Scheduling Algorithms for High-Speed Data Service over CATV

by
Richard R. Rabbat
M.E., American University of Beirut (1996)

Submitted to the Department of Civil and Environmental Engineering in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1998

Author .....
Department of Civil and Environmental Engineering
April 27, 1997

Certified by .....
Sunny(Kai-Yeung) Siu
Department of Mechanical Engineering
Thesis Supervisor

Certified by .....
Steven R. Lerman
Department of Civil and Environmental Engineering
Thesis Reader

Accepted by .....
Prof. Joseph M. Sussman
Chairman, Departmental Committee on Graduate Studies

Scheduling Algorithms for High-Speed Data Service over CATV

by

Richard R. Rabbat

Submitted to the Department of Civil and Environmental Engineering on April 28, 1998, in partial fulfillment of the requirements for the degree of Master of Science

Abstract

Over the past few years, data traffic over the Internet has increased tremendously. Audio, video and simulation data traffic is now common over the World Wide Web. Integrated services networks need Quality of Service (QoS) guarantees, such as minimum bit-rate guarantee, low delay bounds and fairness in the distribution of extra bandwidth. Cable TV (CATV) networks can be used for the delivery of high-speed data service to customers using an available physical infrastructure. Cable companies have been working as part of a consortium on a protocol for CATV. The work that is presented is research conducted to provide QoS guarantee as part of a set of protocols for the delivery of Internet traffic over CATV. The work that we present is research in algorithms that would provide QoS guarantee for CATV networks. We make use of ideas implemented in ATM (Asynchronous Transfer Mode) networks.

We present an in-depth study of the proposed cable network protocol and a literature review of scheduling algorithms. We then discuss our scheduling algorithm that extends the capabilities of the proposed protocol by providing minimum bit-rate guarantee. This scheduling algorithm is simulated in Opnet. We discuss details of the implementation and the results we get from the Opnet simulation. In order for our scheduling algorithm to be effective in an Integrated Services network, we enhance the algorithm to provide a constant bit-rate guarantee, and delay bound on the traffic of Internet Telephony flows. We discuss a dynamic polling mechanism to ensure low delay on certain types of latency-critical flows.

Thesis Supervisor: Sunny (Kai-Yeung) Siu
Title: Assistant Professor

Acknowledgement

I would first like to thank my advisor Sunny Siu, for his directions and guidance during the research and the writing of this thesis. Sunny did the best a student could ask from his advisor.

I would like to thank Steve Lerman, for his unfettered support during my long stay at the Center for Educational Computing Initiatives, as well as his great feedback on my thesis. It was under his leadership that I was able to grow intellectually at MIT.

I would also like the opportunity to thank Cynthia Stewart who sheltered me from all the administrative headaches of MIT. She was the main reason behind my getting "The Coolest Guy in the Department" award from the Head of the Department Rafael Bras.

Thanks to Lisa Latham for reading through the whole draft and correcting all those grammar and English mistakes.

Now the friends - Lebanese, American and international. You made my stay at MIT livable. For that I thank you all and I'd like to mention some of you: Jody, the members of party-com: Saad, Walid, Ziad, Joe, JC and Mahdi. I would also like to mention Issam, Gilly, Jacqueline, Rabih, Cyril, Julio and Brett.

This research was supported by MediaOne. Thank you, Rich Woundy.

To Josephine, Ronald, Ralph

I LOVE YOU!

Table of Contents

1 Introduction 9
1.1 General Discussion 9
1.2 Thesis Outline 13
2 The MCNS Protocol and Scheduling Algorithms 15
2.1 The MCNS Protocol 15
2.1.1 The Cable Network Setup 16
2.1.2 The MAC Sublayer 17
2.2 Scheduling Algorithms 21
2.2.1 First-Come First-Serve (FCFS) [6] 21
2.2.2 Round Robin [7] 23
2.2.3 Deficit Round Robin [7] 24
2.2.4 Virtual Clock [8] 26
2.2.5 Leap Forward Virtual Clock [9] 26
2.3 Recapitulation 28
3 Minimum Bit-Rate Guarantee for MCNS Protocol 29
3.1 Presentation of the Algorithm 29
3.2 The MCNS Implementation in Opnet Simulation 32
3.3 Simulation Results 36
3.4 Recapitulation 39
4 Enhancements for Internet Telephony Traffic 40
4.1 Delivering Delay Bounds 40
4.2 Dynamic Polling 44
4.3 Summary 48
5 Conclusion 50
5.1 Summary of Work 50
5.2 Improvements and Future Work 52
Bibliography 54

List of Figures

Figure 2.1: Connection of various parts constituting the CATV network. 16
Figure 2.2: Physical setup of CMTS and CMs in the cable network. 17
Figure 2.3: Timeline of operation of the upstream communication channel. 19
Figure 2.4: Flows coming to scheduler and being serviced in the FCFS queue 21
Figure 2.5: Round Robin scheduling of data streams. 23
Figure 2.6: Deficit Round Robin example. 25
Figure 3.1: Example of scheduler behavior 31
Figure 3.2: Pseudocode for minimum bit-rate guarantee algorithm. 32
Figure 3.3: Implementation of cable network in Opnet Modeler [10]. 33
Figure 3.4: CMTS_Manager state transition diagram [10]. 35
Figure 3.5: Line chart for bandwidth usage by the different CMs [11]. 38
Figure 4.1: Distribution of bandwidth in MAPs [11] 41
Figure 4.2: Pseudocode for Telephony traffic. 42
Figure 4.3: Setup process for flows with stringent delay bounds. 43
Figure 4.4: Dynamic polling of flows with different delay priorities. 48

List of Tables

Table 3.1: Variables of interest in OPNET simulation. 34
Table 3.2: Results of the simulation [11]. 37

Chapter 1

Introduction

1.1 General Discussion

Now that the Internet is a part of everyday life, research has shifted toward developing networks that can be used to connect individuals at their homes. Traditional methods, such as using of analog phone lines with modems for data transmission, have become slow as the nature of the Internet traffic has shifted to bandwidth-demanding multimedia data, large applications running over the network, and real-time traffic. In addition, the traffic that was mostly one-way has transformed, as clients send substantial quantities of information towards the server, and users engage in delay-critical tasks such as audio and video conferencing.

Multiple technologies have been used to provide high-speed Internet access, such as leased lines, ISDN (Integrated Services Digital Network) lines, DSL (Digital Subscriber Line) and Cable Modems. Leased lines are fast and reliable, they guarantee available bandwidth and have a low delay, but do not constitute an economical choice. They tend to cost more than small businesses and residential users can afford, and are thus oriented primarily to larger businesses. ISDN technology has been commercially available for a long time. Despite its high cost of equipment, the need for high-speed data communication has made ISDN technology very attractive. DSL and Cable Networks are newer technologies. DSL uses the existing copper phone lines, which avoids the need to upgrade the physical network and allows transmission speeds of a few hundred kilobytes per second. Cable networks also make use of an existing physical network (Cable TV), allocating a certain number of channels for digital transmission of data to

provide a high-speed, low cost option for the end-user. The work we are addressing in this thesis relates to this last type of network.

With the increasing demand for QoS on data networks, there is a need to provide service guarantees for networks to allow a high quality of transmission. That quality of transmission can be thought of in terms of low delay, minimum bit-rate guarantee, and dynamic allocation of different levels of service for the customer.

This thesis addresses the data communication requirements of Internet Telephony as well as more traditional data service over cable networks. Our approach to providing quality of service at the MAC sublayer allows upper layer protocols to assume a reliable underlying system. Protocols such as ReSerVation Protocol (RSVP) [1] can then deliver the quality that is needed for the flows that require specific guarantees on bandwidth usage or delay bounds.

RSVP [1], as described in the Internet Engineering Task Force's (IETF) Request for Comment 792, works with routing protocols such as ICMP [2]. On the host side, it asks the network to deliver a certain quality for data flows. On the router side, it follows the path provided by the routing protocols, and requests each node or router on that path to deliver the service it needs for the flow. The routers are responsible for allocating enough resources on the parts of paths they control, and maintaining that resource. RSVP lends itself to usage on different network technologies. It can be used for example on cable networks, because it assumes simplex flows and totally ignores any association between sender and receiver.

A RSVP session does not guarantee any quality of service, but rather tries to coordinate the delivery of QoS to the data flow by working in conjunction with the network itself. A RSVP session should be thought of as an integrator of the network's

efforts to provide QoS. It works over a wide variety of network architectures to get the appropriate level of service. RSVP needs to be used over an entire path from sender to receiver. Its deployment does not need to be global, but rather specific to the needs of certain nodes in the whole network.

This thesis discusses providing QoS at the MAC (Medium Access Control) layer. RSVP would be supported over cable networks, as it is designed to work in conjunction with a sophisticated MAC layer to provide new traffic capabilities. RSVP would run within the transport layer of a network stack and request QoS guarantees from the lower MAC layer, guarantees that this present thesis develops and discusses at length. As previously discussed, there can only be an RSVP session if the rest of the route is RSVP-aware. RSVP can be used to provide QoS between hosts over the cable network itself, and at a later stage, over Internet routes running RSVP. RSVP would ask the cable network server to reserve a certain type of service for different flows, a request processed and attended to by the MAC layer scheduler.

First, the thesis discusses an algorithm that we develop to provide at first minimum bit-rate guarantee to flows, disallowing any one flow from using up all the resources of the network. The server allocates a different bandwidth to each host on the network according to the service agreement with each host. Customers can request more bandwidth and be charged more. The scheduling scheme guarantees bandwidth over all time periods.

Second, we address the issue of how to allocate extra bandwidth. After all the guaranteed bandwidth has been allocated and used, we provide means to distribute the remaining bandwidth fairly to all active hosts or cable modems. For an extra bandwidth of 100 K-bps for example, with 10 cable modems sharing the bandwidth, the scheduling

algorithm should allocate 10 K-bps to each.

The third part of the thesis deals with the servicing of new types of flows with stringent needs in terms of delay bounds. Flows used for Internet Telephony have a constant bandwidth requirement (64 K-bps) but a very small delay bound (around 50 ms) [11], which if not met, makes voice-based communication ineffective. We explore ways of providing that delay bound by developing a dynamic polling policy that allows the scheduler to include a priority scheme: flows with a small bandwidth requirement but a stringent delay requirement will be serviced with a higher priority over flows that only need minimum bit-rate guarantee.

