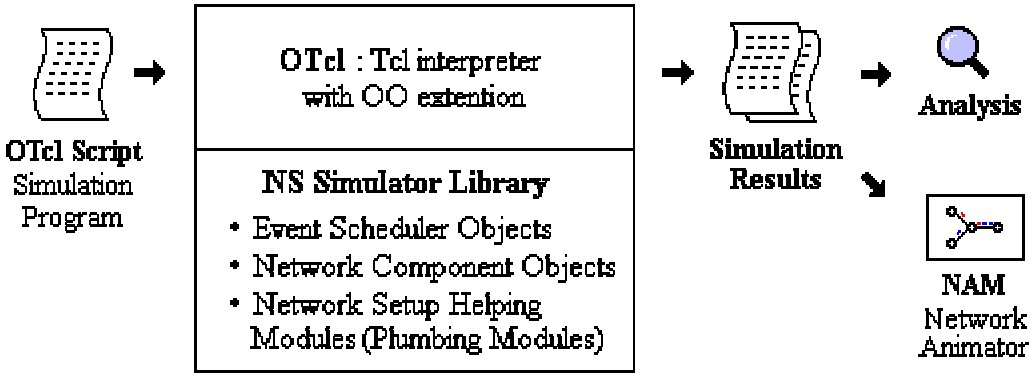


Figure 3.10: User's view of NS

Figure 3.1 shows a simplified user's view of the NS. The OTcl interpreter is the front-end of the NS. To setup and run a simulation, users write an OTcl script that initiates the event scheduler, sets up network topology and configures traffic sources. The event scheduler and

the Network component objects are implemented in C++. There is a compiled class hierarchy of objects written in C++ and an interpreted class hierarchy of objects written in OTcl. There is a one-to-one correspondence between a class in the interpreted hierarchy and a class in the compiled hierarchy.

Event Scheduler

An event in NS is a packet with a unique ID, an expiration time and pointer to the object that handles the event. The event scheduler keeps track of the simulation time and fires all the events in the event-list expiring at the current time by invoking their handlers. Network objects communicate with one another by passing packets. This does not consume actual simulation time. If an object wants to simulate the delay in handling a packet, an event can be issued for the packet to expire after 'delay' time. The current implementation of the simulation engine is **single-threaded** without the support of partial execution of events or preemption. Currently four types of event schedulers are supported: simple linked-list, heap, calendar queue and real-time.

Network Components

Figure 3.2 depicts the partial class hierarchy of NS. The TclObject class is the superclass of OTcl library objects (Scheduler, network component, timers etc). All the basic network component objects that handle packets are derived from the NsObject class. There are

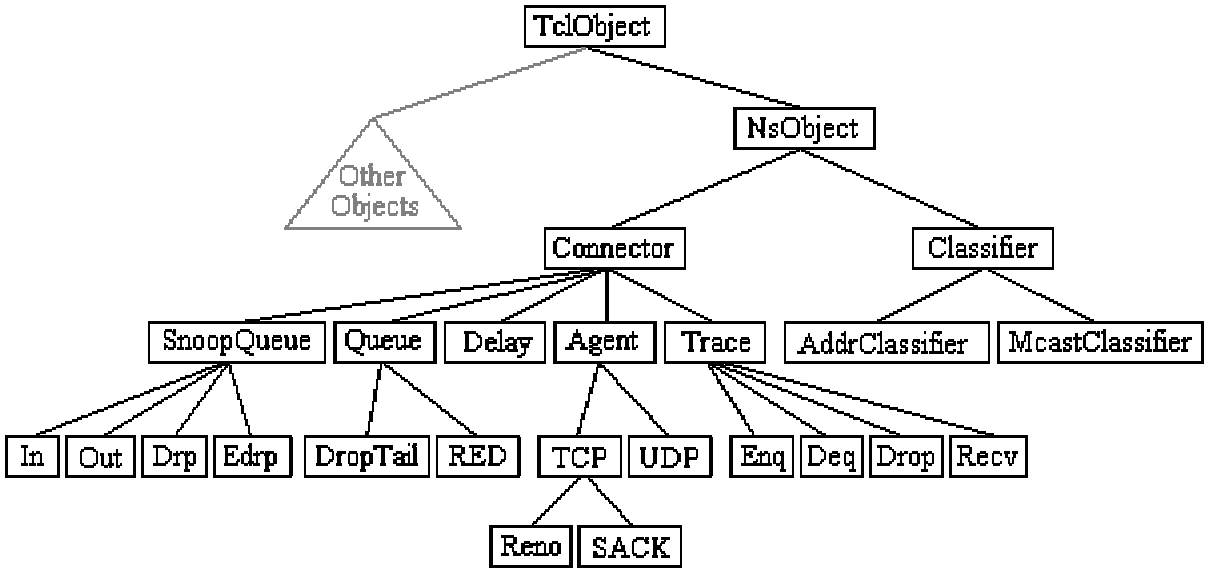


Figure 3.2: Network Simulator Class Hierarchy

two types of basic network objects: *Connector* and *Classifier.* Network objects that have only one output data-path are derived from the *Connector* class, whereas objects, which can have multiple output data paths, are derived from the *Classifier* class.

Nodes

The S*imulator* class provides interfaces to configure, control and operate the simulations. The network topology is created using the standalone classes' *node* and *link.*

Figure 3.3 shows the structure of the node. The function of a node is to receive a packet, process it and forward it to relevant outgoing interfaces. A node is composed of node entry objects, *classifier* and *Agent* objects. Each *classifier* looks at a specific portion of

the packet and forwards it to the next classifier. *Agents* model endpoints of the network where packets are consumed.

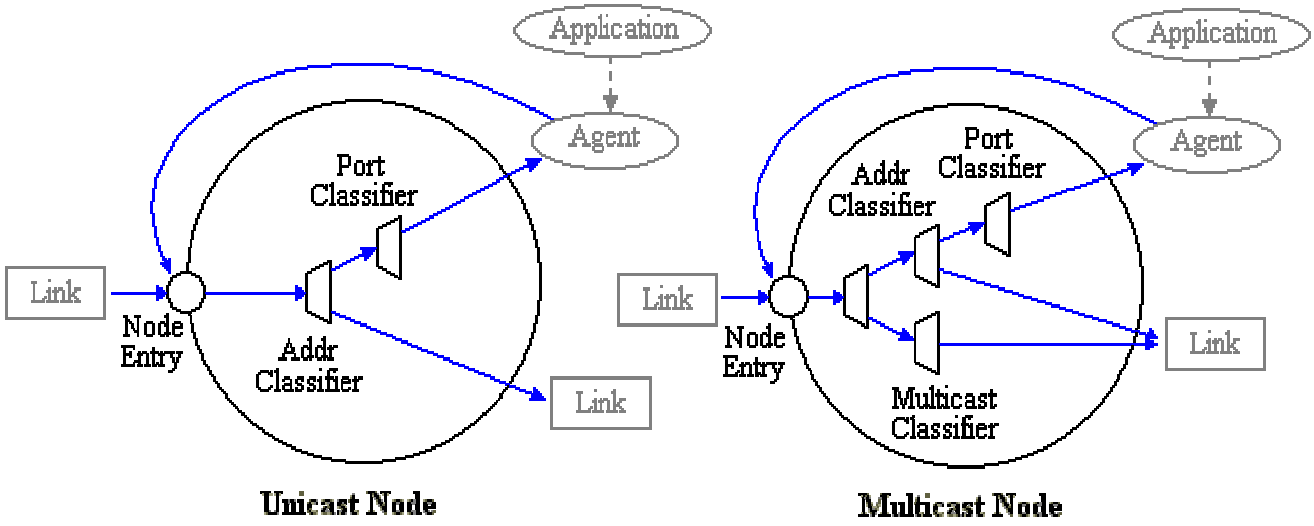


Figure3.3: Network Simulator Node Structure

Links

Links are characterized by bandwidth and delay. When a *duplex-link* is created by the user, two simplex links are created in each direction as shown in Figure 3.4.

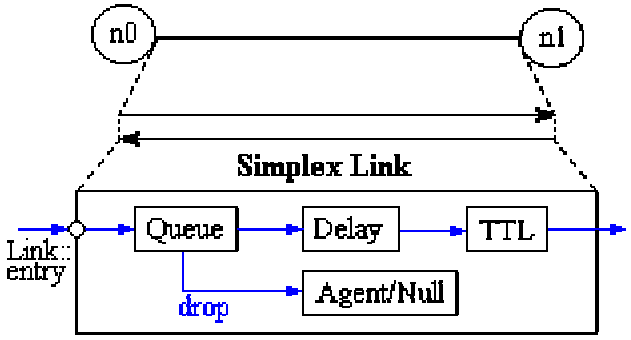


Figure 3.4: Network Simulator Link Structure

The output queue of the node is a part of the *link* object. Deqed packets are passed to the *delay* object to simulate the link delay and the dropped packets are passed to the Null agent

to free their memory. The TTL object updates the Time To Live parameters for each packet received.

Packet

A NS packet is composed of a stack of headers, and an optional data space. (Figure 3.5)

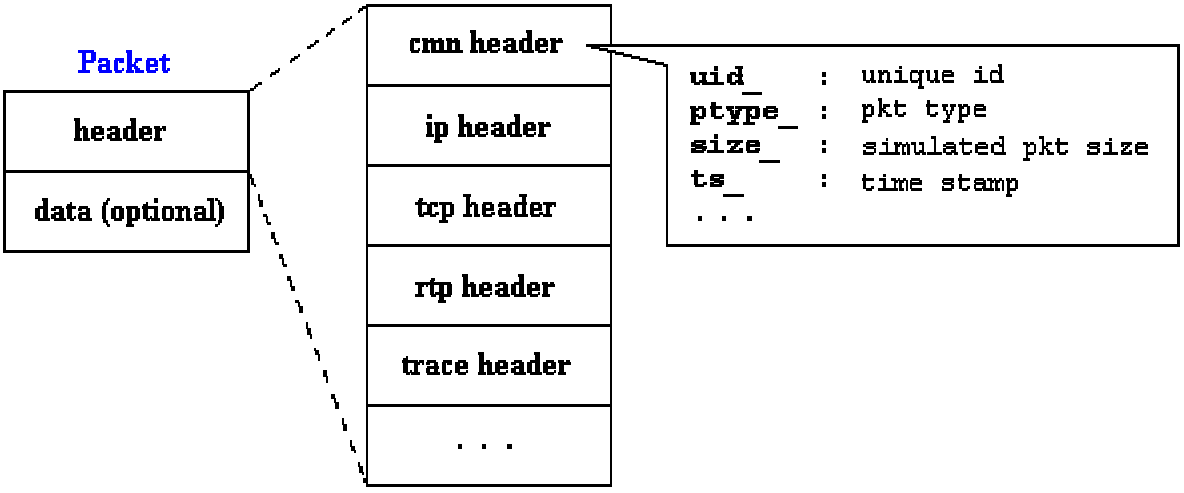


Figure 3.5: Network Simulator Packet Structure

When a simulator object is created, the packet header format is initialized and a stack of all the registered headers is defined and "the offset of each header in the stack is recorded". Whenever a packet is allocated by an agent, a stack composed of all the registered headers is created irrespective of whether or not a specific header is used. Any header can be accessed in the packet using the corresponding offset of the header in the stack [17].

3.2 Network Simulator – 2 Model

3.2.1 Introduction

A NS-2 model for transferring data over cable networks has been developed as part of this thesis. The model supports creation of cable network topologies and provides an

implementation of the Media Access Control (MAC) protocol as per the DOCSIS RF interface specifications [18].

3.2.2 Overview of the Model

Figure 3.6 illustrates a sample network topology using the NS-2 package with our model. NS-2 provides a rich set of node types and application models to accurately simulate a realist Internet environment. We introduce two new node types to NS-2: a cable modem node (CM) and a Cable Modem Termination System node (CMTS). The CM node represents a cable modem deployed at a residential or business location. A CM is limited to a single channel and competes with all the other CMs for upstream bandwidth. The CMTS manages the use of the shared upstream channel and has sole use of the single downstream channel. The simulation model will support any number of CMs to be created over a single DOCSIS network, and further any number of DOCSIS networks can be created.

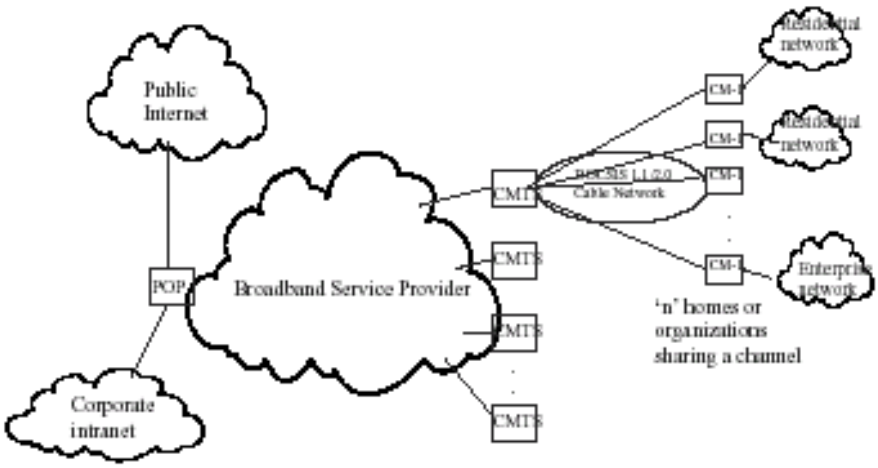


Figure 3.6: DOCSIS Network Topology

The main features of the DOCSIS MAC protocol have been discussed in chapter 2 of the thesis. The main features of our implementation include:

- Time Division Multiplexing (TDM) for the downstream channel
- Time Division Multiple Access (TDMA) for the upstream channel
- Quality of service features:
    - Upstream
        - Packet classification & flow identification
        - Service flow scheduling services
            - Unsolicited grant service (UGS)
            - Real-time Polling (rtPS)
            - Best-effort
        - Fragmentation
        - Concatenation
        - Token-bucket Rate control
    - Downstream
        - Packet classification & flow identification
        - Token-bucket Rate control
- Piggybacking
- Payload header suppression
- Support for one upstream and one downstream channel
- Different types of modulation (data rates)

Following features of the DOCSIS specifications are not supported in the model:

- Dynamic services.

- The physical layer is based on a simple model. Only the effective data rate is modeled. Frame level overheads (preamble, RS bits) are included. The S-CDMA access mode is not supported.
- MPEG video packets interleaving with DOCSIS data packets.
- Link security.

A number of tcl scripts have been provided (presented in the next section) to configure the CM and the CMTS nodes. Once the simulation starts, the CMs will send a Registration request message to the CMTS with the details of the service-flows and related QoS parameters. The CMTS responds by assigning a flow-id to each flow. Once all the flows are registered, the CMTS will start sending a MAP and other management messages periodically. The CM nodes will send data on the upstream channel based on the bandwidth allocated to them. Application generators (sources and sinks) must be created in the model to create the data streams that will send over the DOCSIS network. A number of useful simulation statistics are dumped in a file at the end of the simulation.

3.2.3 User-configurable parameters

In this section, various parameters that have to be configured for creating a cable network topology are presented. The *tcl* scripts to configure these parameters are also discussed.

Upstream Channel Parameters

Following parameters should be configured for the upstream channel:

- Data rate (bits/second)
- Number of ticks per mini-slot: valid values are power of 2

- Upstream Phy. overhead (bits)
- Propagation delay (microseconds)

Based on these parameters, bytes per mini-slot and number of mini-slots per second for the upstream channel is calculated.

Downstream Channel Parameters

Following parameters should be configured for the downstream channel:

- Data rate (bits/second)
- Downstream Phy. overhead (bits)
- Propagation delay (microseconds)

The following script is used to create the cable network topology:

*set docsis [**make-docsislan** "node-list" "Upstream-channel parameters" "Downstream channel parameters" delay LL].*

The node-list contains one CMTS node and a list of CM nodes to be created. The User needs to create a CMTS and atleast one CM node to create a basic topology. The delay parameter specifies the delay (in milliseconds) at the Logical link layer.

CM Node Parameters

The users will be able to create multiple upstream and downstream service flows per CM. Atleast one upstream and one downstream service flow must be created per CM.

The script to create an upstream service-flow is:

*$docsis **configure-upflows** $cm-node "Default-flow Upstream-scheduling-type Destination-node Packet-type PHS-type Fragmentation-enabled Concatenation-enabled Concatenation-*

burst Piggybacking-enabled Grant-size Grant-interval Tolerated-grant-jitter Poll-interval Tolerated-poll-jitter Queue-size Debug".

The list of upstream service flow parameters includes:

Default-flow: Indicates whether this flow will act as a default upstream-flow for the CM, i.e. all the upstream traffic not matching any flows on a CM will be classified on this flow. A value of 1 indicates default-flow and 0 indicates a non-default flow. There must be one default upstream flow per CM.

Upstream scheduling service type: This specifies the upstream scheduling type for the service-flow. The valid values are 0 for UGS, 1 for rtPS and 2 for Best-effort.

Destination-node and Packet-type: The destination-node and the packet-type are used by the classifier to match the packets to a service-flow. Valid values are all the packet-types currently supported in NS-2.

PHS-type: Indicates the rules to apply Payload header suppression. Valid values are 0 for suppress-all, 1 for suppress TCP/IP, 2 for suppress UDP/IP, 3 for suppress MAC and 4 for No suppression.

Fragmentation-enabled: Indicates whether fragmentation is enabled or not for this flow. A value of 1 indicates enabled and 0 indicates disabled.

Concatenation-enabled: Indicates whether concatenation is enabled or not for this flow. A value of 1 indicates enabled and 0 indicates disabled.

Concatenation-burst: This parameter is only used when concatenation is enabled. It indicates the number of packets that must be concatenated together to form a concatenated frame. It must be greater than 0.

Piggybacking-enabled: Indicates whether piggybacking is enabled or not. A value of 1 indicates enabled and 0 indicates disabled.

Grant size and grant interval: These parameters are only valid when the upstream scheduling service is UGS. Indicates the grant size (in bytes) that the CMTS must allocate every grant-interval (in sec).

Tolerated-grant-jitter: This parameter is also valid only when the upstream scheduling service is UGS. Indicates the maximum amount of time that the transmission opportunities may be delayed from the nominal periodic schedule.

Poll interval: This parameter is only valid when the upstream scheduling service is rtPS. Indicates how frequently (in seconds) the CMTS must allocate unicast request grant.

Tolerated-poll-jitter: This parameter is also valid only when the upstream scheduling service is rtPS. Indicates the maximum amount of time that the unicast request interval may be delayed from the nominal periodic schedule.

Queue-size: Specifies the queue size (number of packets) for the service-flow queue.

Debug: The Debug parameter specifies whether debug messages for this flow should be written or not in the LOG file. A value of 1 indicates that debugging is ON and 0 indicates OFF.

The script to create a downstream service-flow is:

$docsis ***configure-downflows*** *$node "Default-flow source-node packet-type PHS-type rate-control rate token-queuelen bucket-size"*

Default-flow: Indicates whether this flow will act as a default downstream-flow for the CM, i.e. all the downstream traffic not matching any flows on a CM will be classified on this flow. A value of 1 indicates default-flow and 0 indicates non-default flow.

Source-node: This specifies the source-node for the service-flow being created.

Packet-type: The packet-type is used by the classifiers to match the packets to a service-flow. Valid values are all the packet-types currently supported in NS-2.

PHS-type: Indicates the rules to apply Payload header suppression. Valid values are 0 for suppress-all, 1 for suppress TCP/IP, 2 for suppress UDP/IP, 3 for suppress MAC and 4 for No suppression.

Rate-control: Indicates whether rate-control is applied on this flow. A value of 1 indicates enabled and 0 indicates disabled.

Rate (in bits/second): This parameter is only valid if rate-control is enabled. It indicates the maximum downstream traffic rate for the flow.

Token-queuelen: This parameter is only valid if rate-control is enabled. Indicates the maximum queue size used in token-bucket rate-control.

Bucket-size (bits): This parameter is only valid if rate-control is enabled. Indicates the maximum burst that can be sent back to back on this flow.

Besides creating upstream and downstream service-flows, CMs must also be configured with the frequency of ranging messages. The script used is:

*$docsis **configure-cm** $node ranging-message-interval cm-id debug*

Ranging message interval: Defines the frequency of ranging messages originating from the CMs (specified in seconds).

cm-id: It is used to uniquely identify a CM when observing LOG messages.

Debug: Specifies whether debug messages for this CM should be written or not in the LOG file. A value of 1 indicates that debugging is ON and 0 indicates OFF.

CMTS Node Parameters

There are a number of parameters that need to be configured for the CMTS. The script used to configure the MAP parameters is:

*$docsis **configure-mapparams** $cmts-node "map-time map-frequency num-management-slots num-contention-slots backoff-start backoff-end map-lookahead"*

Map-time (in seconds): Specifies the time covered by a MAP message.

Map-frequency (in seconds): Specifies the frequency of the MAP message.

Num-management-slots: Indicates the number of management slots to be allocated per MAP.

Num-contention-slots: Indicates the number of contention slots to be allocated per MAP.

Backoff-start: The backoff-start window to be used in the contention phase. Valid value must be a power of 2.

Backoff-end: Specifies the maximum size of backoff window during contention phase. Valid value must be a power of 2.

Map-lookahead: Specifies the maximum number of slots the CMTS can describe over the current map_size. A positive value for this parameter makes the map_time variable.

For configuring other parameters, following script is used:

*$docsis **configure-mgmtparams** $cmts-node "sync-message-interval ranging-message-interval ucd-message-interval Queue-size"*

Sync-message-interval (in seconds): Indicates the frequency of sync messages sent by the CMTS.

Ranging-message-interval (in seconds): Indicates the frequency of ranging messages sent by the CMTS.

Ucd-message-interval (in seconds): Indicates the frequency of ucd messages sent by the CMTS.

Queue-size (in packets): Maximum size of downstream channel queue.

$docsis ***startregistration****:* This command starts the simulation of the registration phase. This must be executed once all the Upstream/Downstream flows have been created for all the CMs and the CMTS has been configured. A complete simulation script has been provided in **Appendix A**.

3.2.4 Simulation Statistics

Users will be able to obtain the following statistics from the simulation.

