ADMISSION CONTROL AND BANDWIDTH ALLOCATION FOR CLASSA TRAFFIC IN RPR NETWORKS

By

Zhenning Xu

Bachelor of Computer Engineering, Shenyang, China, 1997

A thesis

submitted to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Applied Science in Computer Networks

Computer Networks Program

Department of Electrical and Computer Engineering

Toronto, Ontario, Canada, 2004

Author's Declaration:

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Signature:

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Signature:

Ryerson University requires the signatures of all the persons using or photocopying this thesis. Please sign below, and give address and date.

Admission Control and Bandwidth Allocation for ClassA traffic in RPR Networks, Master of Applied Science, 2004, Zhenning Xu, Graduate Program in Computer Networks, Ryerson University

Abstract

In this thesis, we study the admission control and bandwidth allocation methods for classA traffic in RPR networks. First, we investigate the performance of classA traffic under the current RPR protocol. The simulation results show that RPR networks can support low-delay classA traffic even if the networks are congested with classB and classC traffic. The low-delay performance, however, is subject to the condition that the load of classA traffic must be properly controlled. Consequently, an admission control mechanism must be used for classA traffic. In this thesis, several admission control algorithms are studied. They are the Simple Sum algorithm, the Measured Sum algorithm, and the Equivalent Bandwidth algorithm. The simulation results show that the Equivalent Bandwidth algorithm is the most suitable to use as the admission control mechanism for classA traffic.

The admission control mechanism makes admission decision based on the available bandwidth allocated to the classA traffic. The existing RPR standard assumes the bandwidth allocated for classA traffic at each node is fixed. The fixed bandwidth allocation introduces inflexibility and inefficient use of bandwidth for classA traffic. In this thesis, three bandwidth allocation algorithms are proposed to dynamically allocate bandwidth for classA traffic. These algorithms have different levels of complexity and can be applied to different traffic environments. Simulation results show that the proposed algorithms improve the bandwidth efficiency of the RPR networks. The proposed algorithms are also readily integrated with the existing Internet Quality of Services (QoS) paradigms such as Diffserv and RSVP services.

Acknowledgments

I would like to thank my supervisor, Prof. N. W. Ma, for his intellectual support, constructive comments, and the time he spent with me. His great lectures on computer networks have immensely helped me on both this thesis and my career life.

Special thanks to Steven Zhang and Gary Yip for their always available professional advices and willingness to help.

Also, many thanks to Arseny Taranenko for his technical help on OPNET.

And finally, I am again grateful to my girlfriend Anne Luo, for all of her support and encouragement.

Table of Contents

Chapter 3 Dynamic Bandwidth Allocation Using Common Bandwidth Pool

List of Figures

List of Tables

List of Flowcharts

List of Abbreviations

RPR: Resilient Packet Ring

MAC: Media Access Control

ClassA: High Priority Traffic

ClassB: Media Priority Traffic

ClassC: Low Priority Traffic

PTQ: Primary Transit Queue

STQ: Secondary Transit Queue

CIR: Committed Information Rate

EIR: Excess Information Rate

QoS: Quality of Service

IntServ: Integrated Services

DiffServ: Differentiated Services

RSVP: Resource Reservation Protocol

ISP: Internet Service Provider

OAM: Operational and Maintenance (procedure)

GOS: Grade of Service

FIFO: First In First Out

ACK: Acknowledgement (packet)

NAK: Negative Acknowledgement (packet)

B_{G_k}: Guaranteed bandwidth at node k

B_{AG_k}: Available guaranteed bandwidth at node k

B_{C_k}: Common bandwidth pool size at node k

B_{AC_k}: Available common bandwidth pool size at node k

B_{LC_k}: Common bandwidth reserved by local sources at node k

C_i : Equivalent bandwidth of flow i

ΔC_i: Requested bandwidth

C_{RC}: Common bandwidth to be released

BU(n): Bandwidth usage on hop n

RH(n): Requested common bandwidth for reservation/release on hop n

DC(n): Difference between bandwidth usage and guaranteed bandwidth on hop n

Chapter 1
Introduction

It is important to guarantee the high priority traffic (classA traffic) in Resilient Packet Ring (RPR) networks to achieve low end-to-end delay performance. This thesis studies several admission control algorithms to support low delay performance for high priority traffic. Our simulation study shows that with proper control of classA traffic load, the delay of classA traffic can be kept low, even though the network is congested with lower priority traffic (classB and classC traffic). The study also shows that most of the delay is introduced at the source node. Once the traffic enters the network, the delay is minimum and is almost equal to the propagation delay.

Admission Control for classA traffic in RPR networks must be supported by the bandwidth reservation procedure, which is in turn, affected by how the bandwidth is allocated to each node in the network. In the current RPR draft, no reservation algorithm or bandwidth allocation method is proposed. One approach to deal with bandwidth allocation is to allocate fixed bandwidth to each node. However, our study has shown that fixed bandwidth allocation is not efficient. In this thesis, we propose a dynamic bandwidth allocation method using the concept of common-bandwidth pool. Three reservation algorithms based on the dynamic bandwidth allocation method are then proposed and studied. Our results show that the proposed algorithms can provide better bandwidth efficiency than the fixed bandwidth allocation method.

In the rest of this chapter, the basic concepts in RPR networks, classA traffic and Quality of Service (QoS) will be briefly covered.

1.1 Introduction to RPR networks

Resilient Packet Rings (RPR) are data optimized networks, with at least two counter rotating fiber ringlets in which multiple nodes share the bandwidth without the requirement of provisioning circuits [1]. For the best-effort traffic, the nodes on the ring can automatically negotiate for bandwidth among themselves via the Fairness Algorithm. Each station has a topology map of the ring and can send data on the optimal ringlet towards its destination.

RPR layer 2 technologies define a media access control (MAC) protocol that decides the manner in which available bandwidth can be utilized by transmitting stations. The MAC protocol also decides how a station would react to congestion on the media. Finally, the MAC regulates access to the media by buffering and prioritizing packets onto the media. The RPR MAC supports three service classes:

1) High priority traffic (class A): The traffic class requires Committed Information Rate (CIR) services. The service supports guaranteed bandwidth and low latency/jitter applications. Voice, video, and circuit emulation applications can utilize this class of traffic. Within this class, the MAC uses two subclasses, subclassA0 for reserved bandwidth and subclassA1 for reclaimable bandwidth.
2) Medium priority traffic (class B): This traffic class requires CIR services that have less stringent (but still bounded) jitter/latency requirements. Any usages above the CIR are considered EIR (excess information rate) and are subject to the fairness algorithm. Bursty data applications can utilize this traffic class.
3) Low priority traffic (class C): This traffic class is used for best- effort traffic. Nodes negotiate to receive a fair share of the ring capacity using the RPR fairness algorithm. Most of the Internet traffic belongs to this traffic class.

1.2 Quality of Service (QoS)

Networks today are carrying more data than ever in the form of bandwidth-intensive, real-time voice, video, and data, which stretch network capability and resources. So QoS issue plays more and more important role to provide better service to the selected network traffic.

The primary goals of QoS are to provide better and more predictable network service by providing dedicated bandwidth, controlled jitter and latency, and improved loss characteristics. To achieve these goals, several QoS tools have been developed. These tools include traffic policing, traffic shaping [4] and admission control [6]. Below, we will briefly describe these QoS tools.

1.2.1 Traffic policing

Traffic policing allows a network to control the rate of traffic transmitted or received on a network interface. A traffic policer monitors the traffic flow and tries to detect non-

conforming packets. A packet is deemed non-conforming if the transmission of that packet means that the traffic flow violates the agreed policing parameters such as peak data rate, mean data rate and maximum burst length. A non-conforming packet is either discarded or marked. Marked packets are usually treated with lower QoS.

1.2.2 Traffic Shaping

Traffic shaping is similar to traffic policing, except that traffic shaping uses a buffer to store the nonconforming packets, thus the nonconforming packets are delayed, instead of being discarded or marked. If the buffer is full, however, nonconforming packets will also be discarded. Traffic shaping is used extensively to control the traffic flow at the MAC layer of the RPR network.

1.2.3 Admission Control

Admission control is the procedure for the network to determine if it should admit a new flow. A flow is usually characterized by a given source/destination pair. By computing or measuring existing traffic load, the network decides if there are sufficient resources to meet the QoS required by the new connection. The role of any admission control algorithm is to ensure that admittance of a new flow into a resource constrained network does not violate service commitments made by the network to the existing flows.

1.3 QoS Support in the Internet

Two models are used in the Internet for QoS support. They are the Integated Services [2] and Differentiated Services models [3]. These models are briefly described below.

1.3.1 Integrated Services (IntServ): In the IntServ model, routers treat each flow separately according to its required QoS. Each flow makes its own reservation using the Resource Reservation protocol (RSVP). The admission control mechanism in each router interworks with RSVP to decide if a flow is admitted or not. While such architecture provides required QoS per flow, it has significant scalability problems. Routers must process per-flow reservation requests, and must keep many per-flow states. This will place a lot of burden on routers, especially those routers in the Internet Service Provider (ISP) networks, where there

can be tens of thousands of flows.

1.3.2 Differentiated Service (DiffServ):

DiffServ is another approach to provide QoS in the Internet. DiffServ requires no per-flow admission control or signaling, and routers do not maintain any per-flow state. In DiffServ, a potentially large number of traffic flows are merged into a much fewer number of traffic classes. Each traffic class receives different QoS service; while flows in each class receive the same QoS service. Because DiffServ only needs to deal with a small number of traffic classes, it is more scalable than IntServ.

DiffServ is widely used in today's ISP networks. Note that RPR also only define a small number of service classes (three), thus, its structure in QoS support is very similar to DiffServ.

1.4 ClassA traffic in RPR network:

1.4.1 SubclassA0 & SubclassA1:

As it is mentioned before, classA service provides guaranteed bandwidth and supports low latency/jitter applications. Internal to the MAC, classA traffic is partitioned into two subclasses: subclassA0 and subclassA1. The bandwidth reserved for subclassA0 cannot be used (reclaimed) by traffic from other classes even it is not used by subclassA0 traffic; while unused bandwidth reserved for subclassA1 can be reclaimed by traffic from classB and classC. Note that the MAC client has no notion of subclassA0 and subclassA1 services. The client just passes the classA traffic to the MAC layer. It is up to the MAC layer to assign traffic to different subclasses. The amount of bandwidth that can be allocated for subclassA1 depends on the sizes of the secondary transit queues (STQ) in the RPR nodes (STQ will be discussed next). In this thesis, subclassA0 and subclassA1 are combined and treated as a single classA class since both subclasses receive essentially the same QoS treatment. Distinguishing subclassA0 and subclassA1 only has the effect on the performance of classB and classC traffic.

1.4.2 Transmit Operation

A RPR node has to deal with both the transmit traffic and the transit traffic. The transmit traffic is the traffic originated from the local MAC clients; while the transit traffic is received from the upstream RPR node. At the MAC layer, four traffic shapers are used to shape the traffic from four different transmit traffic classes (subclassA0, subclassA1, class B and class C). The shaped traffic is then selected to enter the stage queue based on the priorities of the traffic classes. SubclassA0 has the highest priority; subclassA1 has the second highest priority; while class C has the lowest priority. The RPR MAC datapaths are shown in Figure 1.1.

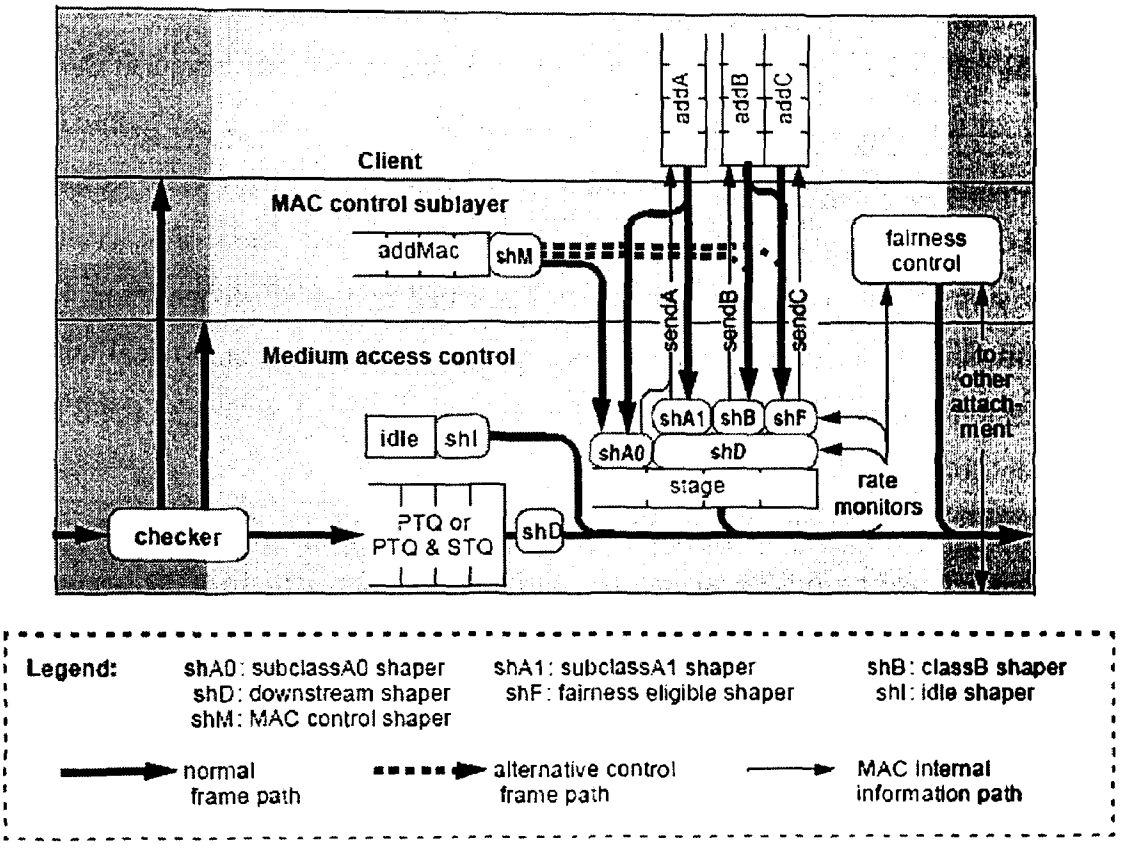


Figure 1.1 RPR MAC datapaths [1]

The client labels its frames as classA, classB and classC. The classA client adds classA flows (addA) through subclassA0 or subclassA1 shaper. The classB client adds classB flows (addB) through classB or fairness eligible shaper. The classC client adds classC flows (addC) through fairness eligible shaper. Fairness eligible traffic (ClassB-EIR and classC) is controlled by fairness eligible shaper and uses the available bandwidth from the unallocated bandwidth and unused reclaimable bandwidth. All unreserved transmit and transit traffic

flows are shaped by the downstream shaper.

Accepted client traffic is placed in the stage queue. The transmit traffic at the stage queue has to compete with the transit traffic for the access of the outgoing link of the node. The transit traffic is buffered in the transit queue. There are two types of MAC transit queueing designs: single-queue and dual-queue. The single-queue design places all transit traffic into a primary transit queue (PTQ), and sends the frames in the queue in the first-in-first-out sequence. The dual-queue design places classA transit traffic into a higher-precedence PTQ, and classB and classC transit traffic into a lower-precedence secondary transit queue (STQ). In this thesis, we will only consider the dual-queue operation for it provides the best QoS support for classA traffic. In the dual-queue operation, the traffic at the PTQ (the transit classA traffic) has the highest priority for transmission. If there is no traffic waiting for the transmission at the PTQ, then the traffic from the stage queue is selected for transmission. Finally, traffic at the STQ will be transmitted if there is no traffic waiting at both PTQ and the stage queue. For the purpose in this thesis, the brief description of the transmission operation above is sufficient. The exact transmission operation, however, is more complicated. It is because the transmission operation also has to deal with the transmission of control frames and the prevention of buffer overflow of the STQ. See the RPR document for more details [1].

1.4.3 Inefficient bandwidth usage problem for classA traffic in RPR network

In the RPR draft, no bandwidth allocation method is proposed. The bandwidth allocated for classA traffic at each node is assumed to be assigned statically through some Operational and Maintenance (OAM) procedures. Once the reserved bandwidth is assigned to an individual node, the node can only use that much bandwidth to transmit classA traffic even though there is more bandwidth available. It makes the bandwidth usage for classA traffic in RPR network not very efficient. This is especially true for subclassA0 type because the bandwidth allocated for this class of traffic, in general, cannot be spatially reused. We will investigate the inefficient bandwidth usage problem in more detail in Chapter 3 and propose three bandwidth reservation algorithms to tackle the problem.

1.5 Organization of the Thesis

The rest of the thesis is organized as follows. In Chapter 2, the effect of traffic shaping in the control of traffic throughput at the RPR MAC layer is studied. In the same chapter, three admission control algorithms are introduced and their effectiveness on the delay performance of classA traffic is investigated. In Chapter 3, we introduce the concept of common-bandwidth pool, where RPR nodes can dynamically reserve bandwidth from it. The introduction of common-bandwidth pool makes the bandwidth usage in RPR networks more efficient. Three bandwidth reservation algorithms are proposed to dynamically reserve and release bandwidth from the pool. They are: Basic Reservation Algorithm, Flow-Based Reservation Algorithm and Hop-by-Hop Reservation Algorithm. Finally, Chapter 4 describes the integration of the proposed dynamic bandwidth allocation algorithm with DiffServ and RSVP.

Chapter 2

The Study of Traffic Shaping and Admission Control for ClassA Traffic

To ensure that the classA traffic has low delay and jitter, admission control and traffic shaping must be used in the RPR networks. Admission control is used to make sure that the network resources are not oversubscribed; while traffic shaping regulates the input traffic to prevent temporarily traffic overload. In this chapter, we first introduce the traffic shaping method used in the RPR MAC layer and study the effect of traffic shaping on regulating the traffic load in the RPR network. We then focus on the admission control. Three admission control algorithms are introduced and their effects on the delay performance of classA traffic are investigated.

2.1 Traffic Shaping

2.1.1 Token bucket implementation

The traffic shapers used at the RPR MAC layer is implemented based on the token bucket algorithm [1]. In the token bucket algorithm, a frame can only be transmitted if there are enough credits (tokens) in the bucket. Every time a frame is transmitted, corresponding number of tokens (*decsize)* will be removed from the bucket. If there are not enough tokens in the bucket upon the arrival of a frame, then the frame cannot be transmitted until more tokens become available. Tokens are added to the bucket at a constant rate, which is the average rate at which frames are transmitted. This is accomplished by incrementing the credits in the token bucket by *incsize* at a constant interval. If no frames are waiting in the transmit buffer, then tokens begin accumulating in the bucket. The tokens stop accumulating when the bucket is full. The number of tokens at a given time represents the maximum number of bytes that can be sent. It is never possible for the number of credits to become negative after subtracting the number of bytes in the transmitted frame. Figure 2.1 illustrates how the credits are accumulated and depleted.

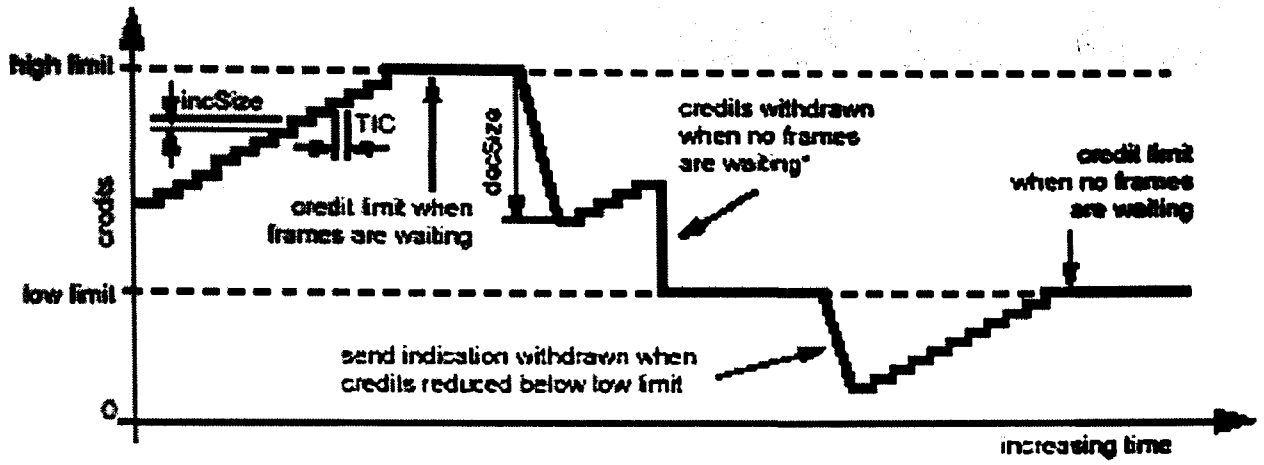


Figure 2.1 Token bucket algorithm used in MAC shaper [1]

There are two threshold values in the token bucket: *loLimit* threshold and *hiLimit* threshold. The *loLimit* threshold is used to generate a rate-limiting indication when the number of credits goes below the *loLimit* threshold. When no frames are ready for transmission, credits are reduced to *loLimit* (if currently higher than *loLimit*) and can accumulate to no more than *loLimit*. The *loLimit* is set to MTU_SIZE (maximum transmission unit size) in order to allow the transmission of a full sized frame without reducing the credits below zero.

The *hiLimit* threshold limits the positive credits, to avoid overflow. When frames are ready for transmission (and are being blocked by transmit traffic), credits can accumulate to no more than *hiLimit*. The *hiLimit* value should be at least MTU_SIZE. If it is set to exactly MTU_SIZE, no bursts are allowed.

2.1.2 Simulation study on traffic shaper

In this section, simulation results are presented to illustrate the effect of using the traffic shaper to regulate the traffic in the RPR network. In all the simulation scenarios, a RPR network with 20 nodes shown in figure 2.2 is used. Each of the nodes is a RPR router. The nodes are connected by dual-ring. The link rate for the transmission link is 622Mbps. And the distance between two adjacent nodes is 14 kilometers.

