ABSTRACT

CONGESTION CONTROL IN RESILIENT PACKET RING NETWORKS

by

Alharbi Fahd

The Resilient Packet Ring (RPR) is a new metro technology; RPR shares SONET's ability in providing fast recovery from link and node failures as well as inherits the cost and simplicity of Ethernet. RPR, like SONET/SDH, is a ring based architecture consisting of two optical rotating rings (uni-directional). In RPR, packets are removed from the ring at the destination so that different segments of the ring can be used at the same time for different flows; as a result, the spatial reuse feature is achieved. Enabling the spatial reuse feature introduces the challenge of guaranteeing fairness among the nodes sharing the same link.

The RPR fairness algorithm is comparatively simple, but it poses some critical limitations. One of the major problems is that the amount of bandwidth allocated by the algorithm oscillates severely under unbalanced traffic scenarios. These oscillations are a barrier to achieving spatial reuse and high bandwidth utilization. Moreover, the current RPR standard uses a single FIFO for each class at the ingress point, thus resulting in the head of line blocking problem. On the other hand, RPR uses the shortest path to route the traffic in the dual ring which is inefficient and unfair.

In this dissertation, the performance of the existing fairness algorithms and their limitations was investigated. Two bandwidth allocation algorithms were proposed to address the fairness issue. Both algorithms were demonstrated analytically and through

simulations were able to achieve fairness and maximize the ring utilization. The Distributed Bandwidth Allocation (DBA) and the Adaptive Bandwidth Allocation (ABA) do not need to maintain information about each node. Instead, they use the local information which makes them scalable for a ring with any number of nodes. The Simple Scheduling Algorithm (SSA) was proposed to avoid the head of line blocking and to maximize the ring utilization at a very low complexity. The SSA algorithm was shown analytically and through simulations to be optimal where the flows achieve their max-min fair rates at a very low computational complexity. Also, the weighted routing algorithm was proposed to maximize the ring utilization by enabling the RPR nodes to transmit in both rings in a weighted manner. The routing algorithm was demonstrated analytically and through simulations was able to maximize the ring utilization.

CONGESTION CONTROL IN RESILIENT PACKET RING NETWORKS

by
Alharbi Fahd

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Electrical Engineering

Department of Electrical and Computer Engineering

January 2005

APPROVAL PAGE

CONGESTION CONTROL IN RESILINT PACKET RING NETWORKS

Alharbi Fahd

Dr. Nirwan Ansari, Dissertation Advisor Professor of Electrical and Computer Engineering, NJIT	Date
Dr. Kevin W. Lu, Committee Member Chief Scientist, Telcordia Technologies	Date
Dr. Edwin Hou, Committee Member Associate Professor of Electrical and Computer Engineering, NJIT	Date
Dr. Lev Zakrevski, Committee Member Assistant Professor of Electrical and Computer Engineering, NJIT	Date
Dr. Roberto Rojas-Cessa, Committee Member Assistant Professor of Electrical and Computer Engineering, NJIT	Date

BIOGRAPHICAL SKETCH

Author:	Alharbi Fahd
Degree:	Doctor of Philosophy
Date:	January 2005

Undergraduate and Graduate Education:

- Doctor of Philosophy in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2005
- Master of Science in Electrical Engineering,
  Fairleigh Dickinson University, Teaneck, USA, 2000
- Bachelor of Science in Electronics,
  College of Technology, Riyadh, Saudi Arabia, 1995

Major: Electrical Engineering

Publications:

F.Alharbi and N. Ansari,
"Low Complexity Distributed Bandwidth Allocation for Resilient Packet Ring Networks," Proceedings of IEEE Workshop on High Performance Switching and Routing (HPSR2004), pp.277-281, April 2004.

F.Alharbi and N. Ansari,
"Adaptive Fairness Algorithm for Resilient Packet Ring Networks," Proceedings of the first IFIP international conference on Wireless and Optical Communication Networks (WOCN2004), pp.86-89, June2004.

F.Alharbi and N. Ansari,
"A Novel Fairness Algorithm for Resilient Packet ring Networks with Low Computational and Hardware Complexity," Proceedings of 13th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN2004), pp 11-16, Mill Valley, Ca, USA, April 24-27, 2004.