CMTS Node Statistics

The following statistics can be obtained on a periodic basis or at the end of the simulation:

Upstream channel bandwidth

Upstream Application load

Downstream channel bandwidth

Downstream Application load

Upstream channel utilization

Downstream channel utilization

Total num. of packets received on Upstream

Total num. of packets sent on Downstream

Total packets dropped in DS transmission queue

Total packets dropped in service-flow queues

Total packets dropped

Loss rate

Number of contention requests received

Number of piggyback requests received

Number of requests granted

Per Service-flow statistics:

Total bytes sent on upstream channel

Total bytes received on downstream channel

Total packets dropped

Loss rate

CM Node Statistics

The following statistics can be obtained on a periodic basis or at the end of the simulation:

Upstream channel bandwidth

Upstream Application load

Downstream channel bandwidth

Downstream Application load

Total num. of packets sent on Upstream

Total num. of packets received on Downstream

Total packets dropped in service-flow queues

Total packets dropped due to max retries

Total packets dropped due to queue overflow

Loss rate

Number of contention requests sent

Number of piggyback requests sent

Avg. Queuing delay

Total num. of first collisions

Total num. of collisions

Total num. of fragments sent

3.2.5 CM Node Design

In this section, a detailed design of the CM node functionality is presented.

3.2.5.1 Introduction

The CM node represents a cable modem deployed at a residential or business location. The main functionality of the CM node is to pass the packets between the CPE and the CMTS and vice-versa. A number of management messages are exchanged between the CMTS and the CMs for various purposes (Upstream channel usage control, ranging, time synchronization etc). Most of the management message functionality is not implemented in the model, as it is not required in a simulation environment. However, these management messages are generated periodically to create the necessary overhead. In our model, the traffic sources representing the CPE will be attached to the CM node itself.

When a packet arrives at the DOCSIS layer of a CM node from the higher layers, it will be classified to a particular service flow based on the source-ip, destination-ip and the packet-type (HTTP, FTP, UDP, etc.). Then the DOCSIS header as shown in Figure 2.5 is added to

the packet. A queue is maintained for each service-flow. Depending on the current allocation for that service-flow, the packet may either be queued or sent. For example, if the packet has been classified to a best-effort service flow, then payload header suppression for that service-flow is performed and the allocation table for best-effort service flow will be examined. If there is no allocation or grant pending, then a request frame will be sent and the packet will be queued. Later when upstream bandwidth is allocated to this flow, the queued packet will be sent. When a frame arrives at the CM over the downstream channel (i.e., sent by the CMTS), the frame is classified as a Management message or a Data-PDU using the DOCSIS header and then the payload is handled accordingly.

3.2.5.2 Service flow implementation

As discussed earlier, any packet going out of the RF interface of the CM must belong to a service-flow. Users configure the service-flows with their associated parameters through tcl scripts (discussed in section 3.2.3). Each CM must configure atleast two default service-flows: one for upstream and one for downstream.

When the simulation starts, every CM node registers itself to the CMTS by simulating the registration phase. The CMTS allocates a flow-id to each service-flow. The CMTS uses this flow-id to make bandwidth allocations to the service-flows. The major components of a service-flow are -

Classifier: A classifier is associated with every service-flow. The classifier uses the source-ip, destination-ip and the packet-type to classify a packet to the service-flow. All the other MAC layer management messages and ARP packets are classified on the default service-flow of the CM.

Queue: A queue is maintained per service-flow. All the packets classified on this service flow are enqued if they cannot be immediately sent.

Allocation-table: The allocation-table maintains the current and the future grants for this service flow. This table is updated whenever a MAP message is received on the downstream channel. Based on the type of grant, either DOCSIS frames, contention request or unicast request will be sent on the upstream channel.

Finite state machine: The model supports three types of upstream scheduling services - Unsolicited Grant Service, Real-time Polling and Best-effort. The type of the scheduling service per service flow is configured by the user and is made known to the CMTS via the registration message. The CMTS implements these service-types by its bandwidth allocation algorithm. At the CM node, depending on the service type, a finite state machine controls all interactions of the service-flow for upstream transmissions. These finite state machines have been implemented using a Procedure-driven approach i.e. one function for each input state. When an event occurs, depending on the current state, the proper function is called to process the input event and update the state variable.

The finite state machine for the three service types is presented in the next section.

3.2.5.3 Unsolicited Grant Service

The following **states** have been defined:

Idle: The idle state depicts that there is no packet to transmit for this service-flow.

Decision: In this temporary state, the allocation-table for the service flow is examined for grants.

Wait_For_MAP: This state represents that a MAP with new allocations is awaited.

To_Send: There is a pending transmission of a packet.

The following events have been defined:

Packet: An upper-layer or management packet has been classified on the service-flow to be sent over the upstream channel.

MAP: A MAP management message has been received on the downstream channel with new allocations.

Send_Timer: The Send Timer has expired.

Send_Pkt: Indicates that a packet has to be sent.

The state machine is depicted in Figure 3.7. A detailed description of the state machine follows.

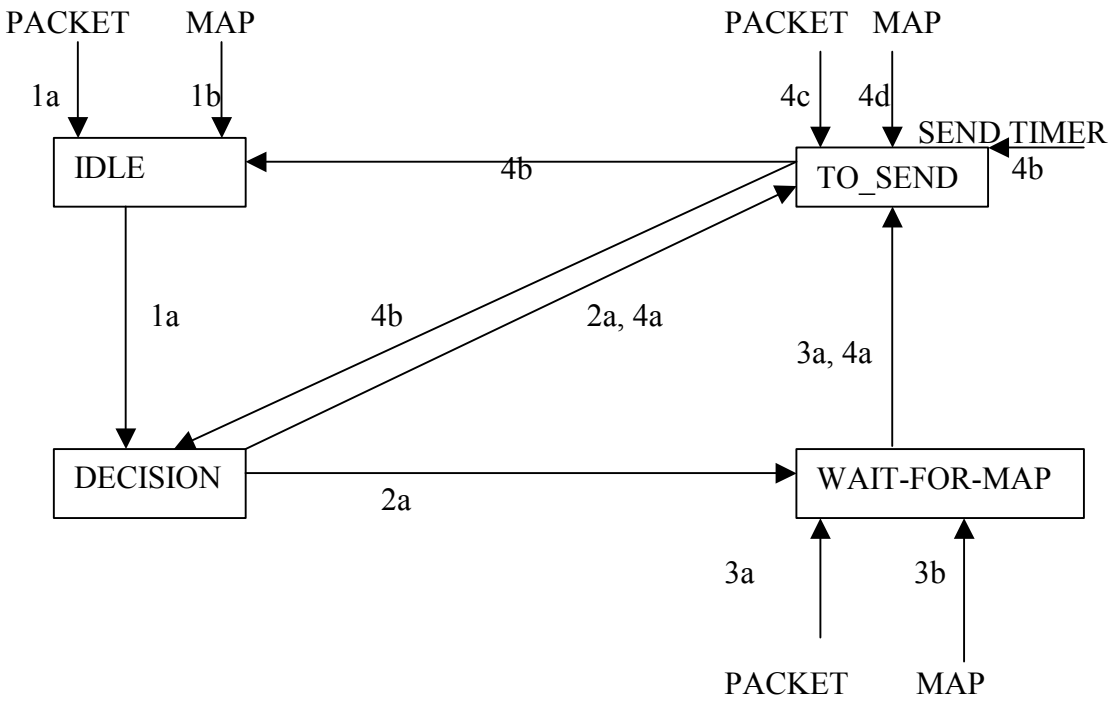


Figure 3.7: FSM for UGS

1. Idle state:

    a. Event: Packet arrival

    Action: The DOCSIS header is added and the packet pointer is stored in the variable current_pkt.

    New state: Decision state with event Packet arrival.

    b. Event: MAP arrival

    Action: Update the allocation-table for this flow.

    New state: No state change.

2. Decision state:

    a. Event: Packet arrival

    Action: Check the allocation table. If no data grants exist the state transitions to Wait_For_MAP. Else, the state transitions to To_Send with

Event Send_Pkt.

New state: Depends upon action routine.

3. Wait_For_MAP state:

   a. Event: MAP arrival

      Action: If there is a data-grant for this flow, then transition to To_Send state with event Send_Pkt. Else, there is no transition.

      New state: Depends upon action routine.

   b. Event: Packet arrival

      Action: Add the DOCSIS headers and enque the packet.

      New state: No state change.

4. To_Send state:

   a. Event: Send_Pkt

      Action: The Send timer is started. It is set to expire when the data grant begins.

      New state: No state change.

   b. Event: Send_Timer

      Action: Send the curr_pkt over the upstream channel. If the queue has become empty, then transition to Idle state.

      Else, a packet is dequed and stored in curr-pkt. The state is changed to Decision state with event packet arrival.

      New state: Depends upon action routine.

   c. Event: Packet arrival

      Action: Add the DOCSIS headers and enque the packet.

New state: No state change.

d. Event: MAP arrival

Action: Update the allocation-table for the flow.

New state: No state change.

3.2.5.4 Real-time Polling

All the states and events described in the UGS section have the same meaning for this finite state machine. In addition, following states and events are defined:

States:

To_Send_Req: Indicates that there is a unicast request grant in the future.

Req_Sent: Indicates that a request has been sent for data grants.

Events:

Request_Timer: The Request timer has expired.

Send_Req: Indicates that a unicast request has to be sent in the future.

The state machine is depicted in Figure 3.8. A detailed description follows:

1. Idle state:

a. Event: Packet arrival

Action: The DOCSIS header is added and the packet pointer is stored in the variable current_pkt.

New state: Decision state with event Packet arrival.

b. Event: MAP arrival

Action: Update the allocation table for this flow.

New state: No state change.

2. Decision state:

   a. Event: Packet arrival

      Action: The allocation table is checked. If there are no data/unicast-request grants, then the new state is Wait_For_MAP. If there is a data-grant in the future, then transition to state To_Send with event Send_Pkt. Else, if there is unicast-request grant in the future, then transition to To_Send_Req with event Send_Req.

      New state: Depends upon action routine.

3. Wait_For_MAP state:

   a. Event: MAP arrival

      Action: The allocation-table is updated. If there is a unicast data grant, then transition to To_Send state with event Send_Pkt. If there is a unicast request grant, then transition to To_Send_Req with event Send_Req. Else, there is no transition.

      New state: Depends upon action routine.

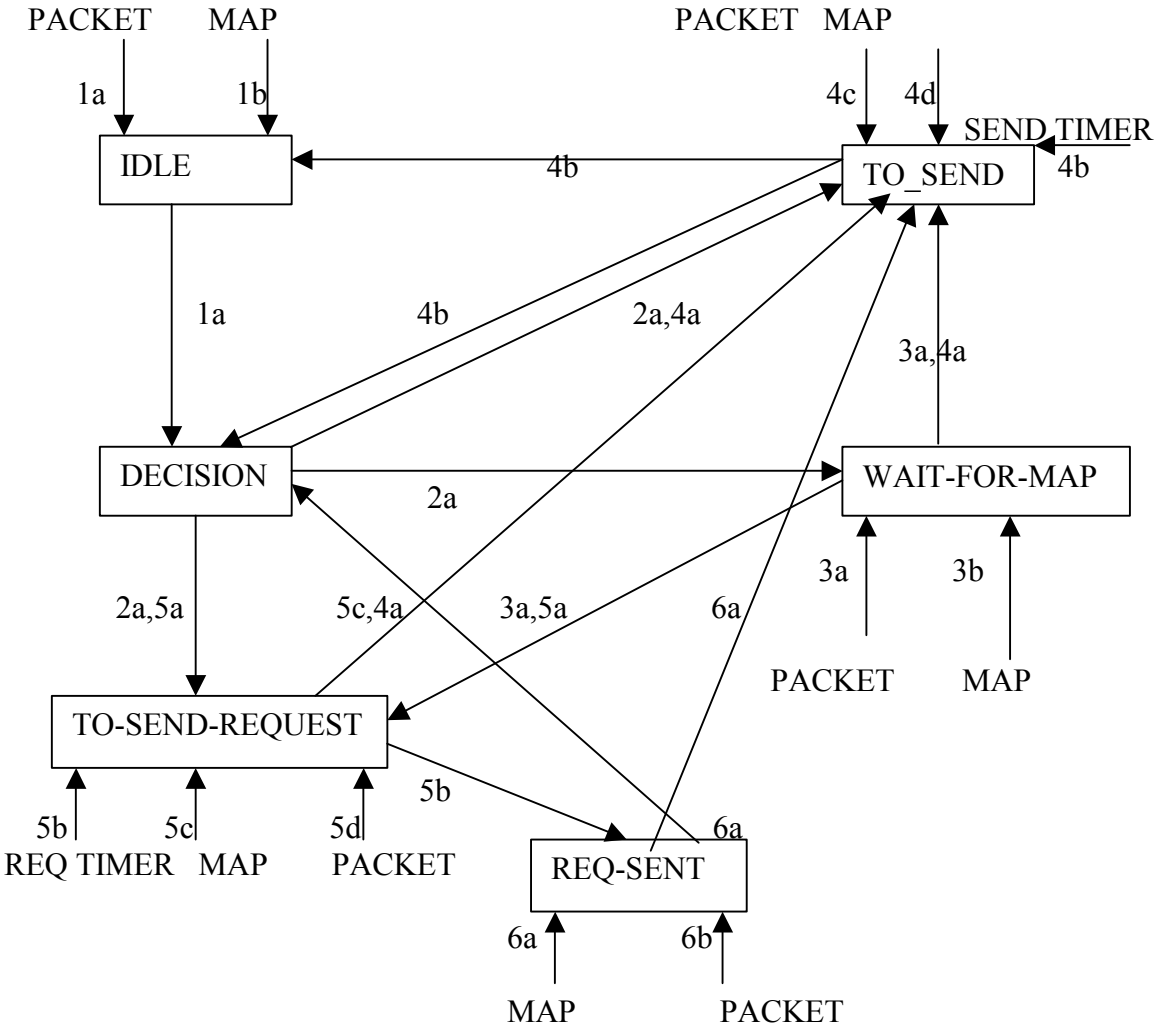


Figure 3.8: FSM for rtPS

b. Event: Packet arrival

Action: The DOCSIS header is added and the packet pointer is stored in the variable current_pkt.

New state: No state change.

4. To_Send state:

a. Event: Send_Pkt

Action: The Send timer is started. It is set to expire when the data grant begins.

New state: No state change.

b. Event: Send_Timer

Action: Send the curr_pkt over the upstream channel. If the queue has become empty, then transition to Idle state. Else, a packet is dequed and stored in curr-pkt. The state is changed to Decision state with event packet arrival.

New state: Depends upon action routine.

c. Event: Packet arrival

Action: Add the DOCSIS headers and enque the packet.

New state: No state change.

d. Event: MAP arrival

Action: Update the allocation-table for the flow.

New state: No state change.

5. To_Send_Req state:

a. Event: Send_Req

Action: Set the Req timer to expire when the allocation starts for sending Unicast Request frame.

New state: No state change.

b. Event: Req timer expires

Action: Send a request frame taking all overheads into consideration.

New state: Req_Sent.

c. Event: MAP arrival

Action: The allocation table is updated. If there is a unicast data grant in the MAP, then the Req timer is stopped and state is transitioned to To_Send with event Send_Pkt. Else, there is no transition.

New state: Depends upon action routine.

d. Event: Packet arrival

Action: The DOCSIS header is added and the packet pointer is stored in the variable current_pkt.

New state: No state change.

6. Req_Sent state:

a. Event: MAP arrival

Action: Update the Allocation-table. If there is a unicast data grant, then transition to To_Send with event Send_Pkt. If there is no data grant pending indicator and the Ack time of the MAP is greater than the time at which the request was sent, then transition to Decision state with event Send_Pkt. Else, there is no state change.

New state: Depends upon action routine.

b. Event: Packet arrival

Action: Add the DOCSIS headers and enque the packet.

New state: No state change.

3.2.5.5 Best-effort Service

In addition to the states and events used in UGS and rtPS section, following new states and events have been defined:

States:

Contention: This state implements the binary-exponential back-off algorithm for contention.

Events:

Unicast_Req: Indicates that a unicast request grant is available for this service flow.

Contention_Req: Indicates that a contention request grant is available for this service flow.

Contention_On: Indicates that the contention phase should be entered.

Contention_Bkoff: Indicates that back-off is required as the contention request was lost.

The state machine is shown in Figure 3.9. The detailed description follows:

1. Idle state:
    a. Event: Packet arrival

       Action : The DOCSIS header is added and the packet pointer is stored in the variable current_pkt.

       New state: Decision state with event Packet arrival.

    b. Event: MAP arrival

       Action: Update the allocation table for this flow.

       New state: No state change.
2. Decision state:
    a. Event: Packet arrival

Action: The allocation-table is checked. If there are no data/unicast-Request/contention grants, then the new state is Wait_For_MAP. If there is a data-grant in the future, then reset the contention_flag and transition to state To_Send with event Send_Pkt. If there is a unicast-request grant in the future, then reset the contention_flag and transition to To_Send_Req with event Unicast_Req.

If there are contention-request opportunities, and the contention_flag is already set, then transition to Contention state with event Contention_Bkoff. Else, the contention_flag is set and there is a transition to Contention state with event Contention_On.

New state: Depends upon action routine.

3. Wait_For_MAP state:

   a. Event: MAP arrival

      Action: The Allocation-table is updated. If there is unicast data grant, then reset the contention_flag and transition to To_Send state with event Send_Pkt. If there is unicast request opportunity, then reset the contention_flag and transition to To_Send_Req with event Unicast_Req. If there are contention slots and the contention_flag is set, then transition to Contention state with event Contention_On. Else, transition to Contention state with event Contention_bkoff. If there are no grants then the state is not changed.

      New state: Depends upon action routine.

   b. Event: Packet arrival

Action: Add the DOCSIS headers and enque the packet.

New state: No state change.

4. To_Send state:

   a. Event: Send_Pkt

      Action: The Send timer is started. It is set to expire when the data grant begins.

      New state: No state change.

   b. Event: Send_Timer

      Action: If Fragmentation is ON for this flow, then send the remaining portion of the packet that is allowed by the current grant size. A piggyback request is also sent if more data has to be sent and there is no grant pending. If the piggyback request has been sent then, the state is changed to Req_Sent. If the entire frame has been sent and the queue is empty, then the new state is Idle state. Else, if the entire frame has been sent and the queue is not empty, then the new state is Decision state with event Packet.

      When fragmentation is not ON and the current grant size is sufficient to send the packet, then the packet is sent on the upstream channel. If a piggyback request was sent on the packet, then the state is changed to Req_Sent. Otherwise, if the queue is empty then transition to Idle state else to Decision state. If the current grant is not sufficient to send the packet, and fragmentation is allowed for this service-flow, then a fragment of the packet is sent with a piggyback request for the remaining portion.

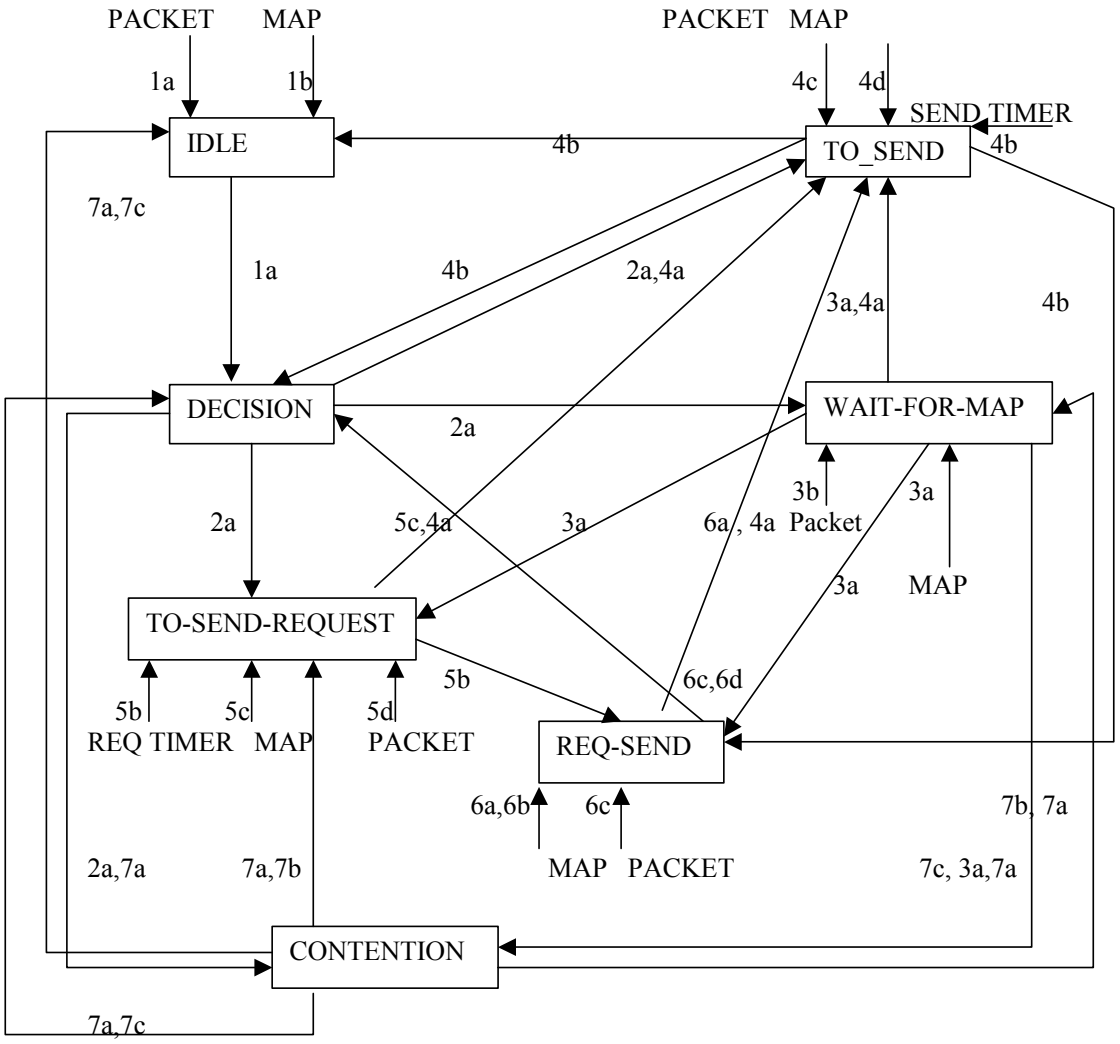


Figure 3.9: FSM for Best-Effort service

The new state is Req_Sent. In case there is another data grant in the future, then Send_Timer is set to expire when the grant begins. There is no transition in this case.

New state: Depends upon action routine.

c. Event: Packet arrival

Action: Add the DOCSIS headers and enque the packet.

New state: No state change.

d. Event: MAP arrival

Action: Update the allocation table for this flow.

New state: No state change.