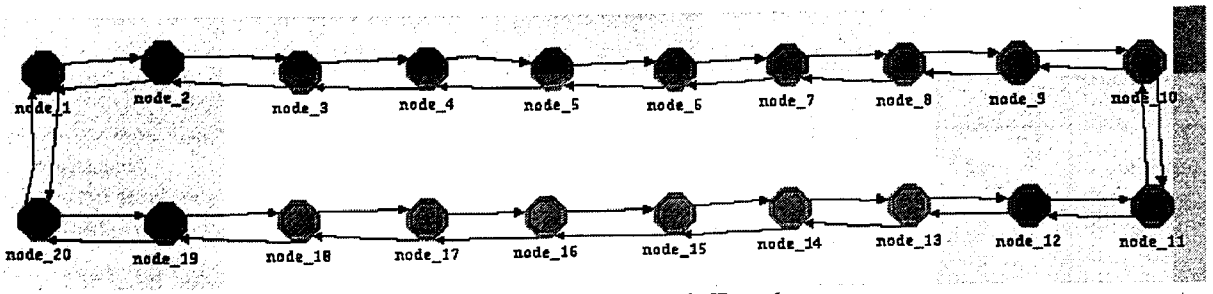


Figure 2.2: RPR Network Topology

Case 2.1: Traffic regulation using classA shaper

In this case, classA traffic shaper at node1 is set to support data rate of 200 Mbps. Node1, however, generates 300 Mbps of classA traffic to node8. The simulation result in Figure 2.3 shows that the shaper has successfully reduced the throughput to 200 Mbps. The excess traffic is queued in the transmit buffer and got delayed. Once the buffer is full, the excess traffic was dropped by the MAC layer.

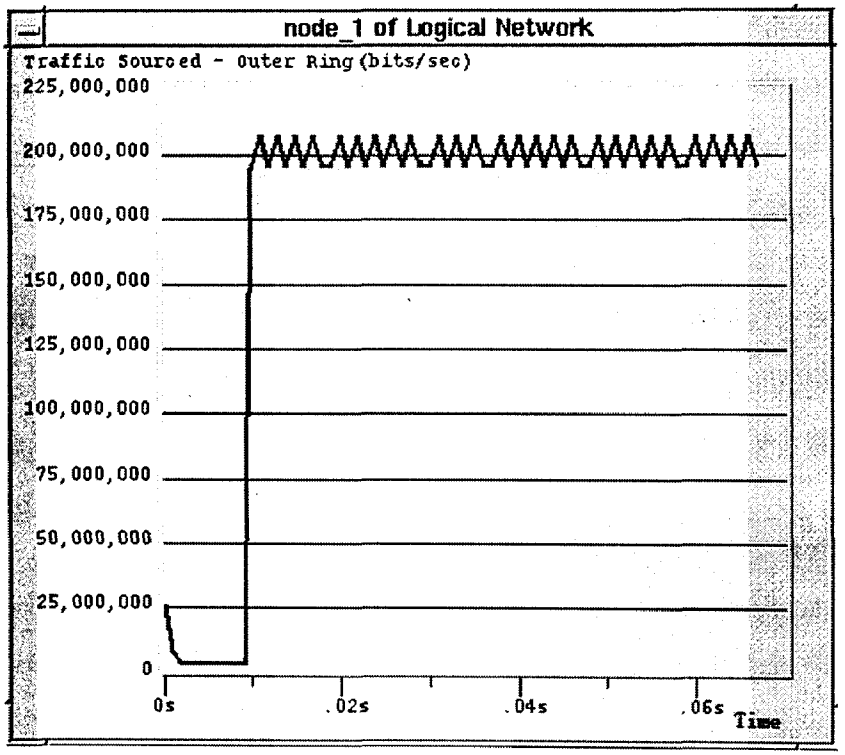


Figure 2.3: Throughput of classA traffic shaped by classA shaper

Case 2.2: Traffic regulation using downstream shaper

Downstream shaper is used to regulate the traffic that does not make any bandwidth reservation. In this simulation scenario, the reserved traffic bandwidth for classA traffic is set

to 200Mbps. Consequently; the downstream shaper is set to support the rate of the unreserved bandwidth, which is (622Mbps - 200Mbps) = 422Mbps. If node1 generates 600Mbps of classC traffic, then only 422Mbps of traffic should be sent. Figure 2.4 shows the simulation result on this scenario.

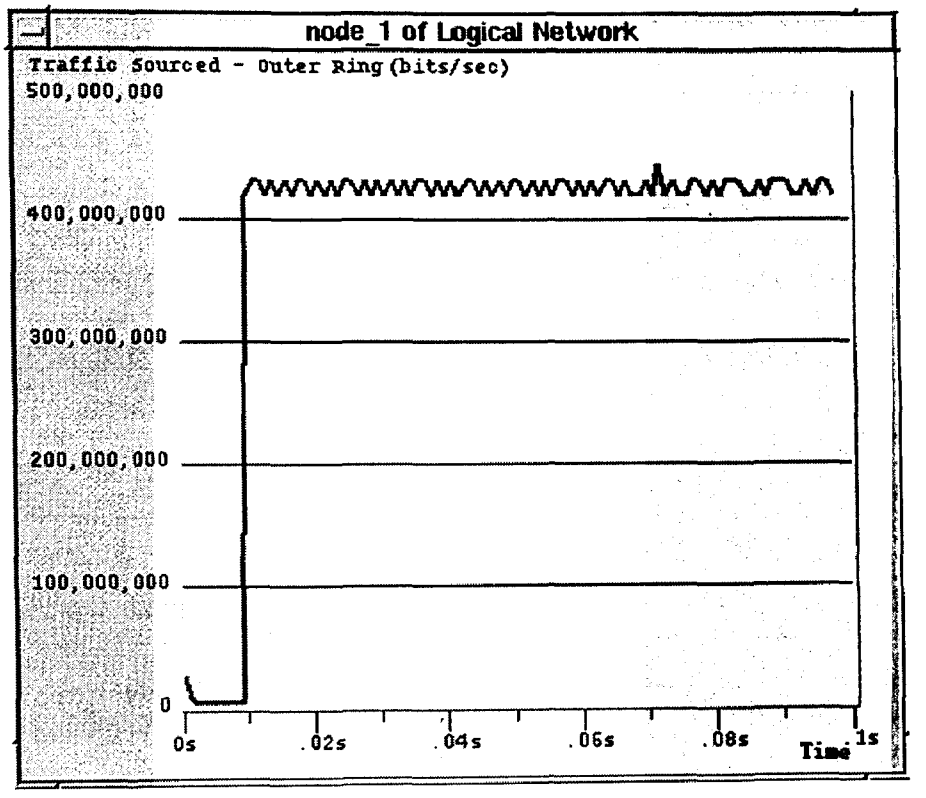


Figure 2.4: Throughput of classC traffic shaped by downstream shaper

Case 2.3: Traffic regulation with both classA and downstream shapers

In this case, the reserved rate for classA traffic is set to 100Mbps, so the unreserved rate is 522Mbps. Here, 300Mbps of classA traffic is generated from node1 to node8; 600Mbps of class C traffic from node 2 to node 9; and 600Mbps of class C traffic from node3 to node 10. Because the link rate is 622Mbps, congestion happens in this scenario. Figure 2.5 shows the simulation results. From the results, we can see that the throughput of classA traffic is restricted to 100Mbps. Node2 and Node3 fairly share the unreserved bandwidth of 522Mbps through the use of the RPR fairness algorithm. Consequently, they both send about 260Mbps of ClassC traffic.

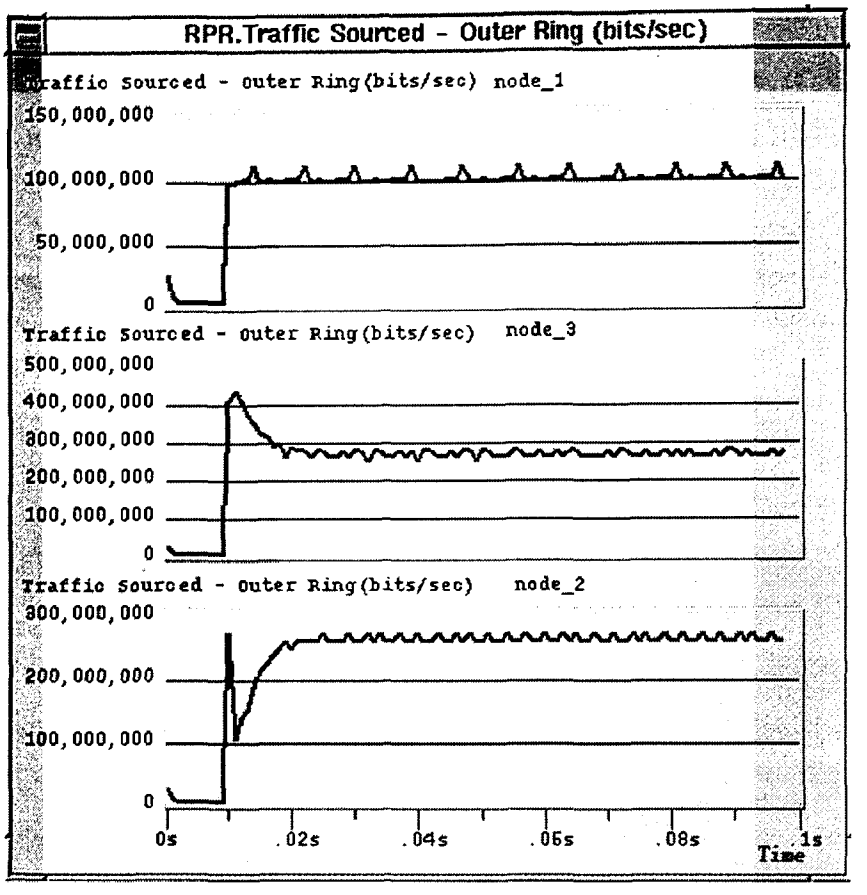


Figure 2.5 Throughputs of traffics shaped by classA and downstream shaper

2.1.3 Conclusion of the simulation study

The simulation results presented in the previous section demonstrate the effectiveness of using traffic shapers to control the traffic load. The results also demonstrate that, with the control of the rate of classA shaper, the bandwidth allocated for classA traffic can be properly reserved and excessive classA traffic is discarded by the shaper. To prevent large portion of classA traffic to be discarded, admission control must be implemented to keep the admitted traffic load below the bandwidth allocated to classA. In the next section, we will study the admission control process.

2.2. Admission Control

In admission control, a traffic flow must first get admitted before the data traffic is allowed to enter the network. The admission decision is usually based on the available network resources and made at the management node or traffic nodes along the path from the source to the destination. There are many admission control algorithms in the literature. Here,

we will study the bandwidth admission control algorithms [6,7]. These algorithms have the advantage of being relatively simple and thus suitable to be implemented in high-speed environments such as RPR.

In bandwidth admission control, a certain amount of reservable bandwidth is allocated to each traffic class. A flow of a given traffic class that wants to get admitted will first provide a set of traffic parameters to the admission control process to assist the process to make the admission decision. The traffic parameters usually include average data rate, peak data rate and maximum burst length. Based on the parameters, the admission process calculates the request bandwidth of the flow and adds this value to the current traffic load. If the sum is less than the reservable bandwidth, then the flow is admitted. There are two basic approaches to bandwidth admission control. The first, which we call the parameter-based approach, make the admission control decision simply based on the traffic parameters provided by the source. Once the admission control algorithm admits the flow, it uses these parameters to calculate the new traffic load. An example of parameter-based approach is Simple Sum. The second, the measurement-based approach, does not rely on the traffic parameter provided by the source to compute the traffic load, but instead, uses the measurement of actual traffic load in making admission decisions. An example of measurement-based approach is Measured Sum.

2.2.1 Simple Sum:

Simple Sum admission control algorithm simply ensures that the sum of all the requested bandwidth does not exceed the reservable bandwidth. Let v be the sum of the reserved bandwidth, u be the reservable bandwidth and r^a be the requested bandwidth for flow a. The Simple Sum admission control admits the new flow as long as it satisfied the following condition:

$$v + r^a < u \qquad \textbf{(2.1)}$$

Simple Sum admission method is easy to implement. But it has some drawbacks. Firstly, because it merely relies on the traffic parameter provided by the source. If the source reserves more bandwidth than it actually uses, the unused bandwidth cannot be reassign to other flows. Secondly, equation 2.1 does not account for the multiplexing effect on the outgoing link. Consequently, the requested bandwidth may be too large if peak data rate is used and too small if average data rate is used.

2.2.2 Measured Sum

Whereas the "Simple Sum" algorithm merely uses the sum of the requested bandwidths of the existing flows to compute the current traffic load, the "Measured Sum" algorithm measures the current load on the link, and makes the admission control based on the measurement and the requesting bandwidth of the new flow. Let v' be the measured load on the link and α be the user-defined utilization target. The Measured Sum algorithm admits the new flow, a, with the requested bandwidth of r^a if the following condition is satisfied:

$$v' + r^a < \alpha u \qquad (2.2)$$

Here we define a utilization target value, α, to keep the link utilization below a certain level to prevent a large delay variation at high link utilization. The Measured Sum algorithm measures the average network load, v', periodically. The choice of the length of the measured period, S, has a significant impact on the performance of the algorithm. If S is too short, the value measured may not truly reflect the actual traffic load of the overall traffic. If S is too large, the measurement process may not react fast enough to the change of traffic load. In either case, a wrong admission decision may result. Consequently, the accuracy of the load measurement depends on the proper choice of S.

2.2.3 Simulation study on the performance of classA traffic

In this section, we study the performance of classA traffic using the two admission control algorithms discussed above. The simulation is based on the topology shown in Figure 2.2. The study focuses on the end-to-end delay performance since classA service is designed to provide services to delay-sensitive traffic. The main components in the end-to-end delay here are propagation delay and queueing delay. To calculate the propagation delay in the simulated RPR network, we note that the distance between two adjacent nodes is 14 kilometers, and the light travels inside the optical fiber at the speed of 200,000 kilometers per second. Thus, the propagation delay between two adjacent nodes is 14km/200,000km/sec = 0.07ms. In the simulation, we choose the packet size to be 1500 bytes including the RPR packet header. We use on-off bursty traffic source in the simulation.

Case 2.4: Delay performance with two classA traffic flows

In the simulation, two classA traffic flows from node1 to node20 starting at 0.01s are

generated. The peak/average rates required by these two flows are 200/100Mbps and 90/45Mbps, respectively. We also generate 600Mbps classC traffic from both node2 and node3 to node19 to create congestion in the network. Here, the "Simple Sum" algorithm makes the admission control decision based on the average rates of the flows. Figure 2.6 shows the end-to-end delay and the classA queue size at node1. The end-to-end delay in this simulation is around 28.6ms. The maximum queue size in the transmit buffer of node1 is about 4M bits. Thus, the maximum queueing delay at node1 is about 26.6ms, which contributes the large portion of the overall delay of 28.6ms. The result can be explained as follows: Since node1 is the source node, the classA transmit traffic is shaped by a classA shaper and experiences substantial delay because of that. Once the traffic gets inside the network, it uses the PTQ and can virtually consume all the bandwidth available (622Mbps), so the delay experienced by the classA traffic is small at the transit node.

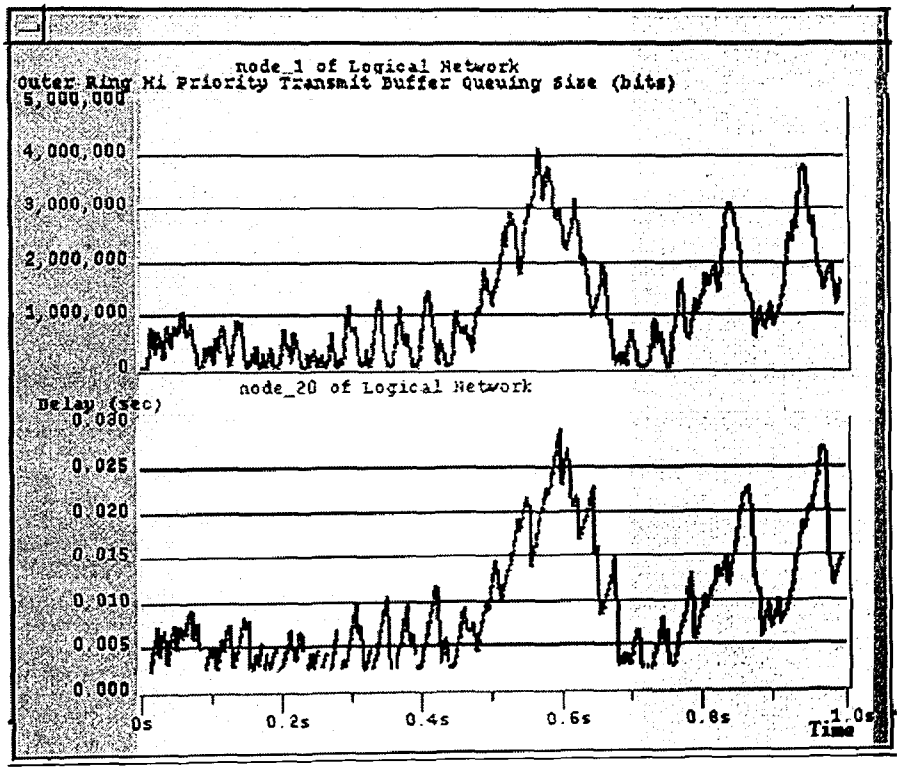


Figure2.6: Delay with two classA traffic flows (Simple Sum)

Case 2.5: Delay performance with multiple classA traffic flows

In this part, we study the delay performance with multiple traffic flows in the network. The total reservable bandwidth for classA traffic in the network is 210Mbps. Ten nodes generate classA traffic flows and share the reservable bandwidth uniformly, so each has

21Mbps. They all start generating traffic flows at 0.01s. The traffic parameters are of these traffic flows are given in the following table:

	Peak Rate	*ON Period*	*OFF Period*	*Average Rate*
Node1-Node20 (classA)	40Mbps	Exponential (10ms)	Exponential (10ms)	20Mbps
Node2-Node19 (classA)	40Mbps	Exponential (10ms)	Exponential (10ms)	20Mbps
Node3-Node18 (classA)	40Mbps	Exponential (10ms)	Exponential (10ms)	20Mbps
Node4-Node18 (classA)	40Mbps	Exponential (10ms)	Exponential (10ms)	20Mbps
Node5-Node18 (classA)	40Mbps	Exponential (10ms)	Exponential (10ms)	20Mbps
Node6-Node18 (classA)	40Mbps	Exponential (10ms)	Exponential (10ms)	20Mbps
Node7-Node18 (classA)	40Mbps	Exponential (10ms)	Exponential (10ms)	20Mbps
Node8-Node18 (classA)	40Mbps	Exponential (10ms)	Exponential (10ms)	20Mbps
Node9-Node18 (classA)	40Mbps	Exponential (10ms)	Exponential (10ms)	20Mbps
Node10-Node18 (classA)	40Mbps	Exponential (10ms)	Exponential (10ms)	20Mbps
Node11-Node1 (classC)	600Mbps	Exponential (10ms)	Exponential (10ms)	300Mbps
Node12-Node1 (classC)	600Mbps	Exponential (10ms)	Exponential (10ms)	300Mbps

Table 2.1: Traffic parameters for multiple traffic flows in the network

Figure 2.7 shows the throughput and delay performances. The simulation results show that the maximum delay is around 115ms. It is much higher than the previous cases. The reason is that the ON state period is longer (0.01s) and the ClassA shaper rate is much lower (21Mbps) than the previous cases. The longer ON state period causes a longer data queue buildup; and the lower shaper rate means it takes more time for the traffic to be drained out of the transmit queue. We can also take a look at the transit queue size in the transit nodes shown Figure 2.8. The transit buffer in the node is used to store the transit frames which are passing through the node via the ring. From the graph, we can see that the transit buffer size in the transit nodes is only about one packet. It means that once a classA frame enters the network, the queueing delay for the frame experienced at a transit node is small. Most of the delay is introduced by the queueing delay at the source node.