F.Alharbi and N. Ansari,
"Distributed Bandwidth Allocation for Resilient Packet Ring Networks," Journal of Computer Networks, accepted on December, 2004.

F.Alharbi and N. Ansari,

"The Weighted Fairness Algorithm for Efficient Routing and Maximum Utilization in Resilient Packet Ring Networks," IEEE Journal of Lightwave Technology, submitted on October, 2004.

F.Alharbi and N. Ansari,

"SSA: Simple Scheduling Algorithm for Resilient Packet Ring Networks," IEE Proceedings on Communications, submitted on October, 2004.

F.Alharbi and N. Ansari,

"Adaptive Bandwidth Allocation for Resilient Packet Ring Networks," to be submitted for publication.

F.Alharbi and N. Ansari,

"Routing in the Resilient Packet Ring Networks," to be submitted for publication.

To my dear parents and my beloved wife

ACKNOWLEDGMENT

I would like to express my deepest appreciation to Dr. Nirwan Ansari, who not only served as my research supervisor, providing valuable and countless resources, insight, and intuition, but also constantly gave me support, encouragement, and reassurance. Special thanks are given to Dr. Kevin W. Lu, Dr. Edwin Hou, Dr. Lev Zakrevski and Dr. Roberto Rojas-Cessa for actively participating in my committee. I would also like to acknowledge the financial support by the government of Saudi Arabia.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

LIST OF FIGURES
(Continued)

LIST OF FIGURES (Continued)

CHAPTER 1

INTRODUCTION

1.1 Background Information

Rings are the most prevalent metro technologies because of their protection and fault tolerance properties, but the current metropolitan ring networking technologies exhibit several limitations. In a SONET ring [1], each node is granted with the minimum fair share, but it is not possible to reclaim the unused bandwidth; moreover, 50% of the potentially available bandwidth is reserved for protection, thus resulting in poor utilization. On the other hand, Gigabit Ethernet assures full statistical multiplexing at the expense of fairness.

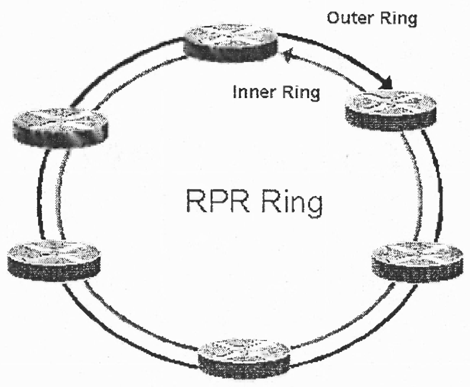


Figure 1.1 The Resilient Packet Ring

The Resilient Packet Ring (RPR), defined under IEEE 802.17, has been proposed as a high-speed backbone technology for metropolitan area networks. RPR is introduced to mitigate the underutilization and unfairness problems associated with the current technologies, SONET and Ethernet, respectively. RPR [6], [7] shares SONET's ability in providing fast recovery from link and node failures as well as inherits the cost and simplicity of Ethernet. Like SONET/SDH, RPR is a ring-based architecture consisting of two optical rotating rings: one is referred to as the inner ringlet, and the other the outer ringlet (Figure 1.1). RPR defines three service classes of user traffics: Class A with guaranteed rate and jitter, Class B with a committed information rate (CIR) and bounded delay and jitter, and the best effort traffic (Class C). In RPR [7], packets are removed from the ring at the destination so that different segments of the ring can be used at the same time for different flows; as a result, the spatial reuse [2], [3], feature is achieved. Enabling the spatial reuse feature (concurrent transfers over the same ring) introduces the challenge of guaranteeing fairness among the nodes sharing the same link

1.2 Motivation

The key performance objectives of RPR are to achieve high bandwidth utilization, optimum spatial reuse on the dual rings, and fairness. The challenge is to design an algorithm that can react dynamically to the traffics in achieving these objectives. The RPR fairness algorithm [7], [8], [9], is comparatively simple, but it poses some critical limitations that require further investigation and remedy. One of the major problems is that the amount of bandwidth allocated by the algorithm oscillates severely under unbalanced traffic scenarios. These oscillations are barrier to achieving spatial reuse and high bandwidth utilization. DVSR [10], [11], was another algorithm proposed to solve the fairness issue with no oscillation at the steady state, but it requires per-source information and has a high computational complexity of $O(N\log N)$, where N is the number of nodes in the ring. Moreover, the current RPR standard uses a single FIFO for each class at the ingress point, and thus the head of line blocking is a potential problem. On the other hand, RPR uses the shortest path to route the traffic in the dual ring which is inefficient and unfair.