5. Req_Sent state:

a. Event: MAP arrival and the contention_flag is set

Action: Update the Allocation table. If there is a unicast data grant, then transition to To_Send with event Send_Pkt. If there is a data grant pending, then the contention-flag is reset. If the Ack time in the MAP is greater than the time at which request was sent and there is no data grant pending or data grant, then increase the backoff window by a factor of 2 and transition to Decision state with Packet event. For all other updates, there is no change in the state.

New state: Depends upon action routine.

b. Event: MAP arrival and contention_flag is not set

Action: Update the Allocation table. If there is a unicast data grant, then transition to To_Send with event Send_Pkt. For all other updates, there is no change in the state.

New state: Depends upon action routine.

c. Event: Packet arrival

Action: Add the DOCSIS headers and enque the packet.

New state: No state change.

6. To_Send_Req :

a. Event: Unicast_Req

Action: Set the Req timer to expire when allocation starts for sending Unicast Request frame.

New state: No state change.

b. Event: Req_Timer

Action: Send a request frame taking all overheads into consideration and transition to Req_Sent state.

New state: Req_Sent.

c. Event: MAP arrival

Action: Update the allocation table. If there is a unicast data grant, then the Req timer is stopped and the new state is To_Send with event Send_Pkt. Else, there is no state change.

New state: Depends upon action routine.

d. Event: Packet arrival

Action: Add the DOCSIS headers and enque the packet.

New state: No state change.

7. Contention:

a. Event: Contention_on

Action: The contention_flag is set. A random number 'r' is generated within the current backoff window. If the number of contention slots available is greater than 'r', then the Req timer is set to expire at the ('r' + 1)th contention slot and the state is changed to To_Send_Req. Otherwise,

'skipped' is updated to ('r' – num of contention slots available) and the state is changed to Wait_For_Map.

New state: Depends upon action routine.

b. Event: contention_slots and the contention_flag is set

Action: If the number of contention slots available is greater than or equal to 'skipped', then set the Req timer to expire at ('skipped' + 1)th contention slot and transition to To_Send_Req state. Otherwise, update 'skipped' to ('skipped' – num of contention slots) and transition to Wait_For_Map state.

New state: Depends upon action routine.

c. Event: contention-slots and the contention_flag is not set

Action: The contention_flag is set. A random number 'r' is generated within the current backoff window. If the number of contention slots available is greater than 'r', then the Req timer is set to expire at the ('r' + 1)th contention slot and the state is changed to To_Send_Req. Otherwise, 'skipped' is updated to ('r' – num of contention slots available) and the state is changed to Wait_For_Map.

New state: Depends upon action routine.

d. Event: Contention_bkoff

Action: If the number of retries is greater than the maximum number of retries, then the packet is dropped and the contention_flag is reset. If the queue is empty, then transition to Idle state, otherwise a packet is dequed and the new state is Decision with event Packet.

If the num of retries is less than the maximum number of retries, then the

action of 7a is repeated.

New state: Depends upon action routine.

3.2.5.6 Receiving frames on the RF interface

On receiving a frame, the payload in the frame is classified as Data PDU or Management message. Management messages are ignored (except for the MAP message). The data packet is classified to a service-flow. Once the packet is classified, and Payload header unsuppression is done, packet is passed to the upper-layer.

3.2.5.7 Concatenation and Fragmentation

If concatenation is enabled for a service-flow, then whenever a request is to be sent, the number of packets in the queue is examined. If the number of packets is greater than or equal to a user-configurable parameter 'Maximum concatenation burst', then a request for these many ('Maximum concatenation burst') packets is sent. If the CMTS grants the complete request then the entire concatenated packet is sent in one DOCSIS frame. If there is a partial grant, then the concatenated packet is subjected to fragmentation as any normal packet.

The model supports fragmentation in Piggyback mode. The rules for fragmentation are followed as given in the Specifications [18] .

3.2.6 CMTS Node Design

The CMTS functionality can be broadly classified into the following categories:

- Receiving frames on the RF interface over upstream channel and forwarding it to the destination.

- Receiving packets from the upper layer and passing it to the CM nodes over the downstream channel.
- Managing the upstream channel bandwidth by periodically sending MAPs such that the QoS requirements of all the flows are met.

All the CM nodes created by the simulation script call the **Register** function of the CMTS with their QoS parameters. The model does not support any kind of authorization module to check whether the system can accept the request or not. It is up to the users to design their simulations in such a way that system is not over-provisioned. The CMTS assigns a unique service-flow ID to each CM and stores the flow parameters in a table. The main parameters of a flow stored at the CMTS include: Upstream scheduling service type and its associated parameters, Fragmentation/Concatenation/Piggybacking capability, Flow-id, Classifier and Payload header suppression profile.

3.2.6.1 Receiving/Sending frames on the RF interface

The CMTS is the sole listener on the upstream channel. It maintains the state of the Upstream channel as either IDLE or BUSY. Whenever it is receiving a frame, the state of the channel is set to BUSY. If the MAC layer at the CMTS gets an indication of another frame while the upstream channel is BUSY, it assumes collision and stops receiving the frame. The following sequence of steps is followed once the frame is successfully received:

1) The frame is classified either as carrying a Management message or a Data PDU using the DOCSIS header.

2) Management messages: The Ranging management messages generated by the CM are to account for the overhead they create and their functionality is not implemented. If the frame is carrying a request for bandwidth, then the scheduler's queue (discussed in bandwidth management section) is updated with the request.

3) If the frame is carrying a single complete data PDU (No fragmentation and concatenation), the packet is classified on a service-flow and payload header unsuppression is done. If the destination MAC address is a broadcast or the address of a CM, then the CMTS sends the packet for transmission on the downstream channel. Otherwise, the DOCSIS headers are stripped and the packet is passed to the network layer for forwarding.

4) If the frame is carrying more than one PDU (concatenation) and there is no fragmentation, then each PDU is treated independently (steps 1 to 3).

5) If fragmentation is not ON for the service-flow and the frame header indicates that it is the first fragment, then the data portion of the frame is stored in a reassembly buffer, fragmentation is turned ON for this flow and the sequence number in the header is noted. The extended header is parsed for piggyback request and the scheduler's queue is updated. All the subsequent packets received should be having expected sequence number with fragmentation header. If a frame is received out of sequence or without fragmentation header when fragmentation is on, then the entire packet is dropped and all the subsequent fragment frames received will be discarded. Once all the fragments have been received (indicated by a last fragment flag), the reassembled packet is treated as a normal data packet. (steps 1 to 4)

Receiving packets from upper layers and passing it on the downstream channel: The CMTS receives a data packet from the upper layer and applies the classifier. The DOCSIS header is

added to the packet to make a DOCSIS frame. If the flow on which the packet is classified is being rate-controlled, then depending upon the available tokens, the frame might be enqued in the service-flow queue waiting for more tokens or it will be passed for transmission on the downstream channel. The downstream channel has an associated queue. If the downstream channel is busy, then the frame is enqued if the queue is not full. Once the downstream channel is idle, the next frame is dequed from the head of the queue to be sent.

3.2.6.2 Upstream channel bandwidth management

One of the main responsibilities of the CMTS is to manage the bandwidth of the upstream channel. The Upstream channel time is divided into slots. MAP messages sent periodically by the CMTS on the downstream channel describe the usage of each of the slots. The CMTS may assign data slots to the CMs based on their request or due to a QoS policy. The bandwidth allocation algorithm must make sure that requirements of all the QoS flows are being met. The DOCSIS specification does not specify any particular algorithm. The model supports an algorithm that meets the QoS requirements of UGS and rtPS flow, provided the system is not over-provisioned. For the best-effort traffic, the requests are satisfied in a FIFO manner. The time described by a MAP and the frequency of the MAP are user-configurable parameters. Similarly, the number of contention-slots and management slots per MAP are user-configurable. We refer to the entity running the bandwidth management algorithm at the CMTS as the ‘scheduler’.

Jobs:

The grant allocation requests by the CMs are modeled as jobs of a non-preemptive soft real-time system. There can be two types of the jobs in the system: Periodic corresponding to UGS periodic data grants and rtPS periodic unicast request grants, and Aperiodic corresponding to rtPS and Best-effort bandwidth requests. Every job has a release time, deadline and a period. The release-time denotes the time after which the job can be processed. The deadline denotes the time before which the job must be processed. For periodic jobs, period is used to determine the next release time of the job. In addition to these parameters, a job has several other parameters like flow-id, number of mini-slots requested, upstream scheduling service type etc.

The scheduler maintains three queues of jobs.

The first queue contains all the periodic jobs of UGS and rtPS upstream scheduling type. The jobs in this queue are maintained in an increasing order of relative deadlines. The second queue contains all the bandwidth requests from the rtPS flows stored in a First-in First-out (FIFO) manner. Similarly, the third queue contains all the bandwidth requests from the Best-effort flows stored in a FIFO manner.

UGS and rtPS jobs: Users can configure the following parameters for a UGS flow - Grant size, Grant interval and max-tolerated-jitter. When a CM registers a UGS flow with the CMTS, the CMTS releases a periodic job in the system with the following parameters: Release time = current time, Deadline = Release time + max-tolerated-jitter, Period = Grant interval. After every period, a new instance of the job is released.

The same equations are used for rtPS with the difference that max-poll-jitter is used to determine the deadline. Bandwidth requests by best-effort and rtPS flows are treated as aperiodic jobs.

The scheduler designed in this model uses the following approaches:

a) For periodic jobs, a fixed-priority algorithm is used. The jobs are assigned priority according to their *relative* deadline.

b) For aperiodic jobs, the requests are served in the order of First-in First-out.

The scheduler tries to ensure that the deadline of each periodic job is met. The pseudo code for the allocation algorithm is presented next:

1) Calculate the number of slots 'n' to be described in this MAP. This calculation is done based on the starting and ending time of the current MAP.

2) The jobs in the first queue are maintained in the increasing order of relative deadlines. The jobs are considered in their priority order. For each job, the number of instances that will be released before the end time of the MAP is calculated. Then, allocation for all these instances is done if slots are available. This step is repeated until all the available slots are exhausted or there are no more periodic jobs before the end time of the MAP.

3) Allocation for a fixed number of contention and management slots is done and the value of 'n' is updated.

4) There is a user-configurable parameter 'proportion' which divides the number of available slots between rtPS requests and Best-effort requests.

5) rtPS requests are allocated in FIFO manner until the number of slots for rtPS is exhausted or there are no more rtPS requests.

6) If there are any unused rtPS slots, then they are added to the available Best-effort slots. Best-effort requests are allocated in a FIFO manner until the number of available slots is exhausted or there are no more best-effort requests. Additionally, there is another user-configurable parameter 'MAP_LOOKAHEAD'(in number of slots). This parameter extends the MAP by these many slots if best-effort requests are available. This parameter is only valid when the system has only best-effort jobs.

7) If all the best-effort slots were not used, then they are used for rtPS requests if there are more rtPS requests in the system.

8) All the remaining slots are marked as unused.

3.2.7 Network Simulator – 2 Implementation Details

The object hierarchy of NS-2 was discussed in section 3.1. The new classes/objects defined for this implementation are derived from an existing Mac object in NS-2. The class hierarchy for the model is shown in Figure 3.10.

The MacDocsis base class is derived from the existing Mac class in NS-2. The MacDocsis class is an Abstract class. Two new classes, MacDocsisCM and MacDocsisCMTS are derived from the base MacDocsis class for implementing the functionality of the CM and the CMTS node respectively.

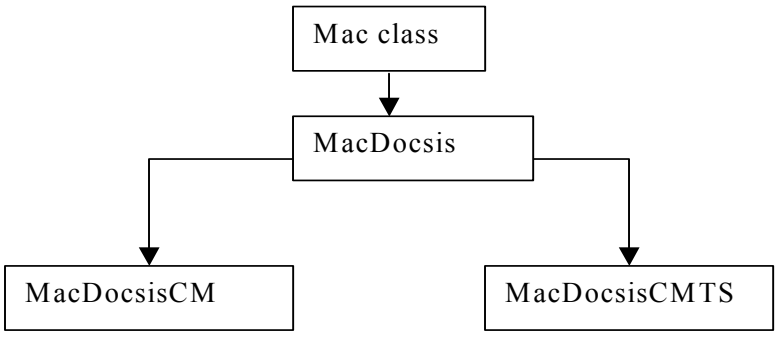


Figure 3.10: DOCSIS Class Hierarchy

In addition, following new packet headers have been added to NS-2: DOCSIS header, Management header, Extended header and MAP header.

In NS-2, the new packet headers that can be declared should be of fixed length. This poses a problem in implementing extended-headers, which may/may not be present in the DOCSIS headers. Also, the number of frames may vary while performing concatenation. To solve this problem, the "data" portion of the NS-2 packet is used. This "data" portion is a variable sized field at the end of a NS-2 packet (after all the headers). This "data" portion is generally not used in most of the implementations, as "real" data need not be transferred for the purposes of simulation.

3.2.7.1 Concatenation Implementation

To send a concatenated frame, a new DOCSIS frame with concatenation header is created. The pointers of all the packets to be sent in the Concatenated frame are copied in the "data" portion of the packet. The size of the Concatenated frame is calculated based on the number

of packets being concatenated. At the CMTS, the packet pointers are used to access the packets.

3.2.7.2 Fragmentation Implementation

To send a fragmented frame, a new DOCSIS frame with fragmentation header and extended header is created. In the "data" portion of the first fragment, a pointer to the original packet is copied and sent. The required number of fragmented frames are then sent with proper sizes. Once the CMTS receives all the fragmented frames with proper sizes, it uses the pointer passed in the first fragment to access the complete packet.

3.2.7.3 Physical Layer Framing

The users configure the upstream and downstream channel physical layer framing overhead using the tcl scripts discussed in section 3.2.2. Whenever the packet is being sent on the upstream/downstream channel, the size of the packet is incremented by the upstream/downstream physical layer overhead to account for physical layer framing.

4. STUDY OF THE MODEL

This chapter is divided into four sections. The first section presents the result of the simulations that were run to verify the basic functionality of the scheduler. The second section presents a study of the scheduler behavior with varying system parameters in the presence of best-effort traffic. The third section illustrates the use of Fragmentation feature by the scheduler. The fourth section presents the results from a performance analysis conducted using the model. The goal of this study was to quantify the performance impact of several key MAC layer system parameters as traffic loads are varied.

4.1 Verifying Scheduler Functionality

The scheduling algorithm implemented in this model was presented in the last section.

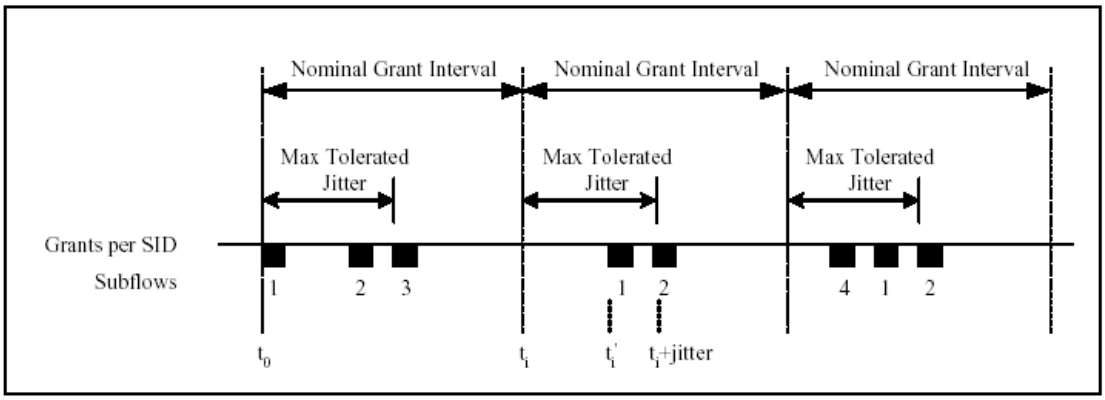


Figure 4.1: Relationship between grant interval and tolerated grant jitter

The scheduler supports the UGS and rtPS upstream scheduling service. To support UGS service type, the scheduler must provide periodic data-grants governed by the service parameters Grant Interval, Grant Size and Tolerated Grant Jitter.

Figure 4.1 shows the relationship between the Grant Interval and the Tolerated Grant Jitter.

The Tolerated Grant Jitter is the maximum difference between the actual grant time (t'_i) and the nominal grant time (t_i). Similarly, for rtPS service type, the scheduler must provide periodic request grants governed by the service parameters polling interval and max-poll-jitter.

To verify that the scheduler meets the requirements of UGS and rtPS service type, three set of experiments were run for both UGS and rtPS service types. The jitter in the grant allocation (difference between the actual grant time and the nominal grant time) is monitored at the CMTS whenever a grant allocation is done. The goal is to show that the jitter observed in grants never exceeds the Tolerated grant jitter (or max-poll-jitter for rtPS).

The first scenario considers the simple case of 1 CM with UGS or rtPS service type. Since there is no other load in the system, the output jitter is expected to be 0 milliseconds (ms). In scenario 2, 5 CMs are created with different UGS or rtPS grant parameters. In scenario 3, the effect of increasing best-effort load is observed on the UGS/rtPS jitter. The number of best-effort CMs are varied from 5 to 450. All the scenarios are based on the the topology shown in Figure 4.2 and model parameters depicted in Figure 4.3. Each periodic data/request grant is referred to as a 'job'.

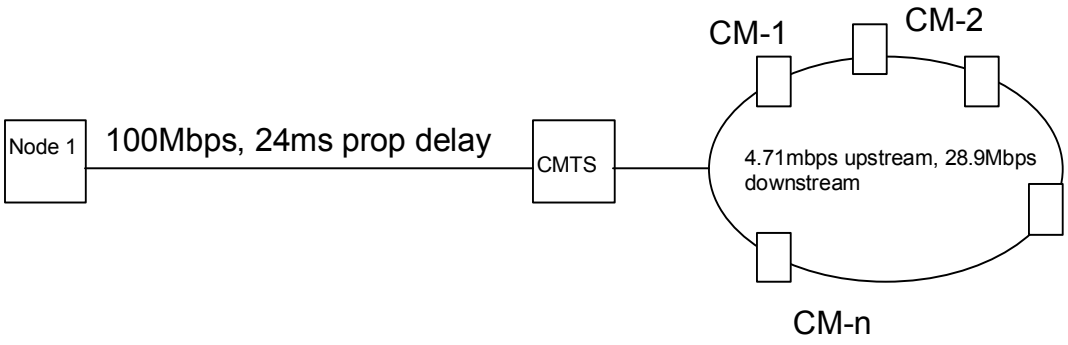


Figure 4.2: Simple Cable Topology

Model Parameters
Upstream bandwidth 4.71 Mbps, 80 bits of physical overhead (preamble, guard time)
Downstream bandwidth 28.9 Mbps, 4 bytes overhead per 188 bytes of data.
4 ticks per minislot (14 bytes/slots)
Map time: 2 milliseconds(80 minislots per map)
Minimum 12 contention slots, 3 management slots
Fragmentation Off, Concatenation Off
Backoff Start: 8 slots, Backoff stop: 128 slots.

Figure 4.3: Model parameters

4.1.1 UGS

For all the UGS scenarios, CMs are configured to generate 500 bytes CBR packets. The CMTS issues a periodic grant of 530 bytes taking all the headers and upstream channel physical layer overhead into account. The 530 bytes grant translates to 38 mini-slots or 950 micro-seconds as per the model parameters. At the CMTS, the jitter between the data-grant start-time and the nominal grant time is recorded. All the UGS jobs are assumed to have the same initial nominal grant time.

Scenario 1 (One CM node with UGS upstream service type):

This is a simple scenario where there is only 1 CM in the system with UGS flow requirements.

The UGS flow parameters were:

Grant interval: 50 ms

Max-tolerated-jitter: 10ms

Output: The jitter was observed to be 0 ms. This is expected as there was no other load in the system.

Scenario 2 (5 CM nodes with UGS upstream service type):

The parameter for the five CMs are shown in Table 4.1.

Table 4.1 UGS flow parameters

CM number	Grant interval(ms)	Max-tolerated-jitter(ms)
1	50	2
2	10	3
3	25	30
4	100	5
5	500	10

The observed jitter is shown in Table 4.2.

Table 4.2: Observed Jitter

CM number	Observed-jitter(ms)
1	0
2	0.19
3	1.79
4	2
5	2.95

Interpretation: The scheduling algorithm is deadline-monotonic with the job having least relative deadline having the highest priority. The relative deadline is equivalent to max-tolerated-jitter parameter of the UGS scheduling type.

In our scenario, the priority order is CM1, CM2, CM4, CM5 and CM3.

So, CM1 jobs never suffer any jitter as there are only UGS periodic jobs in the system and it has the highest priority. CM5 jobs are released at following instants: 0, 500, 1000, 1500 and so on. In all these instants, jobs of CM 1,2 and 4 are always present. Since they have the higher priority, so they get scheduled first. The configuration parameters are such that grant for 2 CMs can fit in a MAP. So, the jobs of CM5 are always delayed by 2 ms (1 MAP_TIME) and grant for CM 4(950 microseconds). So, the jitter of CM5 is 2.95ms.

Similarly, CM4 jobs always have to wait for CM1 and CM2 jobs. So, the CM4 job is delayed by 2ms(1 MAP Time). CM 2 and CM3 jobs suffer delays at only some time instants. E.g., CM2 jobs released at time instants 0,50,100,150,…,500,.. suffer delay due to jobs of CM1. If we consider the avg. jitter till 500ms, then 50 CM2 jobs will be released and they will suffer delay 11 times. So, the avg. jitter is (950microseconds * 11)/50 = 0.2 ms. Similarly, the observed jitter of CM3 can be justified.

The results show the deadline-monotonic behavior of the scheduler.

Scenario 3 (5 UGS flow CMs with varying number of Best-effort CMs):

For this scenario, best-effort traffic is also generated and the number of best-effort CMs is varied from 5 to 450. The best-effort CMs generate FTP traffic in the upstream direction.

The UGS job had the same parameters as in Table 4.1. The observer jitter is shown in Figure 4.4.