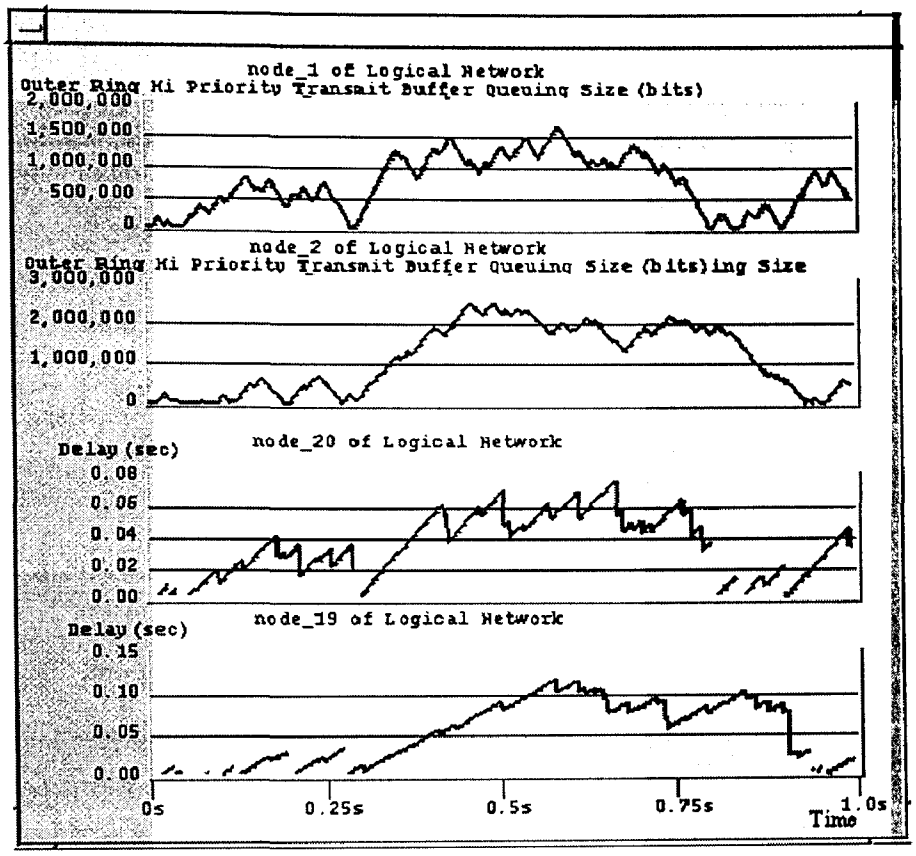


Figure 2.7: Delay performance with multiple flows

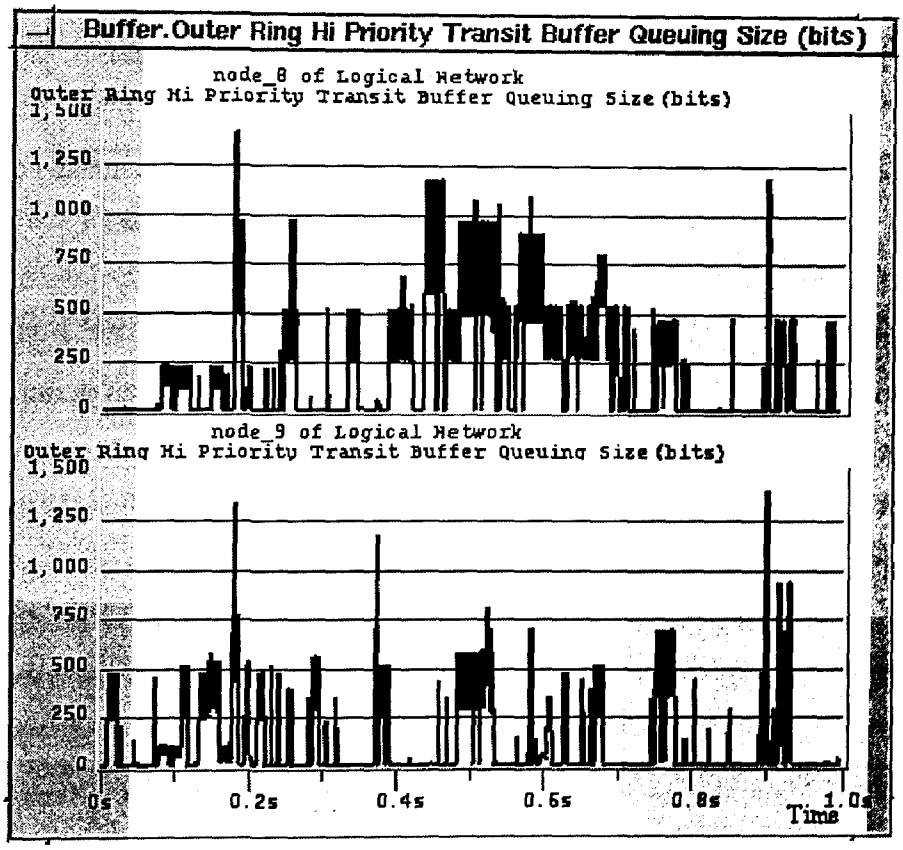


Figure 2.8: High priority transit buffer size in the transit nodes

Case 2.6: Delay performance with multiple traffic sources in one node

In this part, we study the end-to-end delay and the transmit buffer size with multiple traffic sources in one node.

In this test, 150Mbps of bandwidth is reserved for the classA traffic at node 1. Ten traffic sources are setup to generate 10 high priority traffic flows to node20 at 0.01s. The traffic parameters for the traffic flows by the 10 sources are as the following table:

	Peak Rate	*ON Period*	*OFF Period*	*Average Rate*
Source 1	40Mbps	Exponential (10ms)	Exponential (10ms)	20Mbps
Source 2	40Mbps	Exponential (10ms)	Exponential (10ms)	20Mbps
Source 3	40Mbps	Exponential (10ms)	Exponential (10ms)	20Mbps
Source 4	40Mbps	Exponential (10ms)	Exponential (10ms)	20Mbps
Source 5	40Mbps	Exponential (10ms)	Exponential (10ms)	20Mbps
Source 6	20Mbps	Exponential (10ms)	Exponential (10ms)	10Mbps
Source 7	10Mbps	Exponential (12ms)	Exponential (12ms)	5Mbps
Source 8	10Mbps	Exponential (12ms)	Exponential (12ms)	5Mbps
Source 9	40Mbps	Exponential (10ms)	Exponential (10ms)	20Mbps
Source 10	10Mbps	Exponential (12ms)	Exponential (12ms)	5Mbps

Table 2.2: Traffic parameters for multiple traffic sources in one node

We use the "Simple Sum" admission control method in this case. Because the sum of the average rates requested by the 10 traffic sources is 145Mbps, all these 10 flows are admitted into the network. We also define both node11 and node12 to generate 600Mbps (average rate) classC traffic to cause the congestion in the network. Figure 2.9 shows the queue length of the transmit queue at the source node (node1) and the end-to-end delay of the classA packets.

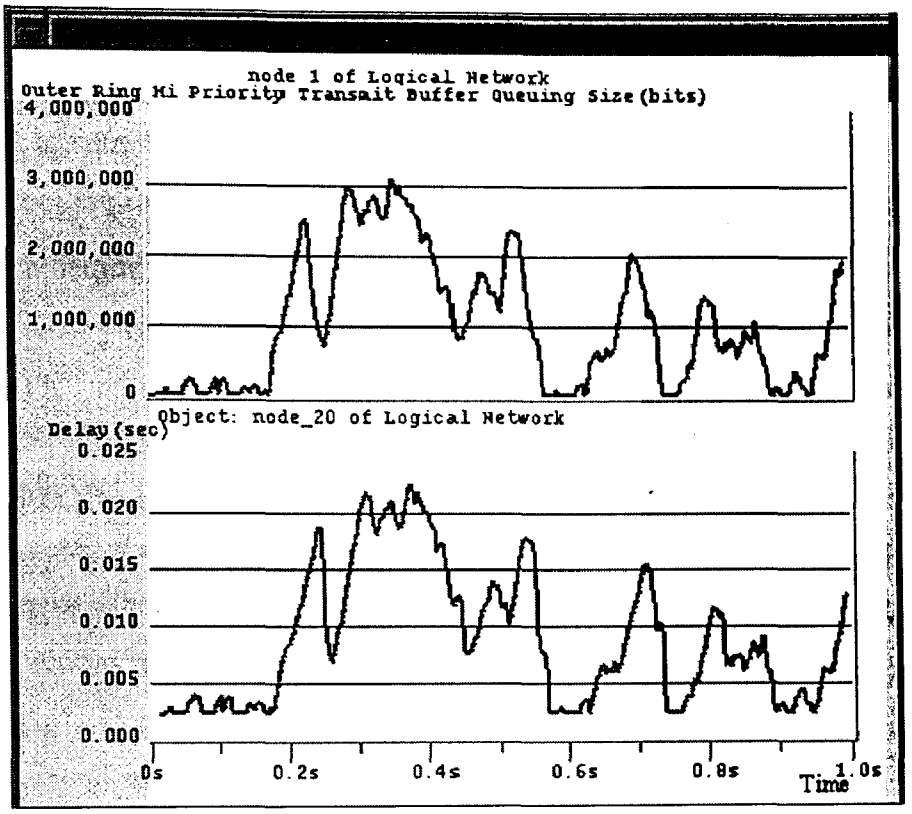


Figure 2.9: Delay performance with multiple traffic sources in one node

The result shows that the maximum delay is around 22ms, which is smaller than the 28.6ms maximum delay found in Case 2.4 where there are only two traffic flows. The higher number of traffic flows introduce a smaller delay because of the multiplexing effect. Simply summing the average bandwidths of the flows cannot capture the multiplexing effect because the multiple flows' aggregate statistical behavior differs from the sum of their individual statistical representations. In the next section, we will utilize the concept of Equivalent Bandwidth in the admission control algorithm to make the algorithm compute the traffic load more accurately.

2.2.4 Equivalent Bandwidth

High-speed network architectures are capable of supporting a wide range of connections with different bandwidth requirements and traffic characteristics. While this environment provides increased flexibility in supporting various services, its dynamic nature poses difficult traffic control problems when trying to achieve efficient use of network resources. One such problem is the issue of bandwidth management and allocation. When connections are statistically multiplexed, their aggregate statistical behavior differs from their

individual statistical representation. One needs, therefore, to define other metrics to represent the effective bandwidth requirement of an individual connection as well as the total effective bandwidth requirement of connections multiplexed on each link. In this section, the concept of equivalent bandwidth is introduced. By using the equivalent bandwidth, the actual bandwidth requirement of the multiplexed connections can be determined more accurately.

2.2.4.1 Equivalent bandwidth derivation

The equivalent bandwidth C(ε) of a set of connections multiplexed on a link is defined as the amount of bandwidth required to achieve a desired grade of service (GOS), such as guaranteed delay value or buffer overflow probability, given the offered aggregate bit rate generated by the connections[8]. Here, we measure the GOS by ε, which is the probability of the buffer overflow.

To calculate the equivalent bandwidth of a single traffic flow, we use the traffic parameters of the traffic source. When a source asks for the permission of connection, it should provide the peak rate (p), the burst period (b), and the utilization (u) which is the fraction of time when the source is active. We define the maximum of the burst period to be b_{max}. Reference [6] provides a simple approximation of C(ε) by $C_s(\varepsilon)$:

$$C(\varepsilon) \approx C_s(\varepsilon) = \sum_{i=1}^{N} C_i \quad , \tag{2.3}$$

where N is the number of multiplexed connections, and C_i is the equivalent bandwidth for the *i*-th connection and is given by the following equation:

$$C_i = \frac{\alpha b_i(1-u_i)p_i - \chi + \sqrt{(\alpha b_i(1-u_i)p_i - \chi)^2 + 4\chi\alpha b_i u_i(1-u_i)p_i}}{2\alpha b_i(1-u_i)} \quad , \tag{2.4}$$

Where $\alpha = \ln(1/\varepsilon)$ and χ is the buffer size; p_i, b_i and u_i are the peak rate, burst period and utilization of *i*-th connection respectively.

In RPR node, we use the following equation to determine the maximum buffer size for a single traffic flow:

$$\chi = b_{max}(p_i - C_i) \ . \tag{2.5}$$

By substituting equation (2.5) into equation (2.4), we get

$$C_i = \frac{b_{max}(1+u_i)p_i - Kp_i + \sqrt{(b_{max}(1+u_i) - K)^2 p_i^2 - 4u_i p_i^2 (b_{max} - K)b_{max}}}{2(b_{max} - K)} \quad , \tag{2.6}$$

Where $K = \ln(1/\varepsilon)b_i(1-u_i)$.

Essentially the calculation of the equivalent bandwidth for multiple traffic flows using equation (2.3) is to sum the equivalent bandwidths for all the traffic flows together. Thus, equation (2.3) uses the same approach as the Simple Sum method does, except that the requested bandwidth of a flow is equivalent bandwidth of that flow, rather than the average bandwidth. By using equation (2.3), we can simplify the bandwidth reservation algorithms proposed in the next chapter. We have also investigated other methods to calculate the equivalent bandwidth provided in references [7]. The main advantage of equation (2.3) is its computational simplicity. However, by simply summing the equivalent bandwidths of individual flows we overestimate the equivalent bandwidths once the flows are multiplexed. Other methods may be more accurate when we calculate the equivalent bandwidth for a large number of connections multiplexed over relatively long burst periods. But those methods are more complicated to implement, and their accuracy is not significantly better compared with that from (2.3) when the number of connections in the network is not too large.

In RPR network, a node makes the admission control decision based on the equivalent bandwidth calculation for a classA traffic flow using (2.3) and (2.6). Let $C(\varepsilon)$ be the sum of the equivalent bandwidths for all the existing flows, C_i be the equivalent bandwidth for the coming flow and u be the outgoing link capacity. Then the equivalent bandwidth admission control admits the new flow as long as it satisfies the following condition:

$$C(\varepsilon) + C_i < u . \qquad (2.7)$$

Note that equation (2.7) has the same form as equation (2.1) of the Simple Sum algorithm. In the next section, we will show the simulation results using the Equivalent Bandwidth admission control algorithm.

2.2.4.2 Simulation result using the Equivalent Bandwidth algorithm

Case 2.7: Delay parameters measured using the Equivalent Bandwidth algorithm

In this case, we use the same simulation setup as in case 2.6. We reserve 150Mbps bandwidth for the classA traffic at node 1. We also have 10 traffic sources to generate 10 classA flows using the same traffic parameters used in case 2.6. The traffic parameters, together with the corresponding equivalent bandwidth, are given in Table 2.3:

	Start Time	*Peak Rate*	*ON Period*	*OFF Period*	*Equivalent Bandwidth*
Source 1	0.01s	40Mbps	Exponential (10ms)	Exponential (10ms)	25.98Mbps
Source 2	0.02s	40Mbps	Exponential (10ms)	Exponential (10ms)	25.98Mbps
Source 3	0.03s	40Mbps	Exponential (10ms)	Exponential (10ms)	25.98Mbps
Source 4	0.04s	40Mbps	Exponential (10ms)	Exponential (10ms)	25.98Mbps
Source 5	0.05s	40Mbps	Exponential (10ms)	Exponential (10ms)	25.98Mbps
Source 6	0.06s	20Mbps	Exponential (10ms)	Exponential (10ms)	12.99Mbps
Source 7	0.07s	10Mbps	Exponential (12ms)	Exponential (12ms)	6.49Mbps
Source 8	0.08s	10Mbps	Exponential (12ms)	Exponential (12ms)	6.49Mbps
Source 9	0.09s	40Mbps	Exponential (3ms)	Exponential (3ms)	25.98Mbps
Source 10	0.10s	10Mbps	Exponential (12ms)	Exponential (12ms)	6.49Mbps

Table 2.3 Traffic parameters for multiple traffic flows in one node with equivalent bandwidth algorithm

With the use of the Equivalent Bandwidth algorithm, the source node can only accept 7 traffic flows from source 1 to source 7, because the sum of equivalent bandwidth for the 7 flows is 149.38Mbps and the source node cannot accept the traffic flow from source 8, source 9 and source 10. Figure 2.10 shows the queue size of the source node and the end-to-end delay.

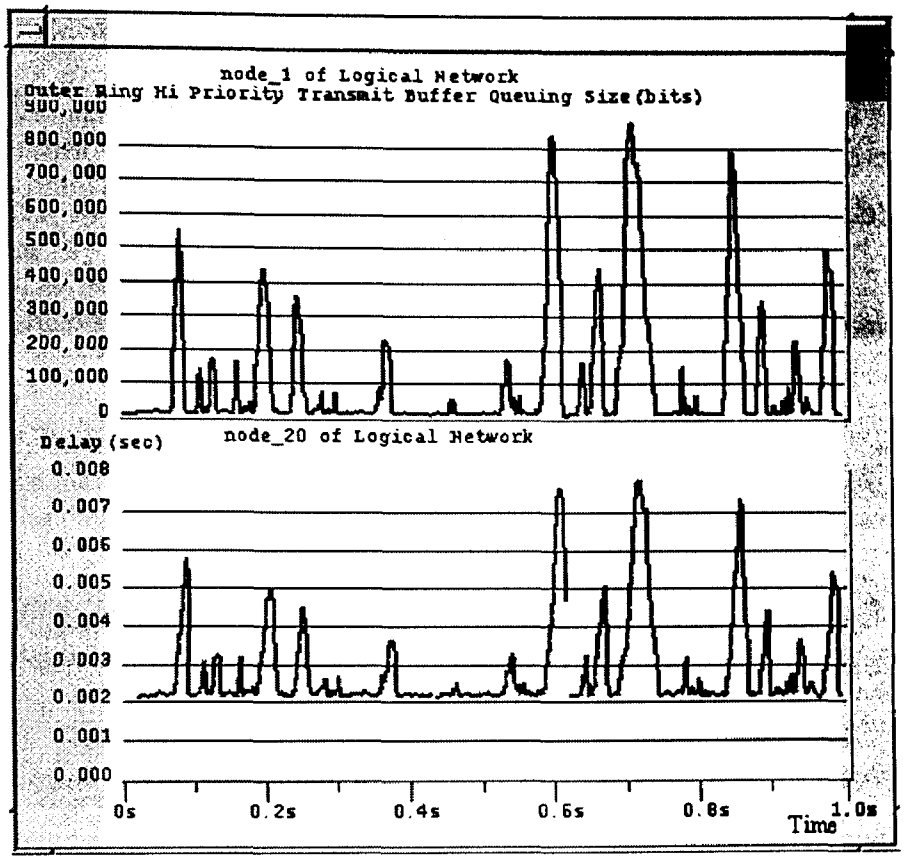


Figure 2.10 Delay performance with Equivalent Bandwidth algorithm

From the figure, we can see that the maximum end-to-end delay in this case is 7.8ms, which is much smaller than the 22ms maximum end-to-end delay introduced by the Simple Sum algorithm in case 2.6. On the other hand, the Equivalent Bandwidth algorithm admits a smaller number of flows. Thus, there is a trade-off between the number of admitted flows and the delay performance. Since classA service is more concerned with the delay/jitter performance. Equivalent bandwidth is a more suitable choice for this type of service. In the rest of the thesis, the Equivalent Bandwidth algorithm will be used as the underlying admission control process.

2.3 Conclusion

In RPR network, it is important to provide high QoS to classA traffic. Here we study different bandwidth admission control methods to control the classA traffic. From our simulation results, we observed that with the use of bandwidth admission control, the delay performance is usually acceptable. We also observed that most of the delay is introduced by

the classA shaper at the source node. After the data traffic enters the network, the delay in the network is almost the same as the propagation delay. Out of the three admission control algorithms studied, we conclude that the Equivalent Bandwidth algorithm is the most suitable to use for classA admission control process.

Chapter 3

Dynamic Bandwidth Allocation Using Common Bandwidth Pool

3.1 Introduction

In the RPR standard, the reserved bandwidth for classA traffic at each node is statically assigned. Once the reserved bandwidth is assigned to a node, the node can only use that much bandwidth to transmit classA traffic even though there is more bandwidth available. For instance, consider the case where a network has 11 nodes, and the reserved bandwidth for classA traffic is 100Mbps in the whole network. If we equally assign reserved bandwidth to each node, each node receives 10Mbps bandwidth because only 10 nodes can use any given link at the same time. Based on this assignment, each node can only send maximum 10Mbps classA traffic even if all the other nodes are not using their shares of bandwidth.

In this chapter, we will investigate a more efficient method in which bandwidth is dynamically allocated. In this method, two types of bandwidth are defined: guaranteed bandwidth and common bandwidth. The guaranteed bandwidth for classA traffic is pre-assigned to each node. Besides an assignment of guaranteed bandwidth to each node, a common-bandwidth pool is created where all the nodes share the bandwidth in the pool. The bandwidth from the common-bandwidth pool is called common bandwidth. If a node requires more bandwidth than its guaranteed bandwidth, it can send request to reserve the extra bandwidth from the common-bandwidth pool along the path of the traffic flow. All the nodes along the path must then perform admission control to decide if the request for common bandwidth should be granted. The admission control adopted in this chapter is based on the equivalent bandwidth algorithms covered in Chapter 2. If there is enough available bandwidth from the common-bandwidth pool to satisfy the request, the request will be granted. The path is successfully setup if all the nodes along the path grant the request. If the available bandwidth from the common-bandwidth pool in any transit node is not enough to grant the request, a negative acknowledgment will be sent back to the source node and the reservation setup fails. After the bandwidth reservation for a flow is successfully set up, the source node can start sending data frames from that flow.

In what follows, we will propose and study three bandwidth reservation algorithms for

classA traffic based on the equivalent bandwidth calculation. These algorithms are: Basic Reservation Algorithm, Flow-Based Reservation Algorithm and Hop-by-Hop Reservation Algorithm.

3.2 Definitions

In this section, we define some important terms that are associated with the reservation algorithms to be discussed in this chapter.

1) Guaranteed Bandwidth (B_{G_k}): The Guaranteed bandwidth is the bandwidth statically assigned to node k in the RPR network used for classA traffic.
2) Available Guaranteed Bandwidth (B_{AG_k}): The available guaranteed bandwidth indicates how much bandwidth from the guaranteed bandwidth pool is available for a new connection at node k. The initial value for B_{AG_k} is the same as B_{G_k}. Once the node admits a new flow, *i*, it adjusts the B_{AG_k} value by using:

$$B_{AG\_k} = \max (0, B_{AG\_k} - C_i) \quad (3.1)$$

Where C_i is the equivalent bandwidth of flow *i*. When B_{AG_k} becomes zero, it means no more guaranteed bandwidth left for new connections.

3) The Requested Bandwidth (ΔC_i): ΔC_i is the bandwidth required from the common bandwidth pool to support the new traffic flow, *i*, and is carried in the bandwidth request packet to set up the reservation. When the equivalent bandwidth from a new traffic flow is greater than the value B_{AG_k}, the source node k calculates the ΔC_i by using:

$$\Delta C_i = C_i - B_{AG\_k} \quad (3.2)$$

4) Common-bandwidth pool size (B_{C_k}): The common-bandwidth pool size specifies the amount of common bandwidth at node k to be shared by all the nodes in the RPR network.
5) Available Common Bandwidth (B_{AC_k}): The available common bandwidth indicates how much bandwidth from the common-bandwidth pool at node k is available for a new connection. The initial value for B_{AC_k} is the same as B_{C_k}. Once node k admits a new connection, it adjusts the B_{AC_k} value by using:

$$B_{AC\_k} = B_{AC\_k} - \Delta C_i \quad (3.3)$$

Note that if B_{AC_k} - ΔC_i is less than zero, the reservation request will be rejected.