Latency-critical flows will be registered with the server as requiring low delay and constant bit-rate and will be serviced according to that agreement. The scheduler will keep track of the bit-rate usage and make sure it is constant over all time periods. It will take flows that try to use more resources than what they are allowed off the priority list. It will flag those flows as ill-behaved and stop reserving resources for them. Instead, it will only service them the way it services flows discussed earlier; it will only start giving them low latency again when they keep their resource requirements lower than or equal to the agreed-upon resources.

In addition to the types of flows discussed above, our work also lays the groundwork for servicing more complex flows over cable networks, such as Variable Bit-Rate (VBR) service for several applications that need this type of assured service. Video On Demand (VOD) and video-conferencing are good examples of such applications where the bit-rate is variable because of the nature of the data. Generally, when doing video compression, two adjacent frames **j** and **j+1** are often very similar. Thus we need a low bit-rate to transmit frame **j+1**. The next frame, **j+2**, could be totally different, and require a high

bit-rate to be transmitted.

1.2 Thesis Outline

This thesis describes the work we conducted implementing QoS guarantees over Cable Networks. It then introduces the issues that arise in this domain, the ways we address them, and the results that we obtain. It also presents future research that could be conducted in the area of cable networks in general and in scheduling over those types of network particular.

Chapter two describes the cable network's underlying architecture, and in particular the MAC sublayer which serves as the container of the system that we research for delivering QoS guarantees. It also presents several approaches from the literature that address the scheduling of clients by the server. Those algorithms are used to provide a breadth of services over different network architectures.

In chapter three we describe the scheduling algorithm that this research has generated and discuss the means by which we can provide a minimum bit-rate guarantee. Then we implement this algorithm using the Opnet simulation software. Chapter three considers the Opnet simulation for the Multimedia Cable Network System (MCNS) protocol in general, then discusses the different enhancements that we perform to implement our scheduling algorithm. Results of the simulation and a comparative study with results from the default scheduling algorithm implemented in the simulation are also shown.

Chapter four explores enhancements to the scheduling algorithm, which allow Internet telephony to be used on the cable network. Mainly we enhance the system to provide a Constant Bit Rate guarantee and an upper delay bound. The chapter discusses in detail the dynamic polling algorithm that allows a select group of flows to make use of those added capabilities.

Chapter five is a conclusion of this thesis. It summarizes the accomplishments related in the earlier chapters, while also providing an extensive agenda for future work that could advance the state-of-the art in cable networks.

Chapter 2

The MCNS Protocol and Scheduling Algorithms

2.1 The MCNS Protocol

A consortium of cable companies has been actively building a protocol specification (Multimedia Cable Network System) for the cable networks. The MCNS protocol [3] describes the system used to transport IP (Internet Protocol) traffic between the Cable Modem Termination System (CMTS) and the Cable Modems (CMs). The physical layout of the entire system is a CMTS connected to the Internet on one end by a high-speed line and to the CM on the other end by the cable network. The CM is then interfaced to the Customer Premise Equipment (CPE). Interfacing CMs and CPEs is typically done by connecting the CM to the output of an Ethernet card on the CPE. The specification does not disallow a cable modem to be designed as an extension card that could fit in the customer's computer. Other machines on the customer premises could be connected to the CMTS by turning the connected computer into a multi-homed computer that would allow IP traffic between the CM and the Local Area Network (LAN). The machine that is connected to the cable network can also act as a router to connect other types of networks to the cable network. The setup is shown in figure 2.1.

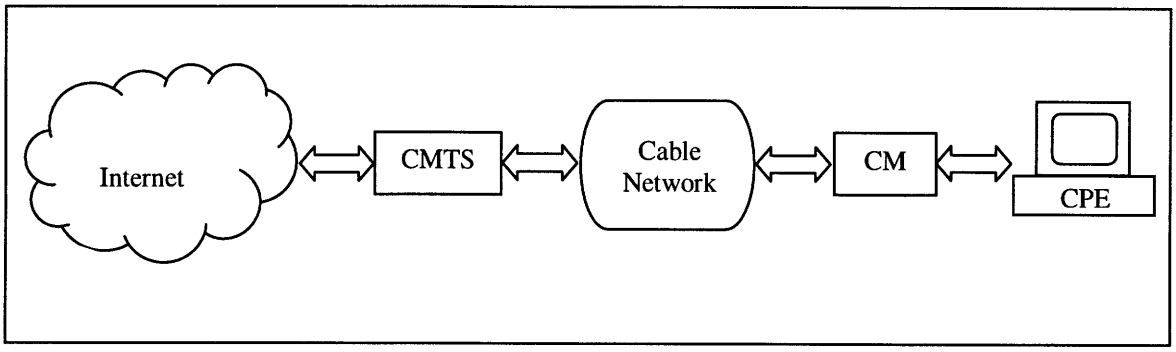

Figure 2.1: Connection of various parts constituting the CATV network.

2.1.1 The Cable Network Setup

The **cable network** is usually a hybrid-fiber/coax (HFC) or an all-coax network. It allows two-way transmission on a typical distance of around 15 to 20 kilometers between the CM and the CMTS, with a maximum distance of up to 160 kilometers. This network separates upstream and downstream frequencies and allows transmission of NTSC analog television or digital data on any channel. Any number of channels can be allocated for either kind of signal in a seamless fashion. The network allows different ways of error recovery and fault isolation so that different kinds of hardware and software can work properly on the shared medium. This would also allow future enhanced hardware to be designed easily for that type of network.

Our interest lies in enhancing the protocol for the entire system, which would allow the exchange of data between CMTS and CMs. The work conducted and discussed in this thesis relates mainly to the MAC layer part of the protocol.

2.1.2 The MAC Sublayer

The MAC sublayer for CATV is close to a typical MAC sublayer in existing LANs but differs on a few points; for example, the channels are simplex and do not fit the assumptions of several IP protocols.

As can be seen in figure 2.2, the CMTS communicates with multiple CMs using a shared cable, an HFC cable in general. CMs use the same channel to send data on to the CMTS.

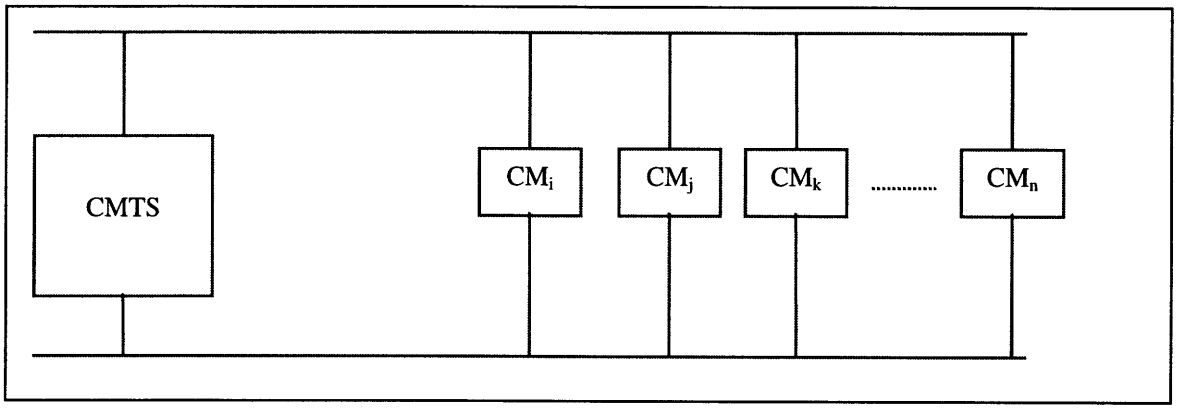

Figure 2.2: Physical setup of CMTS and CMs in the cable network.

The MAC sublayer takes care of a set of functionality in the MCNS protocol. It first allows the CMTS to allocate bandwidth to the different CMs requesting it. The CMTS allocates bandwidth to the CMs by reserving mini-slots in the upstream direction. The mini-slots have a certain time length depending on the modulation. A typical length of a mini-slot is 0.0125 milliseconds, fitting 32 symbols each. Two different modulation techniques are used: Quadrature Phase-Shift Keying (QPSK) and Quadrature Amplitude Modulation (QAM). In the case of QPSK modulation, each mini-slot fits 8 bytes whereas 16QAM modulation fits 16 bytes in one mini-slot. The upstream bandwidth is allocated

for both contention and reservation periods in terms of mini-slots. There is also a small bandwidth allocated to allow new stations to join the system.

Each period is described by the CMTS in an allocation **MAP**. This MAP is broadcast by the CMTS to all CMs to allow the distribution of upstream bandwidth. The CMTS is responsible for allowing and disallowing CMs from transmitting data in the upstream direction. The MAP shows the allocation of bandwidth for the next period, and has a default length multiple of 6.5 milliseconds.

The contention period contains mini-slots that any CM can use, while the reservation period assigns unique transmit opportunities to one CM at each particular time-slot. During the contention period, there is a probability that the data transmitted by a certain CM will collide with other data from one or other CMs, leading the CMTS to apply collision resolution to allow fewer CMs to send data in that period, thus reducing the probability of collision.

In a scenario where a large number of CMs are transmitting data in the contention period, the mini-slots will have a high probability of carrying garbage data, because two or more CMs would have sent upstream data in those mini-slots. It is the responsibility of the CMTS then to have a smaller window of opportunity for CMs to send data: fewer CMs will be able to contend for bandwidth during the next period. This window will diminish in size again if the CMTS senses many collisions in that next period, or the window will be relaxed if it detects a minimal number of collisions.

Figure 2.3 shows the time-line for communication between CM and CMTS, with the MAP generated by the CMTS reaching the CMs so they can use the upstream bandwidth.