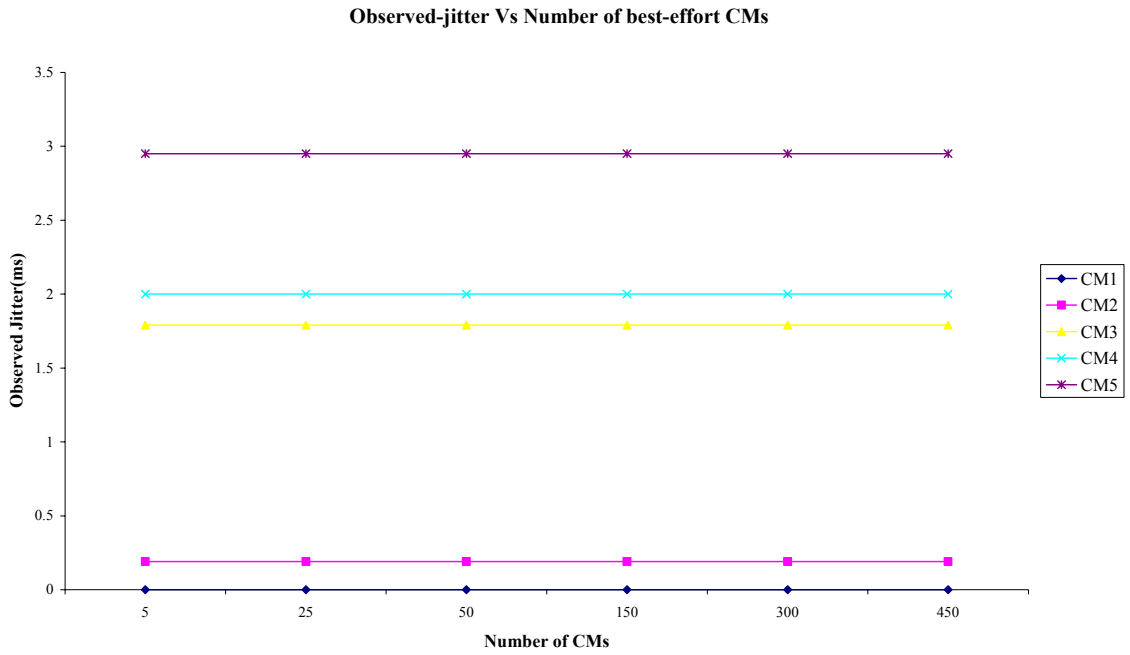


Figure 4.4: Observed jitter vs. Number of CMs

We observe that the jitter remains unchanged from Scenario 2. This is the expected result as the best-effort traffic should not affect the flows that have been promised grant guarantees.

4.1.2 rtPS

For all the rtPS scenarios, CMs are configured to generate 500 bytes CBR packets. The CMTS issues a periodic unicast request grant. According to the system configuration, the request grant requires 2 mini-slots. The size of one mini-slot is 25 microseconds. Hence each CM is assigned 50 micro-seconds of request grant periodically.

At the CMTS, the jitter between the request-grant start-time and the nominal grant time is recorded. All the rtPS jobs are assumed to have the same initial nominal grant time.

Scenario 1 (One CM node with rtPS upstream service type):

This is a simple scenario where there is only 1 CM in the system with rtPS flow requirements. The parameters were:

Polling interval: 50 ms

Max-Poll-Jitter: 1ms

Output: The jitter was observed to be 0 ms. This was expected as there was no other load in the system.

Scenario 2 (5 CM nodes with rtPS upstream service type):

The parameters for the five CMs are shown in Table 4.3.

Table 4.3: rtPS flow parameters

CM number	Grant interval(ms)	Max-tolerated-jitter(ms)
1	50	2
2	10	3
3	25	30
4	100	5
5	500	10

The observed jitter at the CMTS is shown in Table 4.4.

Table 4.4: Observed jitter

CM number	Observed-jitter(us)
1	0
2	10
3	565
4	100
5	150

Again, the result is interpreted using the deadline-monotonic behavior of algorithm. The priority order of the CMs according to the algorithm are: CM1, CM2, CM4, CM5 and CM3.

So, CM1 jobs never suffer any jitter as there are only rtPS periodic jobs in the system and it has the highest priority. CM5 jobs are released at following instants: 0, 500, 1000, 1500 and so on. In all these instants, jobs of CM1, CM2 and CM4 are always present. Since they have the higher priority, so they get scheduled first. So, the jobs of CM5 are always delayed by grants for CM1, CM2 and CM4. So, the observed jitter is 25 * 6 = 150 micro-sec. (Since each request opportunity requires 2 slots and one slot is 25 micro-sec).

Similarly, CM4 jobs always have to wait for CM1 and CM2 jobs. So, the CM4 job is delayed by 25*4 = 100 micro-sec. CM2 and CM3 jobs suffer some delay at only some time instants. E.g., CM2 jobs released at time instants 0,50,100,150,…,500,.. suffer delay due to jobs of CM1. If we consider the delay till 500ms, then 50 CM2 jobs will be released and they will suffer delay 11 times. So, the avg. jitter = (50micro-sec * 11)/50 = 11 microseconds. Similarly, the jitter of CM3 can be justified.

Scenario 3 (5 rtPS flow CMs with varying number of Best-effort CMs):

For this scenario, best-effort traffic is also generated and the number of best-effort CMs is varied from 5 to 450. The best-effort CMs generate FTP traffic in upstream direction.

The rtPS jobs had the same parameters as in Table 4.3. The observer jitter is shown in Figure 4.5.

We observe that the jitter remains unchanged from Scenario 2. This is the expected result as the best-effort traffic should not affect the flows that have been promised grant guarantees.

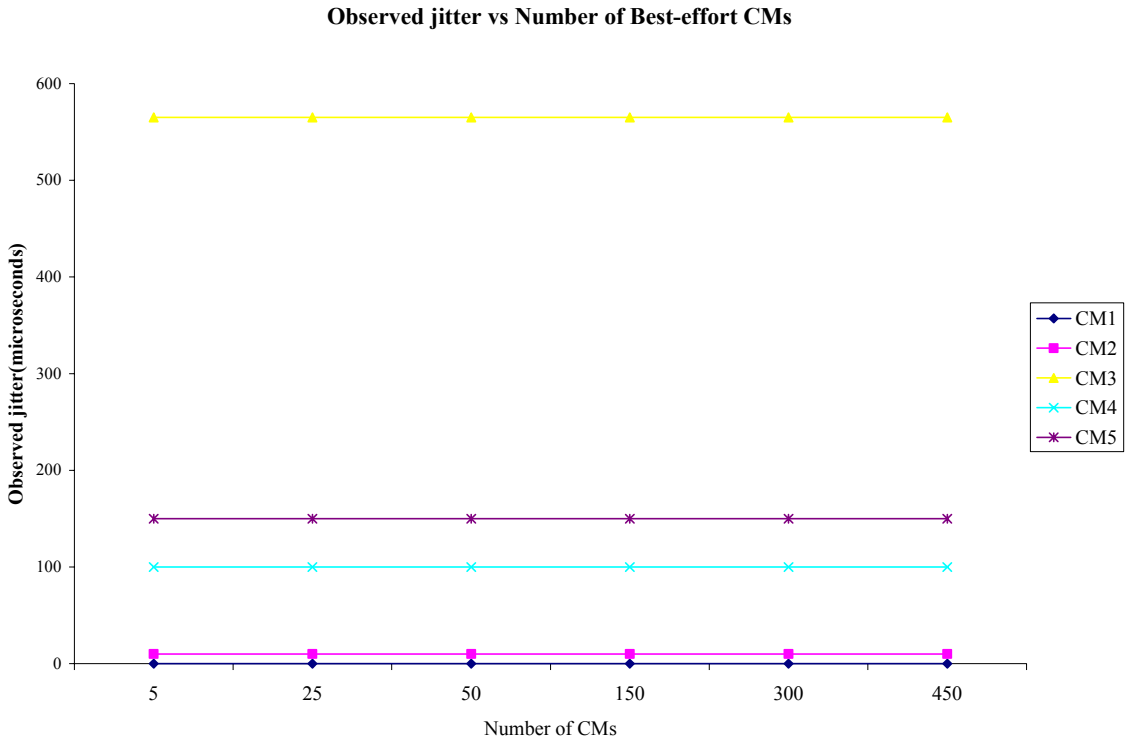


Figure 4.5: Observed jitter vs. Number of CMs

4.2 Scheduling Delay Analysis

In this section we present an analysis of the scheduling delay as various system parameters are varied. The scheduling delay is defined as the delay suffered by a bandwidth-request in the scheduler's FIFO queue. The goal is to observe how various system parameters affect the scheduling delay of bandwidth requests at the CMTS. We only consider the best-effort traffic for the purposes of our analysis. We are interested in observing the effect of MAP_TIME, Concatenation, Piggybacking, Number of contention-slots per MAP and the MAP_LOOKAHEAD as the best-effort load in the system is varied. The number of best-

effort CMs are varied from 5 to 450. The best-effort CMs generate FTP traffic in upstream direction.

Figure 4.2 depicts the topology for all the experiments. The model parameters are listed in Figure 4.6.

Model Parameters
Upstream bandwidth 5.12 Mbps, 80 bits of physical overhead (preamble, guard time) Downstream bandwidth 30.34 Mbps, 4 bytes overhead per 188 bytes of data. 4 ticks per minislot (16 bytes/slots) Map time: 2 milliseconds(80 minislots per map) Minimum 16 contention slots, 3 management slots Fragmentation Off, Concatenation Off, Piggybacking Off, MAP_LOOKAHEAD 255 Backoff Start: 8 slots, Backoff stop: 128 slots.

Figure 4.6: Model parameters

Experiment 1 (MAP_TIME):

MAP_TIME denotes the upstream channel time described per MAP. For best-effort traffic, the model supports a parameter MAP_LOOKAHEAD specified in number of slots during the configuration. This parameter specifies the maximum MAP time scheduler can lookahead. If this is set to 255, then the scheduler will allocate upto 255 slots in the future. The MAP_FREQUENCY parameter specifies how often the CMTS sends a MAP message. Usually these two parameters will be the same (in the 1 – 10 millisecond range).

The MAP_TIME is varied from 1ms to 10 ms and the number of CMs are varied from 5 to 450 for each MAP_TIME. The avg. scheduling delay at the CMTS is monitored.

Figure 4.7 depicts the simulation results.

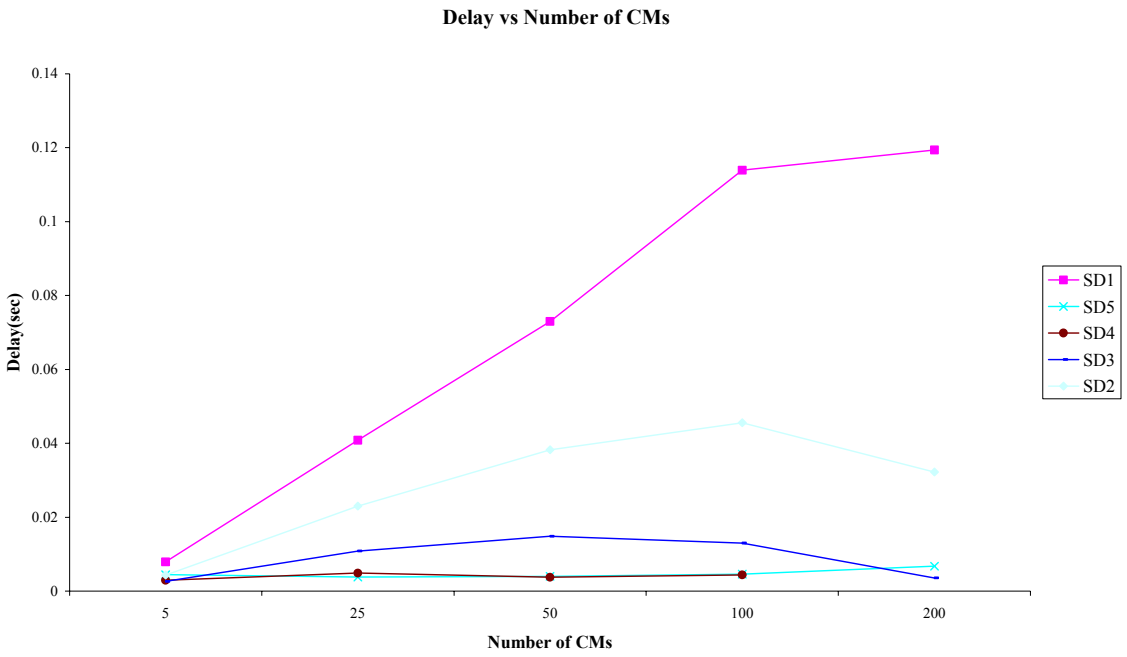


Figure 4.7: Scheduling Delay vs. Number of CMs

SD1: Scheduling delay for MAP_TIME of 1ms

SD2: Scheduling delay for MAP_TIME of 2ms

SD3: Scheduling delay for MAP_TIME of 4ms

SD4: Scheduling delay for MAP_TIME of 8ms

SD5: Scheduling delay for MAP_TIME of 10ms

Analysis:

For low load, the scheduling delay is not affected much by MAP_TIME. However, as the load increases, the scheduling delay decreases with increasing MAP_TIME. This is expected as more number of requests are processed from the scheduler queue per MAP, so the avg. queue size at the scheduler decreases as MAP_TIME is increased. This leads to a smaller avg. scheduling delay.

Experiment 2 (MAP_LOOKAHEAD):

In this experiment, the MAP_LOOKAHEAD parameter is varied. The MAP_LOOKAHEAD is set to two values: 130 and 255. For MAP_LOOKAHEAD 130, one request for 1518 byte TCP packet can be granted per MAP. And for MAP_LOOKAHEAD 255, 2 requests for 1518 byte packets can be granted per MAP. The scheduling delay graph is shown in Figure 4.8.

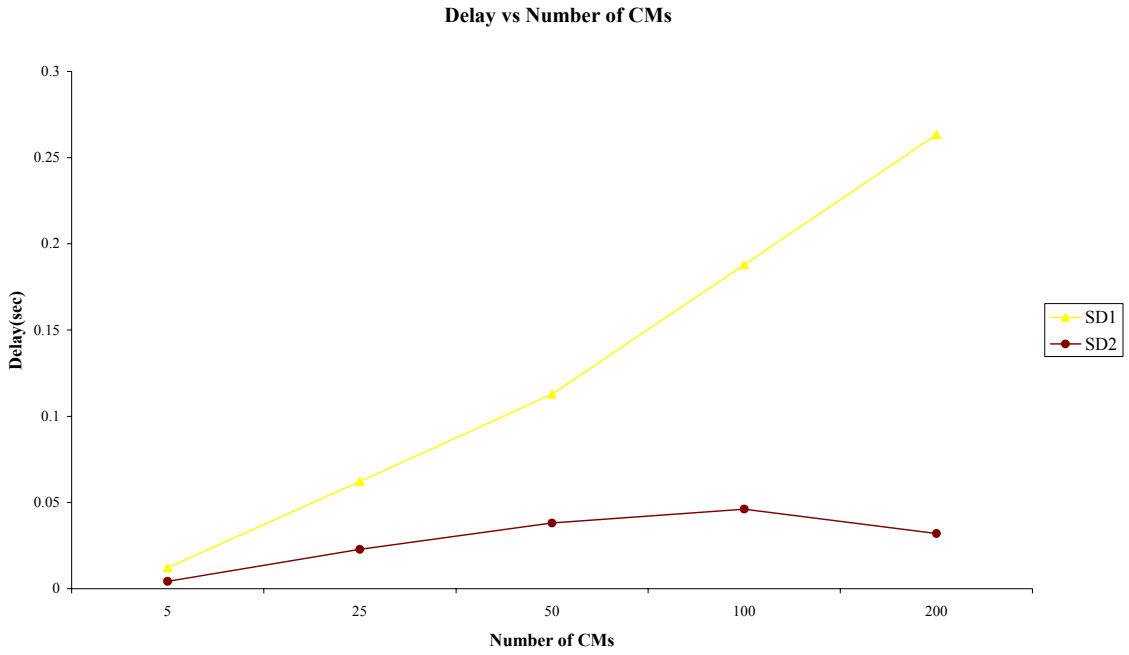


Figure 4.8: Scheduling Delay vs. Number of CMs

SD1: Scheduling delay for MAP_LOOKAHEAD 130

SD2: Scheduling delay for MAP_LOOKAHEAD 255

Analysis:

For low loads (number of CMs less than 5), the scheduling delay is not affected by this parameter. However, with the increase in load, scheduling delay is less when the

MAP_LOOKAHEAD is 255. Since 2 requests are granted per MAP for MAP_LOOKAHEAD 255, so the avg. queue size at the scheduler is less leading to lower avg. scheduling delays.

Experiment 3 (Concatenation):

In this experiment, the impact of concatenation is studied. Two sets of simulations are run. For first set, concatenation is turned-on and for the other concatenation is turned-off. The 'Max-concatenation-burst' parameter is set to 1 when concatenation is on.

The scheduling delay plot is depicted in Figure 4.9.

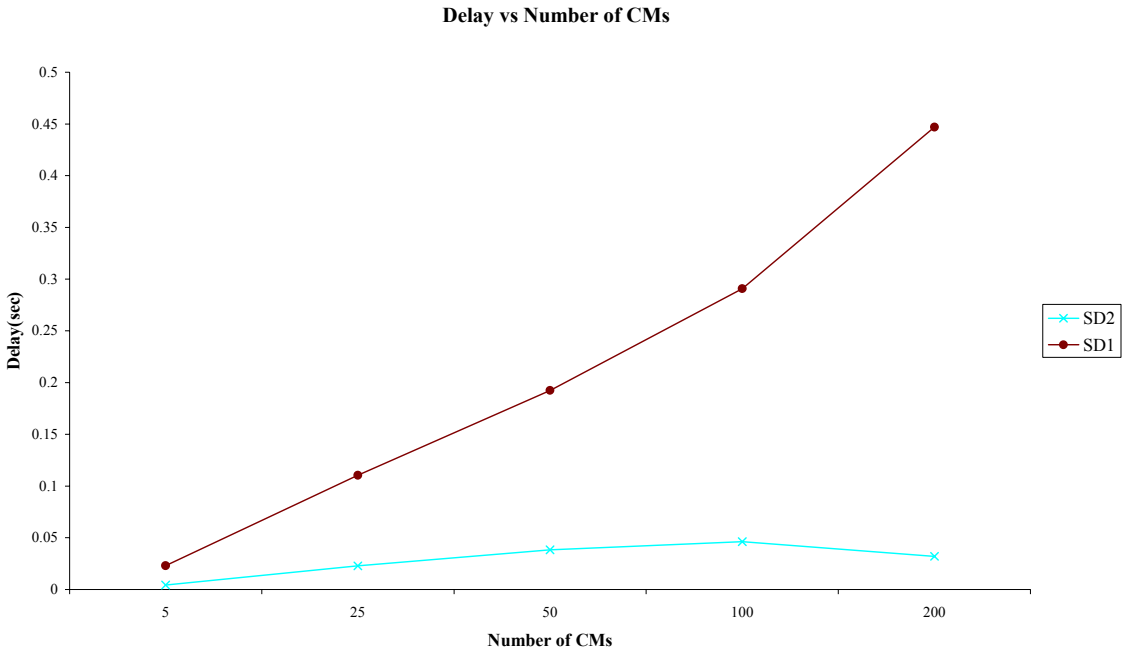


Figure 4.9: Scheduling Delay vs. Number of CMs

SD1: Scheduling delay with concatenation-on

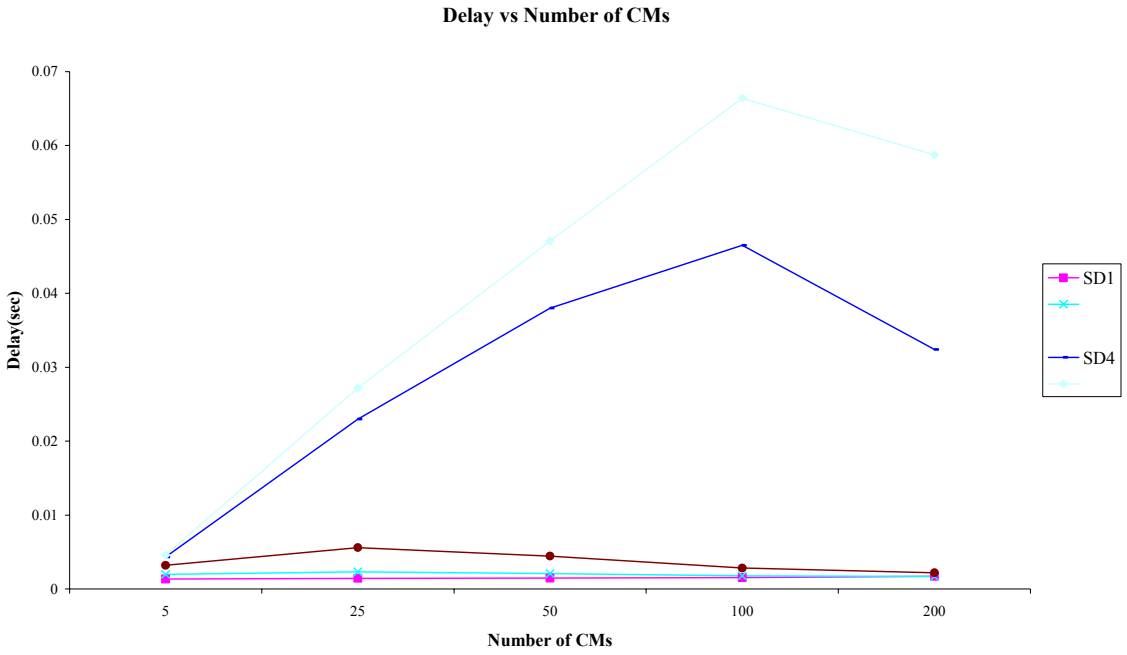
SD2: Scheduling delay with concatenation-off

Analysis:

The scheduling delay is observed to be lower without concatenation for high load. When concatenation is on, each CM requests grants for two 1518 bytes TCP packet in one request. The scheduling delay is higher in case of concatenation as the size of request made by a CM increases. This leads to a higher avg. queuing time in scheduler's queue as lesser number of bigger requests are satisfied by the CMTS.

Experiment 4 (Contention slots):

In this experiment, the number of contention slots per MAP is varied from 3 to 20 slots.



The result is shown in Figure 4.10.

Figure 4.10: Scheduling Delay vs. number of CMs

SD1: Scheduling delay with 3 contention-slots

SD2: Scheduling delay with 6 contention-slots

SD3: Scheduling delay with 9 contention-slots

SD4: Scheduling delay with 16 contention-slots

SD5: Scheduling delay with 20 contention-slots

The scheduling delay is observed to be less for lower values of contention-slots and increases with increase in number of contention-slots.

Analysis: The scheduling delay is less for lower values of contention-slots even with increasing load. This is due to a large number of collisions as the number of contention-slots are less. This was verified by observing the collision-rate at the CMs.

The high rate of collisions leads to lesser number of requests at the CMTS per second and hence lower scheduling delay.

Experiment 5 (Piggybacking):

In this experiment, the effect of piggybacking on the scheduling delay is studied. The result is shown in Figure 4.11.