6) Released Bandwidth (C_{RC}): Released bandwidth is calculated at the source node. It indicates how much bandwidth from the common bandwidth pool should be released downstream when the connection is terminated.
7) Total Bandwidth: the "Total Bandwidth" for a node is the sum of the guaranteed bandwidth and the common-bandwidth pool size, B_{C_k}+ B_{G_k}. Total bandwidth is the maximum bandwidth available to node k.

3.3 State Tables

Each node uses two state tables, the transmit state table and transit state table, to record the reservation states of all traffic flows.

3.3.1 Transmit State Table

The transmit state table is used by the source node to record the traffic parameters for the locally sourced flows. Every locally sourced flow has one entry in the transmit state table. Each entry contains the following fields: source flow ID, destination address, flow bandwidth, request bandwidth and the flag. Further definitions of these fields are given below.

1) Source flow ID: The source flow ID of a traffic flow uniquely identifies that traffic flow from other traffic flows that have the same source node.
2) Destination address: It records the destination address of the traffic flow.
3) Flow Bandwidth: It records the equivalent bandwidth of the traffic flow.
4) Request Bandwidth: It records the requested common bandwidth of the flow.
5) Flag: It indicates whether this flow has reserved bandwidth from the common-bandwidth pool. Flag=1 indicates that a reservation is made; Flag=0 indicates the opposite of that.

3.3.2 Transit State Table:

The transit state table is used to keep track of the transit flows. Every transit flow has one entry in the transit state table. Each entry only contains one field which is the flow ID. The flow ID consists of the MAC address of the source node and the source flow ID. The

flow ID uniquely identifies a flow at the transit node. Note that the transit state table does not record the request bandwidth of a flow. The common bandwidth reservation and release are only controlled by the source node. This characteristic allows a RPR node to handle tens of thousands of flows without requiring significant storage and processing.

3.4 Packet Formats

The reservation algorithm defines 5 control packet types: bandwidth_request, Release_Reservation, acknowledgment (ACK_BR) for bandwidth_request packet, acknowledgment (ACK_RR) for Release_Reservation packet and negative acknowledgment (NAK). The packet formats for the five packet types are the same. These packets are encapsulated by RPR control frame and are processed by each hop along the path from the source to the destination. Figure 3.1 shows the basic packet format:

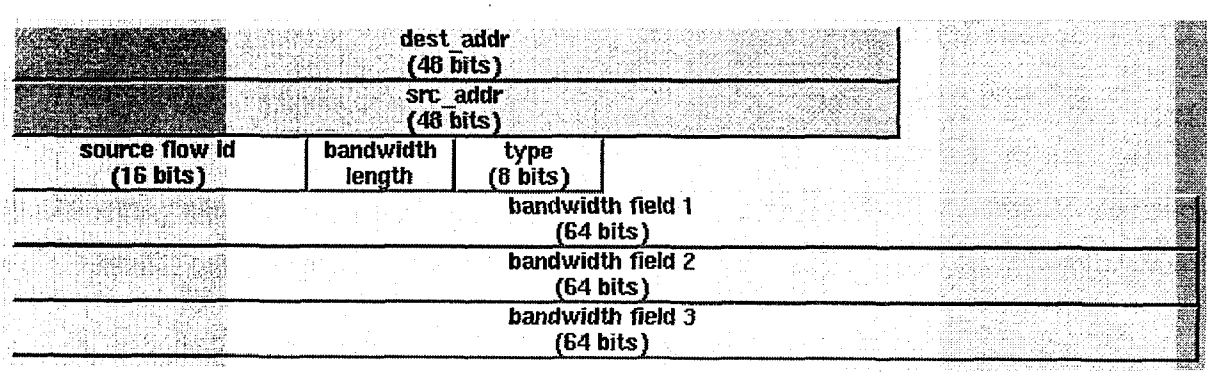


Figure 3.1: Packet Format for dynamic bandwidth allocation method

The "src_addr" field contains the station addresses of the node that generates the control packet. The "dest_addr" field contains the station address of the destination of the control packet. The "bandwidth" field contains the hop address and requested bandwidth for that hop. The first six bytes of the "bandwidth" field contain the "hop address" in the network which is the MAC address of the transit node, while the last two bytes of the "bandwidth" field is the actual "requested bandwidth". The number of bandwidth fields in the Hop-by-hop Reservation Algorithm (discussed in section 3.8) is the number of hops of the flow minus one. The "bandwidth length" field indicates the number of the bandwidth fields in the packet. In the Basic Reservation Algorithm and the Flow-based Reservation Algorithm, the "hop address" is not used and is set to be 0xffffffff. There is only one bandwidth field in the packets for these two algorithms. The "source flow ID" field contains the source flow ID of the traffic flow. When the transit node receives the bandwidth request/release packet, it

combines the MAC address of the source node and the source flow ID carrying in the packet and gets the flow ID of the incoming traffic flow. The flow ID is unique in the whole network. The "type" field is used to identify different packet types. The description of each packet type is given below.

1) Bandwidth_Request Packet:Bandwidth_Request packet (type = 0) carries the requested bandwidth from the source node to the destination node to set up the reservation for the new traffic flow
2) Release_Reservation Packet: When a traffic flow has terminated, the source node generates the Release_Reservation packet (type = 1) downstream to the destination node to release the reservation of the flow.
3) Acknowledgment Packets: The acknowledgment packet (ACK) is generated by the destination node of the flow. It is used to acknowledge the reception of the bandwidth request packet or bandwidth release packet. We define two types of acknowledgement packet: the ACK for Bandwidth_Request packet (type = 2) and the other is the ACK for Release_Reservation packet (type = 3). In the acknowledgment packet, the "bandwidth" field is the amount of common bandwidth to be reserved or released for the flow.
4) Negative Acknowledgment Packet: The negative acknowledgment packet (NAK) is generated by any of the transit nodes along the path from the source node to the destination node who does not has the enough bandwidth available in its common-bandwidth pool to support the new traffic flow. There is no NAK packet for the bandwidth release packet.

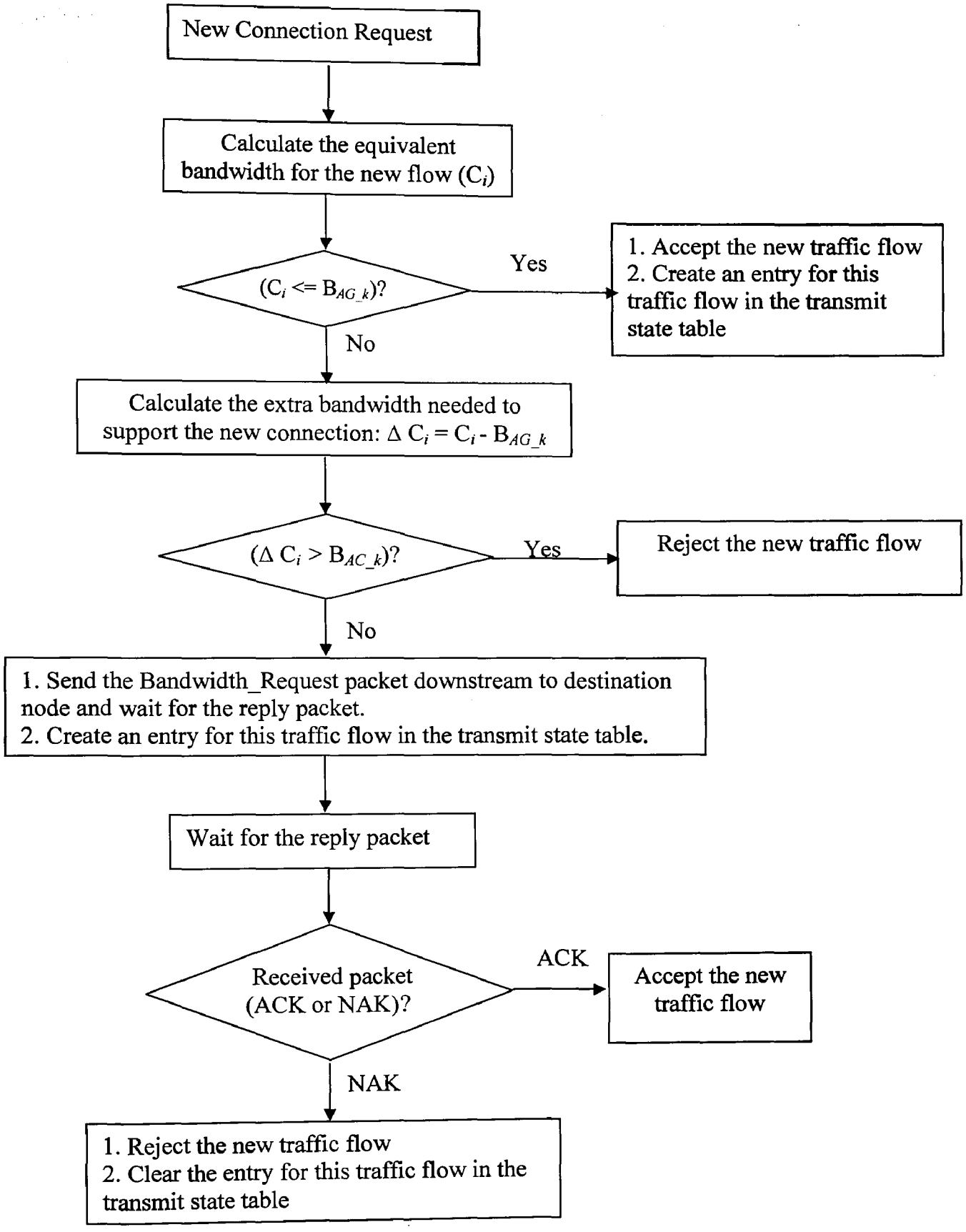
3.5 Basic Reservation Algorithm

In this section the basic reservation algorithm is introduced. In the basic reservation algorithm, when the MAC layer of the source node receives a connection request of a traffic flow from the local MAC client, it decides whether it needs to reserve bandwidth from the common-bandwidth pool for the connection. If it does, it will generate the bandwidth request packet downstream to setup the reservation. When the traffic flow is terminated, the MAC layer decides whether it needs to release the common bandwidth. If it does, it will generate

the bandwidth release packet downstream to release the reservation. In the rest of the section, the algorithm will be described in detail.

3.5.1 Reservation procedure at the source node:

The reservation procedures at the source node, transit node and destination node are different. They will be described separately. First, we will describe the admission control procedure at the source node. We assume that the source node currently supports *i-1* classA connections and a new classA connection request from traffic flow *i* just arrives. The reservation procedure at the source node can be described by the following flowchart. The next few paragraphs give detailed explanations of the flowchart.



Flowchart 3.1: Basic Reservation Algorithm at the source node

When the MAC layer of node k receives a new traffic flow connection request from its upper layer, the MAC layer goes through the following steps according to the above flowchart:

1) When a new connection request for traffic flow *i* arrives, the MAC layer calculates the equivalent bandwidth (C_i) for the new flow using equation (2.6).
2) If C_i is less than B_{AG_k}, the request is granted and no common bandwidth reservation is required. The MAC layer creates an entry in the transmit state table for the new traffic flow and save the C_i and ΔC_i values ($\Delta C_i = 0$ in this case) in the new entry. The information is used to modify B_{AG_k} and B_{AC_k} values when this traffic flow is terminated. The MAC layer also updates B_{AG_k} using equation (3.1). If C_i is greater than B_{AG_k}, go to step (3).
3) Calculate the extra bandwidth needed to support the new traffic flow by using equation (3.2). The MAC layer compares the ΔC_i value with the B_{AC_k} value. If the ΔC_i value is less than B_{AC_k}, the MAC layer generates the Bandwidth_Request packet downstream to the destination node. The requested bandwidth carried in the Bandwidth_Request packet is the ΔC_i value. MAC layer also creates an entry in the transmit state table for the new traffic flow and saves the C_i and ΔC_i values in the "flow bandwidth" and "request bandwidth" fields of the new entry. After generating the Bandwidth_Request packet, MAC layer updates the B_{AG_k} and B_{AC_k} value by using the equations (3.1) and (3.3). This means that an amount of bandwidth equivalent to the requested bandwidth is removed from the common bandwidth pool and is unavailable for new reservation requests. The source node also starts a time-out timer and waits for the reply packet from the downstream node.
   If the ΔC_i value is greater than the B_{AC_k} value, MAC layer rejects the connection request.
4) When the source node k receives the reply packet from its downstream node, it checks the packet type (ACK or NAK). If the packet is an ACK packet, the connection is established. If the packet is a NAK packet, the source node rejects this traffic flow and adjusts the B_{AG_k} and B_{AC_k} values according to the following equations:

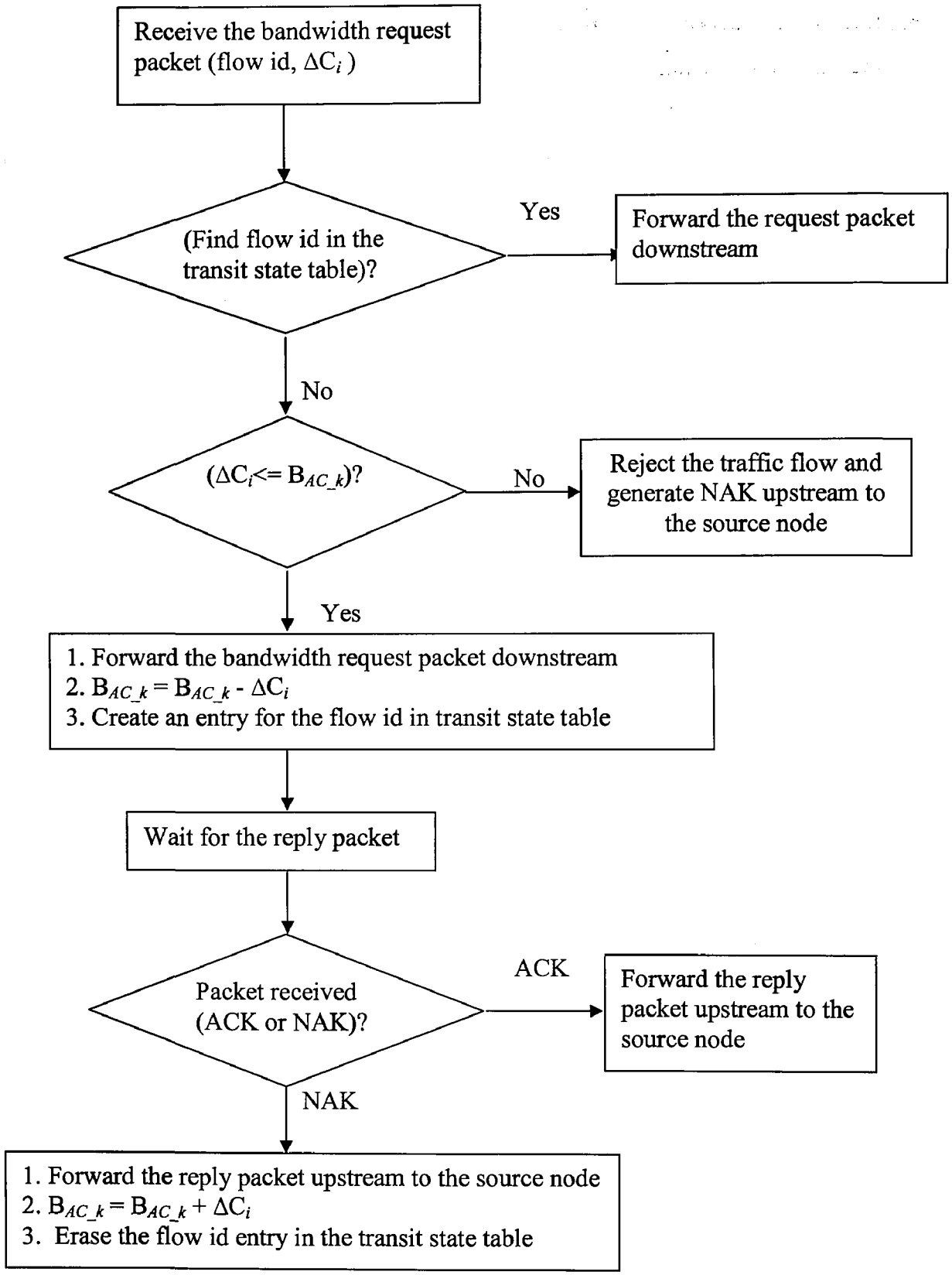
$$B_{AC\_k} = B_{AC\_k} + \Delta C_i \qquad (3.4)$$

$$B_{AG\_k} = B_{AG\_k} + (C_i - \Delta C_i) \qquad (3.5)$$

5) If the retransmission timer started in step (3) times out and no reply packet received, the MAC layer retransmits the Bandwidth_Request packet. The MAC layer expects to receive an ACK or NAK of the request eventually. If it does not receive any reply after many tries, it generates a system error to alert the administration for the possible network problems.

3.5.2 Admission control procedure for the transit nodes:

When a transit node receives the bandwidth request packet, it goes through the following steps to make the admission control decision:



Flowchart 3.2: Admission control procedure at the transit node

When the transit node k receives the Bandwidth_Request packet, first it tries to locate the flow ID in its transit state table. If the transit node finds an entry for the flow ID in the transit state table, it means the node has already set up the reservation for this flow and the node just simply forward this request packet downstream. This situation happens when the Bandwidth_Request packet is retransmitted.

If the transit node k does not find an entry for the flow ID in the transit state table, it compares the requested bandwidth (ΔC_i) with the available bandwidth left in its common bandwidth pool. If the requested bandwidth is less than the available bandwidth in the common bandwidth pool, it means this node has enough available bandwidth left for the new connection. Then the node admits the new traffic flow and forwards the Bandwidth_Request packet downstream. At the same time the node saves the flow ID carried in the request packet into the transit state table and adjusts the size of the common bandwidth pool using equation (3.3).

If the requested bandwidth is greater than the available bandwidth in the common bandwidth pool, the node rejects the new traffic flow and generates NAK packet upstream to the source node.

When the transit node receives the reply packet, it checks the reply packet type. If the reply packet is an ACK packet, the transit node forwards the reply packet upstream to the source node. If the reply packet is a NAK packet, the node also forwards the reply packet upstream to the source node. In addition, it deletes the entry of the flow from its transit state table and releases the common bandwidth reservation by the amount indicated in the NAK packet. The reservation release can be performed simply by adjusting the available common bandwidth (B_{AC_k}) using equation (3.6):

$$B_{AC\_k} = B_{AC\_k} + \Delta C_i \qquad (3.6)$$

3.5.3 Admission control procedure at the destination node:

When the destination node receives the bandwidth request packet, it generates an acknowledgment (ACK) packet back to the source node. The source and destination addresses, the request bandwidth and the flow ID field are directly copied from the corresponding fields in the bandwidth request packet.

3.5.4 Reservation release for the traffic flow:

Whenever a flow is terminated, the source node determines if the terminated flow has reserved common bandwidth based on the flag value stored in the transmit state table. If the flag value is 0, no common bandwidth is reserved for the flow and the source node does not generate the Release_Reservation packet. Instead it just modifies the value using equation (3.7):

$$B_{AG\_k} = B_{AG\_k} + C_i \tag{3.7}$$

If the flag value is 1, the terminated flow has reserved common bandwidth and the source node will generate a Release_Reservation packet towards the destination node to release the bandwidth reservation. The Release_Reservation packet specifies the amount of common bandwidth to be released. Let C_{RC} denoted as the amount of common bandwidth to be released. In the Basic Reservation Algorithm, we have

$$C_{RC} = \Delta C_i \tag{3.8}$$

The values of B_{AG_k} and B_{AC_k} are then adjusted by using equations (3.9) and (3.10):

$$B_{AC\_k} = B_{AC\_k} + C_{RC} \tag{3.9}$$

$$B_{AG\_k} = B_{AG\_k} + C_i - C_{RC} \tag{3.10}$$

When the transit node receives the Release_Reservation packet, it searches for an entry in its transit state table that has the same flow ID. If the entry is found in the transit state table, the transit node knows it has reserved certain bandwidth for this traffic flow and now it need to release the bandwidth as indicated by C_{RC}. The transit node goes through the following steps to release the reservation:

1) It erases the entry from the transit state table;
2) It adjusts the B_{AC_k} value by using the equation (3.9);
3) It forwards the Release_Reservation packet downstream to the destination node.

When the destination node receives the Release_Reservation packet, it sends the ACK for bandwidth release packet to the source node carrying the flow ID of the flow. When the transit node receives the ACK packet for the Release_Reservation packet, it simply forwards the ACK packet upstream to the source node. When the source node receives the reply packet for the Release_Reservation packet, it removes the entry of the corresponding traffic flow in it's transmit state table.

3.6 Retransmission mechanism

Because the MAC layer does not provide reliable frame transfer, a retransmission mechanism is required in the algorithm to deal with the possible control packet loss. The retransmission mechanism will be explained from the angles of the source node, transit node and destination node.

3.6.1 Retransmission mechanism at the source node:

Whenever the MAC layer of the source node sends a bandwidth request or release packet, it starts a retransmission timer. It also saves the source flow ID and the requested bandwidth for this traffic flow in the transmit state table. When the MAC layer receives either the acknowledgment (ACK) or the negative acknowledgment (NAK), it clears the retransmission timer. If the MAC layer has not received any acknowledgment (ACK or NAK) when the retransmission timer expires, it retransmits the packet.

3.6.2 Retransmission mechanism at the transit node:

The transit node does not perform any retransmission. It also does not try to distinguish if a received control packet is an original or a retransmission. Each control packet carries a flow ID. A transit node processes a control packet differently based on whether it can find the corresponding flow ID at its transit state table. The flow IDs are very important for the packet retransmission. Whenever there is a flow ID in the transit state table, it means the transit node has already accepted the traffic flow and reserved certain bandwidth for this traffic flow. Without the flow ID, the same bandwidth may be reserved or released multiple times due to retransmission.