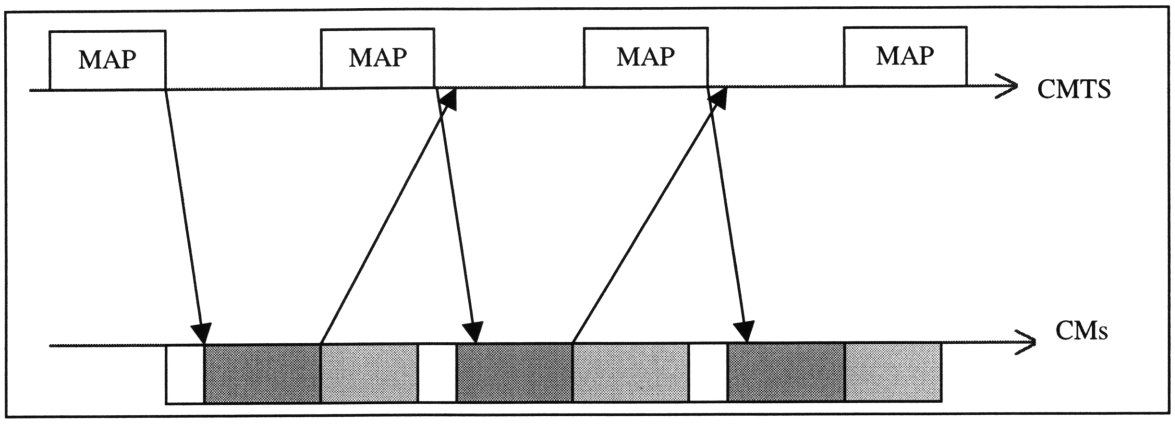

Figure 2.3: Timeline of operation of the upstream communication channel.

The CMTS sends MAPs downwards to all CMs. The CMs make use of the upstream bandwidth accordingly: the dark-shaded area is the reservation period, in which CM **i** can send data in the slots that the CMTS has designated to it in the earlier MAP. The light-shaded area represents the contention period, during which any CM, pending the right to send in that period, could choose to do so. The all-white region is the period where new CMs can join the system. The CMTS does not need to wait for the whole period to be consumed before it broadcasts the new MAP. Rather, it sends MAPs downstream as soon as it can map the next period. In this particular example, a CM that has sent a request for bandwidth will receive a data grant and be able to send data in the period described by the last MAP. The extra delay is needed to ensure smooth operation of the network without wasting bandwidth. This shortcoming is temporary as all CMs can send data requests piggybacked to the data they are sending in the upstream. The delay also applies to new CMs that want to join in, as they would have to wait an extra cycle before sending data in the reservation period. The connected CMs and CMs that need an immediate higher data rate can bypass this requirement by sending data in the contention period and asking for an **ACK** (acknowledgement) on that data from the CMTS. This

method allows both low delay and reliable (CM gets ACKs on data it sends) transmission.

A typical operation of the network at the end-user level starts with an initial ranging request that the CM sends to the CMTS, containing a particular Service ID (**SID**). The SID identifies a unique CM-CMTS mapping. The ranging request allows the setup of the connection. To this request, the CMTS sends a response that includes information about the network, including:

1. Timing adjust information: tells the CM to offset its transmissions so they reaches the CMTS at the expected time
2. Power adjust information: allows the CM to adjust the power level so the transmissions reach the CMTS at the desired power
3. Frequency adjust information: informs the CM to adjust the frequency of its transmissions to the desired frequency
4. Ranging status: tells the CM whether data sent to the CMTS is reaching it within an acceptable delay.

This information is used to allow the CM and the CMTS to synchronize their operation. Now that the CM is registered with the CMTS, it can perform all the above-mentioned operations, such as requesting bandwidth and sending upstream data.

The system discussed does not provide for Quality of Service guarantees. The MAC protocol does not assume a low-level network as does IP [4], but the CMs have no knowledge of the amount of data they can send, what speed that data can be transmitted at, and what delay bound to expect in the CATV network. Scheduling of upstream bandwidth is left to the third-party manufacturer of the CMTS and CM to design and implement. The specification for the MAC layer does provide suggestions as to the

implementation of different classes of service for Integrated Services on the Internet [5], but does not go into details of any system that could provide those classes of service. Basically, the specification has the flexibility to accommodate a wide range of improved scheduling mechanisms. The research in this thesis mainly covers the scheduling aspect of the CATV network as discussed previously. For this purpose, we study the different scheduling algorithms that provide different levels of QoS for the hosts on those networks.

2.2 Scheduling Algorithms

Scheduling algorithms that implement QoS for integrated services over the Internet architecture are discussed in several publications of interest to this thesis. Different algorithms address different services such as minimum bit-rate guarantee, CBR (Constant Bit Rate) service, VBR (Variable Bit Rate), low or fixed delay, etc.. We present some of those algorithms in sections 2.2.1 through 2.2.5.

2.2.1 First-Come First-Serve (FCFS) [6]

In this scheduling scheme, let us consider several flows 1 to n. All incoming flows are stored in a buffer that acts as a FIFO (First In, First Out) queue data structure. This is a process that allows all requests to be processed successively, in the order they were received. The operation of that algorithm is shown in figure 2.4.

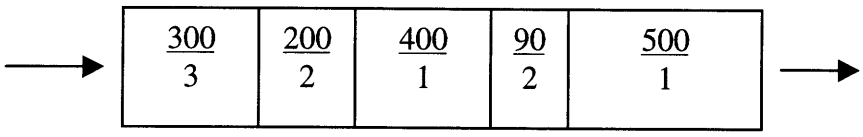

Figure 2.4: Flows coming to scheduler and being serviced in the FCFS queue.

The figure shows operation at the router or server. Each section is denoted by two integers; the upper value denotes the number of bits needed and the lower value denotes

the flow number. In the case of a system that operates on data requests and data grants, those values denote the amount of data bits that are requested by the different active flows. In the case of pure data forwarding, the values are the actual data chunks sent over the datagram network, awaiting forwarding. Flow 1's request for 500 bits has been longest in the queue and will be serviced next. Flow 3's request for 300 bits has just arrived and been enqueued while waiting for all previous requests to be processed first.

The FCFS algorithm would work properly in conditions where all flows are sending requests at the same rate for an equal share of the bandwidth. When these conditions are met, the scheduling is fair and simple as it executes in **O (1)** worst-case running time. This could apply when all flows are similar; e.g. all flows need constant bit-rate guarantee to deliver streaming audio at the same rate to all listeners. Another example of a system where the FCFS algorithm would be both useful and elegant is a phone network, where all flows have the same bandwidth requirement and delay bound.

The problem with this scheme is the heterogeneity of the flows on a data network. Some traffic could be bursty, some other traffic might need constant bandwidth, yet another type would require a very small end-to-end delay, etc.. Therefore, an FCFS queue would not perform efficiently in a data network because flows on a data network do not have similar bandwidth requirements.

To summarize the issues, this algorithm is not fair and cannot guarantee delay bound on the network. Still, it is very important to understand its operation, as it serves as the basis for all other developed algorithms that try to add enhancements to its skeleton to deliver the types of QoS guarantees needed in an advanced communications network. The next scheduling algorithm discussed is the Round Robin (RR) algorithm as well as its variant, the Deficit Round Robin (DRR).

2.2.2 Round Robin [7]

Round Robin adds to FCFS the ability to consider each flow in its separate buffer. Instead of all data requests coming in a serial fashion, all of them come in a parallel fashion as shown in figure 2.5.

Each request from flows **1** through **n** is buffered in a flow-specific queue. The scheduler passes by each buffer in a round robin fashion and services the next request. In the figure, the scheduler is at flow **1** and grants it 300 bits that it requests. It then moves on to flow **2** and grants it 200 bits, eventually getting to flow **n-1** that is served 500 bits and flow **n** that is granted 700 bits. The round starts again over flows **1** to **n**.

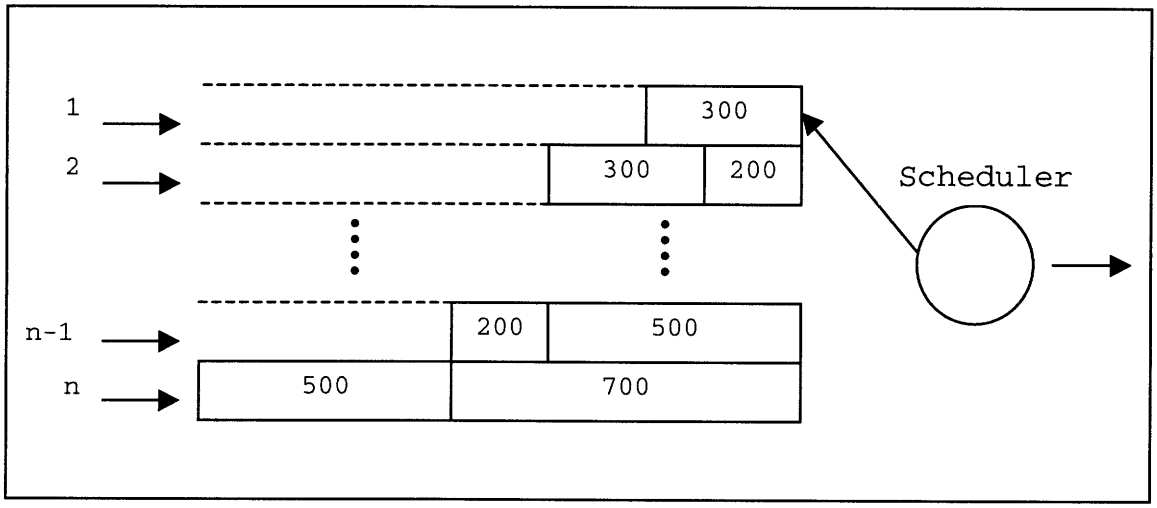

Figure 2.5: Round Robin scheduling of data streams.

Round-Robin allows a big improvement over FCFS. In FCFS, a single stream can clog up the resources of the whole network by sending a series of requests for as much bandwidth as it desires. Therefore, all of its data requests would be serviced before any other flow has the chance to ask for bandwidth. In this scheme, each flow can only receive one grant per scheduler cycle as the scheduler passes by each flow exactly once and grants the bandwidth needed during that cycle.

Round-Robin's improvement over FCFS is that when flows send fast or slow requests for a certain bandwidth, they are serviced equally. Round Robin can be used in networks where flows require a similar share of the bandwidth with no restriction on the data request rate. The worst-case running time for Round-Robin is **O (1)**.

The weakness of the Round-Robin scheduler shows in case flows request different bit rates, such as figure 2.5 depicts. Flow **2** has two requests pending in the queue, one for the amount of 200 bits, the other for 300 bits, while flow **n** is requesting 700 then 500 bits. Round Robin will service both flows equally in terms of number of requests, allowing flow **2** to send 500 bits, and flow **n** 1200 bits in two cycles of the scheduler. For traffic such as real-time traffic with small and numerous datagrams (a telnet session for example), the system is unfair and will only service a very small proportion of the requests of this flow. There is no guarantee for minimum bit-rate. Deficit Round Robin solves that problem as discussed in the next section.