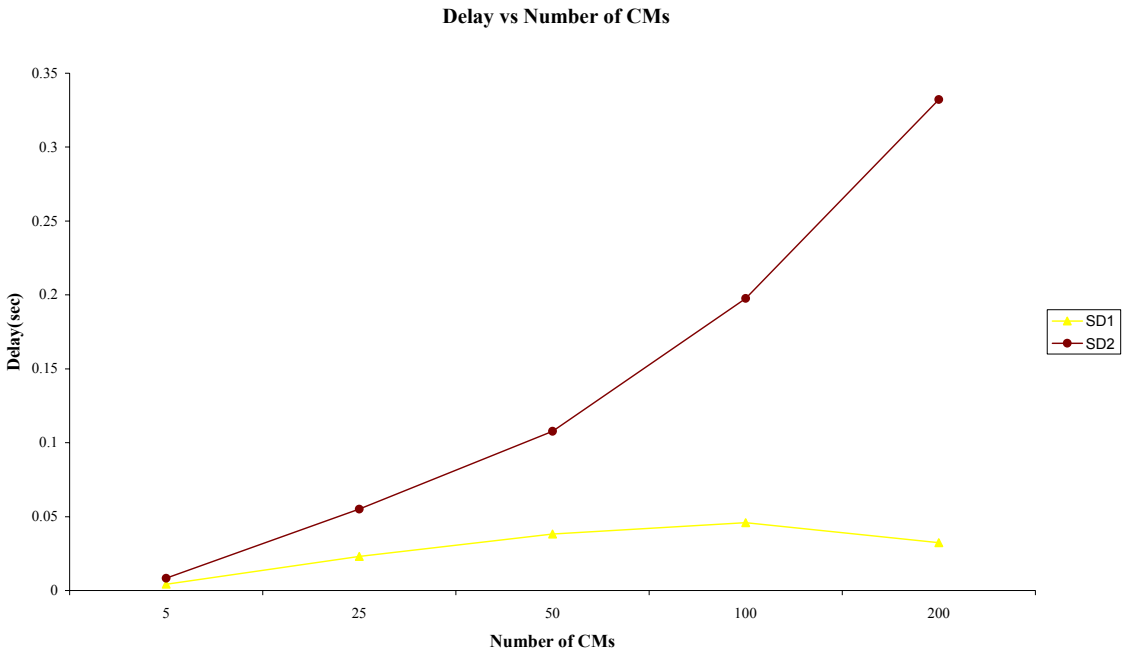


Figure 4.11: Scheduling Delay vs. number of CMs

SD1: Scheduling delay with piggybacking-off

SD2: Scheduling delay with piggybacking-on

The scheduling delay is high with piggybacking.

Analysis: This is expected because with piggybacking the scheduler will receive more number of bandwidth requests as compared to without piggybacking. This leads to higher scheduler queue size and hence higher scheduling delay at the CMTS.

4.3 Use of Fragmentation

The bandwidth allocation algorithm uses fragmentation to increase the upstream channel utilization. Fragmentation is initiated by the allocation algorithm when it allocates a partial

grant to a CM that has fragmentation capability enabled. The use of fragmentation is illustrated with an experiment. The topology used is same as in Figure 4.2. The model parameters are shown in Figure 4.12. Two sets of experiments were run with increasing load. For the first set, fragmentation capability is turned-off for all the CMs. And for the second set, fragmentation capability is on. The number of CMs is varied from 2 to 100. All the CMs send upstream FTP traffic. The packet size is set such that each CM requests 40 mini-slots.

The MAP_TIME is set to 3ms. This translates to 120 mini-slots per MAP. With fragmentation off, atmost 2 requests can be granted per MAP (one request is 40 mini-slots), as 9 slots are reserved for contention and station-management . This leads to a wastage of (120 – 2*40 – 9) = 31 mini-slots per MAP. With fragmentation-on, these 31 mini-slots will be used for partial grant allocation. So, this should lead to a higher upstream channel utilization. This can be confirmed by observing Figure 4.13.

Model Parameters
Upstream bandwidth 5.12 Mbps, 80 bits of physical overhead (preamble, guard time)
Downstream bandwidth 30.34 Mbps, 4 bytes overhead per 188 bytes of data.
4 ticks per minislot (16 bytes/slots)
Map time: 3 milliseconds(80 minislots per map)
Minimum 16 contention slots, 3 management slots
Concatenation Off, Piggybacking Off,
MAP_LOOKAHEAD 0
Backoff Start: 8 slots, Backoff stop: 128 slots.

Figure 4.12: Model parameters

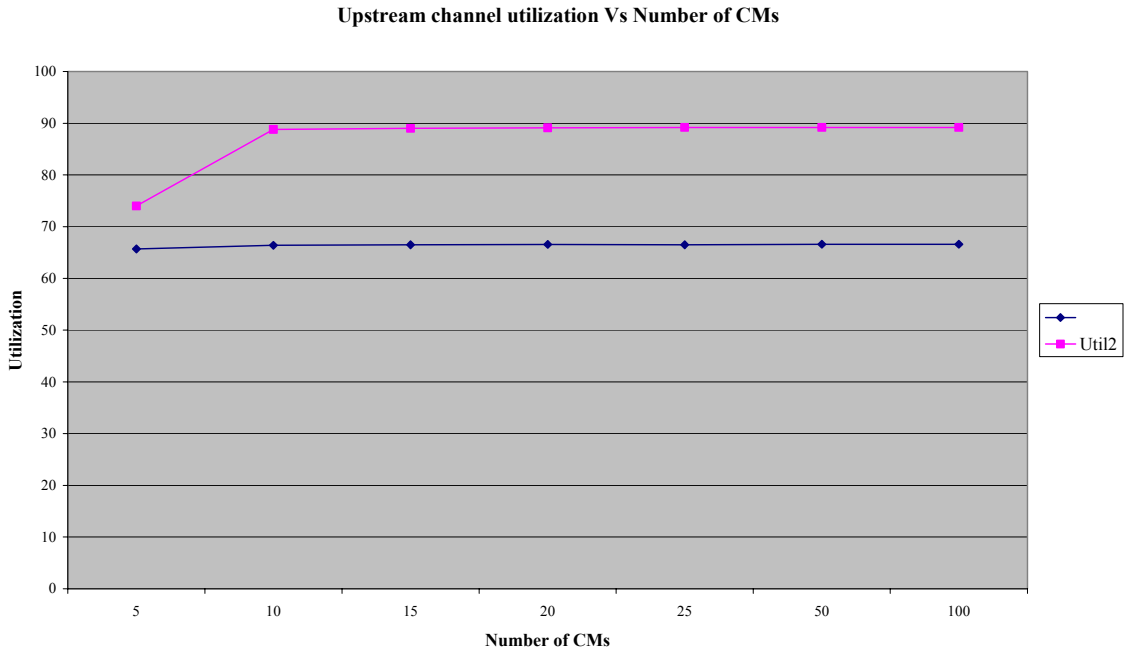


Figure 4.13: Upstream channel utilization vs. Number of CMs

Util1: Upstream Channel Utilization without fragmentation

Util2: Upstream Channel Utilization with fragmentation

4.4 Performance Study

The goal of this study is to quantify the performance impact of several key MAC layer system parameters as traffic loads are varied. The focus is on the best effort traffic.

Figure 4.14 illustrates an upstream transmission of a 1500 byte IP datagram for a particular configuration. Although the nominal MAP size is 80 slots (based on the system parameters listed in Figure 4.16), 96 slots are required to carry the entire packet. The IP packet arrives at the CM at time T-0. The CM sends the bandwidth request message at time T-1 and receives the data grant at time T-2 when the MAP describing MAP time 3 arrives.

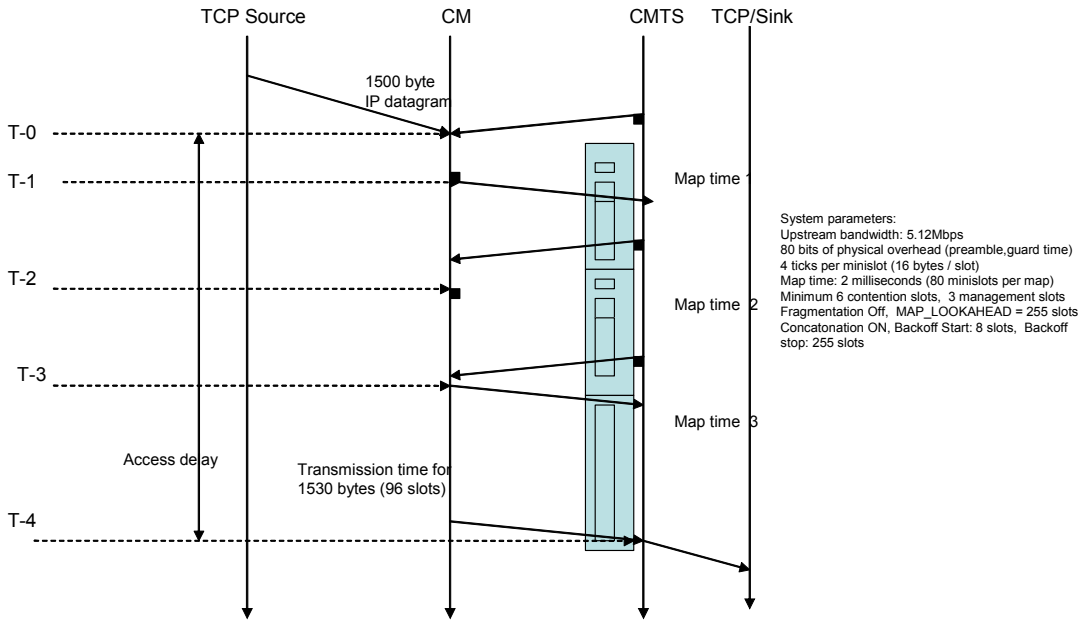


Figure 4.14: Upstream transmission

The CM sends the frame at Time T-3 and is received by the CMTS at time T-4. The time between T-4 and T-0 is the access delay that represents the total time a packet is delayed over the DOCSIS access network. The best case scenario illustrated in Figure 4.14 assumes no system delay. The minimum access delay is roughly 2 MAP times plus the transmission time of the frame. Assuming a MAP time of .002 seconds, framing and physical layer overhead of 30 bytes and an upstream bandwidth of 4.7Mbps (adjusted for FEC overhead), the highest TCP throughput to be expected is roughly 1.8Mbps. This has been verified with the simulation model. In reality, the arriving packet will rarely be allowed to send a contention-based request in the same MAP time as the arrival due to the contention algorithm. Piggybacking can be used to minimize this additional delay. If the CM has multiple packets to send upstream it can piggyback a request for bandwidth on a data frame. This has significant benefits in a busy system as it avoids the contention process leading to lower channel collision rates.

The notion of the MAP_LOOKAHEAD was developed specifically to optimize for TCP's widely used delayed ACK option that leads to frequent back-to-back transmissions of segments. In this case, the CM will issue a concatenated request for a 3000 byte transmission. Again, assuming best case conditions, the minimum access delay would be 2 MAP times plus the transmission time of two concatenated frames. This is roughly 2.6Mbps which again has been validated in the model.

The analysis is based on 3 system scenarios. The first two involve the simple cable network illustrated in Figure 4.15. The third involves a more realistic model based on the 'flexbell' model presented in [19]. Each scenario utilizes a different workload model. Scenario 1 and 2 are used to describe and analyze the model when subject to varying level's of FTP traffic in the upstream and downstream directions respectively. Scenario 3 involves realistic workloads (web, P2P and streaming) primarily in the downstream direction. For each scenario three experiments were run with a different system parameter under study: the MAP_TIME, the number of contention slots per MAP (i.e., CONTENTION_SLOTS) and the contention backoff start range (BKOFF_START) respectively. The MAP_TIME varies between .001 and .01 seconds. The CONTENTIONS_SLOTS varies between 3 and 20 slots per map. The BKOFF_START ranges from 8 slots to 128 slots. For each experiment, specific simulation runs were conducted to show system performance for these values of the system parameters as the traffic load increased (i.e., as the number of CM's increased).

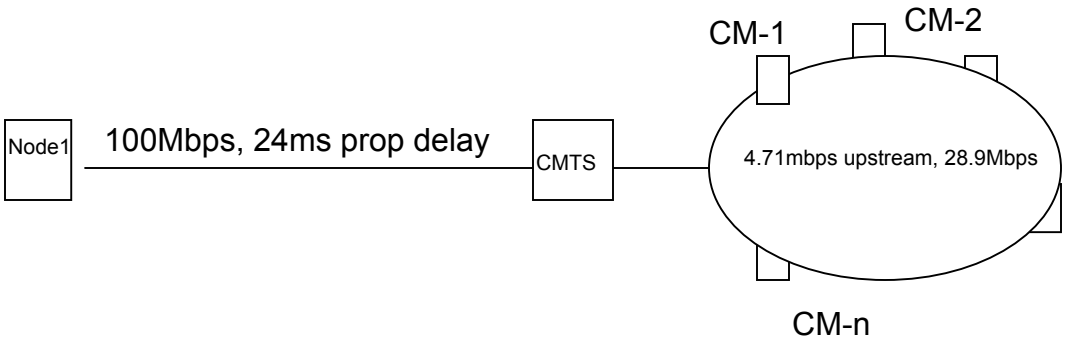


Figure 4.15: Simple Cable Scenario

Model Parameters

Upstream bandwidth 4.71Mbps, 80 bits of physical overhead (preamble,guard time)
Downstream bandwidth 28.9Mbps, 4 bytes overhead per 188 bytes of data.
4 ticks per minislot (16 bytes / slot)
Map time: 2 milliseconds (80 minislots per map)
Minimum 8 contention slots, 3 management slots
Fragmentation Off, MAP_LOOKAHEAD = 255 slots
Concatonation ON
Backoff Start: 8 slots, Backoff stop: 128 slots

Experiment Summary

Experiment 1: Varies the MAP_TIME as follows:

Run#	1	2	3	4	5	6
MAP_TIME	.001	.002	.003	.005	.007	.01

Experiment 2: Varies the CONTENTION_SLOTS as follows:

Run#	1	2	3	4	5	6
#CS's	3	5	8	12	16	20

Experiment 3: Varies the BKOFF_START as follows:

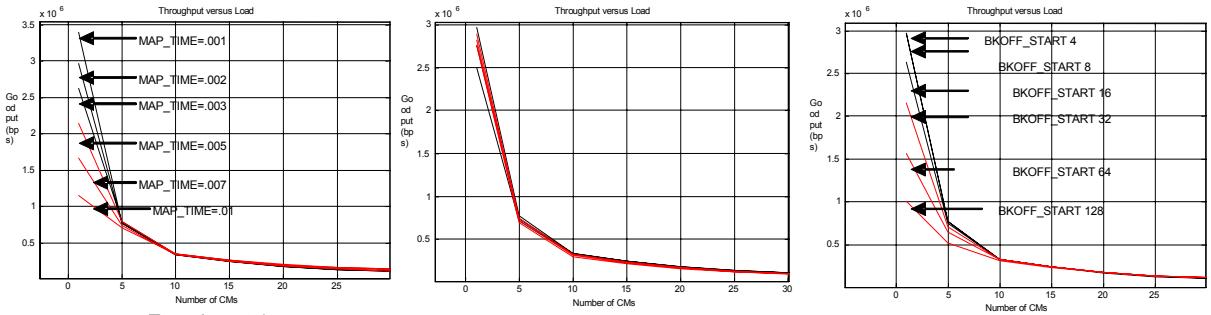
Run#	1	2	3	4	5	6
Backoff start	4	8	16	32	64	128

Figure 4.16: Model parameters and experiment description

4.4.1 Upstream FTP traffic

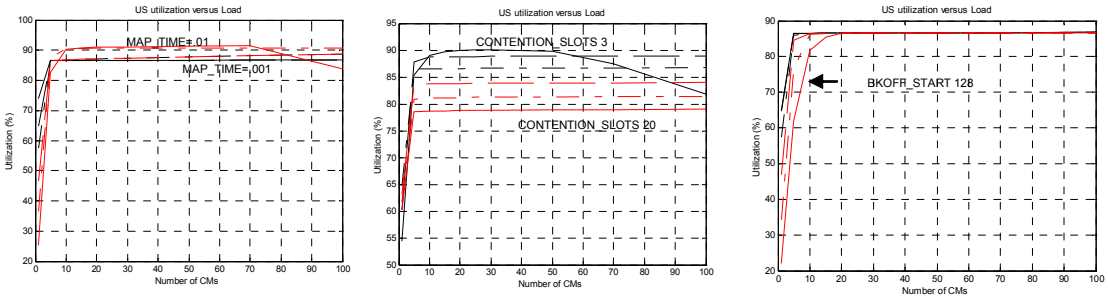
This scenario utilizes the model as described by Figure 4.15 with the FTP clients located at the CM's and the TCP sinks located at node 1. Looking first at the experiment involving the MAP_TIME, a set of 10 simulations run was performed with the MAP_TIME initially set to

.001 seconds and vary the number of CM's from 1 to 50. Next, we set the MAP_TIME to .002 and do another 10 iterations and so on. Figures 4.17a through 4.17c illustrate the throughput experienced by one of the FTP connections as the MAP_TIME, CONTENTION_SLOTS and BKOFF_START are varied. As expected, in an unloaded system (i.e., less than 5 CM's), throughput is highest for small MAP_TIME values. However, for higher loads, the MAP_TIME setting does not make a difference. The number of contention slots does not significantly impact throughput. As the backoff range increases, the throughput for the 1-5 CM case decreases because the average access delay is increased. Figure 4.18 suggests that the CONTENTION_SLOTS does impact utilization. This is because as the number of contention slots increase, the number of slots available for data decreases. The utilization statistic is based on total bytes sent over the wire including frame and physical layer overhead bytes but it does not account for the FEC overhead. Therefore, the actual upstream channel utilization will be up to 8% below what is illustrated in Figure 4.18.



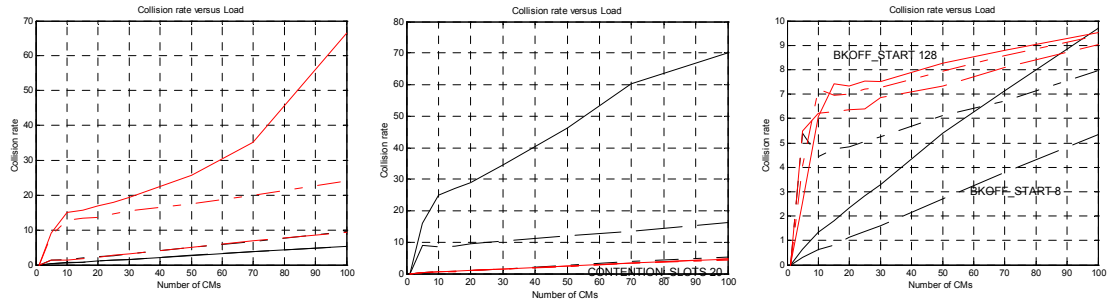
a. Experiment 1 b. Experiment 2 c. Experiment 3

Figure 4.17: Upstream throughputs



a. Experiment 1 b. Experiment 2 c. Experiment 3

Figure 4.18: Upstream utilization



a. Experiment 1 b. Experiment 2 c. Experiment 3

Figure 4.19: Upstream collision rates

Figure 4.19a shows that the collision rate increases with the MAP_TIME. This is because the number of contention slots decrease with increasing MAP_TIME (i.e., the model's parameters specify a fixed number of contention slots per MAP rather than a proportional amount of bandwidth assigned for contention). Figure 4.19b confirms that the collision rate decreases as the number of contention slots increase. The collision rate for experiment 3 tends to increase with contention backoff start times although the level of difference is relatively small. This is attributed to randomness and no conclusions are drawn based on the result.

4.4.2 Downstream FTP Traffic

This scenario utilizes the model as described by Figure 4.15 with the FTP clients located at node 1 and the TCP sinks located at the CM's (one TCP sink per CM). Downstream rate control has been disabled and so each FTP connection attempts to obtain as much bandwidth as possible. Figure 4.20 illustrates a downstream scenario involving the arrival of 4 segments. The sink generates two acknowledgement packets which arrive at the CM very close in time. We assume concatenation is enabled allowing the CM to request bandwidth for the transmission of the two ACK packets in a single frame. In the best case, the ACK's will be received at the CMTS roughly within 2 MAP times plus the transmission time of the data frame. Therefore, assuming a MAP_TIME of .002 seconds, we expect a maximum downstream throughput by any single CM to be limited to about 12Mbps (4 segments metered out every 4 milliseconds). After setting the TCP window size very large (and also making sure the buffers at the CM were large) we did observe a maximum downstream throughput of 12Mbps. Increasing upstream channel capacity to the DOCSIS 2.0 maximum of 30.72Mbps (i.e., 64Qam, 6.4Hzh channel) will not increase the single connection throughput. The MAP_TIME needs to be decreased in order to further optimize the single connection case.