3.6.2.1 Reception of a Bandwidth_Request packet

When the transit node receives a Bandwidth_Request packet, it searches for the Flow ID carried in the Bandwidth_Request packet in the transit state table. If the flow ID is not found in the transit state table, it means the node has not reserved any bandwidth for this traffic flow. In this case, the transit node will perform the normal reservation procedure as described in the previous section.

If the flow id in the received Bandwidth_Request packet is found in the transit state

table, the transit node knows it already reserved the bandwidth for this traffic flow, so it will not make another reservation. It just forwards the Bandwidth_Request packet downstream to destination node without any changes locally.

3.6.2.2 Reception of a Release_Reservation packet

If the node finds the flow ID at the transit state table that corresponds to the flow ID carried by the Release_Reservation packet, it will perform the normal reservation release procedure.

If the flow ID is not found in the transit state table, the transit node knows there is no reservation for this traffic flow at this node and there is nothing to release. So the transit node just simply forwards the Release_Reservation packet downstream.

3.6.2.3 Reception of a reply packet

Since all the ACK and NAK packets are end-to-end significant. The transit node should pass these types of packets upstream towards the source. In addition, if NAK for bandwidth reservation request is received, the transit node will release the requested bandwidth only if it found the corresponding flow ID in its transit table.

3.6.3 Retransmission mechanism at the destination node

The destination node does not perform retransmission. It just replies to requests, either requests for reservation or reservation release, by sending back an ACK. The source/destination addresses, source flow ID fields and the bandwidth field in the ACK are directly copied from those in the request packet. The destination node does not generate NAK packets for it does not reserve bandwidth destined to itself.

3.6.4 Retransmission Examples:

When the Bandwidth_Request packet, Release_Reservation packet or the associated reply packets are lost, the source node will perform retransmission. It is very important to make sure that the same bandwidth is not reserved or released multiple times. Our packet retransmission algorithm handles this problem well. The handlings of the lost packets are illustrated in several examples given below.

All the examples are based on a RPR network with 6 nodes. Node 1 tries to set up a reservation from node 1 to node 6. The flow id for this traffic flow is 10, and the requested bandwidth is 10Mbps. We use the following notation to identify the type of control packet and the content in the packet:

(Source flow ID, Request Bandwidth, Type)

Based on the above notation, the Bandwidth_Request packet, Release_Reservation packet, ACK packet and NAK packet can be described as (10, 10M, R), (10, 10M, RR), (10, 10M, A) and (10, 10M, N), repectively.

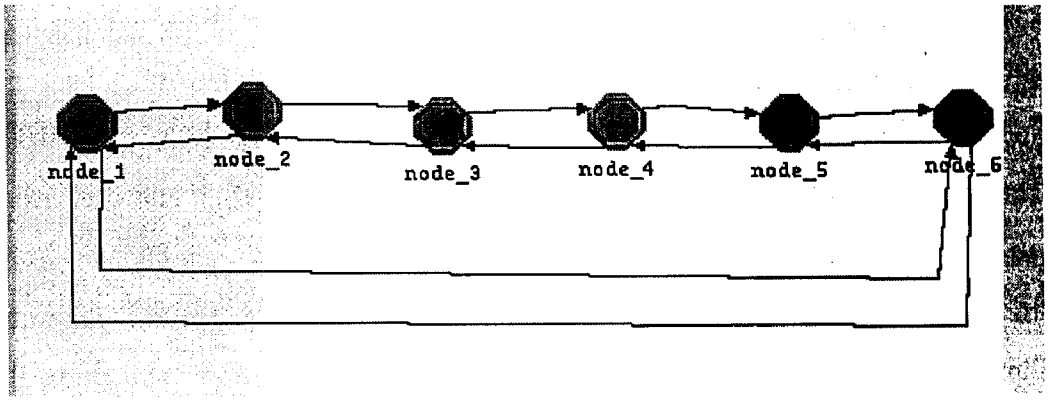


Figure 3.2 RPR Network Setup

Example 1: A Bandwidth_Request packet is lost on the link between node 3 and node 6.

In this case, when node 3 first receives the Bandwidth_Request packet (10, 10M, R), it looks for the flow ID "MAC_{node1} + 10" in the transit state table. Suppose there is no such flow ID found in the table, node 3 will create an entry for the flow ID "MAC_{node1} +10" into the local table, adjusts the B_{AC_3} value with "$B_{AC\_3} = B_{AC\_3}$ - 10M", and forward the Bandwidth_Request packet downstream to node 6. If the request packet is lost on the way from node 3 to node 6, the retransmission timer for this request at node 1 will expire. Node 1 generates another Bandwidth_Request packet (10, 10M, R) to node 6. When node 3 receives this retransmission packet, it finds the flow ID "MAC_{node1} +10" in its transit state table. It then deduces that it has already reserved bandwidth for this traffic flow and it just forwards the packet downstream. The same bandwidth is not doubly reserved in this case. When node 6 receives the retransmission packet, it generates the ACK packet back to node 1.

Example 2: The ACK packet is lost on the link between node 3 and node 1.

In this case, node 3 receives the ACK packet (10, 10M, A) from node 6. Node 3 just forwards the packet upstream to node 1 without any changes locally. If the ACK packet gets

lost on the way from node 3 to node 1, node 1 retransmits the Bandwidth_Request packet (10, 10M, R). When node 3 receives the retransmission packet, it finds the flow ID "MAC_{node1} +10" in its transit state table. Node 3 just forwards the retransmission packet downstream to node 6. When node 6 receives the retransmission packet, it generates another ACK packet (10, 10M, A) back to node 1. The loss of an ACK causes the retransmission of Bandwidth_Request packets. Again, our algorithm prevents repeated reservation of the same flow.

Example 3: The NAK packet is lost on the link between node 3 and node 1.

In this case, node 3 receives the NAK packet (10, 10M, N) from downstream node and it finds the flow ID "MAC_{node1} +10" in its transit state table. Node 3 knows the set up for this traffic flow has been failed. Thus, it removes the entry with flow ID "MAC_{node1} +10" from its transmit state table, adjusts the B_{AC_3} value with "$B_{AC\_3} = B_{AC\_3} + 10M$", and forwards the reply packet upstream to the source node.

The NAK packet gets lost on the way from node 3 to node 1. Node 1 generates another Bandwidth_Request packet (10, 10M, R) to node 6. When node 3 receives this retransmission packet, it searches for the flow ID "MAC_{node1} +10" in its local table and can not find the corresponding flow ID in the table. Thus, node 3 reserves 10Mbps bandwidth for this traffic flow, and forward the request packet downstream to node 6. In this case, node 3 still only reserves the 10Mbps bandwidth for the traffic flow once. Eventually, the NAK packet will be retransmitted and Node 3 releases the bandwidth reservation again.

Example 4: The Release_Reservation packet lost in the links between node 3 and node 6.

Node 1 generates the Release_Reservation packet (10, 10M, RR) to node 6 to release the reservation. When node 3 receives the Release_Reservation packet (10, 10M, RR), it finds the flow ID "MAC_{node1} +10" in its transit state table. Node 3, thus, deletes the entry with flow id "MAC_{node1} +10" in the transit state table, adjusts the B_{AC_3} value with "$B_{AC\_3} = B_{AC\_3} + 10M$", and forwards the Release_Reservation packet downstream to node 6. This packet gets lost on the way from node 3 to node 6. After the retransmission timer for release reservation times out at node 1, node 1 generates another Release_Reservation packet to node 6 carrying the same information (10, 10M, RR). When node 3 receives this retransmission packet, it searches for the flow ID "MAC_{node1} +10" in its transit state table. Because node 3 has deleted the entry with flow ID "MAC_{node1} +10" in the last step, it cannot find it now. Node 3 just

forwards the Release_Reservation packet downstream to node 6 without any changes locally. The 10Mbps bandwidth is not released twice in this case.

From the above examples, one can see that our algorithm works well to deal with the loss of control packets.

3.6.5 Guaranteed Bandwidth Hole problem

In the reservation procedure, a node will use up its guaranteed bandwidth before making reservation of common bandwidth. It is because the guaranteed bandwidth is more restrictive for it cannot be reserved by the other nodes even if it is not used by the node that owns it. Extending this concept, a desirable reservation condition is that a node should not have any common bandwidth reservation at any time if its available guaranteed bandwidth is non-zero. Condition 1 below describes the desirable operating condition:

$$B_{LC\_k} > 0 \text{ only if } B_{AG\_k}=0, \quad \text{(Condition 1)}$$

Where B_{LC_k} is the amount of common bandwidth reserved by the locally sourced flows at node k.

Unfortunately, the Basic Reservation Algorithm sometimes introduces the situation that does not satisfy condition 1. More specifically, situation may arise such that $B_{LC\_k} > 0$ even if $B_{AG\_k} > 0$. This undesirable situation is termed as guaranteed bandwidth hole problem. A simple example is given here to illustrate how the problem arises.

Consider a node k with B_{G_k} of 10 Mbps and B_{C_k} of 50 Mbps. Suppose the node makes two reservations for the locally sourced flows: flow 1 and flow 2. Flow 1 requests a reservation of 8 Mbps first, and later Flow 2 requests a reservation of 7 Mbps. After the two reservations, B_{AG_k} and B_{LC_k} have the values of 0 and 5 Mbps, respectively. Note that Flow 1 did not make any common bandwidth reservation. Now suppose Flow 1 is terminated and since it did not make any reservation from the common-bandwidth pool, the node just releases the bandwidth back to the guaranteed bandwidth pool. The new B_{LC_k} and B_{AG_k} values are 5 Mbps and 8 Mbps, respectively. In other words, the node has some guaranteed bandwidth unused but at the same time occupied bandwidth from the common-bandwidth pool. This obviously violates condition 1. In the extreme case, the node may uses up all the available common bandwidth but leaves its guaranteed bandwidth untouched. This is an undesirable situation for other nodes cannot make reservation of the unused guaranteed bandwidth. In the

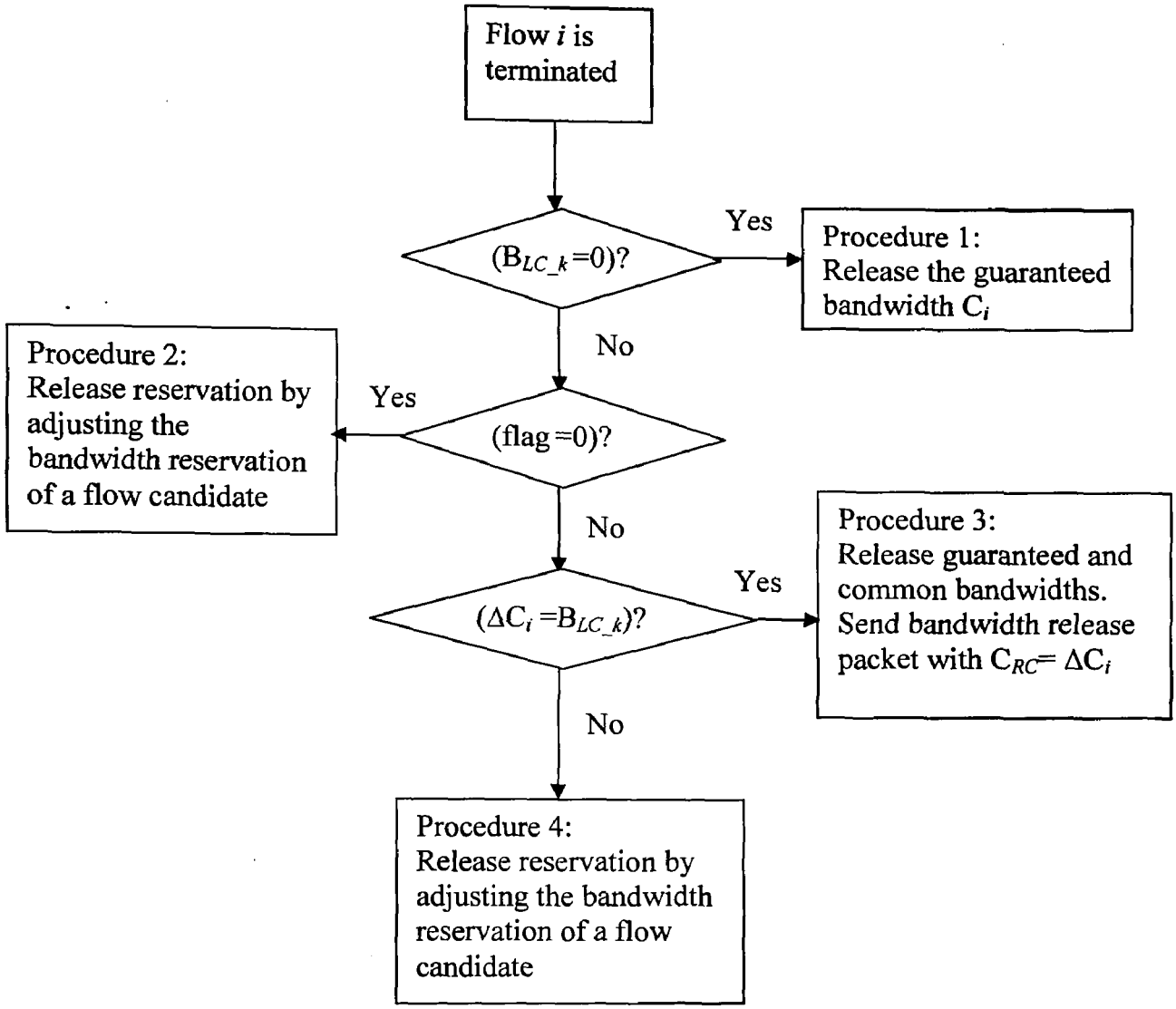
next two sections, two modified reservation algorithms will be presented to deal with the guaranteed bandwidth hole problem.

3.6.6 General Observations for the basic reservation algorithm

Even the basic reservation algorithm may introduce the guaranteed bandwidth hole problem, it is still an attractive algorithm due to its simplicity. Furthermore, the guaranteed bandwidth hole problem does not affect the effectiveness of the reservation for the types of flows that come with large number and require small bandwidth reservation individually. A good example of this type of flows is VoIP connections. This type of flows creates small guaranteed bandwidth hole that only exists in short duration for a hole will be quickly filled up by a new flow request.

3.7 Flow-Based Reservation Algorithm

The flow-based reservation algorithm is used to reduce the effect caused by the guaranteed bandwidth hole problem. The central idea of the algorithm is to adjust the existing flows to "fill up" the guaranteed bandwidth hole left by a newly terminated flow. Following is the flowchart that describes the algorithm performed by the source node.



Flowchart 3.3 Flow-based Reservation Algorithm at the source node

The algorithm consists of four different bandwidth release procedures. Procedure 1 deals with the case where $B_{LC\_k} = 0$. In this case, the source node simply releases the guaranteed bandwidth reserved by the terminated flow (flow *i*). Procedure 3 is another simple case where $\Delta C_i = B_{LC\_k}$. This case implies that all the current common bandwidth reservation was reserved by flow *i*. Thus, the source node simply releases the guaranteed and common bandwidths using the release procedure from the basic reservation scheme.

Procedures 2 and 4 are more complicated. They involve the bandwidth reservation adjustment of the existing flows. To adjust the bandwidth reservation of the existing flows, the source node first calculates how much common bandwidth, C_{RC}, it can release. C_{RC} can be

determined using the following equation:

$$C_{RC} = \min \{B_{LC\_k}, C_i\} \quad (3.11)$$

Once C_{RC} is determined, the source node searches for the flow candidates (the concept of flow candidate will be discussed later) that it can use to adjust the bandwidth reservation. Depending on the availability of the flow candidates and the characteristics of these flows, the amount of common bandwidth released could be less than or equal to C_{RC}. In the rest of this section, the mechanism for bandwidth adjustment is first described. Then the criteria of choosing flow candidates for Procedures 2 and 4 are given.

3.7.1 Reservation Adjustment

Assume that the source node has already identified flow j as the flow candidate for reservation adjustment, it will then check if C_{RC} is less than or greater than or equal to ΔC_j. If C_{RC} is greater than or equal to ΔC_j, the source node simply releases the common bandwidth of flow j downstream and updates the corresponding entry of flow j in the transmit state table. Specifically, ΔC_j and flag are both set up zero. If C_{RC} is less than ΔC_j, the source node will first make another reservation for flow j with different flow ID and the common bandwidth reservation of ΔC_j', where

$$\Delta C_j' = \Delta C_j - C_{RC}. \quad (3.12)$$

If the new reservation is successful, the source node will tear down the old reservation for flow j. The amount of common bandwidth released is the difference between ΔC_j' and ΔC_j, which is C_{RC}. Also note that if the amount of bandwidth released is less than C_{RC}, the adjustment procedure can be repeated for other flow candidates. The adjustment process will stop if the total amount of released bandwidth is equal to C_{RC} or if no more flow candidate is available.

3.7.2 Identification of Flow Candidate

The identification of flow candidate for reservation adjustment depends on the release procedure used.

First let consider Procedure 2. Procedure 2 is performed under the situation where the terminated flow did not make any common bandwidth reservation. In this case, the source node can choose any other flow sourced by the node to adjust the reservation. To maximize

the reservation efficiency, the source node should choose the flow with the largest span (i.e., largest number of hops). It is because by releasing common bandwidth reservation of the flow with the largest span, more nodes will release the common bandwidth, thus, more bandwidth is available for other traffic flows.

Bandwidth release in Procedure 4 is performed under the condition that the common bandwidth reservation by the terminated flow is greater than zero but less than B_{LC_k}. In this case, the selection of flow candidate is more restrictive. In this case, the candidate must have the same destination. If it is not, the mechanism may release common bandwidth over more number of hops than it should in the case where the candidate has a longer span than the terminated flow, and lesser number of hops in the case where the candidate has a shorter span.

There are other useful guidelines to select the flow candidates. One obvious guideline is to select the candidate with the reserved common bandwidth equal to C_{RC}. This selection will avoid the overhead of establishing a new reservation. Another guideline is to choose the flow with the largest bandwidth. This choice has the potential to reduce the number of flows that need to be adjusted.

3.7.3 Examples

To demonstrate the Flow-Based Reservation Algorithm, we use the RPR network with six nodes presented in Figure 3.2.

In this network, we assign 10Mbps guaranteed bandwidth and define 50Mbps common-pool bandwidth at each node. We have five traffic flows in this network. The traffic flow parameters are described in table 3.1. In the following sections, we will study several cases about the reservation release.

	Start time	Source	Destination	Equivalent Bandwidth (Mbps)	B_{AG_2}, B_{AC_2} and B_{LC_2} values after the flow is admitted		
					B_{AG_2} (Mbps)	B_{AC_2} (Mbps)	B_{LC_2} (Mbps)
Flow 1	0.1s	Node2	Node 4	5	5	50	0
Flow 2	0.2s	Node2	Node 4	6	0	49	1
Flow 3	0.3s	Node 2	Node 5	10	0	39	11
Flow 4	0.4s	Node 1	Node 6	10	0	29	11
Flow 5	0.5s	Node 2	Node 4	8	0	21	19

Table 3.1 Traffic flow parameters used for Flow-base Reservation Algorithm examples

3.7.3.1 Reservation release example for Procedure 2:

First let us consider the example for Procedure 2 where the terminated flow did not make any common bandwidth reservation.

At 0.6s, flow 1 is terminated at node 2. Node 2 will go through the following steps to adjust the reservation. First, node 2 calculates the C_{RC} value using equation (3.11):

$C_{RC} = \min \{5, 19\} = 5$ Mbps

Next, node 2 will identify the flow candidate. As flow 1 did not make any common bandwidth reservation, node 2 will choose an existing flow sourced by node 2 with the largest span to adjust the bandwidth reservation. In this example, node 2 will choose flow 3 to adjust the reservation. In the adjustment procedure, node 2 makes another reservation for flow 3 using a different flow ID with the common bandwidth reservation of 5 Mbps (10 Mbps – 5 Mbps). After the new reservation has been set up, node 2 tears down the old reservation for flow 3. After the old reservation has been released, the B_{AC_2} value becomes 26 Mbps and B_{LC_2} value becomes 14 Mbps.

Compare to the B_{AC_2} and B_{LC_2} values before flow 1 was terminated, we can see the equivalent bandwidth of the terminated flow has been successfully released without the guaranteed bandwidth hole problem.

3.7.3.2 Reservation release example for Procedure 4:

Next, we consider the example for the Procedure 4 where the common bandwidth

reservation by the terminated flow is greater than zero but less than B_{LC_k}.

At 0.6s, flow 2 is terminated at node 2. Node 2 will go through the adjustment procedure to release the common bandwidth reservation. First, node 2 calculates the C_{RC} value using equation (3.11):

$C_{RC} = \min \{6, 19\} \approx 6$ Mbps

Next, node 2 will identify the flow candidate. Based on the selection guidelines, node 2 will select the flow candidate that has the same source and destination as the terminated flow and has the largest common bandwidth reservation (node 2 cannot find a candidate with the common bandwidth reservation same as C_{RC}). In this example, flow 2 will select flow 5 to adjust the bandwidth reservation. To do that, node 2 makes a new reservation for flow 5 with a common bandwidth reservation of 2 Mbps. After the new reservation has been set up, node 2 tears down the old reservation for flow 5. After the old reservation for flow 5 has been released, we have $B_{AG\_2} = 0$, $B_{AC\_2} = 27$ Mbps and $B_{LC\_2} = 13$ Mbps. Again, by compare with the B_{AC_2} and B_{LC_2} values before flow 2 was terminated, we can see that the equivalent bandwidth of the terminated flow has been successfully released without the guaranteed bandwidth hole problem.