1.3 Contributions

In this dissertation, we will first focus on the fairness and bandwidth allocation issue. Two bandwidth allocation schemes are introduced. First, the Distributed Bandwidth Allocation (DBA) is proposed to achieve fairness at a very low complexity of O (1) and does not require per-source information which makes it scalable for any ring network. Second, an Adaptive Bandwidth Controller (ABC) is developed to adjust the fair rate based on the state of congestion. The adaptive algorithm provides significantly better performance than the RPR fairness algorithm. Second, we introduce the Simple Scheduling Algorithm (SSA) to avoid the head of line blocking associated with the current RPR traffic shaping scheme and maximize the ring utilization at a very low complexity. Finally, we propose two weighted routing algorithms to maximize the ring utilization by enabling RPR nodes to forward packets in both rings in a weighted manner. The first routing algorithm is based on the RIAS fairness concept and referred to as the Routing Algorithm for RPR (RA-RPR). The second routing algorithm is the Weighted Fairness Algorithm (WFA) which adapts the per-flow fairness concept.

1.4 The Structure of the Dissertation

Chapter 2 provides an overview of the RPR technology and focuses on the main features and design objectives. Chapter 3 investigates the existing fairness algorithms for RPR and their limitations. In Chapter 4, we introduce our Distributed Bandwidth Allocation (DBA) algorithm. In Chapter 5, we develop the adaptive bandwidth allocation algorithm. Chapter 6 focuses on the traffic shaping issue. In Chapter 7, we investigate the limitation of the shortest path routing policy and propose alternative solutions. We conclude and discuss our future work in Chapter 8.

CHAPTER 2

THE RESILIENT PACKET RING

RPR is a ring-based architecture consisting of two optical rotating rings (uni-directional). In RPR, packets are removed from the ring at the destination so that different segments of the ring can be used at the same time for different flows; as a result, the spatial reuse feature is achieved. This chapter gives an overview of the RPR Technology [7].

2.1 Layer Model

The RPR layer model and its relationship to the Open Systems Interconnect (OSI) reference model is illustrated in Figure 5.1.

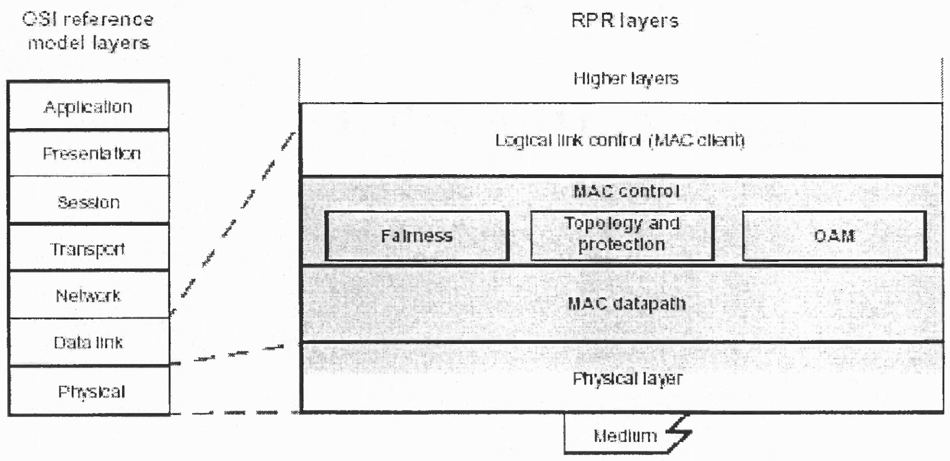


Figure 2.1 The Layer model

The MAC control sublayer controls the exchange of data between the MAC and its client. It also maintains information related to the state of the network such as ring topology and protection information. The MAC data path sublayer provides data transfer functions for each ringlet such as transmitting, forwarding , and striping frames. The PHY service interface is used by the MAC to transmit and receive frames on the physical media.