2.2.3 Deficit Round Robin [7]

Deficit Round Robin improves on Round Robin by providing minimum bit-rate guarantee. In order to achieve that, it adds a simple data structure that keeps track of the bandwidth usage by the different flows. Each flow is allocated a certain value of bandwidth **BW(i)** that states the number of bits a flow is allowed over each time period, the round-robin round. A Deficit Counter (**DC**) is initially set to 0. It is incremented by **BW(i)** every round. Each time flow **i** is serviced by **bits(i)**, **DC(i)** decreases by **bits(i)**. All consecutive datagrams that await servicing in the queue of flow **i** will be serviced while **DC(i)** still allows that, until no such packets exist. We have the constraint that **DC(i)** should remain non-negative at all times. The round will go to the next flow.

Let us return to the example in 2.2.2, to discuss how the algorithm would work in this

case. The values in the deficit counter show the number of bits that each flow can send. The scheduler when reaching for example flow 2, increments **DC(2)** by **BW(2)** = 100 to 700. It can service both packets of 200 and 300 bit lengths in that queue. **DC(2)** will therefore drop by (200+300) to 200. The next packet has a length 300 which is larger than the new value of **DC(2)** so the scheduler will not service that packet but proceed to other flows and service packets waiting in their queues if the packet lengths are smaller than the DC at each particular flow. In that respect, the scheduler reaches flow **n-1** where it increments **DC(n-1)** by **BW(n-1)** = 250; the scheduler in this case will not allow the 500 bit packet to be serviced because DC(n-1) < 500. It will go on with its round to flow **n** and apply the same process.

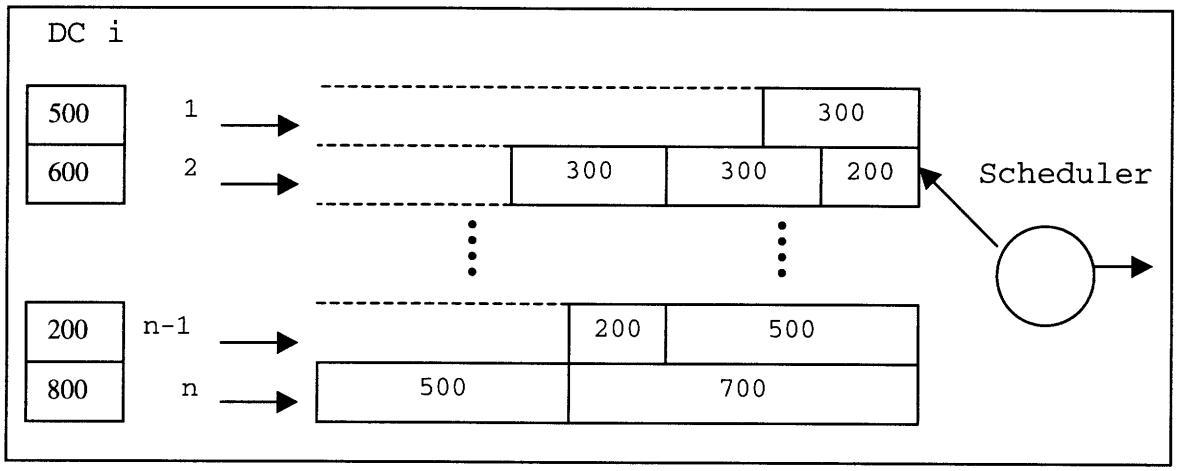

Figure 2.6: Deficit Round Robin example.

The operation of the scheduler with the help of the deficit counter allows it to deliver a minimum bit-rate guarantee, as it can ensure all flows are receiving sufficient bandwidth over each round. To disallow flows that go idle for a long period to seize all the network resources once they start sending data again, the scheduler always resets the DC of empty queues.

The shortcoming of this scheduling algorithm is that it cannot guarantee a small delay

to any flow. In addition, it does not assume that extra bandwidth would be available, and therefore does not try to distribute it to the different flows. In fact, with extra bandwidth at hand, there is no way to increase **BW(i)** to allow that extra bandwidth to be utilized. This functionality could be added, but is not part of the specification per se.

2.2.4 Virtual Clock [8]

The Virtual Clock scheduling algorithm has a worst-case computation time of **O (log n)**. It allows routers to distribute bandwidth fairly among different flows. The principle behind the algorithm is to tag each packet. That tag is the clock time at which the packet should have been sent. The server maintains this virtual clock. All packets get serviced in non-decreasing order of their tag values. Each packet that is up for servicing by the server is tagged by a number indicating that it should have been serviced before the virtual clock reaches that time. Each packet serviced increments the virtual clock by (packet length / total bandwidth). In a well-behaved network with packets sent by their tag time, the algorithm can guarantee a low delay bound. For flow **f**, the packet **j** reaching the head of the queue will have a tag **T** calculated in terms of:

- the arrival rate **A** of the packet,
- the previously calculated tag **T** of packet **j-1**,
- the length $\mathbf{l}^{\mathbf{j}}$ of the packet,
- the rate **r** of servicing of that flow.

The equation that calculates this tag is as follows:

$$T(p_f^j) = \max\{A(p_f^j), T(p_f^{j-1})\} + \frac{l_f^j}{r_f}$$

Tags are calculated using this equation and allow for a proper bit-rate guarantee. The

scheduling of packets using that scheme is faster than the more traditional Fair Queuing, but leads to unfairness in the case of bursty traffic. In that case, there will be a large difference between the server clock and the flow tag. Thus traffic that has been idle for a while will be able to use most of the resources of the network, keeping other flows from getting serviced. This results in a system that can service the needs of flows that are non-bursty and have a reasonably close bandwidth requirement. Leap Forward Virtual Clocks address those two problems as shown below.

2.2.5 Leap Forward Virtual Clock [9]

Leap Forward Virtual Clock solves the problems of the Virtual Clock. In this case, the algorithm corrects ill-behaved flows by putting them offline temporarily. Flows that do not keep to a certain bandwidth (denoted as the throughput condition) are thought of as oversubscribed. In this case, instead of applying the equation above to determine the tag time of the packet in that flow, the server will just keep it in a holding area. So, it treats well-behaved flows as having higher priority, and services these flows first before getting to the lower priority flows in that holding area. The flows in the holding area will then be serviced.

If all flows become oversubscribed, the server can leap forward and transfer at least one ill-behaved flow to the queue of well-behaved flows to ensure continued use of the bandwidth available. Transferring flows to the well-behaved queue is more conservative and is only done when the bandwidth needed for that flow falls below a critical level. The following equation governs transfer from status of well-behaved to ill-behaved:

$$t_f - \Delta_f \geq t_s$$

with $\mathbf{t}_s$ being the server clock, Δ_f being the time needed to send the largest packet at the minimum bit-rate guarantee, and $\mathbf{t}_f$ being the tag for flow **f**. When flows are moved from

the ill-behaved category to well-behaved queue, the following equations must hold:

$$t_f \leq \tau + \Delta_f + t_s$$

where τ is the time needed to transfer the largest packet in the system.

So, ill-behaved flows need to display "good behavior" before being permitted to send data packets in the system.

The running time for the Leap Forward Virtual Queue is **O (log log n)**. It provides fairness between contending flows and does not result in the bursty behavior displayed by classical Virtual Clocks.

2.3 Recapitulation

We have presented the MCNS protocol, its advantages, and the shortcomings that have led to our undertaking this research in scheduling algorithms for cable networks. We have also presented some algorithms used to schedule traffic over packet-switched networks to provide QoS guarantees. These have different QoS capabilities but serve as a pre-requisite for devising an algorithm that works on cable networks. That algorithm is the subject of chapter three, together with the results of a simulation that we ran on the Opnet Modeler™ network simulation software.

Chapter 3

Minimum Bit-Rate Guarantee for MCNS Protocol

3.1 Presentation of the Algorithm

The main focus of our research was to implement QoS guarantees to allow for different levels of service. We have devised an algorithm to guarantee minimum bit-rate for flows. Flows identified by their SID (Service ID) do not need to have the same bandwidth using this scheme. Each one can be guaranteed a different bit-rate that the CMTS will deliver. We need to make sure that they do not overload the network. Our algorithm takes care of discrepancies and misbehavior of flows. Essentially, the CMTS and CMs that need minimum bit-rate guarantee first negotiate the minimum bandwidth that a flow from the CM will use. This negotiation can be done in one of two ways:

1. Statically, where the customers have requested and signed up for a certain bit-rate at account setup. They will be served that throughput for all the sessions.
2. Dynamically, with the customer and server negotiating to determine a certain bandwidth allocation for the duration of the connection. If the customer has different bandwidth requirements, depending on the time of the connection, the bandwidth that a particular CM is allocated could be allowed to fluctuate. The CMTS can keep track of the usage on a periodic basis and charge the customer the appropriate amount.

Irrespective of the allocation mechanism, after the negotiation process, the server will possess information about the bandwidth that it should provide for the particular flow. All flows with minimum bit-rate guarantee will be monitored in a table with their respective bit-rates. In addition, each record in the table will also have a field for the bit

usage of its corresponding flow. Every time a flow requests bandwidth from the CMTS, the CMTS allocates a number of mini-slots to the particular SID. The number of mini-slots is a factor of the number of bits needed to meet the requested throughput over a period T. Then the CMTS updates the bandwidth usage of that SID in the record entry for that SID. At any time, there will be **n** requests pending from the different flows. The CMTS calculates priorities of these flows by looking at the current bit usage. Flows with lower bit usage will have a higher priority of being served and vice versa. Every several MAPs, meaning after T seconds, table entries for current bit usage are updated by subtracting from them T times the guaranteed minimum bit rate.

The priority system used is both easy to design and implement. The extra bandwidth should also be taken into account, allowing the scheduler to distribute it fairly among the different cable modems, that is, all cable modems would be able to get an equal share of the extra bandwidth. Fairness is important in our scheme to allow cable modems that need bandwidth beyond the guaranteed bit-rate to take an equal part, and thus be able to profit from any extra bandwidth. Distribution of extra bandwidth is important in case we have either flows that are not requesting a reservation or flows that have a low bandwidth need. When extra bandwidth is available, we should allow other flows to use it. The equation that governs that behavior is as follows for a flow **k**:

$$Bit\ usage\ [k] \leftarrow Bit\ usage\ [k] - T * minimum\ bit\ rate\ [k]$$

Every time period T, this subtraction is performed and the scheduler re-prioritizes the flows from smallest to highest bit usage. Figure 3.1 shows an example of the algorithm for two flows **j** and **k**, with different requests coming for bit grants.