3.7.4 General observations of using Flow-Based Reservation Algorithm

The Flow-Bsed Reservation Algorithm is more complicated than the Basic Reservation Algorithm, but it alleviates the guaranteed bandwidth hole problem. The Basic Reservation Algorithm may be used in the situation where lots of small bandwidth connections come and go fast. In this situation the bandwidth hole is small and filled up quickly. Thus, there is no need to make the reservation algorithm more complicated. However, in the situation where there are a number of large-bandwidth flows with long connection life times, the Flow-Based Reservation Algorithm may be more suitable to use. It is because the guaranteed bandwidth hole is larger and lasts longer in this case. The Flow-Based Reservation Algorithm will help to eliminate or reduce the size of the hole.

Note that the Flow-Based Reservation Algorithm cannot completely eliminate the guaranteed bandwidth hole problem when the MAC layer of the source node cannot find the suitable flow candidate. To completely eliminate the guaranteed bandwidth hole problem, a Hop-by-hop Reservation Algorithm is introduced in the next section.

3.8 Hop-by-Hop Reservation Algorithm

The Flow-Based reservation algorithm only solves the guaranteed bandwidth hole problem partially. Since the reservation is released per flow, not per hop. Here, a hop is defined as a link between two adjacent nodes. The address of the hop is the address of the outgoing interface of the transmitted node. When a reservation is reserved or released, every hop along the path from the source to the destination makes the same amount of bandwidth reservation or release. This forces the flow-based algorithm, in some cases, to only select the flows that have the same source/destination as the terminated flow for bandwidth adjustment. If no such flow exists, the guaranteed bandwidth hole cannot be filled.

The per-flow reservation also leads to an inefficient bandwidth allocation. The following example illustrates the inefficiency. Let consider a RPR network with 6 nodes as shown in figure 3.2.

A guaranteed bandwidth of 5 Mbps and a common bandwidth of 50 Mbps are assigned to each node. Suppose node 1 generates two traffic flows; flow 1 and flow 2. Flow 1 requests a bandwidth of 5 Mbps; while flow 2 requests a bandwidth of another 5 Mbps. Destinations of flow 1 and flow 2 are node 4 and node 5, respectively. Node 1 made a reservation for flow 1 first. After the reservation, $B_{AG\_1} = 0$. Later node 1 made another reservation for flow 2. This reservation causes nodes 1, 2, 3, and 4 to reserve 5 Mbps of bandwidth from the common-bandwidth pool. Note that there are 5 Mbps of guaranteed bandwidth owned by node 1 at the hop between node 4 and node 5 that is not used. In fact, the guaranteed bandwidth at this hop will not be available for any other flows as long as the reservation of flow 1 is not released. This situation is not desirable for the reason that if the guaranteed bandwidth of that hop was available for flow 2, then flow 2 did not need to make reservation from the common-bandwidth pool at that hop, which, in turn, allowed more common bandwidth at that hop to be available to other flows. Clearly, a hop-by-hop reservation approach will provide a more flexible and efficient reservation mechanism.

3.8.1 Introduction to the Hop-by-Hop Reservation Algorithm:

In the hop-by-hop reservation algorithm, every node maintains a bandwidth usage table to record the total bandwidth reserved on every hop for the flows sourced from this node.

The bandwidth usage table is used for the local transmit traffic only, not for the transit traffic. For the N nodes network, the total number of entries in the table is N-1.

When the source node receives a new traffic flow request, it checks the bandwidth usage table to decide how much common bandwidth it should request on every single hop towards the destination node for the new flow. The requested bandwidth at each hop may not be the same. The bandwidth reservation is done hop-by-hop. Using this approach, node 1 in the example from the previous section did not need to request the 5Mbps common bandwidth reservation on the link between node 4 and node 5 for flow 2. Because the bandwidth usage on the link between node 4 and node 5 in the bandwidth usage table would be zero after the reservation of flow 1, thus, flow 2 could use the guaranteed bandwidth on the link between node 4 and node 5 instead.

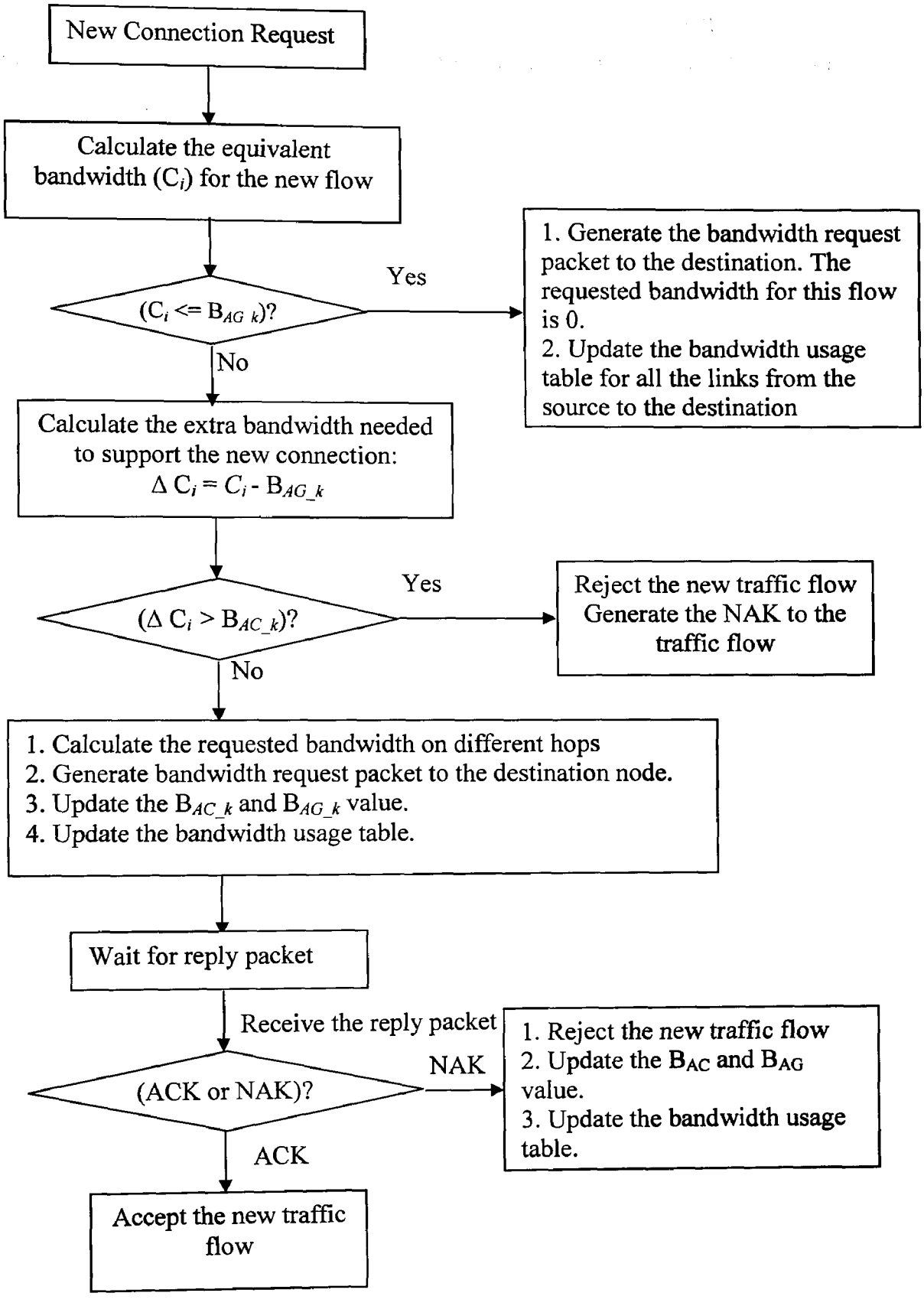
To reserve different bandwidth on different hops, we need to add a node address field and the bandwidth field for each hop between the source and the destination in the bandwidth request packet. The node address field identifies the hop (The output interface of the node) and the corresponding bandwidth field indicates the requested bandwidth to be reserved or released on that hop. When the source node generates the bandwidth request packet, it puts the different requested bandwidth values on different bandwidth fields. When the transit node receives the bandwidth request packet, it looks into the bandwidth fields and gets the requested bandwidth for its outgoing link. The transit node may admit or reject the new traffic flow by comparing the requested bandwidth with the available common bandwidth of that link.

When the traffic flow is terminated, the source node needs to release the reservation for this flow downstream and adjust the bandwidth reservation for other flows on every hop. We cannot just simply release the same bandwidth value on all the hops downstream to the destination because the source node may reserve different bandwidths on different hops before. To properly release the bandwidth on each hop, the source node should calculates released bandwidth for each hop using the algorithm described in later sections. The source node puts different released bandwidth values on different bandwidth fields in the Release_Reservation packet and sent the bandwidth release packet downstream to the destination. When the transit node receives the bandwidth release packet, it gets the released bandwidth value for its outgoing link and adjusts its available common bandwidth

accordingly.

3.8.2 Reservation setup process in the Hop-by-Hop reservation algorithm

The following flowchart summarizes the Hop-by-Hop Reservation Algorithm performed at the source node.



Flowchart 3.4: Hop-by-hop Reservation Algorithm at the source node

Let us define BU(n) and RH(n) as the bandwidth usage on hop n and the request common bandwidth for reservation/release on hop n. The bandwidth usage table holds the values of BU(n) for n = 1, 2, … , N-1. When the MAC layer of the source node receives a new traffic flow connection request from its client, it goes through the following steps:

1) The MAC layer calculates the equivalent bandwidth for the new flow C_i using the equation (2.6).
2) If C_i is less than B_{AG_k}, MAC layer generates the Bandwidth_Request packet to the destination node. The requested bandwidth for this flow is 0. The algorithm sets up the reservation for every flow including the flows using the guaranteed bandwidth. Recall that the source node does not make the reservation for the flows using guaranteed bandwidth in Basic Reservation Algorithm and Flow-Based Reservation Algorithm, because the source node may not need to release the reservations for the terminated flows in these two algorithms. For Hop-by-hop Reservation Algorithm, it is more efficient for the source node to release every terminated flow even if the flow did not make any common-bandwidth reservation. We will talk about the reservation release process later. MAC layer updates the bandwidth usage table for all the links from the source to the destination using the equation (3.13):

   $$BU(n) = BU(n) + C_i \qquad (3.13)$$

   The MAC layer also updates the B_{AG_k} value using the equation (3.1). If C_i is greater than B_{AG_k}, go to step (3).
3) Calculate the extra common bandwidth needed (ΔC_i) to support the new traffic flow by using equation (3.6). If the ΔC_i value is greater than the B_{AC_k} value, MAC layer rejects the connection request for the coming traffic flow and generates the NAK message to the traffic source. If the ΔC_i value is less than B_{AC_k}, go to step (4).
4) In this step, MAC layer generates Bandwidth_Request packets to set up the reservations. MAC layer calculates the requested bandwidth values on different links according to entries in the bandwidth usage table. If the bandwidth usage value on of hop n is grater than the guaranteed bandwidth of the source node, the requested bandwidth for hop n is the same as C_i. If the bandwidth usage value of given hop n is less than the guaranteed bandwidth for the source node, MAC layer calculates the requested bandwidth on this hop using the equation (3.14):

$$RH(n) = C_i - (B_{G\_k} - BU(n)) \quad (3.14)$$

After the MAC layer calculates all the requested bandwidth on all the links, it generates the bandwidth request packet to the destination node. The bandwidth request packet carries different requested bandwidth values in different requested bandwidth fields.

After generating the bandwidth request packet, MAC layer updates the B_{AG_k} and B_{AC_k} values by using equations (3.1) and (3.3). MAC layer also updates the bandwidth usage table for all the links from the source to the destination using the equation (3.13). Then MAC layer waits for the reply packets for the bandwidth request packets.

5) When the source node receives the reply packet, it checks the reply packet type. If the packet is a NAK packet, the MAC layer rejects this traffic flow and adjusts the B_{AG_k} and B_{AC_k} value using equations (3.5) and (3.6). The MAC layer also updates the bandwidth usage table for all the hops from the source to the destination using the equation (3.15):

$$BU(n) = BU(n) - C_i \quad (3.15)$$

If the packet is an ACK packet, the MAC accepts the traffic flow.

The reservation mechanism at the transit node and the destination node in the Hop-by-Hop Reservation algorithm is the same as the previous two algorithms.

3.8.3 Reservation release process in the Hop-by-Hop Reservation Algorithm

When a flow is terminated, the MAC layer of the source node starts the reservation release process. The MAC layer calculates the released bandwidth on every single link downstream to the destination using the following algorithm:

(1) If the bandwidth usage value on a given link n is less than the guaranteed bandwidth, the released bandwidth is zero on this link.

(2) If the bandwidth usage value on a given hop n is greater than the guaranteed bandwidth, MAC layer calculates the difference ($DC(n)$) between the bandwidth usage value and the guaranteed bandwidth using equation (3.16):

$$DC(n) = BU(n) - B_{G\_k}. \quad (3.16)$$

The MAC layer then calculates the released bandwidth on hop n ($C_{RC}(n)$) using (3.17):

$$C_{RC}(n) = \min \{DC(n), C_i\} \quad (3.17)$$

After the source node sends the bandwidth release packet carrying the released bandwidths on different hops downstream to the destination node. The source node updates the bandwidth usage table using equation (3.18):

$$BU(n) = BU(n) - C_i \quad (3.18)$$

When the transit node receives the bandwidth release packet, it looks into the packet and obtains the released bandwidth for its outgoing link. Then the transit node modifies the available common pool size using the same mechanism we described in the previous two algorithms.

3.8.4 General Observations

The Hop-by-hop Reservation Algorithm can completely eliminate the guaranteed bandwidth hole problem. But it is more complicated to implement. The source node needs to maintain an extra bandwidth usage table and make a number of calculations for each reservation or release request. Furthermore, the Bandwidth_Request or Release_Reservation packet for a flow can be quite large if the number of hops for that flow is large. On the other hand, the Hop-by-Hop Reservation Algorithm provides the most advantage in a situation where there are only few connections and each connection consumes a large amount of bandwidth. In addition, this algorithm does not require the search of flow candidates and the bandwidth adjustment of the flow candidates, unlike the Flow-Based Reservation Algorithm.

3.9 Simulation Results

This section presents some simulation results that illustrate the performance of the reservation algorithms proposed in this chapter. All the simulation uses the 6-node topology shown in Figure 3.2. All the traffic flows go through the outer ring. The total bandwidth reserved for class A traffic is 100Mbps. We define the guaranteed bandwidth to each node as 10Mbps and put 50Mbps bandwidth into the common bandwidth pool. So each node can use maximum 60Mbps bandwidth for its classA traffic. All the traffic sources are the on-off sources.

Case 3.1: Dynamic Bandwidth Allocation

This case demonstrates how the dynamic bandwidth allocation method admits more class A traffic. The result in this case also demonstrates that the delay for class A traffic is quite small with proper admission control even the RPR network is congested with class C traffic.

In this case, we set up 5 class A traffic flows from node1 to node 4 and one class A traffic flow for each of the following node pairs: node 6-> node 5, node 5 -> node 4, node 4 -> node 3 , node 3 -> node 2 and node 2 -> node 1. We also generate 600 Mbps class C traffic flows from node 2 to node1. The traffic parameters of the Class A traffic flows are shown in the following table:

	Start Time	***Source***	***Destination***	***Peak Rate***	***Burst Period***	***Utilization***	***Equivalent Bandwidth***
Flow 1	0.10s	Node 1	Node 4	30Mbps	0.004s	0.4	16.58Mbps
Flow 2	0.12s	Node 1	Node 4	20Mbps	0.006s	0.6	14.70Mbps
Flow 3	0.14s	Node 1	Node 4	20Mbps	0.006s	0.6	14.70Mbps
Flow 4	0.16s	Node 1	Node 4	20Mbps	0.006s	0.5	12.99Mbps
Flow 5	0.18s	Node 1	Node 4	30Mbps	0.004s	0.5	19.48Mbps
Flow 6	0.10s	Node 6	Node 5	15Mbps	0.008s	0.5	9.74Mbps
Flow 7	0.10s	Node 5	Node 4	15Mbps	0.008s	0.5	9.74Mbps
Flow 8	0.10s	Node 4	Node 3	15Mbps	0.008s	0.5	9.74Mbps
Flow 9	0.10s	Node 3	Node 2	15Mbps	0.008s	0.5	9.74Mbps
Flow10	0.10s	Node 2	Node 1	15Mbps	0.008s	0.5	9.74Mbps

Table 3.2 Traffic parameters for classA traffic flows in simulation case 3.1

The traffic parameters for the Class C traffic flow from node 2 to node 1 are shown in the following table:

State Time	***Peak Rate***	***Burst Period***	***Utilization***
0.05s	1200Mbps	0.001s	0.5

Table 3.3 Class C traffic parameters used for simulation case 3.1

At node 1, the equivalent bandwidth for the traffic flows 1-4 together is 58.99 Mbps. So these 4 traffic flows are admitted. The traffic flow 5 at node 1 is rejected because the requested bandwidth for flow 5 is more than the available common-pool bandwidth. The other 5 classA traffic flows sourced from the rest of the 5 nodes are all admitted because the equivalent bandwidth for each flow is 9.74Mbps which is less than the guaranteed bandwidth. In this case, the equivalent bandwidth of the total flows for the hops between node1 and node4 is almost 100Mbps that is the maximum reserved bandwidth for classA traffic, and there is 600Mbps classC traffic from node2 to node1. Figure 3.3 shows the simulation results of classA transmit queue size at node 1 and the end-to-end delay from node 1 to node 4. It can be seen that the maximum end-to-end delay for classA traffic is only about 5 ms, even though the network is congested with classC traffic.

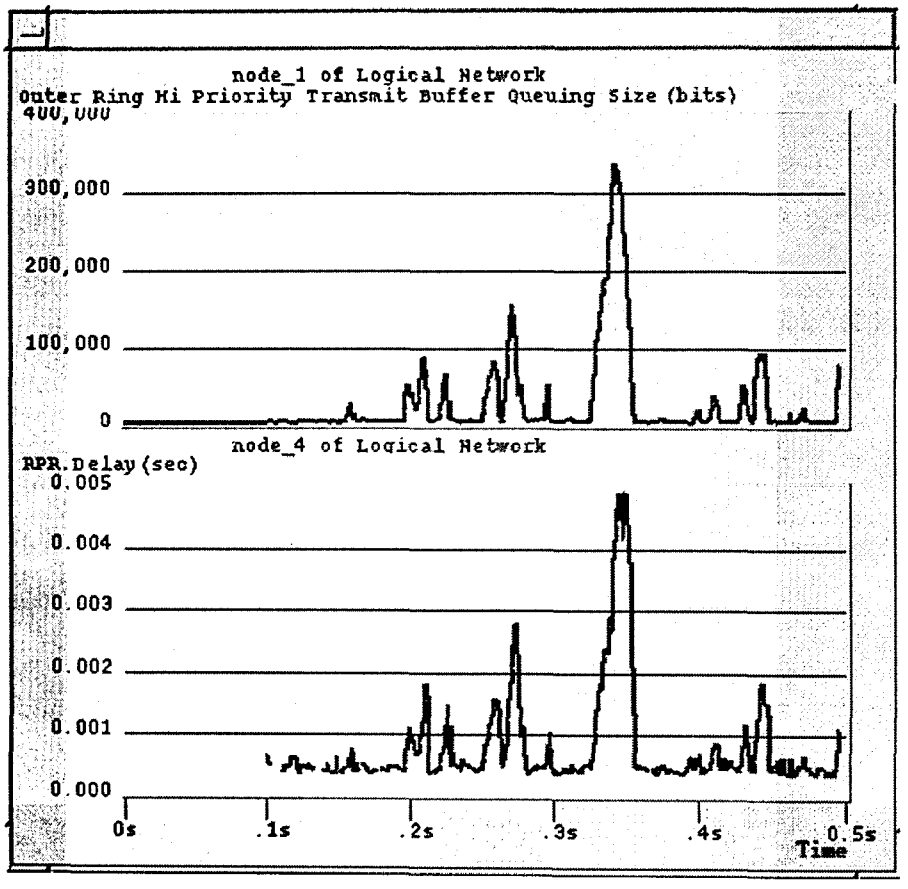


Figure 3.3: Delay performance with dynamic bandwidth allocation method

Figure 3.4 shows the high priority transit queue size in nodes 3 and 4.The maximum queue size is equivalent to one packet. This illustrates that the major contribution of the end-

to-end delay for classA traffic is at the source node. The delay experienced by a classA packet at the transit node is negligible as observed in the previous chapter.

This case also demonstrates the effectiveness of using dynamic bandwidth allocation method. By dynamically allocating bandwidths for different flows, node 1 can send up to 60 Mbps classA traffic while 10 Mbps of bandwidth can still be allocated to each of the other 5 nodes. Without the dynamic bandwidth allocation method, only 20 Mbps bandwidth could be statically assigned to each node (assume bandwidth is uniformly assigned) and node 1 would have to reject all the flows, except flow 1. The method can also allow the support of more connections in the network. For instance, if every connection requires 1Mbps bandwidth and we statistically assign 20Mbps bandwidth to each node, the maximum number of the connections that can be supported is 20*6 = 120. If we use dynamic bandwidth allocation method, we can support maximum 60*6 = 360 connections in the network in the case where all the connections are between adjacent nodes.