2.2 Ring Structure

RPR is a ring-based architecture consisting of two optical rotating rings: one is referred to as the inner ringlet, and the other the outer ringlet (Fig. 1.1). Unlike FDDI [5], where the secondary ring is idle during the normal operation, both rings in RPR are used for transmitting data and control packets. This allows RPR to maximize the ring utilization. At each ringlet, a node receives packets from its upstream neighbor and transmits them to its downstream neighbor.

2.3 Spatial Reuse

RPR does not use token like Token Ring [4] and FDDI [5] to control access to the ring. Instead, RPR is a buffer insertion ring (Figure 2.2). To avoid collision when the node is transmitting a packet from its local buffers, the arriving packets from upstream nodes wait in the transit buffers. The node arbitrates the service among the local and transit traffic according to the fairness algorithm and avoid transit buffer overflow. Moreover, packets are removed from the ring at the destination so that different segments of the ring can be used at the same time for different flows; as a result, the spatial reuse feature is achieved (Figure 2.3). Enabling the spatial reuse feature (concurrent transfers over the same ring) maximizes the ring utilization but at the same time introduces the challenge of guaranteeing fairness among the nodes sharing the same link.

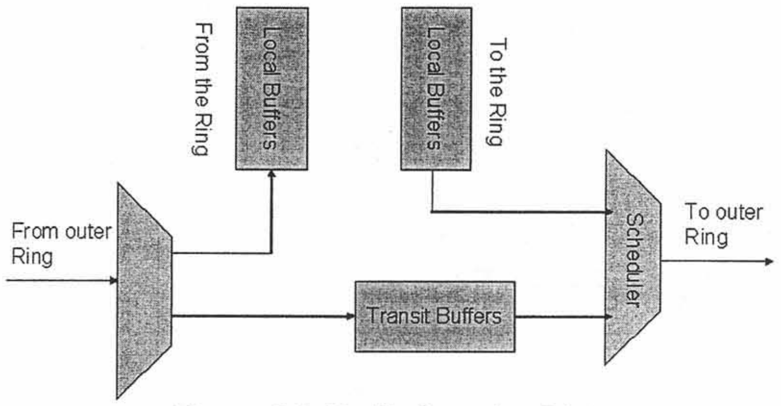


Figure 2.2 Buffer Insertion Ring

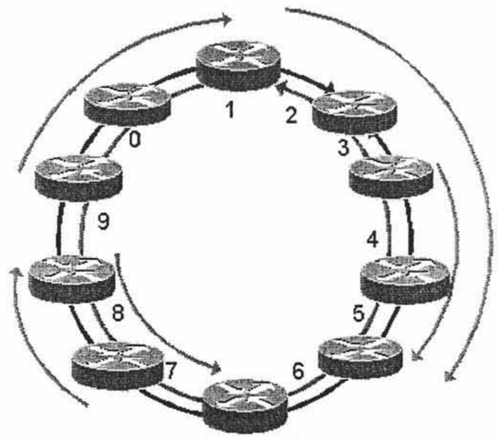


Figure 2.3 Spatial Reuse

2.4 Frame Format

The resilient packet ring (RPR) defines four different types of frames. These four types of frames and their important fields are introduced here.

2.4.1 Data Frame

Figure 2.4 shows the data frame. The following is a short description of the functionalities of the data frame fields.

ttl: An 8-bit (time to live) field that specifies the maximum number of hops the frame is expected to cover before reaching the destination. This field provides a mechanism to ensure that frames do not circulate forever on the ring.

ri: A (ringlet identifier) bit that identifies the ringlet onto which the frame was originally transmitted.

fe: A (fairness eligible) bit that marks whether the frame is subject to the fairness algorithm. A value of 0 indicates that the frame is not fairness eligible, while a value of 1 indicates that the frame is fairness eligible.

ft: A 2-bit (frame type) field that identifies the type of the frame

sc: A 2-bit (service class) field that identifies the service class of the frame.

we: A (wrap eligible) bit that indicates whether the frame is eligible to be wrapped during a wrap condition

p: parity bit

da: A 48-bit (destination address) field that specifies the station(s) for which the frame is intended.

sa: A 48-bit (source address) field that specifies the local station sending the frame.

ttlBase: An 8-bit ttlBase field that is set to the initial value of the ttl field upon transmission of a data frame.

hec: A 16-bit (header error check) field that is a checksum of the header.