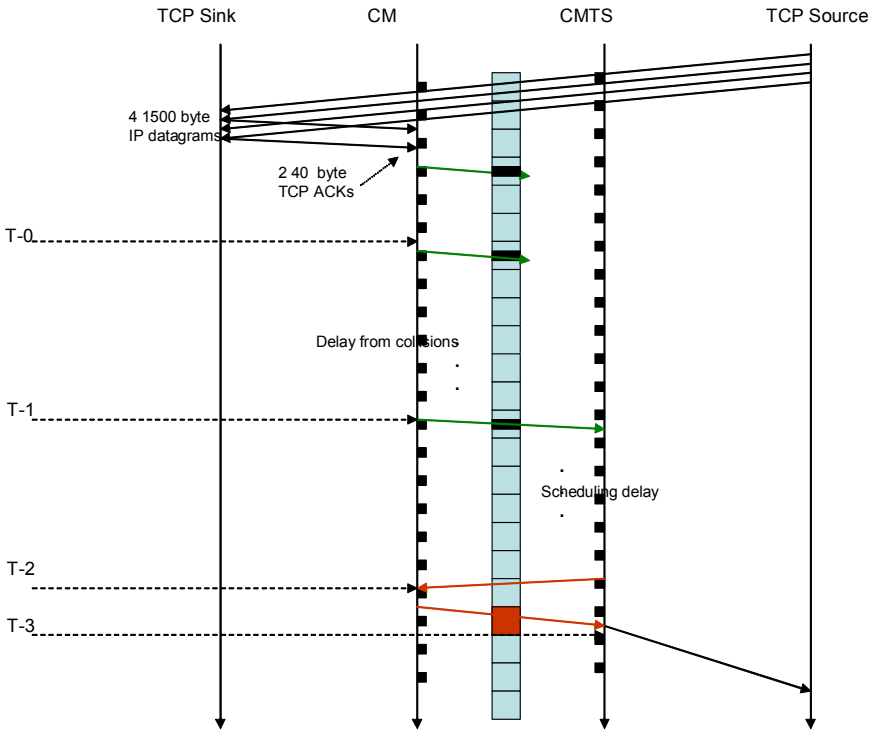
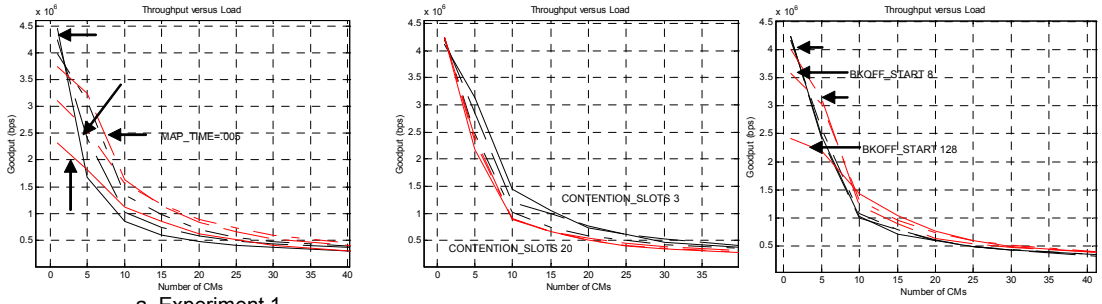
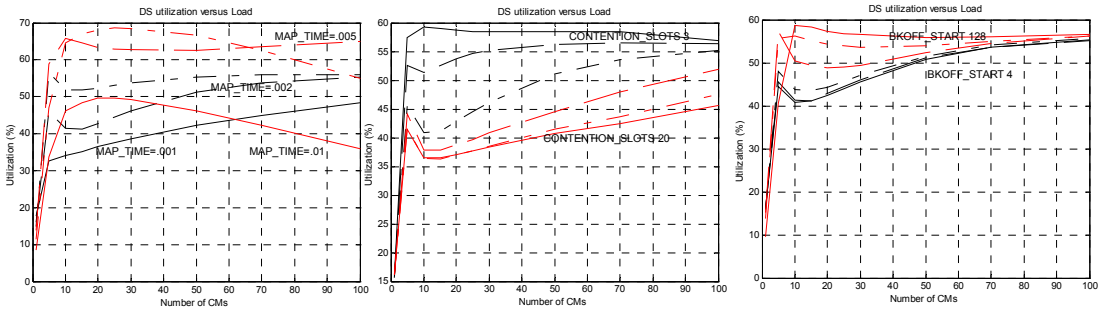


Figure 4.20: Downstream scenario



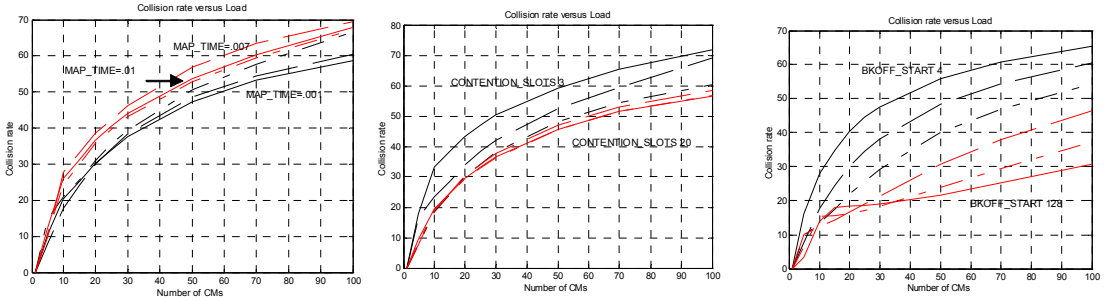
a. Experiment 1 b. Experiment 2 c. Experiment 3

Figure 4.21: Downstream throughput



a. Experiment 1 b. Experiment 2 c. Experiment 3

Figure 4.22: Downstream channel utilization



a. Experiment 1 b. Experiment 2 c. Experiment 3

Figure 4.23: Downstream collision rates

Figures 4.21 through 4.23 illustrate results of the three experiments. Figure 4.21a illustrates that throughput for the runs with 1 or 2 CMs is optimized with a MAP_TIME of .001. However once there are 5 or more CM's competing, larger values of MAP_TIME tend to perform slightly better. For a MAP_TIME setting of .002 seconds with 1 CM active we see a throughput of about 4.2 Mbps. The TCP connection window is set to 22 segments which is not sufficient to allow the connection to fully utilize available bandwidth. Figure 4.21a illustrates that when there is more than 1 CM, we see that the runs with lower MAP_TIME settings experience a lower throughput. Smaller settings of MAP_TIME lead to bursty TCP dynamics which increases the TCP loss rate and reduces the link utilization (confirmed in Figure 4.22a).

Figure 4.22b shows that utilization is lower when the number of contention slots is large. As in the upstream case, the explanation is because the amount of bandwidth available for data is lower. Figure 4.22c shows higher DS channel utilization for larger backoff start range values. This is because the collision rate is much lower (Figure 4.22c) minimizing the inefficiency caused by contention. The collision rate is lower for the smaller MAP_TIME (Figure 4.23a) because loss occurs in these cases. Figure 4.23b confirms that collision rates will be higher with a lower number of contention slots.

4.4.3 Web traffic

In this scenario, the three experiments described in Figure 4.16 were performed using the network illustrated in Figure 4.24. In each experiment, the number of CM's is increased between 50 and 300. Each CM generates a certain level of downstream web traffic, interacting with one or more servers (see Figure 4.25 for web traffic settings). The network and web traffic models are based on the "Flexbell" model defined in [19]. In addition to web traffic, we configure 5% of the CM's to generate downstream low speed UDP streaming traffic (56Kbps), 2% of the CM's to generate downstream high speed UDP streaming traffic (300Kbps) and 5% of the CM's to generate P2P traffic (i.e., using an exponential on/off model with a setting to download on average 4 Mbps with an idle time of several seconds in between downloads (our model is based loosely on [20]).

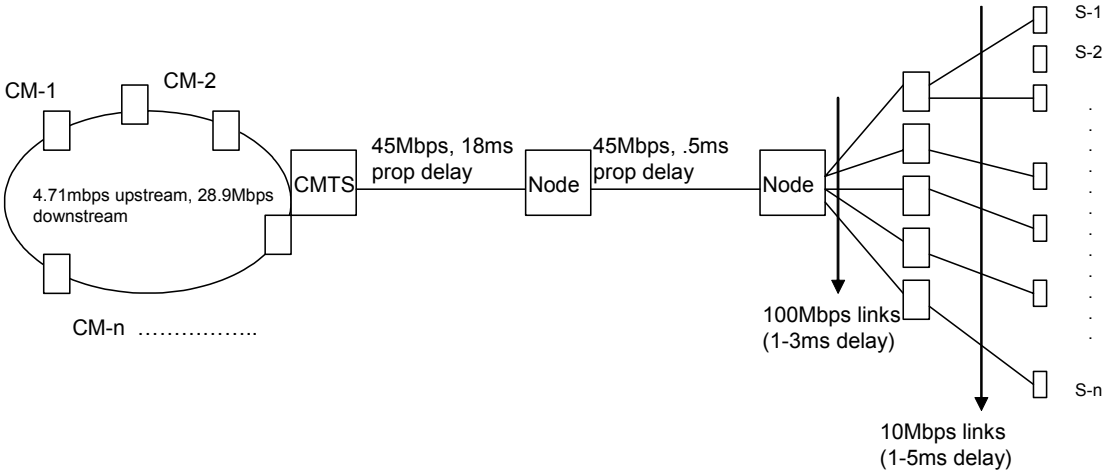
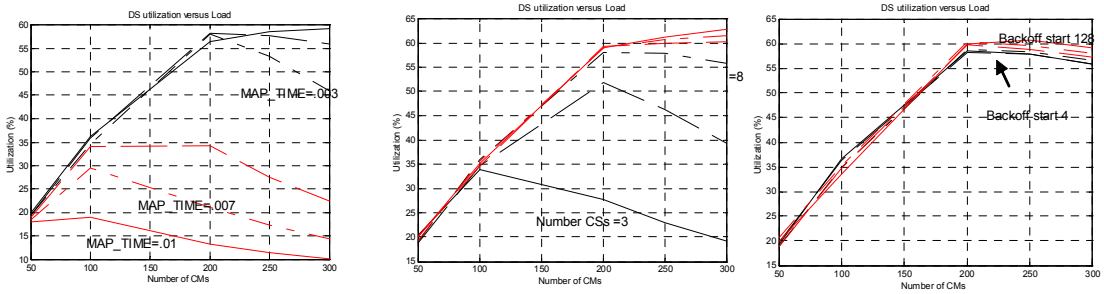


Figure 4.24: Web Traffic Topology

Web Traffic Model Parameters
Inter-page: pareto model, mean 10 and shape 2
Objects/page: pareto model, mean .5 and shape 1.5
Inter-object: pareto model, mean .5 and shape 1.5
Objectsize: pareto model, mean 12 (segments) and shape 1.2

Figure 4.25: Model Parameters



a. Experiment 1 b. Experiment 2 c. Experiment 3

Figure 4.26: Downstream web scenario - Downstream utilization

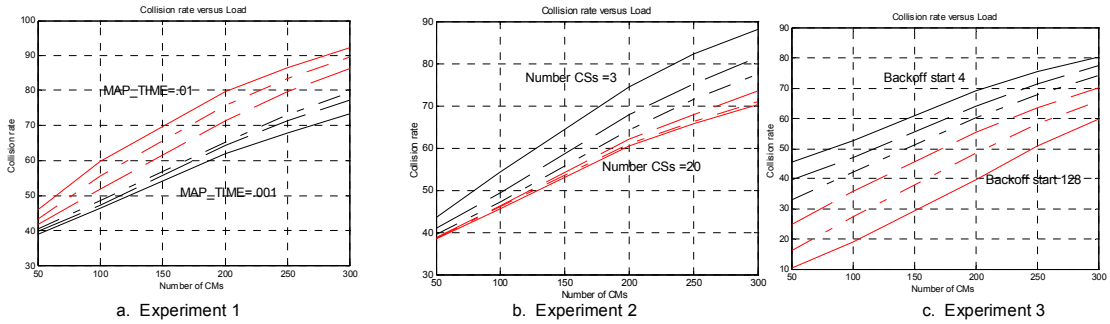


Figure 4.27: Downstream web scenario - Upstream collisions

Figures 4.26 and 4.27 illustrate the main results for experiments 1,2 and 3 for the scenario. The experiment 1 results show that performance significantly deteriorates as the MAP_TIME increases. For a MAP_TIME of .01, the system is virtually unusable as the CM's spend most of their time in collision recovery. Similar results occur when the CMTS does offer enough contention slots (i.e., experiment 2 results). The experiment 3 results show that although the BKOFF_START parameter does not have a setting that leads to extremely poor performance, it does have an impact. Figure 4.26c best illustrates the tradeoff. At low usage levels, system delay is lowest for small BKOFF_START times (i.e., because the delay before making the initial contention request is lower). But as the number of CM's increase, larger BKOFF_START ranges result in slightly lower delay. The explanation is illustrated in Figure 4.27c, the collision rates decrease with large BKOFF_START ranges.

4.4.4 Summary of Results

We find that while all three of the system parameters under investigation directly impact system performance, the MAP_TIME is perhaps the most critical. Based on our analysis, we

find that the MAP_TIME should be less than or equal to .002 seconds. For the upstream FTP scenario, TCP throughput decreases as the MAP_TIME increases.

For the downstream web scenario, performance drops significantly as the MAP_TIME increases primarily because the number of contention slots decreases. If we were to increase the number of contention slots (i.e., as a percentage of bandwidth), the higher MAP_TIME values would not result in extreme performance degradation.

But this would not change the fact that upstream access delay increases in proportion with the MAP_TIME setting. The drawback to small MAP_TIMES is the overhead incurred by the CMTS, the CM's and downstream channel due to frequent MAP messages having to be generated and sent over the wire. All three parameters impact the frequency of collisions. We found collisions to be highest in the downstream web scenario (60-90% depending on the load). There is a clear performance tradeoff with the number of contention slots. Too few contention opportunities at high downstream loads (i.e., when the utilization exceeds 50%) can lead to significant performance degradation as the upstream channel won't have the capacity for the ACKs. However too many contention slots during periods of low usage will decrease utilization. We found that a value of at least 12 contention slots is needed although this is a parameter that could be dynamically adjusted. The contention backoff start range also has a clear tradeoff. A low starting range helps keep the upstream access delay low, especially during periods of low usage. A higher starting range can reduce the collision rate at higher loads. We found that the starting range should be at least 8 contention slots.

Our base configuration for our analysis used the following settings: a MAP_TIME of .002, a CONTENTION_SLOTS of 8 and BKOFF_START of 8. With these settings, we observed high data rates for the low load upstream FTP scenario. As the load increases we observed

high channel utilization with very low collision rates. In the downstream FTP scenario we saw rather low utilization (40-55% depending on the load) due to bursty TCP dynamics.

5. SOLUTION TO THE BANDWIDTH-HOG PROBLEM

In this chapter, a solution to the bandwidth-hog problem in Cable Networks is presented.

5.1 Introduction

Cable networks are based on a shared infrastructure with several homes or businesses sharing a local access pipe. If one home or business is using its connection to transfer large amounts of data, performance for all other homes or businesses that rely on the same access pipe is affected. The 'elephants' (the very few subscribers who consume the majority of bandwidth) represent a tremendous concern as they significantly impact the quality of service observed by the 'mice' (the majority of subscribers who consume low bandwidth). In this chapter we present an algorithm to solve this problem.

The algorithm controls the rates of the heavy users when they start affecting the performance of other users. The algorithm monitors the per-subscriber usage of the channel over a period of time and identifies heavy users among them. If the heavy users start affecting the ‘mice’ users, then they are rate-controlled for a punishment period.

5.2 Related work

There has been a lot of work in the area of rate control in communication networks. In this section, a brief overview of some of the important results is presented.

ERICA algorithm was proposed for congestion avoidance in ATM networks. This algorithm is for ABR service class in ATM networks. Under this scheme, each network switch monitors their load on each link, and based on a load factor, available capacity and number of active virtual channels, the end source is advised about the rates it should transmit. The

algorithm achieves efficiency, fairness, controlled queuing delays, and fast transient response [21].

[22] presents another approach to controlling high bandwidth flows at congested router. RED-PD (RED with preferential dropping) is a mechanism which keeps states of just high-bandwidth flows and uses the RED packet drop history at the router to detect high-bandwidth flows in times of congestion. Then packets of these high bandwidth flows are dropped during times of congestion. As states of only heavy flows are maintained, this has the advantage of having lesser overhead than per-flow scheduling mechanisms. The authors also suggest that best-effort traffic may not require the level of fairness provided by the per-flow scheduling schemes.

[23] presents an analysis of performance of several grades of data, voice and video sessions over a cable or DSL based access network. Each session has a minimum and maximum bandwidth data-rate. Under congestion, fair and prioritized rate-degradation is considered. A bufferless model is presented in which the excess traffic is lost under congestion (or the session user transfers less data to keep the session duration independent of congestion) and a buffered model in which no traffic is lost and session duration is elongated under congestion to keep the total data transfer in a session independent of congestion. The bufferless model is a generalization of the standard rate envelope multiplexing (REM) model and the buffered model is a generalization of the M/G/c processor-sharing model.

[24] presents a novel technique, TCP rate control, to augment the end-to-end TCP performance by controlling the sending rate of a TCP source. TCP rate control technique affects the factor affecting sending rate of TCP source, namely window size, round trip time and rate of acknowledgements. The acknowledgment rate is modulated and the ack number and the window size is modified in the acknowledgment to affect these factors.

[25] presents a discussion of congestion control from a game-theoretic perspective. It is assumed that users are selfish and independent and central control is exerted only at the network switches. The effect of various switch service disciplines on the operating point resulting from selfish users is studied. Specifically, the effect of various service disciplines on the efficiency and fairness of the operating point is presented. It is shown that no service discipline can guarantee optimal efficiency. Also, the authors prove that Fair-share service discipline is the best and FIFO service discipline is the worst.

5.3 Our Approach

There has been an increased need for participation of network itself to control congestion rather than relying on end-users to provide end-to-end congestion control. As presented in [2], there can be three approaches for controlling the best-effort traffic:

- Per-flow scheduling mechanisms: The bandwidth of each best-effort flow is regulated to approximate min-max fairness.
- Routers or Network elements promote the use of end-to-end congestion control as the primary mechanism to share best-effort traffic by providing incentives to flows using these mechanisms.

- Pricing mechanisms can be used to control sharing.

The authors argue that during congestion, router mechanisms are needed to identify and control the bandwidth of selected best-effort flows. These flows can be unresponsive to congestion, "not TCP-friendly" or using very high bandwidth during congestion. A flow whose long-term arrival rate exceeds that of any conformant TCP in the same circumstances is defined as "not TCP-friendly". An unresponsive flow fails to respond to congestion. All the other flows not falling in any of these categories are not restricted during congestion. This approach provides a concrete incentive to end-users and application developers to use end-to-end congestion control for best-effort traffic.

In our algorithm, we have taken this approach of promoting end-to-end congestion control by providing incentives to flows. The algorithm controls the rates of heavy users when they start affecting the performance of other users. The algorithm monitors the per-user usage of the channel over a period of time and identifies heavy users among them. These heavy flows are then restricted for a punishment period.

In a cable network environment, typically a cable service provider would want to restrict the maximum bandwidth of a subscriber to a static rate to ensure that it does not hogs all the bandwidth. If the value of this static-rate is too low, then the channel might be highly under-utilized during low network load. If this value is too high, then few heavy flows might hog all the bandwidth affecting other users. With our approach, a cable service provider may be able to configure very high maximum bandwidth-limits or no limits for the subscribers. If

few heavy subscribers try to hog the bandwidth over a period of time, then they will be punished by the algorithm.

An alternative to this approach is the per-flow scheduling mechanism discussed earlier. This approach can ensure min-max fairness. But the overhead of such an approach is much more, and they provide no incentives to best-effort flows for using end-to-end congestion control mechanisms [2].

5.4 Algorithm

The algorithm allows the subscribers to achieve high data-rate and ensures that few heavy bandwidth flows do not hog the bandwidth. In a cable network environment, a subscriber may set-up multiple flows with the CMTS for data-transfer. Our algorithm can be used to either restrict a bandwidth-hogging 'flow' or a 'subscriber'. For the rest of the discussion, we assume that a subscriber sets up one flow for data transfer. The implementation changes for a per-subscriber control are also discussed below.

The algorithm can be divided into three components: Detecting when the heavy flows are affecting other traffic, Identifying these heavy flows, and the type and duration of punishment given to these heavy flows. In the next few sections, we discuss these components. The algorithm has been implemented and tested for the downstream channel.

5.4.1 Detecting impact of heavy flows

An important aspect of the algorithm is to act only during times of congestion (when performance of other flows is affected by the heavy flows). Given that the majority of the

application traffic flowing over the Internet is HTTP data, we use the web performance metric presented in [26]. The Web Response Time (wrt) metric used in our algorithm is defined as follows: A client periodically issues an HTTP request to a web server located at the other end-point of the network. The client is located at one end-point of the cable network on a sample CM, and the server is located at the other end-point. A wrt sample is the amount of time from when the client issues the request to when the entire web object has been successfully received by the client. A configurable parameter 'wrt monitoring interval' (t-wrt) denotes the time-interval used to collect wrt samples and an assessment of the impact of network performance on the end-user is done at the end of every t-wrt.

As suggested in [26], we use the rule of thumb that once the average web response time statistic approaches 1 second, it can be concluded that the network performance is negatively impacting end-user experience. In addition to this rule, changes in avg. wrt are also used to determine the impact on the end-user. If the new avg. wrt changes over a threshold limit (thr-wrt) over the last avg. wrt, then again we conclude that network performance is affecting the end-user performance.

The impact on the end-user may or may not be due to presence of heavy flows. The rate-control part ensures that if the impact on end-user performance has been caused by heavy flows, then they are punished.

The advantage of using an application level performance metric over the traditional ping based metrics are presented in [26]. To assess the impact of heavy users on the network and end-users, Upstream/Downstream channel utilization can also be used as an indicator of the congestion. Whenever the utilization increases above a threshold limit, congestion can be assumed.

5.4.2 Identification of heavy flows

The algorithm monitors the avg. rates of the flows over a configured time-interval (from few second to hours). Whenever congestion is detected in the upstream or downstream channel, the rate-control component of the algorithm is invoked. Then based on the number of active flows and the channel capacity, the fair-share that each flow should get is calculated. E.g., for a 30 Mbps channel with 100 flows, the fair-share of each flow should be 300 kbps. All the flows that are above a threshold limit of the fair share are marked as heavy flows. This threshold limit is user-configurable. The parameters that have to be configured are:

a) Rate-monitoring interval (t-lt) : This parameter indicates the time over which the average rate of a flow is monitored (t-lt). The avg. rate monitored during this interval is used to decide whether the flow is heavy or not. This parameter can be configured to a range of values. A small value of t-lt will lead to punishment of flows which have been heavy for a small period of time. If t-lt is configured to be long (30 min or 1 hour), then it will lead to punishment of the users which were heavy for a large period of time. So, this parameter allows to choose the type of behavior desired from the algorithm. If the algorithm is used on a per-subscriber basis, then the avg. rate of a subscriber is the sum of the avg. rate of all the flows of the subscriber.

b) Heavy User Threshold (thr-heavy): All the flows which have avg. rates above (thr-heavy)*fair-share are marked as heavy flows. If this is set to very high values, it will lead to a punishment of very heavy flows. E.g., if this parameter is set to 10 and if the fair-

share for each flow should be 300 kbps, then all the flows having avg. bandwidth above 3 Mbps (300 kbps * 10) will be marked as heavy flows.

5.4.3 Type and duration of the Punishment

Once the heavy flows have been detected, the avg. rates of these heavy flows is brought down to the fair-share value over one t-lt. These heavy flows are now rate-controlled for a fixed configurable amount of time. During their punishment period, the heavy flows never get more than the fair share of bandwidth according to the network conditions. All the other flows are not affected by the algorithm. The parameters that have to be configured are:

a) Rate-update interval (t-ruf): This parameter denotes the frequency of the rate-updates of the heavy flows. The equation used to bring down the rates of the heavy flows is:

New_rate = (Avg_rate) – (Avg_rate – fair_share)/(t-lt)

Where Avg_rate: Avg. rate of the flow over last t-lt

Using this equation, the rate of the heavy flows is updated every t-ruf. This decrease in the rates is stopped if the New_rate is near fair-share.

d) Punishment period (in units of t-lt): Indicates the time period after which the punishment will be over for heavy flows and any sort of rate-control on these heavy flows will be removed. In a real system, this parameter might give the flexibility to the cable operators to divert these heavy flows to off-peak times.