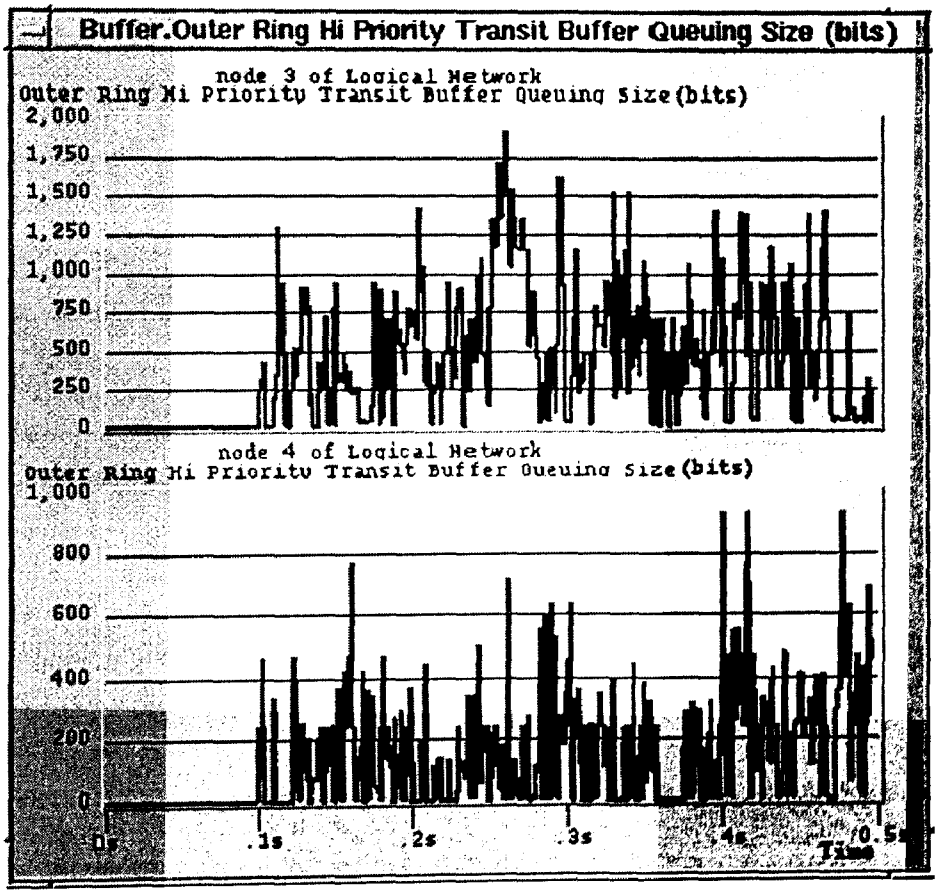


Figure 3.4 High priority transit buffer sizes in transit nodes

Case 3.2: Bandwidth reservation and release process using Basic Reservation Algorithm

In this case, the bandwidth reservation and release process using the Basic Reservation Algorithm is simulated. Four classA traffic flows with the following traffic parameters are used in the simulation:

	Start Time	*Stop Time*	*Source*	*Destination*	*Peak Rate*	*Burst Period*	*Utilization*	*Equivalent Bandwidth*
Flow 1	0.10s	0.50s	Node 1	Node 4	30Mbps	0.004s	0.4	16.58Mbps
Flow 2	0.12s	0.45s	Node 1	Node 5	20Mbps	0.006s	0.6	14.70Mbps
Flow 3	0.14s	0.40s	Node 1	Node 4	20Mbps	0.006s	0.6	14.70Mbps
Flow 4	0.16s	0.35s	Node 1	Node 4	20Mbps	0.006s	0.5	12.99Mbps

Table 3.4 Traffic parameters for classA traffic flows in simulation case 3.2

Figure 3.5 shows the simulation result for available common bandwidth at node2. The simulation results for the available common bandwidth at node 1 and node 3 are the same. The results demonstrate that the Basic Reservation Algorithm handles the bandwidth reservation and release process correctly.

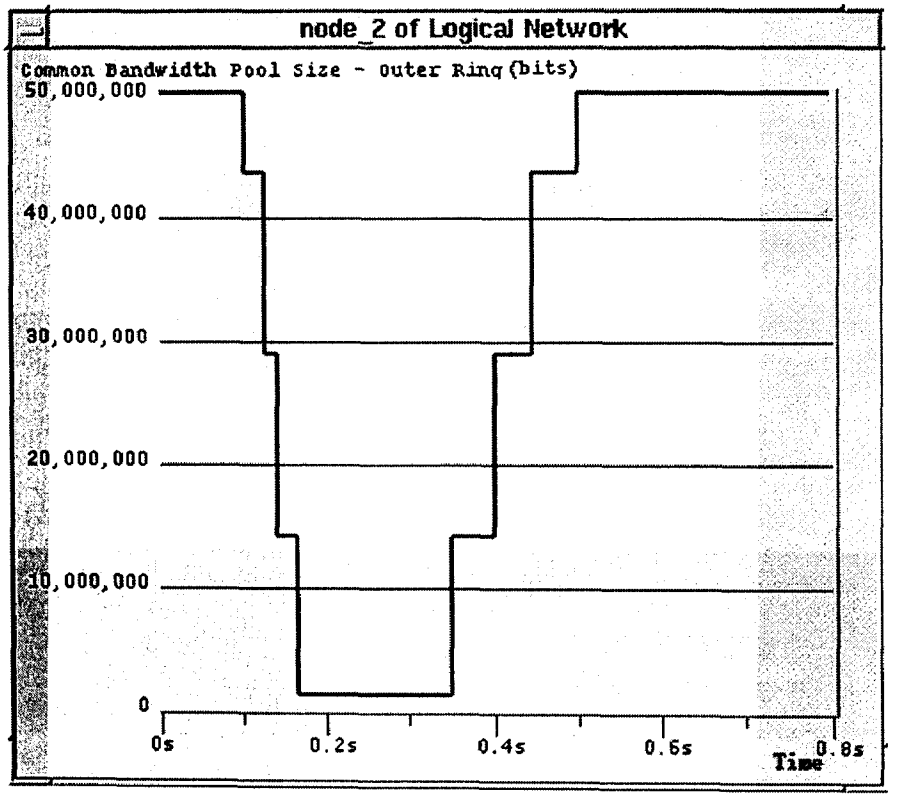


Figure 3.5: Available common bandwidth pool size with Basic Reservation Algorithm

Case 3.3: Guaranteed Bandwidth hole problem of the Basic Reservation Algorithm

This case illustrates the guaranteed bandwidth hole problem. Four class A traffic flows with the following traffic parameters are used in this simulation:

	Start Time	*Stop Time*	*Source*	*Destination*	*Peak Rate*	*Burst Period*	*Utilization*	*Equivalent Bandwidth*
Flow 1	0.10s	0.34s	Node 1	Node 4	10Mbps	0.004s	0.4	5.52Mbps
Flow 2	0.12s	-	Node 1	Node 5	20Mbps	0.006s	0.6	14.70Mbps
Flow 3	0.14s	-	Node 1	Node 4	20Mbps	0.006s	0.6	14.70Mbps
Flow 4	0.16s	-	Node 1	Node 4	20Mbps	0.006s	0.5	12.99Mbps

Table 3.5 traffic parameters used for classA traffic flows in simulation case 3.3

Figure 3.6 shows the simulation result for the available common bandwidth at node 2. The simulation results for the available common bandwidth at node 1 and node 3 are the same.

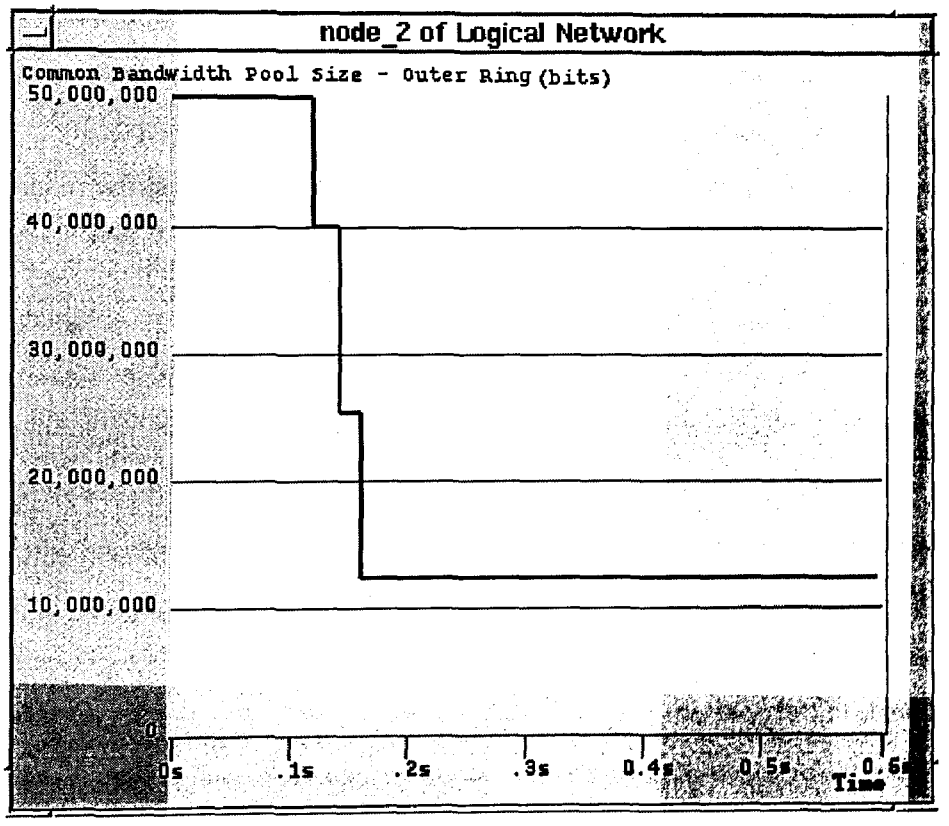


Figure 3.6: Guaranteed bandwidth hole problem with Basic Reservation Algorithm

When node 1 made a reservation for flow 1 (at 0.1s), it accepts this traffic flow without

making common bandwidth reservation; instead, this flow only uses the guaranteed bandwidth. Thus, the available common bandwidth remains to be 50 Mbps. After the successive reservations for flows 2, 3 and 4, the available common bandwidth is reduced to12Mbps. At 0.34s, flow 1 is terminated, but node 1 does not release the bandwidth for this traffic flow because this flow only used guaranteed bandwidth, and the available common bandwidth remains to be 12 Mbps. Consequently, bandwidth of 5.52 Mbps is not available for reservation to class A traffic flows from nodes other than node 1. In the next case, the Flow-Based Reservation Algorithm is used to deal with the problem.

Case 3.4: Flow-Based Reservation Algorithm

In this case, we try to solve the bandwidth hole problem observed in the case 3.3. Figure 3.6 shows the simulation results for the available bandwidth at node 2 using the Flow-based Reservation Algorithm. The simulation results for the available common bandwidth at node 1 and node 3 are the same.

In this case, after flow 1 was terminated, node 1 started to adjust the bandwidth reservation using the Flow-Based Reservation Algorithm. Because flow 1 did not make any common bandwidth reservation, node 1 chose the existing flow sourced by node 1 with the largest span. Node 1 used the traffic flow 2 to adjust the reservation.

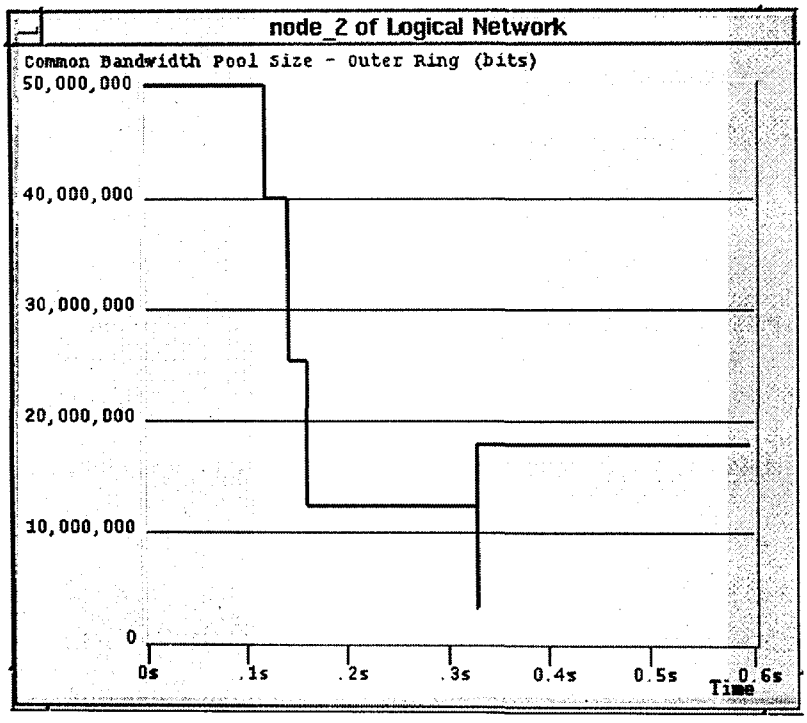


Figure 3.7: Available common bandwidth size with Flow-based Reservation Algorithm

Firstly, node 1 set up a new bandwidth reservation downstream to node 4 asking for 14.70-5.52=9.18 Mbps of bandwidth. After the new reservation has been successfully set up, the B_{AC_k} values at node 1, 2 and 3 are 2.66 Mbps. Then node 1 started to release the bandwidth used by flow 2 downstream. After the bandwidth is successfully released, the B_{AC_k} values at node 1, 2 and 3 were 17.5 Mbps. The result shows that the bandwidth reserved by flow 1 (5.52 Mbps) is properly released by the Flow-based Reservation Algorithm and the guaranteed bandwidth hole problem is avoided.

Case 3.5: Inefficient bandwidth allocation problem

This case illustrates the inefficient bandwidth allocation problem observed in the Basic Reservation and Flow-Based Reservation Algorithms. In the simulation, node 1 generates several traffic flows to node 4 and node 5. The traffic parameters are shown in the following table:

	Start Time	***Destination***	***Peak Rate***	***Burst Period***	***Utilization***	***Equivalent Bandwidth***
Flow 1	0.10s	Node 4	30Mbps	0.004s	0.4	16.58Mbps
Flow 2	0.12s	Node 5	20Mbps	0.006s	0.6	14.70Mbps
Flow 3	0.14s	Node 4	20Mbps	0.006s	0.6	14.70Mbps
Flow 4	0.16s	Node 4	20Mbps	0.006s	0.5	12.99Mbps

Table 3.6 traffic parameters for classA traffic flows in simulation case 3.5

Figure 3.8 shows the simulation results for the available common bandwidth at node 2 and node 4. At 0.1s, node 1 sets up 6.58Mbps bandwidth reservation for flow 1. After 0.1s, the common bandwidth pool size at node 2 is 43.4Mbps, and the available guaranteed bandwidth at node 1 becomes zero. At 0.12s, node 1 sets up 14.7Mbps bandwidth reservation for flow 2. After 0.12s, the available common bandwidth at node 2 and node 4 are 29.7Mbps and 35.3Mbps, respectively. Node 4 has reserved 14.7Mbps bandwidth for flow 2, even though 10 Mbps of guaranteed bandwidth owned by node 1 remain used. If that 10 Mbps of bandwidth could be utilized, node 4 would have 10 Mbps more of available common bandwidth that can be used to support other traffic flows. In the next case, the Hop-by-Hop Reservation Algorithm is applied for a more efficient use of the guaranteed bandwidth.

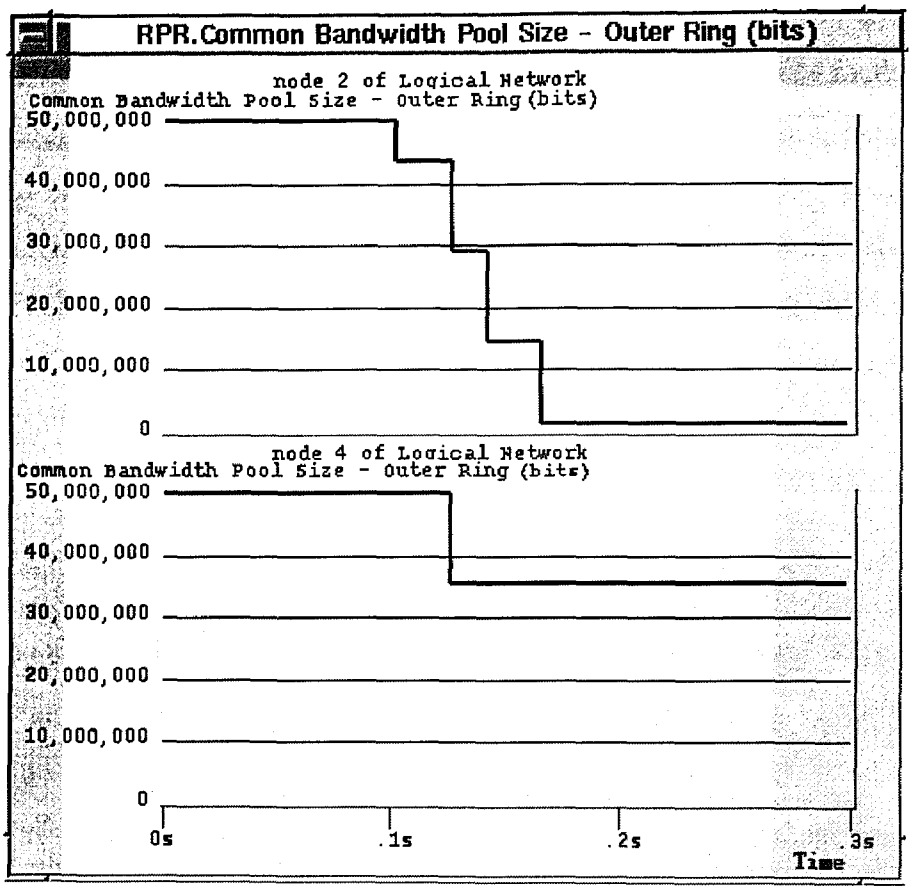


Figure 3.8: Inefficient bandwidth allocation problem with Basic Reservation Algorithm

Case 3.6: Solving the inefficient bandwidth allocation problem by using Hop-by-hop Reservation Algorithm

In this case, the simulation has the same setup as in case 3.5, except that the Hop-by-Hop Reservation Algorithm is used. Figure 3.9 shows the simulation results. Based on the Hop-by-Hop Reservation Algorithm, node 1 only reserves 4.7 Mbps of common bandwidth on the link from node 4 to node 5. Thus, the available bandwidth at node 4 is 10 Mbps larger than that in case 3.5.

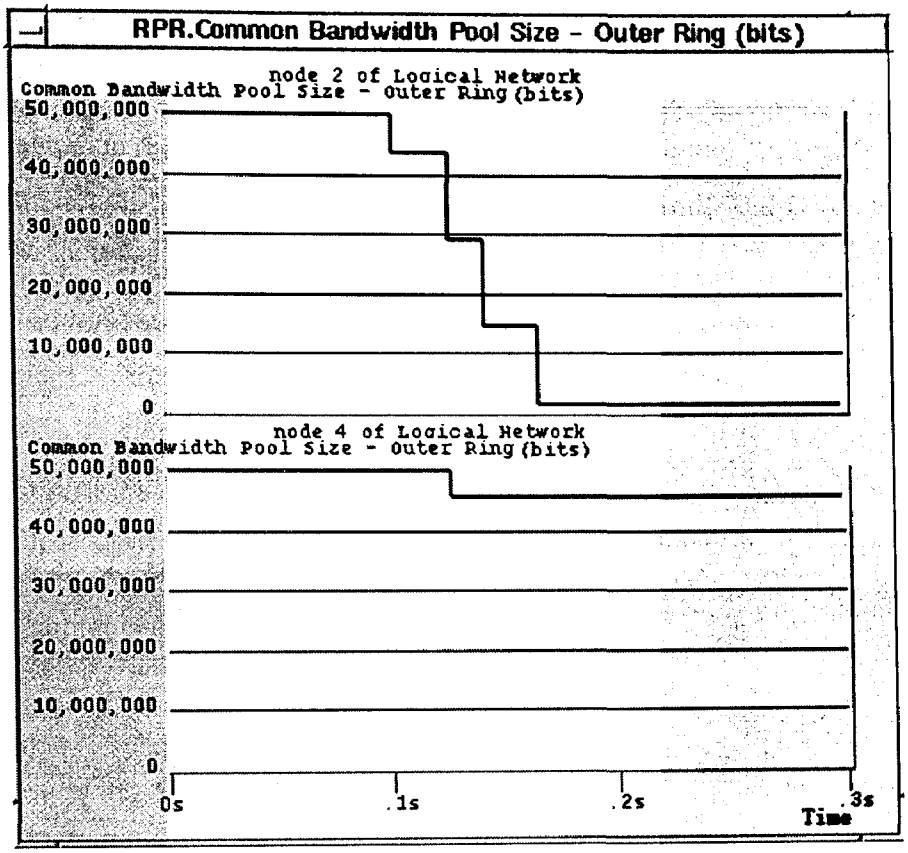


Figure 3.9: Inefficient bandwidth allocation problem solved by Hop-by-hop Reservation Algorithm

Case 3.7: Bandwidth reservation and release using the hop-by-hop reservation algorithm

In this case, 7 traffic flows are generated. The traffic parameters for the traffic flows are shown in the following table:

	Start Time	*Stop Time*	*Source*	*Destination*	*Peak Rate*	*Burst Period*	*Utilization*	*Equivalent Bandwidth*
Flow 1	0.10s	0.50s	Node 1	Node 4	30Mbps	0.004s	0.4	16.58Mbps
Flow 2	0.12s	0.45s	Node 1	Node 5	20Mbps	0.006s	0.6	14.70Mbps
Flow 3	0.14s	0.40s	Node 1	Node 4	20Mbps	0.006s	0.6	14.70Mbps
Flow 4	0.16s	0.35s	Node 1	Node 4	20Mbps	0.006s	0.5	12.99Mbps
Flow 5	0.10s	0.50s	Node 2	Node 1	15Mbps	0.008s	0.5	9.74Mbps
Flow 6	0.36s	0.60s	Node 3	Node 6	20Mbps	0.006s	0.5	12.99Mbps
Flow 7	0.20s	0.55s	Node 3	Node 6	15Mbps	0.008s	0.5	9.74Mbps

Table 3.7 Traffic parameters for classA traffic flows in simulation case 3.7

Figures 3.10 and 3.11 show the simulation results of the available common bandwidth at nodes 2, 3, 4 and 5. Using these figures, the reservations setup and release using the hop-by-hop reservation algorithm will be described in detail below.