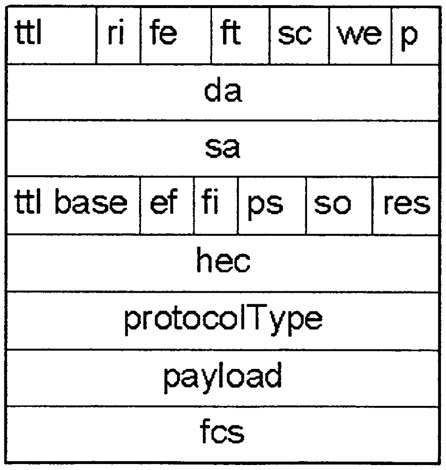


Figure 2.4 Data frame

2.4.2 Fairness Frame

The fairness frame, shown in Figure 2.5, is sent to MAC neighbors to inform upstream nodes about the state of congestion. The value of the fairRate field is computed using the fairness algorithm (Chapter 3).

The frame format for fairness frames is different from the frame format for data frames. Fairness frames are not sent to specific destination nodes, but are sent to a station's nearest neighbor or broadcast to the entire ring. Therefore, the destination address does not contain any useful information and is omitted. The size of the fairness frame is rather small in order to reduce their effective bandwidth consumption.

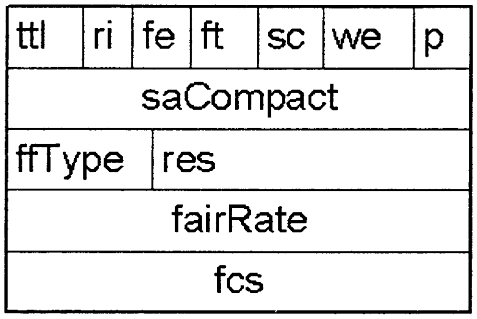


Figure 2.5 Fairness frame

2.4.3 Control Frame

Control frames are used to propagate topology and recovery information. The format of the control frame is similar to the data frame but is distinguished by the *ft* field value. There are several control frames in RPR such as topology, protection, and OAM (Operations Administration and Maintenance) frames.

2.4.4 Idle Frame

The idle frame format is similar to the fairness frame. The idle frame is sent to MAC neighbors to adjust the rate synchronization between the station and its neighbors and to allow for a strict control of bandwidth allocation around the ring.

2.5 MAC Data Path Sublayer

The MAC layer defined by RPR (Figure 2.1) consists of the MAC control sublayer and the MAC data path sublayer. The MAC data path controls transmitting, forwarding, and striping data from each ring. There is a separate data path for each ringlet as shown in Figure 2.6.