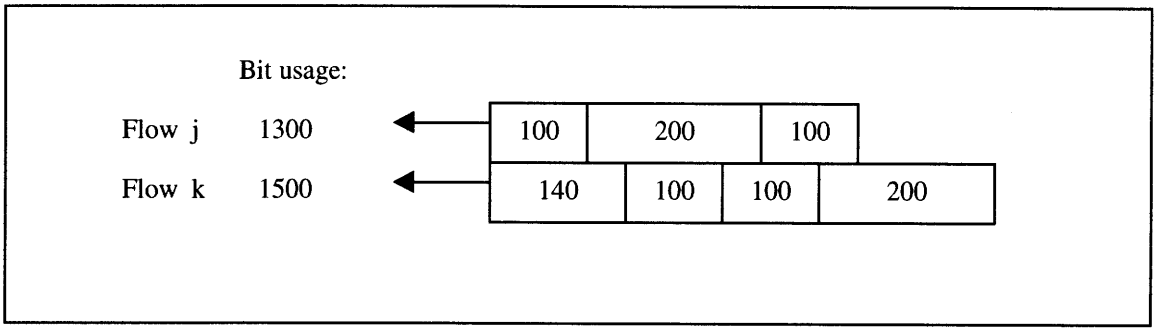

Figure 3.1: Example of scheduler behavior.

Assume that the flows have the same minimum bit-rate guarantee. Flow **j** with less bit usage, will be serviced first by granting it 100 bits then 200 bits as it has a higher priority. If at this time the scheduler tries to reprioritize the flows, it will delete from each flow's bit usage the value for the allowed bit usage during the **T** period. If it finds that flow **j** has used up more bits, it will assign a higher priority to flow **k.** Flow **k** will get its new data requests processed before **j**. The time period T is kept small enough to allow a proper working of the algorithm, but also large enough to keep the scheduler from having to perform a lot of processing.

The scheduler also resets the bit usage for all flows to zero for every large number N of MAPs, say 1000 MAPs. This guarantees that numbers for bit usage will not grow indefinitely and the algorithm will work over a long period of time. With no proper resetting, a flow that idles for a certain period T could use up all the network resources. Figure 3.2 shows the pseudocode for that minimum bit-rate guarantee algorithm.

```
// At setup
Choose large N and moderately small number of MAPs
   before priority update u
Assign to or negotiate with CMs guaranteed minimum
   bit rate for all SIDs
Assign random prior priorities from 1 to n for
   SIDs 1 to n
Initialize Bit usage [k] to 0 for all k
// When sending next MAP
While (bandwidth available in next MAP)
Begin
     For all flows k
     Begin
          Reserve bandwidth for flow k:
          Update Bit usage [k] = Bit usage [k] +
            granted bits [k]
     End
End
If (u MAPs have been transmitted)
Begin
     Bit usage[k] = Bit usage[k] - minimum bit
        rate[k]
     Sort priority of SIDs according to new bit
      usage
     // highest priority SID is one with lowest
        Bit usage[k]
End
     If (N MAPs have been serviced)
          Reset all Bit usage[k] to 0
```

Figure 3.2: Pseudocode for minimum bit-rate guarantee algorithm.

In our particular simulation example, we chose **N = 1000** and **u = 20**.

3.2 The MCNS Implementation in Opnet Simulation

The simulation was built using Opnet Modeler simulation software developed by MIL 3, Inc. and used extensively by the networking community for researching protocols and networks. A model of the MCNS MAC protocol was implemented using Opnet to test its functionality and the theoretical results. Mil 3, together with CableLabs (an MCNS

consortium member), developed the implementation of the MCNS MAC protocol. This simulation includes at the network level, a head-end (that contains the CMTS), 20 CMs, a network cloud from which downstream data originates and towards which upstream data goes. In addition, a data collection node collects data about the state of the network to produce the measurements needed. Figure 3.3 shows the actual Opnet setup at the network level. In this particular setup, CMs are at a random distance from the head-end to study the effect that transmission delays have on the overall efficiency and effectiveness of the protocol.

For this example simulation, table 3.1 shows the different values of interest to the scheduler that we have used. We limit our work to unicast traffic as the model we were using was still in the early stages of development and did not have support for multicast traffic. It is noteworthy to see that bandwidth available for downstream traffic is about 30 megabits/sec, while the upstream bandwidth is much smaller, reaching about 2.56 megabits/sec for the purposes of the simulation.

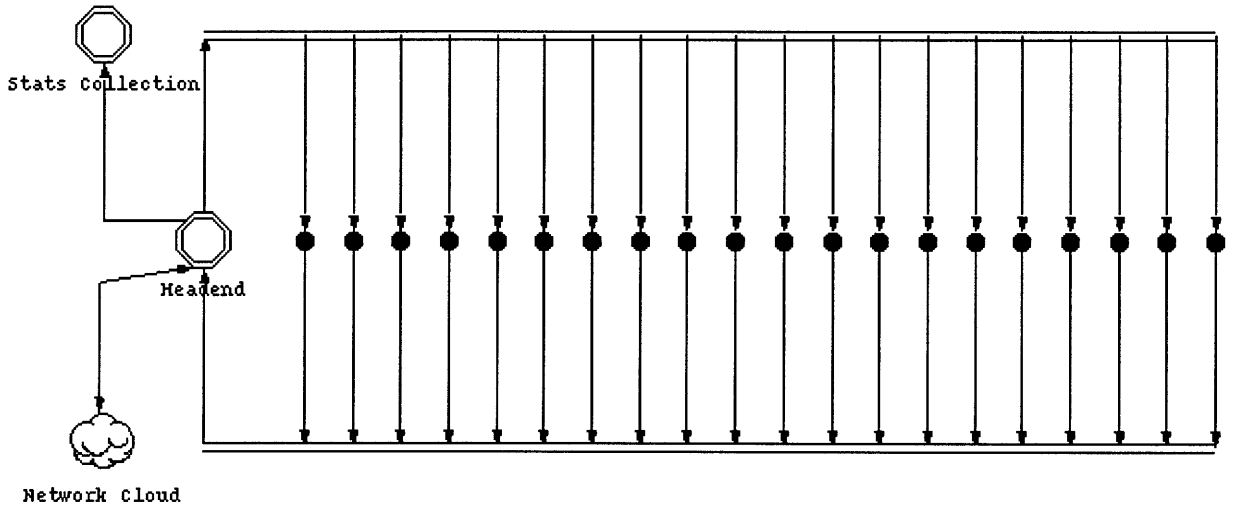

Figure 3.3: Implementation of cable network in Opnet Modeler [10].

The simulation runs for about 10 simulated seconds. Our scheduler is implemented at the head-end, and more specifically at the CMTS_Manager. The MAP timing is the sum of three time values:

1. The limit of mapping by the MAP (.00299 sec)
2. The processing time at the CM (.00001 sec)
3. The downstream interleave delay (.0005 sec)

The value .0035 sec is used as a gauge for taking 20 MAPs into consideration when we want to update the priorities of the CMs. The scheduling algorithm therefore recalculates these priorities every .0035 * 20 = 70 ms leading to no noticeable deceleration. The algorithm recomputes the priorities by applying the algorithm above. The table in the head-end was built as an array of structures. The algorithm described in section 3.1 was implemented at the CMTS level, with no changes to the CM, as required in the MCNS specification [3]. In order to keep the design of the cable modems simple, there is no change in the implementation of the protocol at the CMs.

Table 3.1. Variables of interest in Opnet simulation.

Downstream bandwidth	30 megabits/sec
Upstream bandwidth	2.56 megabits/sec
Mini-slot size	16 bytes
Time between update of priorities	70 ms
Simulation time	10 sec
MAP timing	.0035 sec

Figure 3.4 shows the state transition diagram for the manager. The system starts by performing a ranging of all CMs to determine their distance and exchange information about the physical network. Sync and Upper Channel Description (UCD) times are interrupts that serve to keep the network in sync and to give information to the CMs about the state of the network in order to fine-tune the frequency and delay in transmission.

Upstream and downstream data traffic allows a good estimate of the delay that requests incur and allow for a good statistics collection in the simulation. The maintenance MAP is obviously for maintenance purposes. The **SEND_NEXT_MAP** interrupt is most important for our simulation, as this is the domain that our improved scheduling needs to have control over. The CM sends requests to the CMTS, and when mini-slots are allocated to it, it uses them to send data upstream.

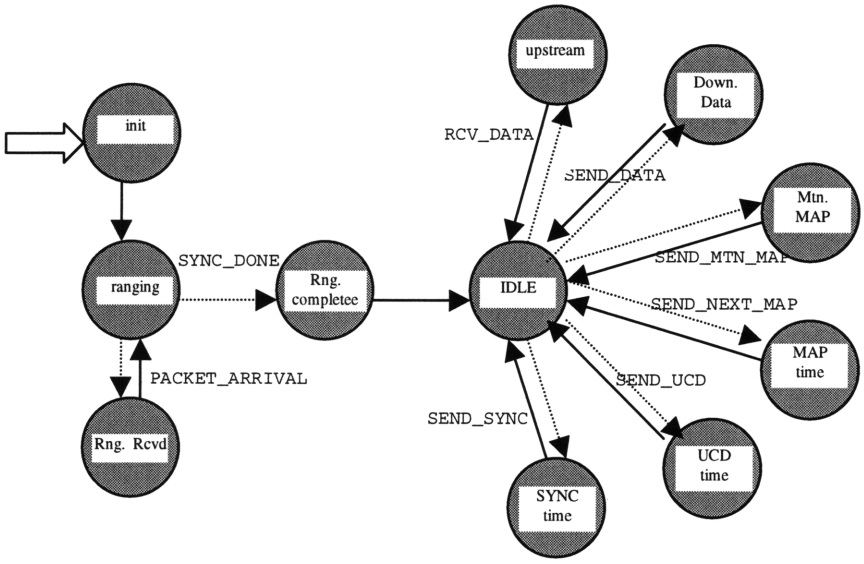

Figure 3.4: CMTS_Manager state transition diagram [10].

The added computational work on the CMTS consists of both maintaining the data structures for the CMs and sorting the flows according to the priority scheme discussed

above. The simulation sets up a random minimum bit-rate guarantee to each CM at initialization, but as was mentioned earlier, that bandwidth could be either pre-assigned or reserved dynamically.