1. At 0.1s, node 2 makes a reservation for flow 5. Since node 2 uses the guaranteed bandwidth for the reservation, it does not affect the available common bandwidth.
2. Between 0.1 s and 0.16s, nodes 2, 3 and 4 has reserved the common bandwidth for flows 1-4. After the reservation setup, the available common bandwidth at node 2 and node 3 almost becomes zero; while node 4 only reserved 4.7 Mbps bandwidth for flow 2. It is because only flow 2 actually uses the link between node 4 and node 5. There is no reservation at node 5 for flows 1-4.
3. At 0.2s, a reservation for flow 7 is made. Node 3 uses its guaranteed bandwidth to support the reservation.
4. At 0.35s, flow 4 is terminated. Both nodes 1, 2 and 3 release 12.99Mbps of common bandwidth.
5. At 0.36s, node 3 generated flow 6. Then node 3, 4 and 5 reserve 12.99Mbps bandwidth for flow 6.
6. At 0.40s, flow 3 is terminated. Node 1, node 2 and node 3 release 14.70Mbps of common bandwidth.
7. At 0.45s flow 2 is terminated. Node 1, node 2 and node 3 release 14.70Mbps of common bandwidth; while node 4 release 4.70Mbps of common bandwidth.
8. At 0.50s flow 1 and flow 5 are terminated. After the release of flow 1, Node 1, node 2 and node 3 release 6.58Mbps of common bandwidth. In the case of flow 5, since node 2 does not make any common-bandwidth reservation, no release of common bandwidth occurs.
9. At 0.55s flow 7 is terminated. Node 3, node 4 and node 5 release 9.74Mbps of common bandwidth.
10. At 0.60s flow 6 is terminated. Node 3, 4 and 5 release 2.99Mbps of common bandwidth.

From the above simulation results, we can see that the hop-by-hop reservation algorithm handles the reservation setup and release well. It completely solves guaranteed

bandwidth hole problem.

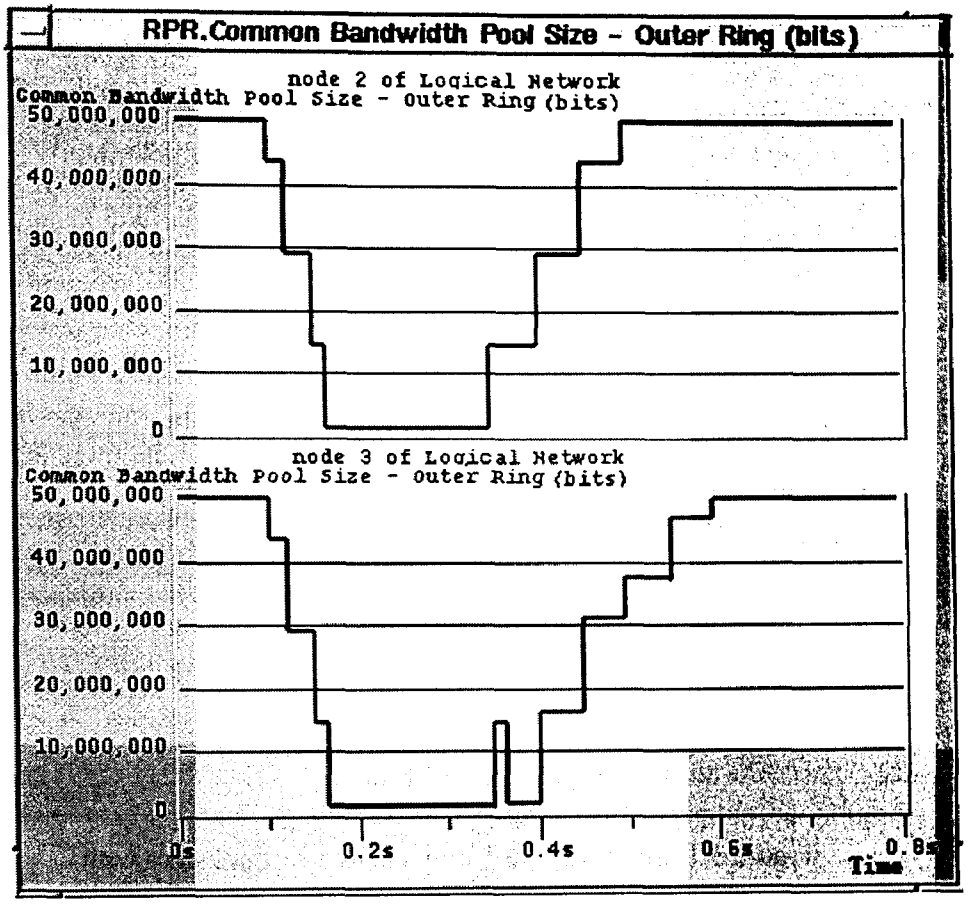


Figure 3.10: Available common bandwidth pool sizes with Hop-by-hop Reservation Algorithm

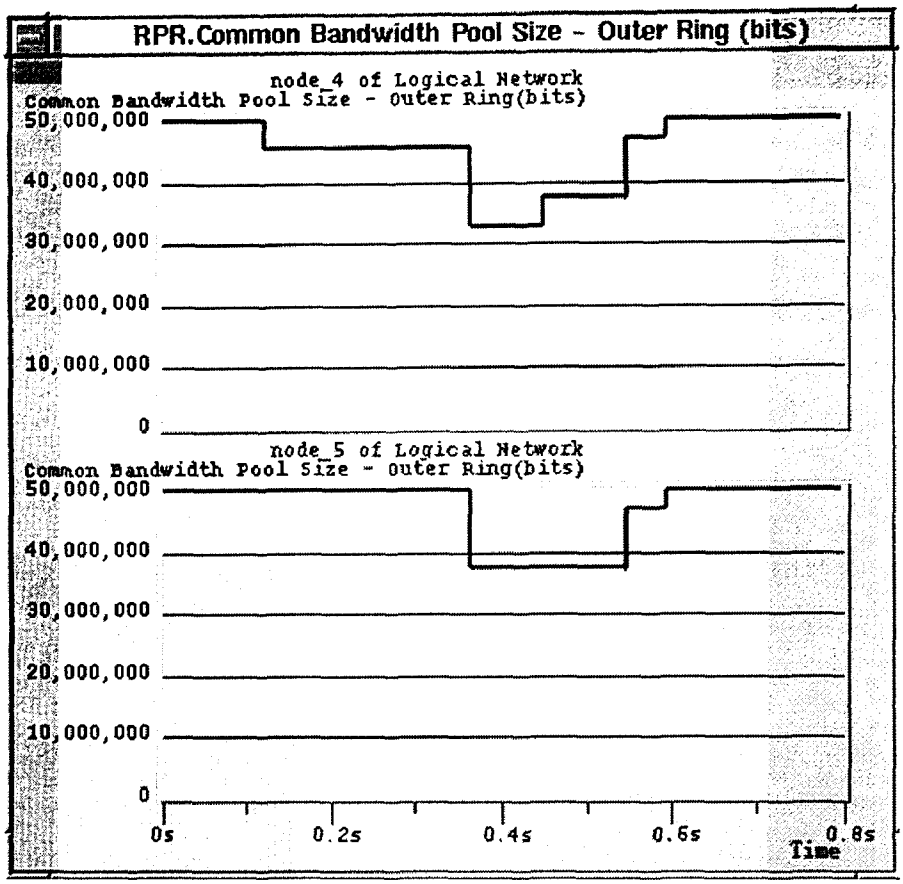


Figure 3.11: Available common bandwidth pool sizes with Hop-by-hop Reservation Algorithm

3.10 Conclusion

The dynamic bandwidth allocation using common bandwidth pool method can make the bandwidth usage in the RPR network efficiently. We introduced three bandwidth reservation algorithms in this paper. Each of them has its own advantage and drawback. The Basic Reservation Algorithm is easy to implement, but it may cause the guaranteed bandwidth hole problem and exhibits the inefficient bandwidth allocation problem during the bandwidth reservation process. The Flow-based Reservation Algorithm is more complicated than the Basic Reservation Algorithm. It can eliminate the guaranteed bandwidth hole problem in most cases. But it still has the inefficient bandwidth allocation problem during the bandwidth reservation process. The Hop-by-hop Reservation Algorithm is the most complicated one. It can completely eliminate the guaranteed bandwidth hole problem and the inefficient bandwidth allocation problem. Note that the three algorithms can be deployed in the same network. It is because only the source node needs to decide which algorithm to use. The

transit node and the destination node use the same mechanism for all three algorithms.

Chapter 4 Integration with DiffServ and RSVP

4.1 Integration with Differentiated Services (DiffServ)

RPR and DiffServ have similar QoS structure. The best-effort class, the assured forwarding classes, and the Expedited forwarding class in DiffServ can be mapped to RPR classC, classB, and classA service classes, respectively. In addition, the reservation algorithms proposed here can be incorporated with the bandwidth allocation method used by the DiffServ. More specifically, the DiffServ bandwidth allocation method can use the proposed reservation algorithms to reserve bandwidth at the RPR MAC layer. Figure 4.1 illustrates the integration between the DiffServ bandwidth allocation mechanism and the RPR reservation algorithm.

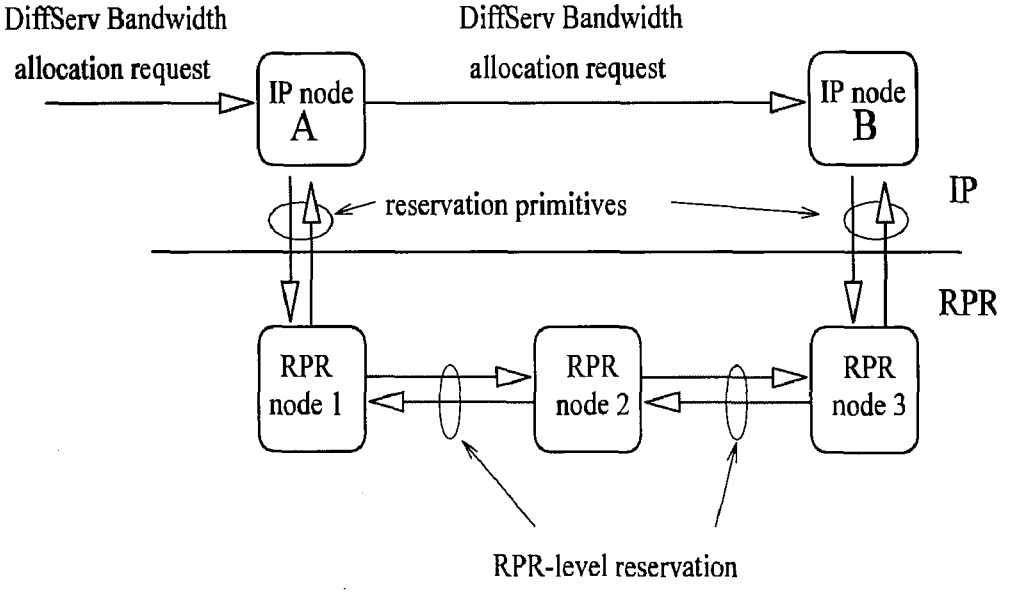


Figure 4.1: Interworking between DiffServ and RPR reservations

In the figure, IP node A receives a DiffServ bandwidth allocation request. The request can be generated by a bandwidth booker or some other management entity. Suppose the request is to allocate certain amount of bandwidth for some traffic class from the link from IP node A to IP node B. To make the requested bandwidth allocation, a bandwidth reservation at the RPR MAC layer from node 1 to node 3 is required. The proposed reservation algorithms can be used for the reservation. Note that IP node A (B) and RPR node 1 (3) are logical entities. Usually, they are co-located in the same physical router.

Since the bandwidth allocation is relatively static under the DiffServ paradigm, the Hop-by-Hop Reservation algorithm seems to be the best choice among the three proposed reservation algorithms. It is because the Hop-by-Hop Reservation algorithm provides the most efficient mechanism for bandwidth reservation. At the same time, the major drawback of the algorithm, that it is relatively complex to implement, does not incur a significant processing overhead under the DiffServ operation for reservation requests do not happen very often.

4.2 Integration with Integrated Services (IntServ)

The resource ReSerVation Protocol (RSVP) [9] is the signaling protocol used for reservation in IntServ-enabled networks. The reservation principles adopted by the proposed reservation algorithms and RSVP are quite different. In RSVP, it is the receiver that initiates the reservation; in the proposed reservation algorithms, it is the sender that initiates the reservation. The sender-initiated reservation is chosen for the proposed reservation algorithms because the sender is the only node that can decide how much common bandwidth reservation is required for each flow.

RSVP uses soft-state approach to maintain the reservation. That means the reservation must be refreshed periodically; otherwise, the reservation will be released automatically. The proposed reservation algorithms in contrast uses hard-state approach, in which, the reservation will persist until it is released explicitly by exchanging reservation release control packets among nodes. The hard-state approach is chosen because RPR routes do not change as frequently as those in the general IP networks. Consequently, the ability of the soft-state approach in adapting the changes of routes does not provide that much advantage in the RPR environment. On the other hand, the soft-state approach incurs the overhead of sending reservation packets periodically for each flow. The overhead may significantly affect the performance of the RPR networks, since we expect the networks will handles tens of thousands of flows at any given time. The hard-state approach has no such overhead.

Even RSVP and the proposed reservation algorithms uses different approaches for reservation, nevertheless, RSVP can still incorporate the proposed reservation algorithms to make reservation at the RPR path between two RSVP-capable routers. Figure 4.2 illustrates the interworking between RSVP and the proposed reservation algorithms.
In the figure, the two RSVP-capable routers first exchange RSVP Path and Resv messages

(steps 1 and 2). When IP node A receives the Resv message, it will invoke reservation at the MAC layer (steps 3 and 4). This leads to the exchanges of request and ack messages at the MAC layer (steps 5 and 6) based on the proposed reservation algorithms. Finally, if the reservation is successful, node A will receive the confirmation (steps 7 and 8).

Since RSVP Path and Resv messages are periodically exchanged between the two routers, a convergence sub-layer is required to distinguish the original RSVP Path and Resv messages from the subsequent RSVP Path and Resv messages. When the RSVP Path message for the already established connection is generated by the RSVP process, the sub-layer will not invoke another reservation. Instead, it will automatically generate the RSVP Resv message back to the process. In this way, no new reservation or extra traffic is created in the RPR network. In addition the convergence sub-layer should also issue RSVP Path messages to the downstream IP node (node B in Figure 3.3) periodically. The RSVP Resv messages received in response to the RSVP Path messages will be accepted by the convergence sub-layer and silently discarded. In the event where the reservation has not been refreshed within certain time-out period (this may happen, for instance, when node A has not received any RSVP Path message from its upstream node in a given time-out period), the convergence sub-layer will issue a release primitive, which in turn, invoke the bandwidth release process at the MAC layer based on the proposed reservation algorithms.

It must be emphasized that this chapter only provides preliminary outlines on how to integrate the proposed reservation algorithms with DiffServ and IntServ system. More careful study must be done to complete the full integration.

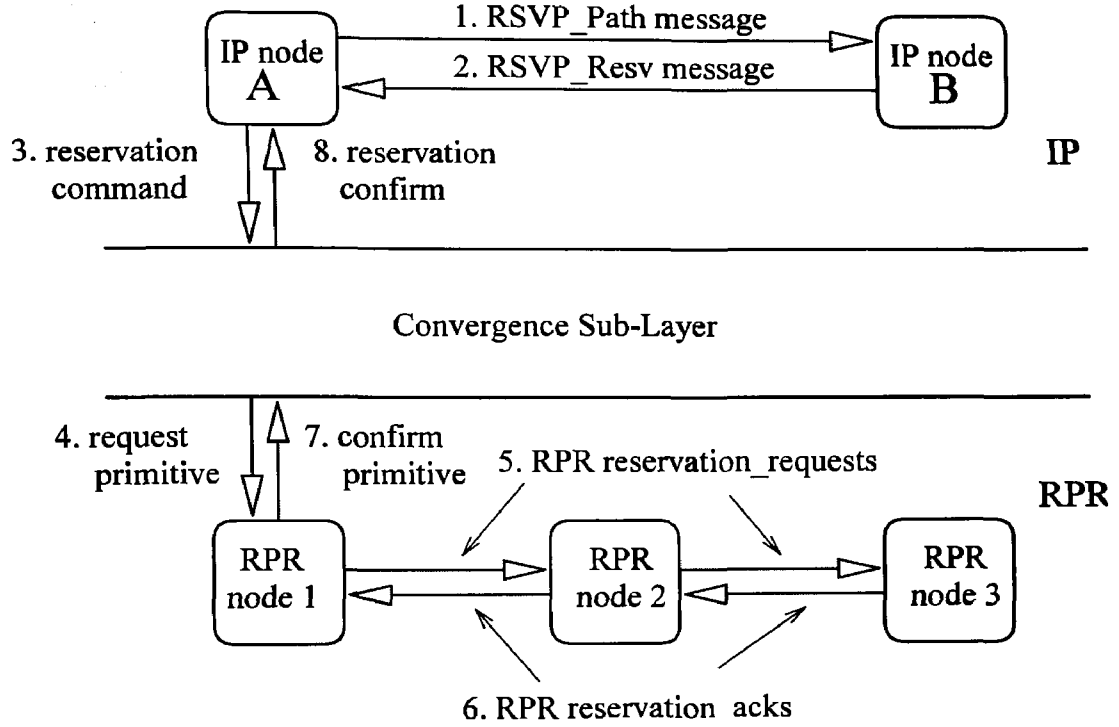


Figure 4.2: Interworking between IntServ and RPR reservations

Conclusion and Future Work

In this paper, we studied the admission control methods used in RPR networks and proposed the dynamic bandwidth allocation method for classA traffic in RPR network. The simulation results show that with the use of admission control, the delay of classA traffic can be kept small. Most of the delay is introduced by the processing and queueing delay at the source node. Once the traffic enters the RPR network, the total delay introduced by the transit nodes is almost the same as the propagation delay. We studied three admission control methods: Simple Sum, Measured Load and Equivalent Bandwidth. Simple Sum is the simplest one and it is easy to implement, while Equivalent Bandwidth method should be used in the situation where many classA traffic flows are multiplexed.

We implemented three reservation algorithms for the dynamic bandwidth allocation method: Basic Reservation Algorithm, Flow-based Reservation Algorithm and Hop-by-hop Reservation Algorithm. Basic Reservation Algorithm is easy to implement but it is also easy to cause the guaranteed bandwidth hole problem and insufficient bandwidth allocation problem. Flow-based Reservation Algorithm can eliminate the guaranteed bandwidth hole problem in many cases but it still cannot solve the insufficient bandwidth allocation problem. Hop-by-hop Reservation Algorithm is the most complex one but it can completely eliminate the guaranteed bandwidth hole problem and the insufficient bandwidth allocation problem. The three algorithm and be implemented in the same network because only the source node decides which algorithm to use; the transit nodes behave the same in all three algorithms. The dynamic bandwidth allocation method is also integrated well with the existing Internet Quality of Services (QoS) paradigms such as DiffServ and RSVP services.

In this thesis, we only focus the admission control and bandwidth allocation methods for the classA unicast traffic in RPR networks. The admission control and bandwidth allocation methods for multicast traffic in RPR networks need further study. In addition, this thesis has not addressed the situation when there is an equipment or facility failure and the two rings are wrapped to form a single ring. In this case the existing reservations will be disrupted and new reservations may be required. A more efficient reservation method to deal with this situation is desirable.

Reference

[1] IEEE Draft P802.17/ D2.3 June 16, 2003

[2] RFC 2998: A Framework for Integrated Services Operation over Diffserv Networks

[3] RFC 2983: Differentiated Services and Tunnels

[4] Isli Hjálmtysson and Alan G. Konheim, "Policing and Traffic Shaping at the User-Network-Interface (UNI)", *the Journal on Telecommunication Systems* Vol. 6, Nos. 3,4, February 1997, pp. 261-288.

[5] G. Bianchi, F. Borgonove, A. Capone, L.Fratta and C.Petrioli, "Endpoint Admission Control with Delay Variation Measurements for QoS in IP Networks", *ACM SIGCOMM Computer Communications Review*, Volume 32 , Issue 2 (April 2002)

[6] Sugih Jamin, Scott J. Shenker and Peter B. Danzig, "Comparison of Measurement-based Admission Control Algorithms for Controlled - Load Service", *Proc. IEEE INFOCOM '97*, April 1997

[7] Sally Floyd, "Comments on Measurement-based Admissions Control for Controlled-Load Services", Technical report, Lawrence Berkeley National Laboratory, July 1996. **URL** ftp://ftp.ee.lbl.gov/papers/admit.ps.Z

[8] Roch Guerin, Hamid Ahmadi and Mahmoud Naghshineh, "Equivalent Capacity and Its Application to Bandwidth Allocation in High-Speed Networks", *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 7, pp 968-981, 1991.

[9] RFC 2210: The Use of RSVP with IETF Integrated Services

[10] Sassan Pejhan, Mischa Schwartz, Life Fellow and Dimitris Anastassiou, "Error Control Using Retransmission Schemes in Multicast Transport Protocols for Real-Time Media", *IEEE/ACM Transactions on Networking (TON)*, Volume 4, Issue 3 (June 1996)

[11] Nasser M.Marafih, Ya-Qin Zhang and Raymond L. Pickholtz, "Modeling and Queueing Analysis of Variable-Bit-Rate Coded Video Sources in ATM Networks", *Circuits and Systems for Video Technology, IEEE Transactions* on Volume 4, Issue 2 (Apr 1994)

[12] Hans-Peter Schwefel and Lester Lipsky, "Impact of Aggregated, Self-similar ON/OFF Traffic on Delay in Stationary Queueing Models", *Performance and Control of Network Systems III*, August 1999, pp. 184-195

[13] Jeong-Soo Han, Seong-Jin Ahn and Jin-Wook Chung, "Study of delay patterns of

weighted voice traffic of end-to-end users on the VoIP Network", *International Journal of Network Management*, Volume 12, Issue 5 (May 2002).

(2)

BL-104-159