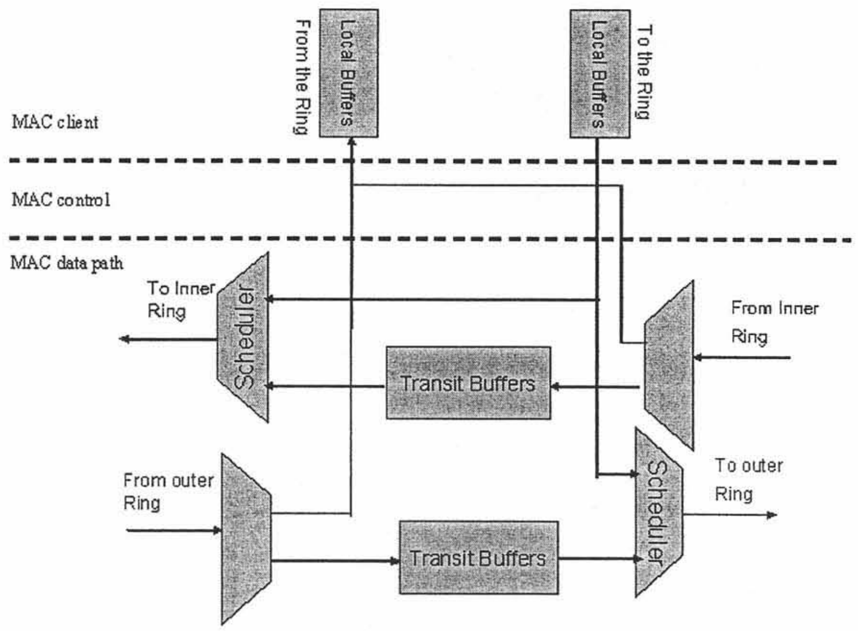


Figure 2.6 MAC layer

2.5.1 Service Class

RPR defines three service classes for user traffics. First, Class A is divided into ClassA0 and ClassA1; the difference is that the reserved bandwidth for ClassA0 can be reclaimed. Class A provides guaranteed rate and jitter, and used by applications with strict delay and jitter, such as video and audio applications. Second, Class B provides committed information rate (CIR) and bounded delay and jitter. Finally, the best effort traffic (Class C) with rate limited by the fairness algorithm (*fairness eligible traffic*) provides no guarantees delay.

2.5.2 Frame Transmission

To ensure hardware simplicity and that the transit path is lossless, the RPR node does not include per–ingress or per-flow queues on the transit path; instead, it supports two scheduling modes. In the single–queue mode (Figure 2.7), the transit path is a single FIFO and the transit traffic has a strict priority over the station traffic. On the other hand, in the dual-queue mode (Figure 2.8), the transit path consists of two queues: Primary Transit Queue (PTQ) for Class A traffic, and Secondary Transit Queue (STQ) for Class B and Class C traffic; in this mode, PTQ will be served first. When PTQ is empty, STQ has strict priority over the station traffic when the queue length exceeds the STQ threshold; otherwise, the station traffic is served in the following order: Class A, then Class B. If the station (node) has no Class A or B traffic, then Class C traffic will be served.