3.3 Simulation Results

The simulation was run over 10 virtual seconds. Table 3.2 shows results of the simulation. For all 20 connected and active CMs, random values for a minimum guaranteed bit-rate are chosen. Column 1 enumerates each CM from 1 to 20. The data collected in each row corresponds to that particular CM. Column 2 tabulates the minimum bit-rate guarantee for each flow. The numbers chosen are random values that the scheduler has to guarantee. We collect data over a period of 1 second. In column 3, we show the number of bits used by each CM over that time period, thus the achieved bit-rate of each flow when we are not using the algorithm that we have developed. We instead use FCFS queuing in this case. Some of the values in column 2 are similar because the random number generator in Opnet frequently generates the same values. FCFS gives largely varying results in terms of bit-rates to each flow. Using a different random number generator, we can get a different set of numbers for the bit-rates with a large variance. The rate of flow 1 (104,228 bps) is lower than the minimum bit-rate guaranteed (104,781 bps). Other rates are barely above the desired bit-rate (for example, flows 5, 12 and 20) while some other flows receive a lot of extra bandwidth (for example, flows 8, 9, 10 and 14). Column 4 shows the achieved bit-rate when we use the algorithm proposed. We can see that all values in column 4 are consistently greater than the values tabulated in column 2. This means that the scheduler does achieve the required minimum bit-rate for each flow.

This minimum bandwidth guarantee is the first desired objective of the scheduling

algorithm. Column 5 depicts the extra bandwidth as the difference between achieved bit rate and minimum bit-rate guarantee (column 3 – column 2) when we do not use the algorithm that we developed. Column 6 is the difference between column 4 and column 2, where we have used our algorithm. The average of that extra bandwidth is around 19.8 K-bits/sec in the case of columns 5 and 6. Standard deviation is about 2.2 K-bits/sec when using our algorithm and 14.4 K-bits/sec for FCFS. We do achieve reasonable fairness in the distribution of the extra bandwidth when using our algorithm. The equation provided in the pseudocode works both for guaranteeing minimum bit-rate and distributing the extra bandwidth fairly.

Table 3.2: Results of the simulation [11].

1	2	3	4	5	6
CM	Minimum bit rate	FCFS	Our algorithm	Extra BW	Extra BW
				Column 3 – column 2	Column 4 – column 2
1	104781	104229	123723	-552	18942
2	103960	110498	124247	6538	20286
3	72285	109714	93381	37429	21096
4	104408	110498	123462	6090	19054
5	105937	109714	121631	3777	15694
6	67145	109714	89457	42569	22312
7	91174	109714	111691	18540	20517
8	78967	109714	98874	30747	19907
9	81804	109714	103582	27910	21778
10	74651	110498	94950	35847	20299
11	92398	109714	112737	17316	20339
12	107642	109714	121892	2072	14250
13	91868	110498	112214	18630	20346
14	80708	110498	102013	29790	21305
15	85691	110498	107244	24807	21553
16	95859	110498	115876	14639	20017
17	68377	109714	89981	41337	21604
18	77039	109714	98351	32675	21312
19	104636	109714	123200	5078	18564
20	107290	109714	122939	2424	15649
			average:	19883	19741
			stdev:	14382.3	2201.4

In the results table presented, the high value for the standard deviation is due to the fact that several flows were not backlogged. Rather they sent requests within the allocated minimum bit rate guarantee.

Greedy flows can benefit the most from this scheduling scheme. For this reason, in our calculations, only flows that are backlogged should be counted towards the standard deviation, to prove the correctness of our algorithm in providing fairness among all the flows. For that reason, if we do not count flows 5, 12 and 20 that were requesting very low bandwidth (as seen in column 5 of table 3.2) in calculating the average and standard deviation, the results will be more representative of our claim. Average will go up to 20.5 k-bit/sec but the standard deviation will go down to 1.0 k-bit/sec, showing that the algorithm is fair for backlogged traffic.

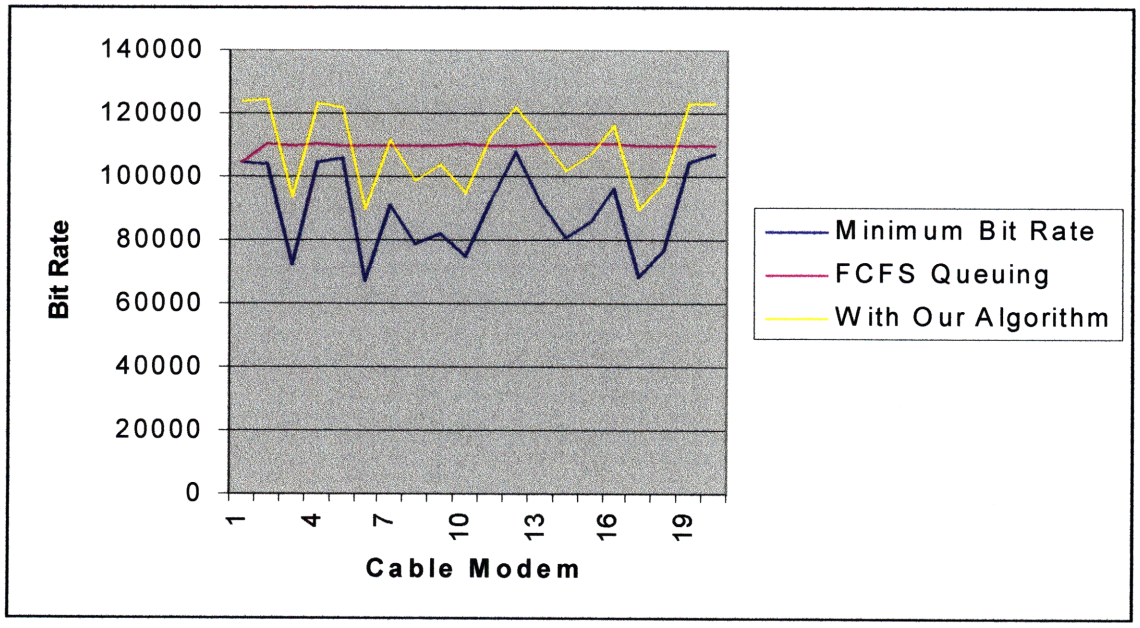

Figure 3.5: Line chart for bandwidth usage by the different CMs [11].

The results of the calculations also show in the line chart of figure 3.5. We show the guaranteed bit-rate (in column 2 of table 3.2) and the achieved bit-rate both with and

without our algorithm (columns 3 and 4 respectively of table 3.2). The achieved bit-rate with the use of our algorithm follows the minimum bit-rate guaranteed and achieves it for all CMs. CM 1 does not achieve the minimum bit-rate guarantee and extra bandwidth is not allocated fairly between the different cable modems.

Thus we can see that for different bandwidth requirements, our algorithm performs well and allows us to provide minimum throughput guarantee and fairness in the distribution of extra bandwidth.

3.4 Recapitulation

We have presented our QoS algorithm and shown how our Opnet simulation is implemented to study its effectiveness and applicability to the cable network protocol. We have discussed the results that prove that our claim is correct. We now discuss issues with the provision of more QoS guarantees to satisfy flows with stringent delay bounds, such as Internet Telephony flows.

Chapter 4

Enhancements for Internet Telephony Traffic

In general, we can identify different types of flows for different applications that each require a certain QoS in terms of available bandwidth, fairness or delay bounds. An important type of flow is IP Telephony that has moderate bandwidth requirement (64 K-Bit/sec uncompressed) but a stringent delay requirement. Another example is a telnet session, where only a few characters are sent over the flow at times, but where the user needs immediate feedback. These flows contrast with flows that use as much bandwidth as possible but do not require stringent delay bounds (for example, an FTP session). Because of the type of data sent in data services and Internet telephony flows, any significant delay would be perceptible to the user. An upper limit for the delay in the case of real-time voice is considered to be 50 ms [11].

4.1 Delivering Delay Bounds

We distinguish between two qualities of flows: best-effort flows that get a minimum bit-rate guarantee and a fair allocation of the extra bandwidth, and latency critical flows that only need constant bit-rate allocated to them while having a delay bound guarantee. Let us identify these flows as $\mathbf{f_j}$ for latency critical flows and $\mathbf{F_j}$ for best-effort flows. The way the CMTS will be servicing flows will be dependent on their type. At connection time, each SID **j** will be identified by either $\mathbf{f_j}$ or $\mathbf{F_j}$, depending on the type of service the CM agrees upon with the CMTS. The CMTS will keep track of the bandwidth requirement for all $\mathbf{f_j}$s. It will distribute the bandwidth allocation in the next MAP as follows (refer to figure 4.1):

1. Contention period, that all CMs to contend for;

2. Best-effort flows $\mathbf{F_j}$, that need only be guaranteed a minimum bit-rate;
3. IP telephony flows $\mathbf{f_j}$ that are assigned a percentage of the total available bandwidth;

Figure 4.1: Distribution of bandwidth in MAPs [11].

In order to support flows that have stringent delay requirements, we propose a scheduling scheme illustrated by the following pseudocode:

```
// At CMTS side
Set maximum bandwidth for latency-critical flow f_j
   to BWc_max;
BWc_max = α * tot.  BW; // 0 < α < 1
Total bandwidth allocated to latency-critical
   flows is initialized: BWc = 0;
// To have a new flow setup
When CMTS receives request for new flow setup f_j
   from CM_j
Begin
   If (BWc + BW(f_j) > BWc_max)
      Refuse request and return
   Else
      Update BWc += BW (f_j)
   Reserve number of mini-slots in MAP needed to
      satisfy BW (f_j)
End

// Operation of CMTS-CM pair for f_j
CM_j sends data in assigned mini-slots and
   piggybacks request for next MAP at end of data
   if it needs continued bandwidth
CMTS reserves bandwidth in subsequent MAPs, until
   no Request
// Moving a flow from f to F
If requested bits (CM_j) > BW (f_j)* T (mapped by a
   MAP)
Begin
   Refuse request;
   Flag flow f_j as being best-effort traffic flow
End
```

Figure 4.2: Pseudocode for Telephony traffic.

When a request for a new delay-stringent flow comes to the CMTS from CM_j, the CMTS makes sure it can accommodate such a bandwidth request. This test makes sure that the network is not overloaded, and will be able to accommodate all bit rate guarantees to subscribed flows. In case the CMTS cannot allocate such bandwidth, it will refuse the request to join the session and the CM will have to try again with either the

same or lower QoS requirements. Upon success of this join session operation, the CMTS will ACK the join request. Figure 4.3 shows the flowchart for the setup process. When a delay-sensitive flow **f** is not granted the request to start transmitting data, it can choose whether it can use best-effort service instead.