5.4.4 Pseudocode of the algorithm

The pseudocode of the algorithm is presented next:

Web Response Time monitor:

Every t-wrt,

1) Compute the avg-wrt

2) new-wrt = (avg-wrt - normal-wrt)/(normal-wrt);

If (new-wrt > thr-wrt) or (new-wrt is approaching 1 seconds), call Rate-control function.

Every t-lt:

1) Compute Avg. rate of all the flows(R-avg(i))

2) If heavy flows are being rate-controlled and their punishment period is over, then

remove rate-control.

Rate-control function()

{

Compute fair-share = (channel capacity)/ (number of active flows);

For all flows for which (R-avg(i) – fair-share) / (fair-share) > thr-heavy

{

Update their instantaneous rates(R-max(i)) every t-ruf sec.

using R-max(i) = R-max(i) - (R-max(i) – fair-share)/(t-lt);

where fair-share is also updated every t-ruf.

}

}

5.5 Simulation Results

The algorithm was implemented using our simulation model for the downstream channel. The same algorithm can be implemented to prevent the problem in the upstream channel. Several simulations were run with different settings of the parameters discussed to verify whether the algorithm achieves its goals. In the next section, we present some of the results from the simulations.

Scenarios:

For all the scenarios, the topology shown in Figure 5.1 is used.

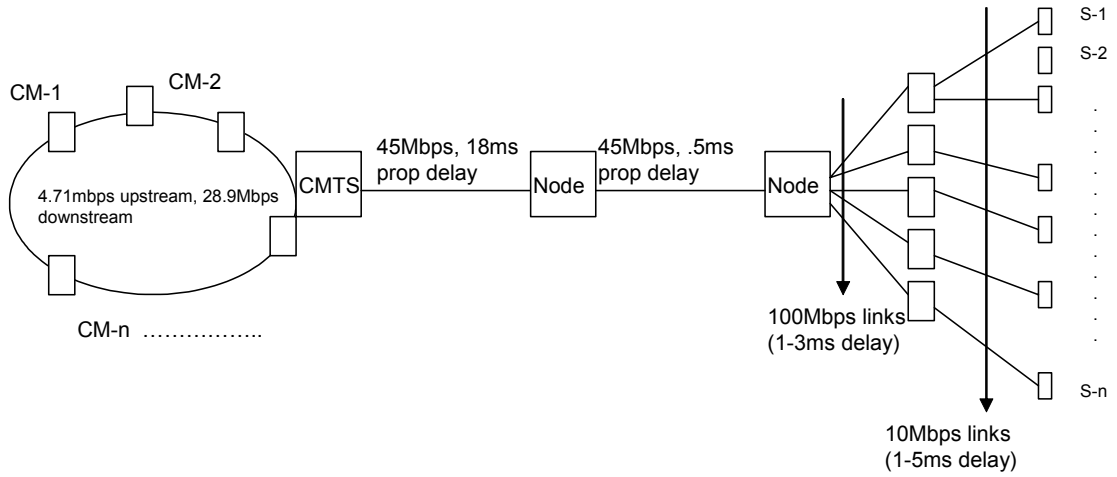


Figure 5.1: Simulation Topology

To simulate normal web-user, some CMs generate a certain level of downstream web traffic, interacting with one or more servers (see Figure 5.2 for web traffic settings).

Web Traffic Model Parameters

Inter-page: pareto model, mean 10 and shape 2
Objects/page: pareto model, mean 3 and shape 1.5
Inter-object: pareto model, mean .5 and shape 1.5

shape 1.2

Figure 5.2: Model Parameters

To simulate the heavy flows, few CMs are configured to generate downstream high speed UDP traffic. All the heavy users start and stop at the same time.

Scenario 1:

The parameters for this scenario were:

Number of CMs: 130, Number of heavy users: 6, Number of normal web-users: 100, Number of users generating downstream low speed traffic: 24.

When the simulation starts, the web-users are started. The heavy users start at around 500 sec generating around 4 Mbps of traffic. The remaining 24 CMs are started from 500 sec with a gap of 10 sec between start times of the CMs. Each CM generate around 100 kbps traffic on downstream channel.

The algorithm parameters are:

t-lt: 250 sec, t-wrt: 250 sec, t-ruf: 1 sec, thr-wrt: 70, thr-heavy: 80, Punishment period: 5 t-lt.

The results for the simulations are presented in Figure 5.3 and 5.4.

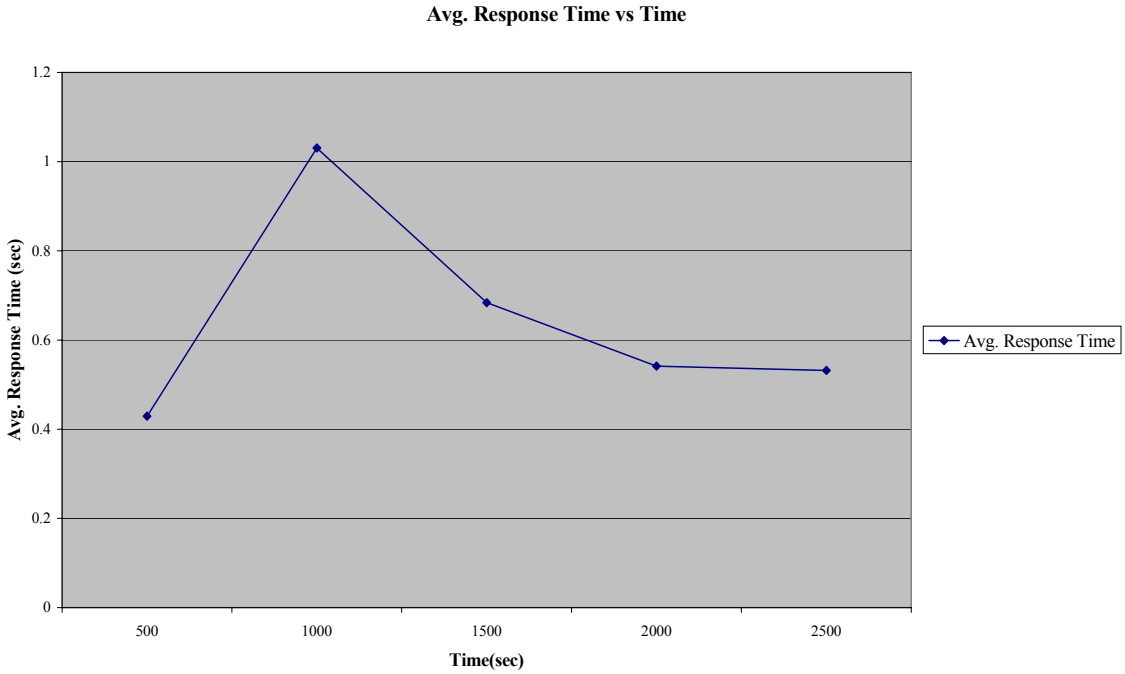


Figure 5.3: Avg. web response time

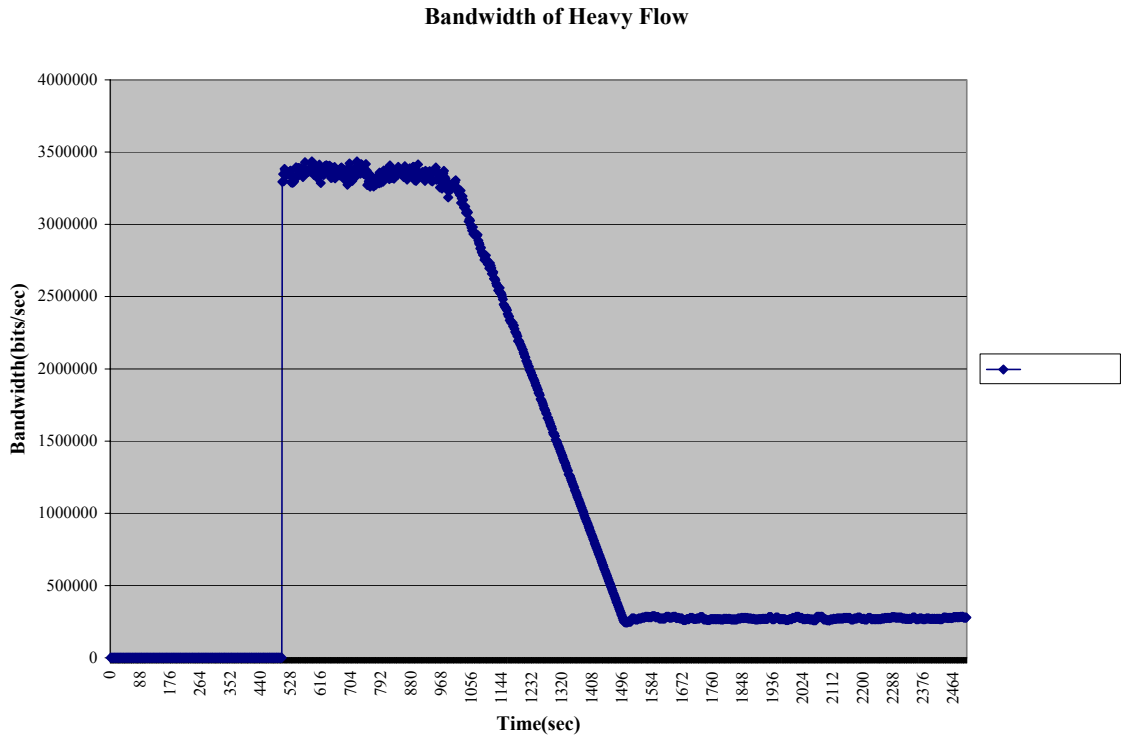


Figure 5.4: Bandwidth of Heavy Flow vs. Time

The avg. response time increases when the heavy users start generating traffic. The rate-control of heavy flows is started at around 1000 sec and it is stopped at 1500 sec when they reach the fair-share. Since the punishment time is not reached, so the heavy flow continues to operate at the fair-share determined by network conditions. Since, the traffic conditions are not changing in this scenario, the heavy flows continue to operate at same bandwidth.

Scenario 2:

The parameters for this scenario were:

Number of CMs: 130, Number of heavy users: 6, Number of normal web-users: 100, Number of users generating downstream low speed traffic: 24.

When the simulation starts, the web-users are started. The heavy users start at around 500 sec generating around 4 Mbps of traffic. The remaining 24 CMs are started from 0 sec with a gap of 10 sec between start time of the CMs. Each CM generate around 100 kbps traffic on downstream channel. These flows are then stopped starting from 800 sec with a gap of 20 sec between each CM.

The algorithm parameters are:

t-lt: 250 sec, t-wrt: 250 sec, t-ruf: 1 sec, thr-wrt: 70, thr-heavy: 80, Punishment period: 5 t-lt.

The results for the simulations are presented in Figure 5.5 and 5.6.

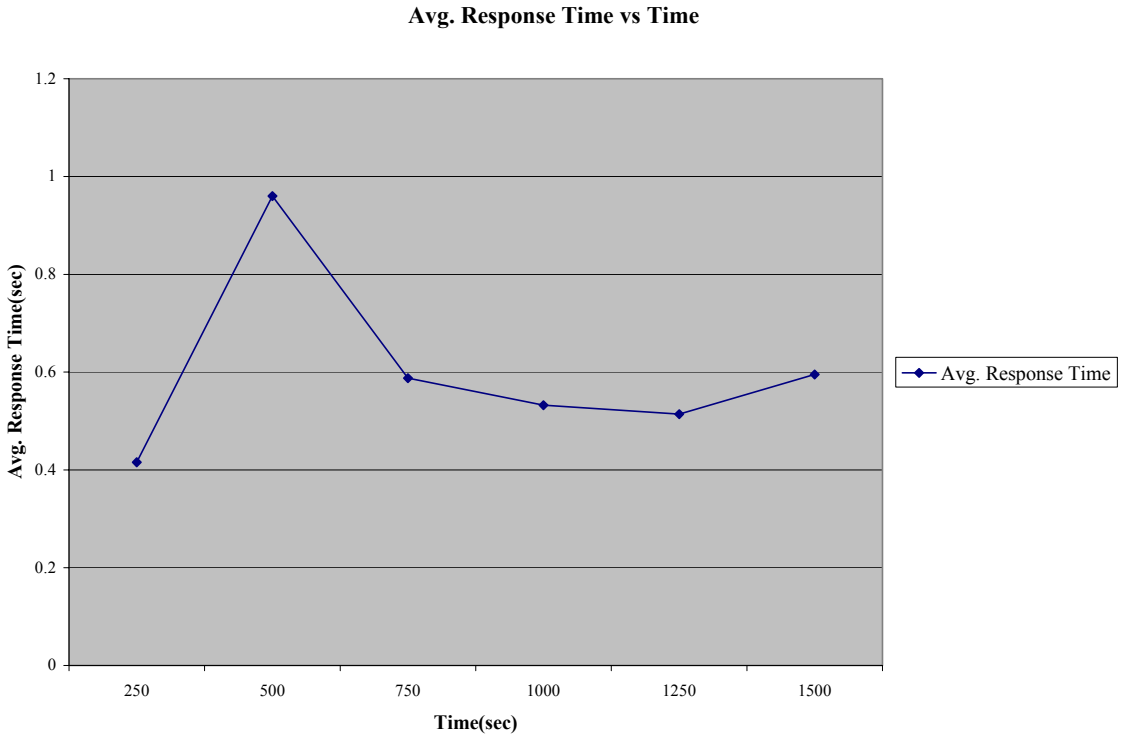


Figure 5.5: Avg. web response time

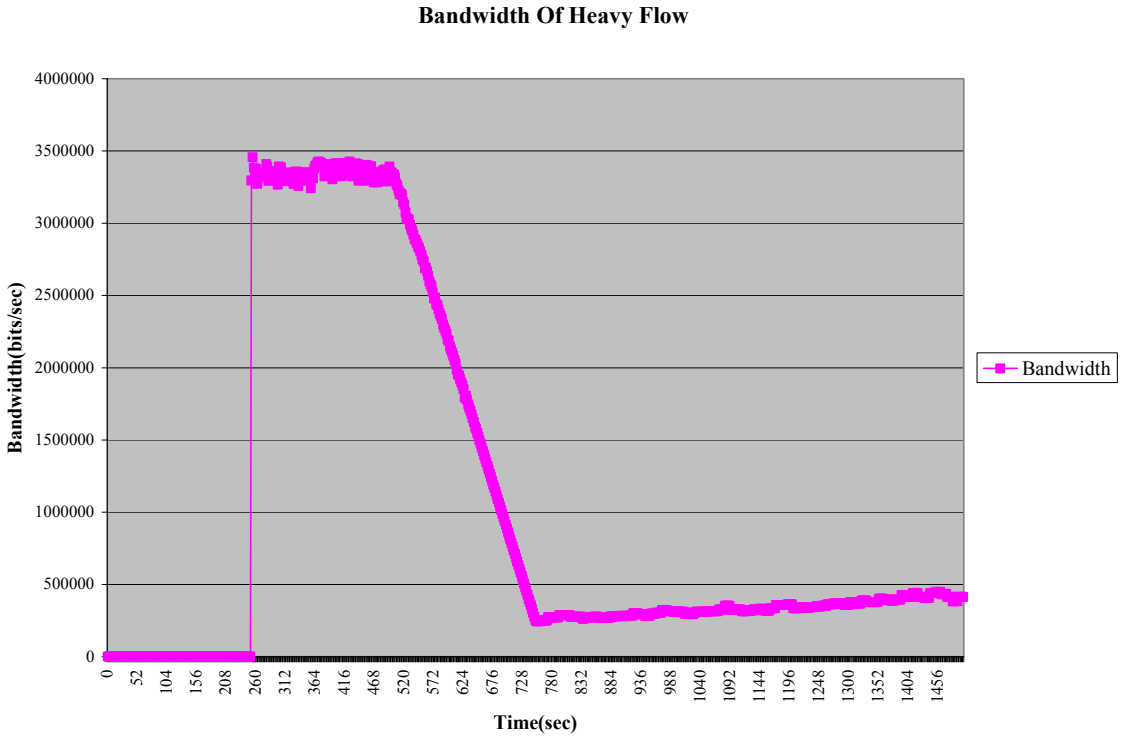


Figure 5.6: Bandwidth of Heavy Flow vs. Time

The results for this scenario are similar to results of scenario1. As the CMs doing low speed UDP traffic stop after 800 sec, we see a gradual increase in the bandwidth of the heavy flows.

Scenario 3:

The parameters for this scenario were:

Number of CMs: 160, Number of heavy users: 6, Number of normal web-users: 100, Number of users generating downstream low speed traffic: 54.

When the simulation starts, the web-users are started. The heavy users start at around 3000 sec generating around 3.8 Mbps of traffic. The remaining 54 CMs are started from 10000 sec with a gap of 10 sec between start times of the CMs. Each CM generate around 150 kbps

traffic on downstream channel. These flows are then stopped starting from 13500 sec with a gap of 10 sec between each CM. Also, the parameter for web-traffic is such that the number of web-users are reduced to very few after 10000 sec. The goal is to show that if the network is under-utilized, then heavy users can get high bandwidth even during their punishment phase to keep the utilization of the channel high.

The algorithm parameters are:

t-lt: 3000 sec, t-wrt: 3000 sec, t-ruf: 1 sec, thr-wrt: 70, thr-heavy: 80, Punishment period: 5 t-lt.

The results for the simulations are presented in Figure 5.7 and 5.8.

Figure 5.7: Avg. web response time

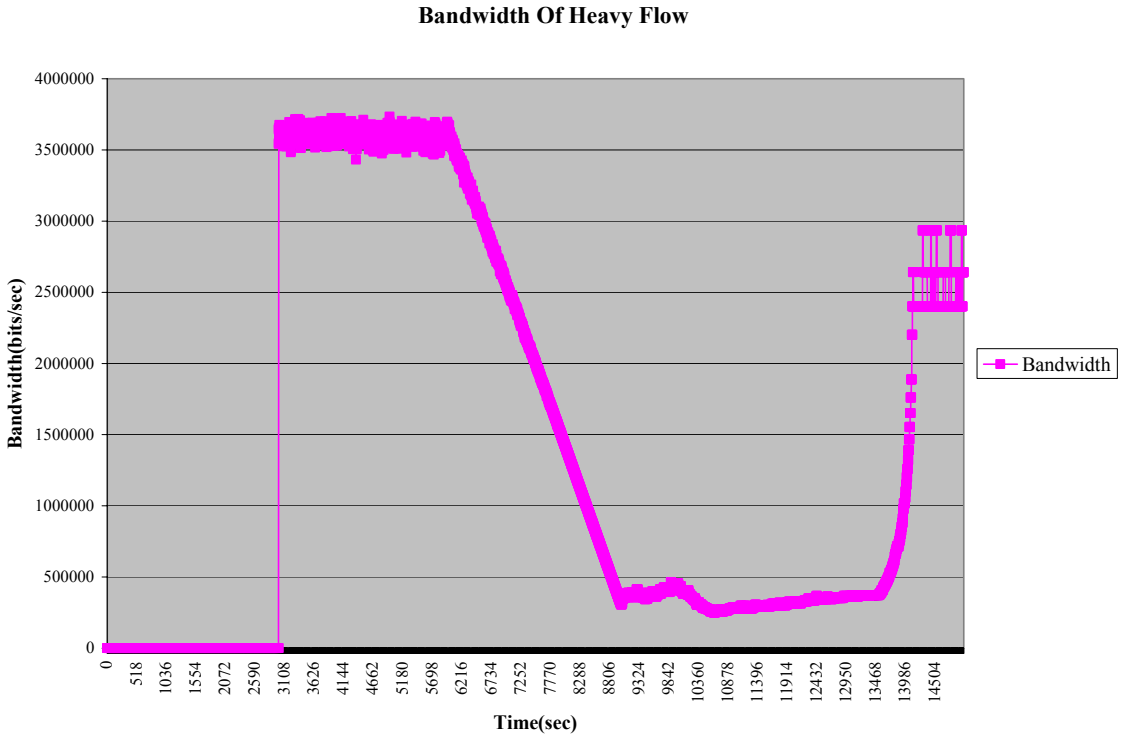


Figure 5.8: Bandwidth of Heavy Flow vs. Time

This scenario illustrates the use of algorithm to punish flows which have been heavy for a long time. After 13400, the load in the system is very less. So, we observe an increase in the bandwidth of the heavy user.

Scenario 4:

The parameters for this scenario were:

Number of CMs: 100, Number of heavy users: 6, Number of normal web-users: 94, When the simulation starts, the web-users are started. The heavy users start at around 500 sec generating around 4 Mbps of traffic.

The algorithm parameters are:

t-lt: 500 sec, t-wrt: 500 sec, t-ruf: 1 sec, thr-wrt: 70, thr-heavy: 80, Punishment period: 2 t-lt. The goal is to verify that the punishment ends for the heavy flows.

The results for the simulations are presented in Figure 5.9 and 5.10.