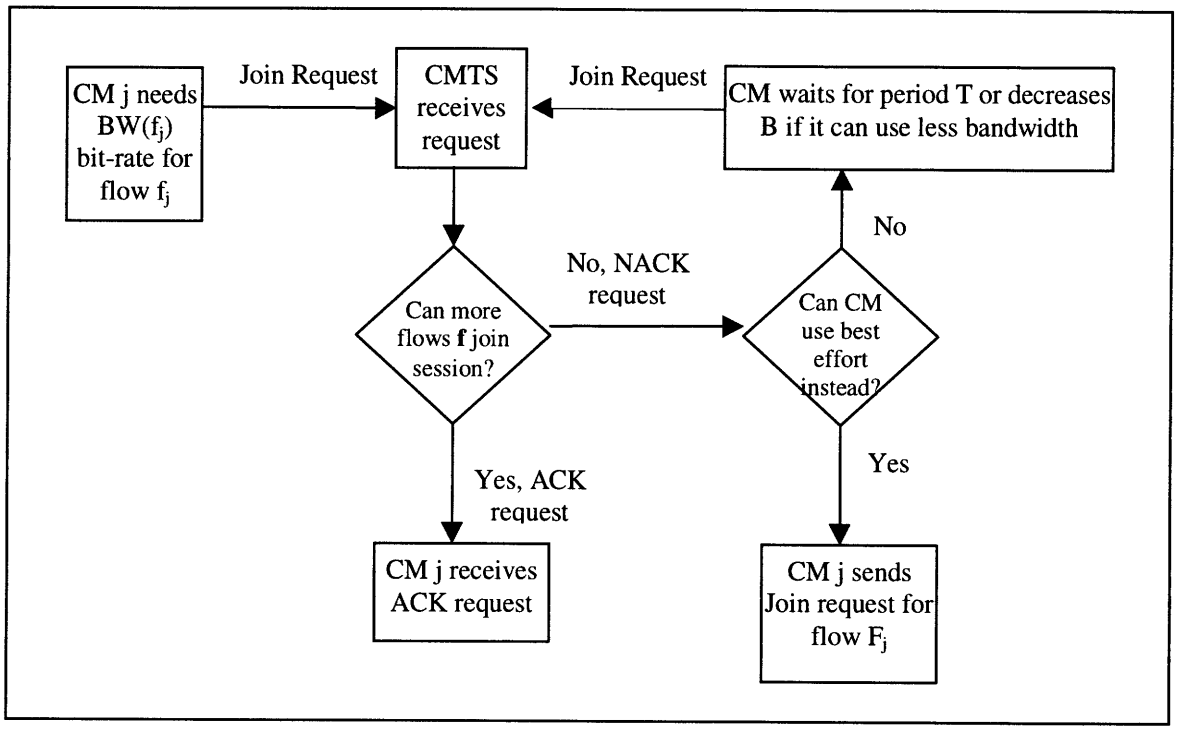

Figure 4.3: Setup process for flows with stringent delay bounds.

If the CM can use a best-effort service, it will send a join request for flow F; otherwise, it will either wait for a time period T to try again or decrease the bandwidth requirement that it had previously, and send the join request again to the CMTS. After this handshaking process, all CMs that do get a grant for a delay-sensitive flow would start sending data in their allocated mini-slots. The CMTS keeps track of bandwidth requests for all latency critical flows. When it receives a data request from a certain CM, it checks whether the bandwidth the CM is asking for is beyond the guaranteed bit rate to that particular CM. It could then either deny the request or send it to the best-effort part of the scheduler that would try to accommodate it on a priority basis, as discussed in

chapter 3. The CM, in order to ensure an uninterrupted allocation of bandwidth, should piggyback requests to the data that it sends in the mini-slots allocated by the CMTS.

Idling CMs will not be given a reservation during the next MAP, and thus there will be no lost bandwidth in the proposed scheme from over-allocation. There are two ways to stop servicing a flow:

1. The CMTS can either stop reserving mini-slots as soon as it stops receiving piggybacked requests from that flow or wait for the CM to send data requests again during the contention period.
2. Alternatively, it can conduct polling dynamically to allow an idling flow to send data requests again, thus reducing delay in the request/grant process, as discussed in section 4.2.

4.2 Dynamic Polling

When a flow does not request bandwidth during a certain mini-slot, it does not have the opportunity to ask again for bandwidth unless it contends for it during the contention period. A lightly loaded network would not have difficulty in accommodating data transmission during contention, but a heavily used network will exhibit a lot of collisions and retries during the contention period. In the case of light loads, the probability that more than one CM tries to send a request in the same mini-slot is small, as many contention slots are not used, leading to a higher probability of success in the transmission of requests from CM to CMTS. The expected number of mini-slots that will have collisions will be high in the case of a high load on the network, and only a few CMs are able to get their requests successfully transmitted and processed. There would be a higher probability that requests from latency critical flows do not go through, leading to a high delay at first in fulfilling the bandwidth requirement when a CM is no

longer idling. The contention period approach is more bandwidth-efficient than static polling. Using static polling, a CMTS would have to send **n** request opportunities for **n** idling CMs, even when the **n** CM connections are obviously inactive. **n** request opportunities in each MAP lead to inefficient usage of upstream bandwidth. The contention period, though not very reliable, allows for a better bandwidth usage. The contention resolution policy used in cable networks is to deny certain CMs the use of the contention period when it is heavily utilized, based on the following windowing scheme:

- When the CMTS first starts transmitting and receiving data, all CMs are allowed to use the contention period.
- When more collisions occur, the window of opportunity becomes smaller, and the number of CMs allowed to transmit in that period decreases exponentially. In a worst-case scenario, only one CM would be allowed to transmit data requests using the contention period.
- The more transmission successes in the contention period, the more the window grows to accommodate a higher number of CMs.

We can think of two levels of QoS. On the one hand, we can consider flows that we want to service with a reasonable delay, while keeping a certain amount of flexibility when these flows become idle then want to send data again. On the other hand, there are flows that would absolutely need low delay at any time even if they were idling. To address these two issues, whenever a CM that is latency-critical becomes idle, the CMTS will start polling it in subsequent MAPs. Polling gives opportunity for the CM to send a request for future bandwidth in an assigned mini-slot that the CMTS provides to idling CMs. For flows that have a fixed delay need, the CMTS will conduct polling of these flows until they request data again. This wastes some bandwidth, but is required to fulfill

the agreement between the client host and the server that mandates uninterrupted service. We do however get improved operation over static polling that wastes bandwidth at all times, and the operation of the contention period that is unreliable in terms of QoS. For other flows that try to get a low delay, the CMTS keeps them in another list. The CMTS monitors the contention period to see if it is overused. We determine that by the number of collisions that occur.

If for **m** mini-slots, we get **c** collisions, **s** successes and **m-c-s** free mini-slots, then over-used contention period happens as follows:

$$c/m \rightarrow 1$$

$$s/m \rightarrow 0$$

$$(m-c-s)/m \rightarrow 0$$

Here, the collision rate is high, the success rate is low and the number of empty (unused) slots is small. The CMTS then polls all idling CMs because there is low probability that an idling flow could send a request in a contention slot successfully. Those flows will be able to send their requests in reserved space. When the contention period becomes underutilized, we get the following behavior:

$$c/m \rightarrow 0$$

$$s/m \rightarrow 0$$

$$(m-c-s)/m \rightarrow 1$$

The CMTS will be monitoring a contention region being very highly utilized or underutilized. It will take decisions appropriately: when the contention region starts getting less traffic, the CMTS will start taking off those CMs that have been idling the longest. At the limit, for an unused contention region, the CMTS does not provide any polling, while for a heavily used contention region, the CMTS provides polling to all

latency-critical flows. This allows the system to be flexible and efficient. Latency critical flows would be able to send data with low delay. The polling scheme can apply to both **f** and **F** flows, to relieve the contention period. Priority goes to delay-sensitive flows. Let us consider an example that would present the proposed dynamic polling.

Suppose we have **j** idling flows **f** of that are delay-critical, and **k** idling flows **F** that are best effort flows. Of the **j** delay-critical flows, there is agreement between **l** flows $\mathbf{f_{hp}}$ and the CMTS to poll them at all times when they are idling. The other **f-l** flows can be considered low-priority $\mathbf{f_{lp}}$. Only flows $\mathbf{f_{hp}}$ will be polled when the network is lightly utilized. With a higher utilization, the CMTS will start polling some flows of type $\mathbf{f_{lp}}$. In this case a window of opportunity for polling can be used that grows exponentially to accommodate more flows. The more utilized the contention period is, the higher number of CMs that get polled. When all flows $\mathbf{f_{lp}}$ are being polled and the contention period is still largely utilized, flows of type **F** will also be polled. Conversely, with a lower utilization of the contention period, the number of polled flows drops exponentially too. The priority for dropping the flows off the polling list consists of dropping flows **F** first, then flows $\mathbf{f_{lp}}$ until no flows other than $\mathbf{f_{hp}}$ flows are being polled.

The dynamic polling scheme is both very efficient and very easy to implement. It allows the scheduler to provide flows that need that kind of service with stringent delay bounds, while allowing the scheduler to give low delay bounds to more flows as well to alleviate congestion. With the use of that type of polling, the scheduler makes sure that most of the flows can get their requests delivered. In both cases, the bandwidth needed for that service is minimal. In contrast, consider the case where many flows would be trying to send data requests; those requests would collide with one another, the CMTS will not fulfill requests and MAPs will basically not be able to map the next T period.

This would lead to high delays and low network efficiency. Figure 4.4 shows the enhanced operation resulting from the use of dynamic polling in the scheduler.

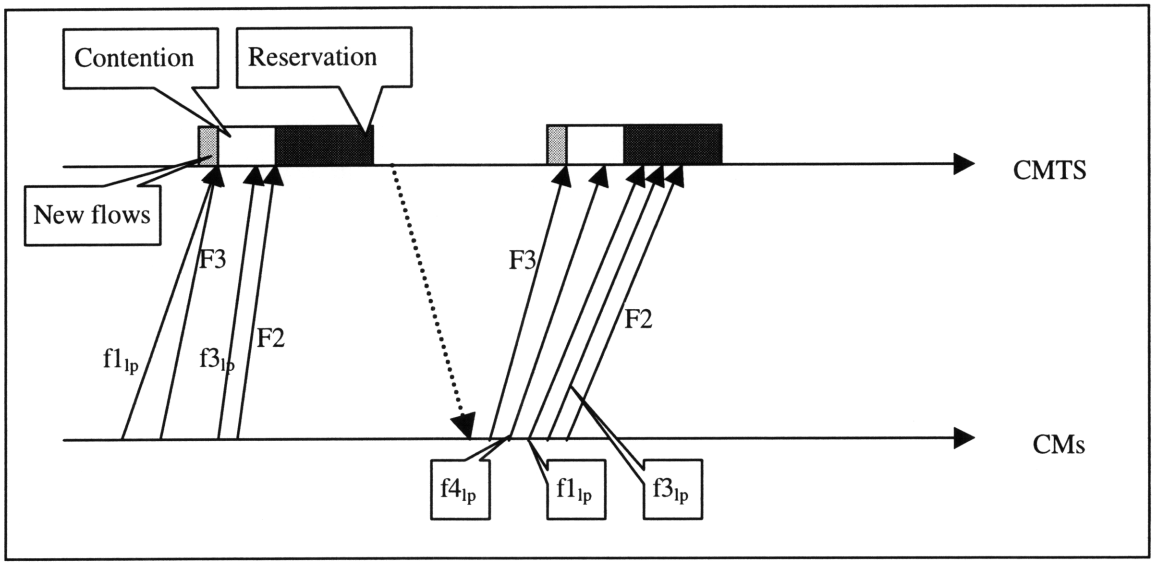

Figure 4.4: Dynamic polling of flows with different delay priorities.

Before the first depicted transmission, all flows with high priority are being polled, but others are not. Flows **F2**, **F3**, $\mathbf{f3_{lp}}$ and $\mathbf{f1_{lp}}$ are idle and decide to ask for bandwidth. When flows **F2**, **F3**, $\mathbf{f3_{lp}}$ and $\mathbf{f1_{lp}}$ send data requests in the contention period of the MAP, the contention period becomes busy and both flows $\mathbf{f1_{lp}}$ and **F3** use the same mini-slot. Both **F2** and $\mathbf{f3_{lp}}$ do get their requests granted. The CMTS conducts polling of a certain number of idling flows. Flow $\mathbf{f1_{lp}}$ will get a slot by polling. It would then be able to use it. This is depicted in the next data transmission from CMs to the CMTS. In the case of flow **F3**, it is considered to be lowest priority with respect to polling, and would only get polled in the case of heavy usage of the contention period. Other idle flows that did not send data requests could send them in the contention period, or if the polling process picks them for unsolicited grants, could send data in the slots they have been granted.

4.3 Summary

This chapter has explained our approach towards providing small delay bounds to

delay-critical applications. We identified several types of traffic such as best-effort flows that can be serviced with minimum bit-rate guarantee, telephony flows that have constant bit-rate needs, but also low delay requirements, and other data traffic with very stringent delay needs. Each of the flows is serviced according to its need. The flexibility of the scheduling system allows it to service all of these types of flows with the proper QoS requirements. We make use of an intelligent polling design that allows the network to incur minimal loss of bandwidth while the scheduler provides the low delay that certain flows need.

Chapter 5

Conclusion

5.1 Summary of Work

This thesis has discussed different issues related to scheduling and delivering QoS to cable networks. We first described the MCNS protocol, backed by the cable industry as the potential standard for cable networks. This protocol uses the inherent physical layer to provide a certain level of services at the MAC layer. The assumptions of a good underlying physical layer allow the MAC interface to be complex. The MAC layer does provide the hooks for giving higher layers service guarantees because of this underlying assumption of a good quality hardware layer, with high bandwidth availability. In fact, the bandwidth availability is dependent on the allocation of channels for data transmission. As a first step, only one channel is allocated, but the allocation of more bandwidth is both feasible and normal, to allow more bandwidth with the increase in need for speed. On the other hand, hosts that need increased QoS can be serviced accordingly as the network can accommodate a breadth of services implemented by software, either at layer 2 (the layer that includes the MAC) or layer 3 (where IP protocol and reservation protocols operate).

In our presentation of the MAC layer, we discussed the asymmetry between upstream and downstream bandwidth and their subsequent allocation to the CMs on the network. Due to the availability of a small bandwidth on the upstream compared to the downstream more effort has to be put in delivering QoS guarantees. Our work was part of an effort to provide bandwidth and fairness guarantees for CMs.

First we covered the MCNS protocol in detail, then talked about issues in scheduling

bandwidth. In that context, we reviewed different scheduling technologies found in the literature. Those included First-Come First-Served, Round Robin, Deficit Round Robin, Virtual Clock and Leap Forward Virtual Clock scheduling. All these algorithms address different levels of QoS and give extensive insight on how to address our problem of delivering QoS over cable networks.

We then presented an algorithm that we developed to achieve minimum bit-rate guarantee and to distribute extra bandwidth fairly between CMs on the session. This algorithm keeps track of the minimum bit rate guarantee of each CM, the current bit rate usage of the particular CM, and the remaining bandwidth still needed to serve the CM. The scheduler will then give a higher priority to CMs that are further from fulfilling their service guarantee. Higher priority flows are serviced before lower priority flows, in order to achieve the minimum bit rate guaranteed to those flows. On the other hand, the remaining bandwidth is distributed equally among CMs using the same priority scheme. We implemented the scheduling algorithm using the OPNET network simulation software and a model of the MCNS protocol built in OPNET. Results of the simulation of the algorithm show that it does achieve its claim in terms of minimum bit rate guarantee and fairness in distributing the extra bandwidth.

We then described extensions to the scheduling algorithm in the domain of delay bounds. In order to deliver telephony traffic and data services that have stringent delay requirements with the appropriate latency, we described a polling mechanism that would identify two types of flows, latency-critical flows and best-effort flows. Latency critical flows have CBR need but stringent delay requirements. Best-effort flows try to use as much bandwidth as possible, by requesting a lot of bandwidth, to accommodate the transport layer, whether UDP or TCP. For example, an FTP session is best served by

getting as much bandwidth as possible. Above the minimum bit-rate guarantee, flows of this type can get the fair share of extra bandwidth, but can also send data in the contention period, with the added precaution that this data might collide with other data from CMs using the contention period themselves. Polling latency critical flows periodically allows the CMs with those flows to send data requests if they have any. We do not consider the use of that bandwidth as a waste, but as an agreement to dedicate that bandwidth to a certain flow that needs to be honored by the CMTS. Bandwidth needs to be continually provided to flows requiring such a service. Polling of other latency critical flows can be conducted dynamically. The CMTS polls flows based on the usage of the contention period. When the contention period is heavily used, the CMTS will poll more flows than when the contention period is clearly unused. The CMTS will dynamically poll more flows as the contention period is more heavily used, based on a windowing scheme that adds more CMs to the polling list. This scheme achieves a small delay bound for the flows that need such a QoS.

5.2 Improvements and Future Work

Future work in the area of QoS over cable networks that we are planning to do includes the simulation of delay bounds for different types of flows. We need to provide more numerical results in that domain. In that respect, we intend to build a simulation of the enhanced polling protocol that registers flows for polling either dynamically in the case of non-delay critical applications or statically for flows that need stringent delay bounds. Our approach to this problem serves both conflicting constraints of having good delay bounds on the one hand, and saving as much bandwidth as possible for data traffic on the other hand.

Another area of interest and that we would like to pursue is the provision of VBR-like

(Variable Bit-Rate) service for traffic such as real-time video traffic such as streaming video and video-conferencing. The dynamic change in bandwidth requirement makes the problem interesting and challenging. We would like to research ways of dealing with jitter among other things. Another issue that we would like to tackle in that domain is to simulate the behavior of the scheduling algorithm in the presence of flows with different QoS requirements transmitting data concurrently. We would like to be able to formulate the problem as a multicommodity flow problem.

The delivery of a breadth of services for Internet traffic makes the technology engaging. For that reason, it is very important to make the QoS scheduling algorithm work within the framework of a larger approach that involves different layers of the network stack. To allow such a mechanism, we publish the capabilities of the MAC layer to higher layers where RSVP negotiates the service needed by the application layer. It then makes the appropriate resource reservation with the MAC layer for the needed service.

One last topic that we discuss is a system that provides extra bandwidth for all the flows when the network is being heavily used. For that matter, the allocation of extra channels for data traffic should be researched, and a good strategy needs to be built that will allow on-the-fly channel allocation and de-allocation at the physical layer of cable networks.

References

[1] Braden, R., L. Zhang, S. Berson, S. Herzog and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997, available at http://ds.internic.net/rfc/rfc2205.txt.

[2] Postel, J., "Internet Control Message Protocol - DARPA Internet Program Protocol Specification", RFC 792, USC/Information Sciences Institute, September 1981, available at http://ds.internic.net/rfc/rfc792.txt.

[3] MCNS Holdings, Data-Over-Cable Interface Specifications, Radio Frequency Interface Specification, SP-RFII01-970317, Interim Specification, March 97.

[4] Postel, J., "Internet Protocol - DARPA Internet Program Protocol Specification", STD 5, RFC 791, Defense Advanced Research Projects Agency, September 1981, available at http://ds.internic.net/rfc/rfc791.txt.

[5] Braden, R., D. Clark and S. Shenker, "Integrated Services in the Internet Architecture: an Overview", RFC 1633, ISI/MIT/Xerox PARC, July 1994, available at http://ds.internic.net/rfc/rfc1633.txt.

[6] Bertsekas, D. and R. Gallager, Data Networks, Prentice-Hall, Inc., New Jersey: 1987.

[7] Sreedhar, M. and G. Varghese, Efficient Fair Queuing Using Deficit Round-Robin, IEEE/ACM Transactions on Networking, 4(3), June 1996.

[8] Lixia Zhang, Virtual Clock: A new traffic control algorithm for packet-switched networks, ACM Transactions on Computer Systems, 9(2), May 1991.

[9] Suri, S., G. Varghese and G. Chandranmenon, Leap Forward Virtual Clock: A New Fair Queuing Scheme with Guaranteed Delay and Throughput Fairness, IEEE Infocom '97 - 16th Conference on Computer Communications, 1997.

[10] Narayanaswamy, S. MCNS Design Specifications, Mil 3, Inc., January 1997.

[11] Woundy, R., R. Rabbat and S. Siu, QoS Presentation, DOCSIS Monthly Meeting, October 1997.