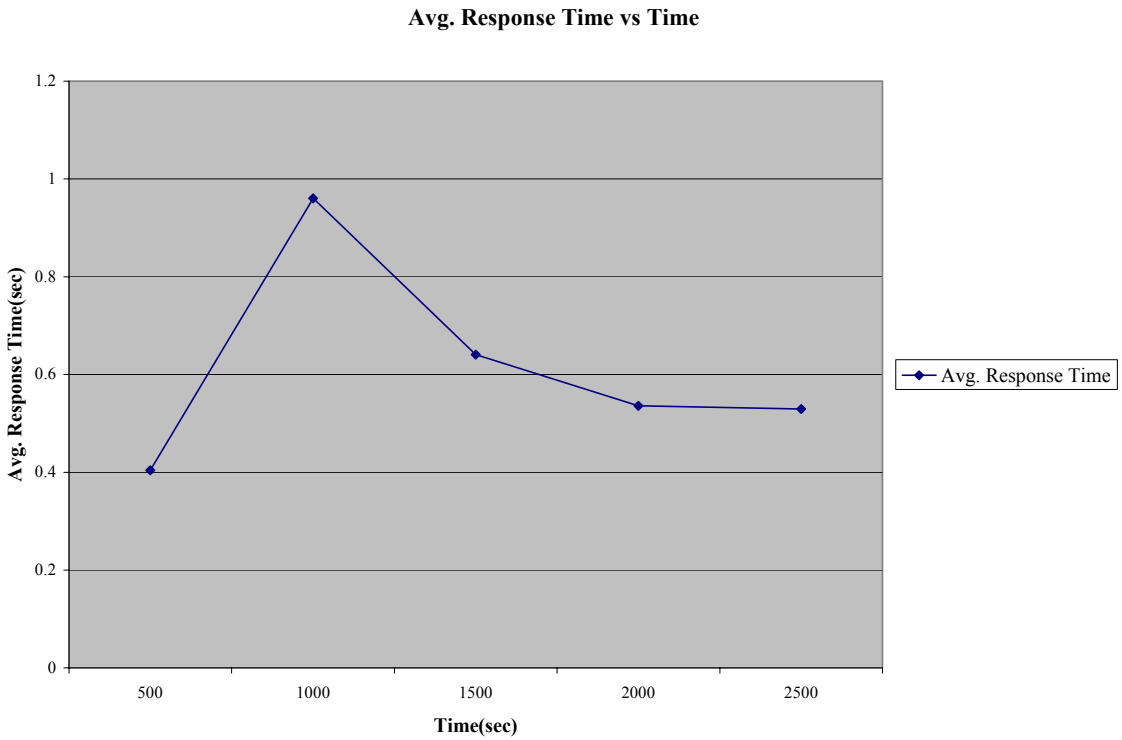


Figure 5.9: Avg. web response time

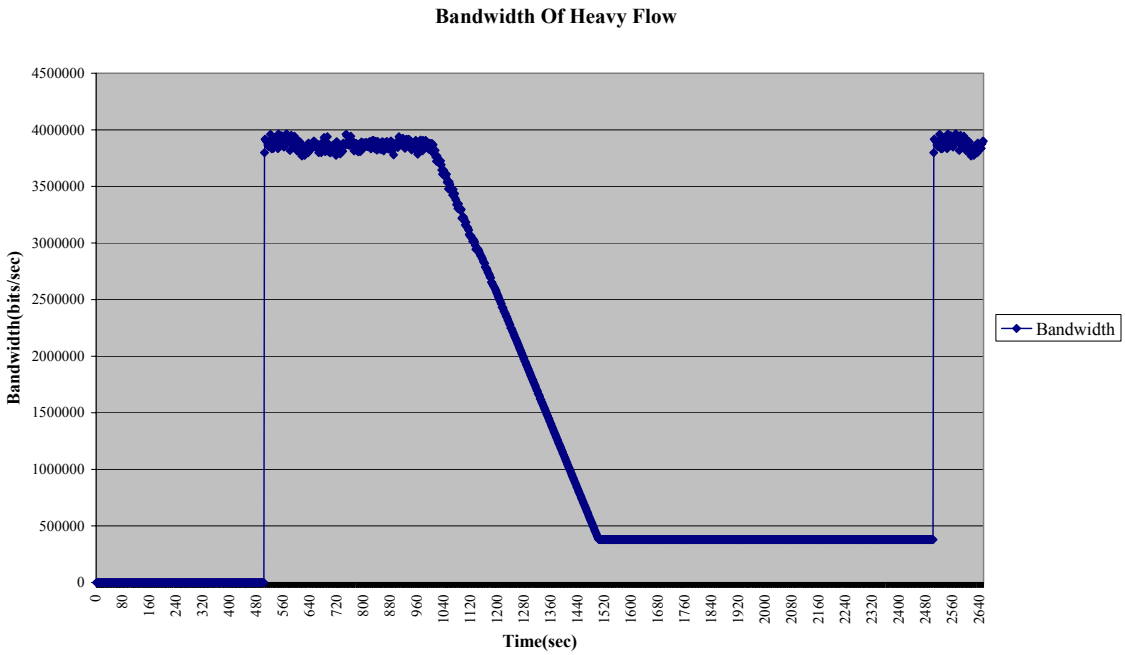


Figure 5.10: Bandwidth of Heavy Flow vs. Time

In this scenario, the punishment of heavy flow ends soon. Since, the heavy flows are still present, they again occupy the same level of bandwidths they had before the punishment began.

5.6 Discussion

The goal of the algorithm is to punish heavy users when they start affecting the performance of other low users of bandwidth. The algorithm only controls the bandwidth of the users consuming too much bandwidth and it does not affects the rest of the traffic. Once heavy users are detected in the system, the algorithm brings them down to a fair-share bandwidth. However, other flows which have not been profiled as heavy users can continue to operate at any bandwidth. It does not achieves fairness like ERICA or other fair-share algorithm. But as suggested in [22], such degree of fairness might not be required for best-effort traffic. Also,

the implementations of such algorithms involves more overhead as every flow needs to be rate-controlled.

The congestion in the upstream and downstream channel may be caused due to several reasons like presence of heavy flows, too many users etc. The algorithm has been designed to handle the case where congestion is caused by presence of heavy flows. The presence of heavy flows starts affecting the performance of other users as they compete for the common resources with other flows. Specifically, packet drops and queuing delay at the downstream channel queue and upstream channel contention delays will increase due to the presence of the heavy flows.

The rate-monitoring interval (t-lt) can provide a lot of flexibility to a cable service provider in implementing different policies. E.g., if t-lt is low (few seconds to few minutes), then the algorithm ensures fairness among the flows on a very short time scale. However, for example if the cable service provider wants to detect the users who have setup file-sharing servers, then the t-lt would be kept long (1 or 2 days). These long-term heavy users can be punished whenever congestion is detected in the network.

Once the heavy flows are detected, they are brought down to a fair-share value using following equation:

New_rate = (Avg_rate) – (Avg_rate – fair_share)/(t-lt)

The rates are brought down gradually to prevent a sudden drop in the rates of the heavy users. The fair_share value is computed as (channel capacity)/(Number of active flows). A flow is considered to be active if it transmits atleast one byte in the last monitoring interval (t-ruf).

Once the heavy flows have been brought down to fair-share, their rate is adjusted every t-ruf to a new fair-share value. The reason for operating the heavy users at fair-share is to prevent extreme degradation in the performance of heavy flows. The fair-share will also ensure that heavy flows are no longer unfair to other flows. Moreover, the fair-share will automatically tune the bandwidth of the heavy flows according to the network condition. So, for instance, if the number of users are high, heavy users will have less bandwidth. But if the number of users are less, then heavy users will get high bandwidth. Since, we only consider the number of active flows in calculating fair-share, it is possible that the channel might remain under-utilized. For instance, assume there are 100 users using 30 kbps bandwidth. The fair-share will be 300 kbps for a 30 Mbps channel. Now, the heavy users will continue to operate at 300 kbps even though channel is under-utilized (Other 100 users are only using 3 Mbps). The reason for doing this is due to greedy nature of heavy flows. If we decide to calculate fair-share based on the usage of the channel, rather than the number of users, then it is probable that the problem (few heavy users occupying most of the bandwidth) will arise again. In our example considered above, the heavy users can use 27 Mbps of excess capacity assuming other 100 users continue to operate at 30 kbps. This will lead to the same situation for which the algorithm has been designed.

The heavy flows continue to operate at the fair-share for a punishment time. The punishment time might be used by cable operators to divert the heavy users traffic to off-peak times. There might arise situation where new flows might originate when other heavy flows are being rate-controlled. If these new flows are also profiled as heavy, then they will be given the same punishment as other heavy flows are being given. This will ensure fairness in the

treatment of heavy flows. Also, the algorithm maintains the subscriber's id of the heavy flows being punished. This is to ensure that if a heavy flow stops in the middle of punishment and starts again, then it is still punished for the remaining punishment period.

If the punishment period is too small for heavy flows, then they might again start to operate at high rates. Soon they will be rate-controlled again. This might lead to oscillations and an unstable system. To prevent this, the punishment period should be kept high. Also, the rates of heavy flows should be monitored over a large time interval before profiling them as heavy flow. Once congestion is detected due to heavy flows, the algorithm will remove the congestion within one t-lt. The algorithm does not exerts any control on non-heavy flows and gives fair treatment to heavy flows. The heavy flows are always guaranteed atleast fair-share bandwidth. Also, after the punishment period no control is exerted on these heavy flows.

There can be many other approaches to solve this problem. One possible approach is to allocate a fixed download/upload limit for each user and charge them if they exceed this limit. This model imposes a download/upload limit on each user to prevent excessive use of channel. Our algorithm has the advantage that it does not imposes any such limit on the users and prevents excessive usage of channel by few rogue flows. Service-tier model allows cable operators to create different service tiers with different prices and download/upload speeds. Each service tier has a minimum and maximum bandwidth limit. The advantage of our algorithm is that it does not limits users to a maximum bandwidth. Depending on the network conditions and the nature of the source, a flow can achieve high download/upload speeds.

6. CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

A NS-2 model for the Media Access Control (MAC) specifications of the DOCSIS 2.0 protocol was designed and implemented. The model extends NS-2 with support for creating high speed data over cable network topologies. The implementation supports most of the important features of the protocol. The scheduling algorithm currently implemented can be easily replaced by other users with their own implementation. The model can be used to explore various research issues in cable networks. Some of the research issues have been identified in section 6.2. The performance study helped in identifying the effect of various system parameters on the performance of the system.

The congestion control algorithm is shown to be effective against bandwidth-hog problem in cable networks. This provides a good alternative to pricing schemes like pay per usage or service-tiers. The algorithm will allow subscribers to achieve very high data-rate and will prevent heavy flows from hogging the bandwidth.

6.2 Future work

The basic model developed as part of this thesis can be extended to support the features not currently implemented. Specifically, the important extensions possible are:

1) Support for Dynamic service establishment

2) Support for S-CDMA physical layer

3) Quality of Service support for Downstream channel

A number of research issues can be explored using the model:

1) A predictive bandwidth allocation algorithm : The idea is to reduce the request/grant delay by proactive grant allocation by the CMTS. The CMTS should be able to predict the future requirements of all or some flows, and proactively provide grants to them. This will also reduce the number of collisions in the system.

2) Algorithm for dynamic assignment of Contention slots: Our model supports a fixed number of contention slots per MAP. Ideally, the number of Contention slots per MAP should be decided by the current network conditions.

3) Algorithm for dynamic MAP times: Our model supports a fixed MAP time. An algorithm needs to be developed which will use the current network conditions to determine the MAP size.

4) Scheduling algorithms for Service-tiers: Cable service providers are moving towards a service-tiered approach where each service-tier has a maximum and minimum bandwidth guarantees. The idea is to develop a scheduling algorithm to support various service-tiers.

7. LIST OF REFERENCES

[1] The Network Simulator, http://www-mash.cs.Berkeley.edu/ns/.

[2] Floyd, S., and Fall, K., "Promoting the Use of End-to-End Congestion Control in the Internet", IEEE/ACM Transactions on Networking, August 1999.

[3] Cablelabs, "DOCSIS Overview", http://www.cablemodem.com/downloads/slideshow.ppt

[4] Taylor A. S. "Characterisation of Cable TV Networks as the Transmission Media for Data", IEEE Journal in Selected Areas in Communication, March 1985, Vol. 3, No 2, pp. 255-265.

[5] Large D. "Effects of network topology on data transmission" Commun. Eng. Des., pp. 36-44, June1983.

[6] CATV Data networks, http://www.linktionary.com/c/cabledata.html.

[7] Victor Rangel, C. Smythe, P. Tzerefos , S. Cvetkovic and S. Landeros, "A Comparison of the DVB/DAVIC, DOCSIS AND IEEE 802.14 Cable Modem Specifications", ICC 2000, Acapulco, May 2000.

[8] DAVIC, http://www.davic.org

[9] IP over Cable Data Network, http://www.ietf.org/html.charters/ipcdn-charter.html.

[10] ITU-T Recommendation H.222.0 (2000) and ISO/IEC 13818-1:2000, Information technology - generic coding of moving pictures and associated audio information systems.

[11] DOCSIS 2.0, http://www.cedmagazine.com/ced/2002/0202/02e.htm.

[12] Data-Over-Cable Service Interface Specifications, CableModem to Customer Premise Equipment Interface Specification, SP-CMCI-I08-020830, http://www.cablemodem.com/specifications.

[13] Data-Over-Cable Service Interface Specifications, Cable Modem Termination System – Network Side Interface Specification, SP-CMTS-NSI-I01-960702, http://www.cablemodem.com/specifications.

[14] Data-Over-Cable Service Interface Specifications, Operations Support System Interface Specification, SP-OSSIv2.0-I03-021218, http://www.cablemodem.com/specifications.

[15] IEEE Std 802-1990, Local and Metropolitan Area Networks: Overview and Architecture.

[16] Data-Over-Cable Service Interface Specifications, Baseline Privacy Plus Interface Specification, SP-BPI+-I09-020830, http://www.cablemodem.com/specifications.

[17] NS by Example, http://nile.wpi.edu/NS/

[18] Data-Over-Cable Service Interface Specifications, Radio Frequency Interface Specification, SP-RFIv2.0-103-021218, http://www.cablemodem.com/downloads/specs/SP-RFIv1.1-I09-020830.pdf.

[19] A. Feldmann, et. Al., "Dynamics of IP Traffic: A study of the role of variability and the impact of control", SIGCOM99.

[20] S. Saroiu, P. Gummadi, S. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems", Multimedia Computing and Networking (MMCN), Jan 2002.

[21] Shivkumar Kalyanaraman, Raj Jain, Sonia Fahmy, Rohit Goyal, and Bobby Vandalore, "The ERICA Switch Algorithm for ABR Traffic Management in ATM Networks" IEEE/ACM Transactions on Networking, Vol. 8, No. 1, Feburary 2000, pp. 87-98.

[22] Ratul Mahajan, Sally Floyd, and David Wetherall, "Controlling High-Bandwidth Flows at the Congested Router", 9th International Conference on Network Protocols (ICNP), November 2001.

[23] Gagan L. Choudhary, “Analysis of combined voice/data/video operation in cable and DSL access networks: graceful degradation under overload”, Performance Evaluation, Volume 52, Issue 2-3, pp. 89-103.

[24] Shrikrishna Karandikar , Shivkumar Kalyanaraman , Prasad Bagal , Bob Packer, TCP rate control, ACM SIGCOMM Computer Communication Review, v.30 n.1, January 2000.

[25] S. Shenker, "Making Greed Work in Networks: A Game-Theoretic Analysis of Switch Service Disciplines," in SIGCOMM Symposium on Communications Architectures and Protocols, (London, UK), pp. 47-57, Sept. 1994 .

[26] J. Martin, A. Nilsson, “Service Level Agreements for IP Networks”, IEEE Infocom 2002, June 2002.

APPENDIX A: Sample TCL Configuration Script

The script creates a simple topology of two cable modems engaged in Downstream FTP. The network topology is shown below.

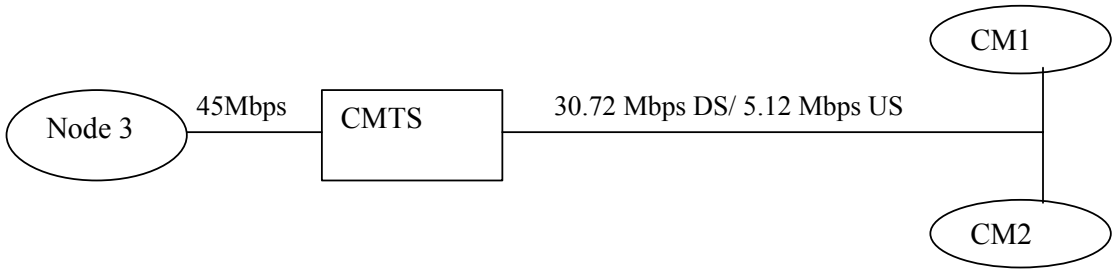


Figure A.1: Sample network topology

Tcl Script:

```
#This assumes  1.6Mbhs channel, 16Qam
set UPSPEED 5120000
#DOCSIS 2.0 with 256 Qam 40.455528MBPS
set DOWNSPEED 40455520

set CONCAT_FLAG 0
set FRAG_FLAG 0
set PIGGYBACKING_FLAG 1
set NUMBER_CONTENTION_SLOTS 9
set NUMBER_MANAGEMENT_SLOTS 3
set BKOFF_START 3
set BKOFF_END 8
set MAP_LOOKAHEAD 0
set MAP_TIME .003
set MAP_FREQUENCY .003
set SERVICE_FLOW_QUEUE_SIZE 20



#Invoke a new ns object
set ns [new Simulator]
set nf [open out.nam w]

set stoptime 100.5
```

```
set printtime 100.0
set tcpprinttime 100.2

#build basic topology (n0 is the CMTS, n1 is CM1, n2 is CM2)
set cmts [$ns node]
set cm1 [$ns node]
set cm2 [$ns node]
set n3 [$ns node]

$ns duplex-link $cmts $n3 45Mb 24ms DropTail

lappend CMnodelist $cmts
lappend CMnodelist $cm1
lappend CMnodelist $cm2

# Create the docsis network


# make-docsislan <node-list> <upstream-channel params> <downstream-channel params> <ll-
# delay> <ll-type> <ifq-type>

# <upstream-channel params = "updatarate ticks maxburst upstream-overhead prop-delay">

#<downstream-channel params = ""downdatarate downstre#am overhead prop-delay">

set docsis [$ns make-docsislan $CMnodelist "$UPSPEED 4 255 80 5us" "$DOWNSPEED 32 5us"
2ms LL]

# configure-cm priority rng_msg_interval cm_id debug on/off
$docsis configure-cm $cm1 1 1.0 1 0
$docsis configure-cm $cm2 1 1.0 1 0


# configure-upflows node "schedtype dst-node pkt-type phs-type frag-enable concat-enable
# concat_threshhold piggyback-enable grant-size grant-interval queue size debug on/off"

$docsis configure-defaultup $cm1  "2 $n3 2 4 $FRAG_FLAG $CONCAT_FLAG $CONCAT-
BURST $PIGGYBACKING_FLAG 0 0 0 $SERVICE_FLOW_QUEUE_SIZE 0"

$docsis configure-defaultup $cm2  "2 $n3 2 4 $FRAG_FLAG $CONCAT_FLAG $CONCAT-
BURST $PIGGYBACKING_FLAG 0 0 0 $SERVICE_FLOW_QUEUE_SIZE 0"

# configure-defaultdown node "dst-node pkt-type phs-type rate-control on/off rate tokenqlen
#  bucket"

$docsis configure-defaultdown $cm1  "$n3 1 4 0 256000 35 24448"

$docsis configure-defaultdown $cm2  "$n3 1 4 0 256000 35 24448"

#configure-mgmtparams cmts-node sync-msg-interval rng-msg-interval ucd-msg-interval
# Dsqlimit
```

```
$docsis configure-mgmtparams $cmts 2.0 1.0 3.0 50


# configure-mapparams cmts-node time-covered map-interval num-SM per map
# num-contention-slots per map bkoff-start bkoff-end proportion Map_lookahead

$docsis configure-mapparams $cmts $MAP_TIME $MAP_FREQUENCY $NUMBER_MANAGEMENT_SLOTS $NUMBER_CONTENTION_SLOTS $BKOFF_START $BKOFF_END 0.5  $MAP_LOOKAHEAD

#Trace the BW
TraceCMBW $docsis $ns $cm1 1.0 CM-BW-cm1.out
TraceCMBW $docsis $ns $cm2 1.0 CM-BW-cm2.out

$ns at $printtime "dumpCMStats $docsis $ns $cm1 CMstats.out"
$ns at $printtime "dumpCMStats $docsis $ns $cm2 CMstats.out"

#Trace CMTS throughput
TraceCMTSBW $docsis $ns $cmts 1.0 CMTS-BW.out


#Trace the DS and US utilization as observed by the CMTS
TraceDOCSISUtilization $docsis $ns $cmts 1.0 CMTS-DS-util.out $DOWNSPEED $UPSPEED

# Dump other cmts stats
$ns at $stoptime "dumpCMTSStats $docsis $ns $cmts CMTSstats.out $DOWNSPEED $UPSPEED


# Start the simulation(starts the registration phase)
$docsis startsim

# Set up the FTP flow

set tcp1 [new Agent/TCP/Reno]
set tcp2 [new Agent/TCP/Reno]

set sink1 [new Agent/TCPSink/DelAck]
set sink2 [new Agent/TCPSink/DelAck]

$tcp1 set class_ 2
$sink1 set fid_ 11
$tcp1 set cxId_ 1
$sink1 set sinkId_ 1

$tcp2 set class_ 2
$sink2 set fid_ 12
$tcp2 set cxId_ 2
$sink2 set sinkId_ 2

$ns attach-agent $cm1 $sink1
```

```
$ns attach-agent $n3 $tcp1

$ns attach-agent $cm2 $sink2
$ns attach-agent $n3 $tcp2

$ns connect $tcp1 $sink1
$ns connect $tcp2 $sink2


$tcp1 set window_ 22
$tcp1 set fid_ 10
$tcp1 set packetSize_ 1460

$tcp2 set window_ 22
$tcp2 set fid_ 11
$tcp2 set packetSize_ 1460

set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1

set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2

$ns at 1.101 "$ftp1 start"
$ns at 1.101 "$ftp2 start"

$ns at $stoptime "finish

proc finish {} {
global ns nf
   $ns flush-trace
   close $nf

   exit 0
}

#Start the simulator
$ns run
```

Output files produced:

CM-BW-CM1.out, CM-BW-CM2.out: These files contains the downstream and upstream bandwidth of CM every monitoring interval. The format of the output is :

Current_time Upstream bandwidth Downstream bandwidth

CMstats.out: This file contains the following statistics for all the CMs configured in the simulation. These statistics are dumped at the end of the simulation.

Total number of bytes received, Total number of bytes sent, Total number of packets sent, Total number of packets dropped, Loss rate, Total number of collisions, Collision rate, Total number of packets dropped due to collisions, Total number of packets dropped due to service-queue overflow, Total number of application bytes on Upstream channel, Total number of application bytes on Downstream channel, Total number of piggyback request sent, Total number of contention request sent, Total number of first collision, Avg. queuing delay.

CMTS-BW.out: This file contains the downstream and upstream bandwidth of the CMTS every monitoring interval. The format of the output is :

Current_time Upstream bandwidth Downstream bandwidth

CMTS-DS-util.out: This file contains the average utilization of downstream and upstream channel every monitoring interval. The format of the output is:

Current_time Upstream utilization Downstream utilization

CMTSstats.out: This file contains the remaining statistics. These statistics are dumped at the end of the simulation. The format of the file is:

Total bytes received on Upstream channel, Total bytes sent on downstream channel, Total number of application bytes on Upstream channel, Total number of application bytes on Downstream channel, Total number of packets dropped in Downstream transmission queue, Loss rate, Downstream utilization, Upstream utilization, Number of contention request received, Number of piggybacked request received, Number of contention request granted.

Per SID stats:
Total bytes sent on Downstream, Total bytes received on Upstream, Total number of Packets dropped in Token queue, Total number of packets dropped, Loss rate