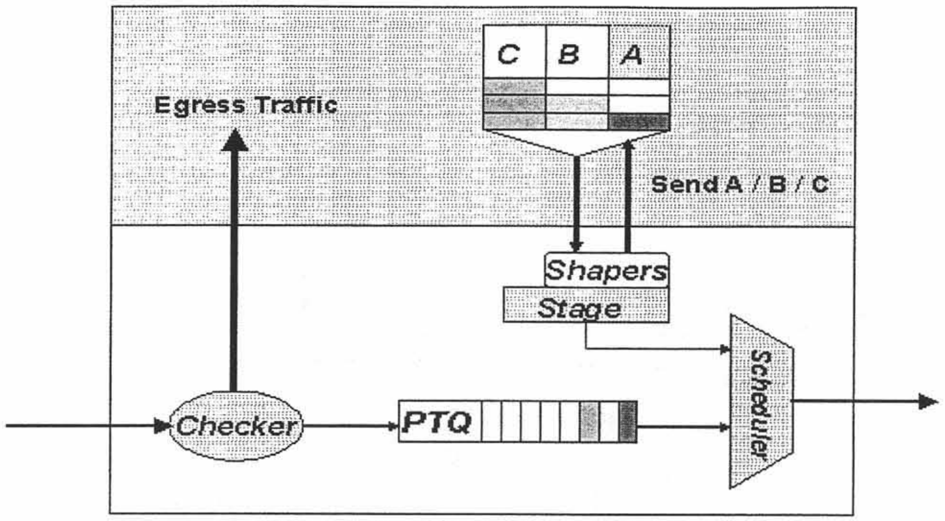


Figure 2.7 Single Queue Mode

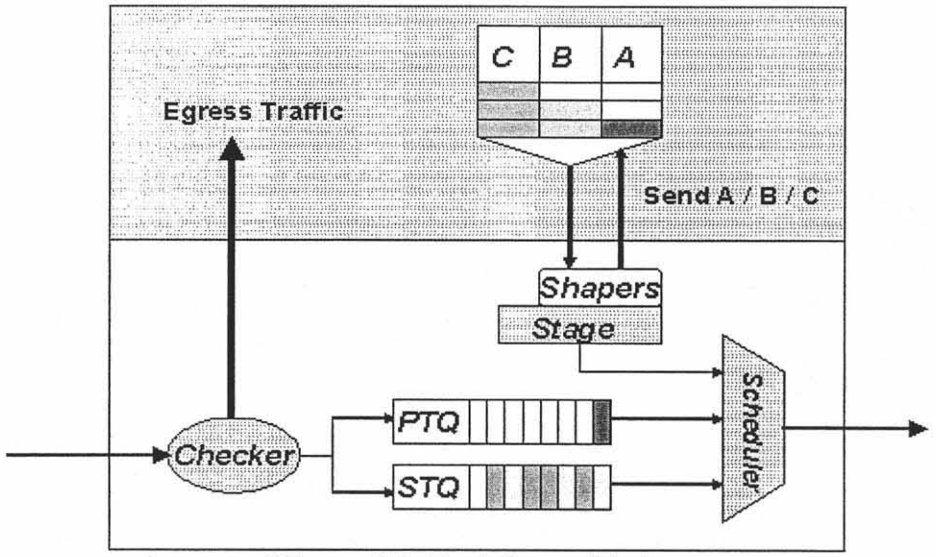


Figure 2.8 Dual Queue Mode

2.5.3 Frame Reception

Each node inspects the destination address of the received packet from upstream nodes. If this node is the destination, the packet is copied to the MAC control sublayer and then passed to the MAC client. On the other hand, if the packet is a transit packet, the *ttl* field is decremented and packet with *ttl* value equal to zero will be discarded. Otherwise, the packet will be forwarded to the respective transit buffer as detailed in Section 2.5.2.

2.6 MAC Control Sublayer

The MAC control sublayer, illustrated in Figure 2.9, supports the following activities:

a) Fairness algorithm and protocol

b) Protection database and protocol

c) Topology database and protocol

d) Operations, Administration, and Maintenance (OAM) functionalities

e) Ringlet selection

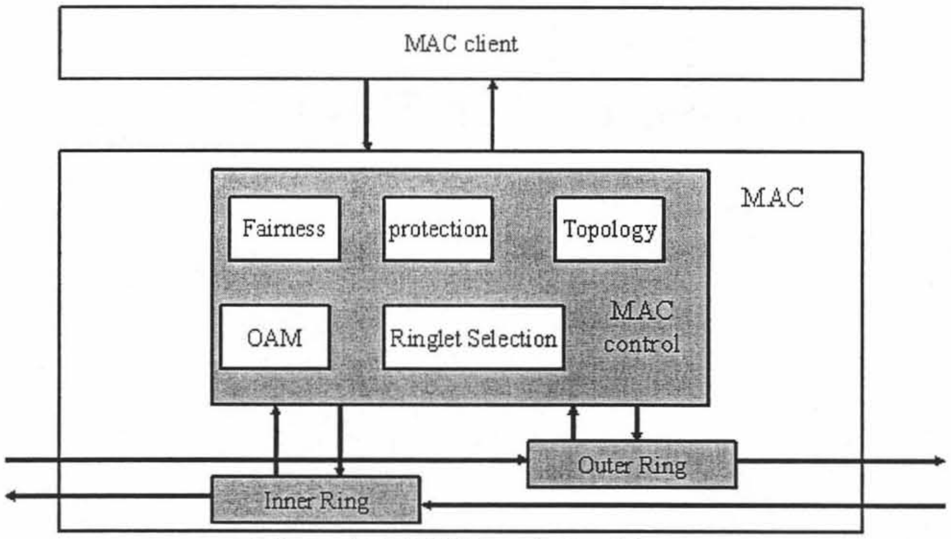


Figure 2.9 MAC control sublayer

2.7.1 Fairness

The goal of the fairness algorithm is to achieve the Ring Ingress Aggregated with Spatial Reuse (RIAS) fairness concept [10], [11], where the level of traffic granularity at a link is defined as an ingress-aggregated (IA) flow, i.e., the aggregate of all flows originated from the same node but destined to different nodes. At the state of congestion, all nodes should be able to send the same amount of data on the congested link relative to the other nodes. For example, consider the scenario shown in Figure 2.10 where the available bandwidth is equal to 100Mbps. If the per-flow fairness concept is used, each flow transmits at rate equal to 20Mbps. On the other hand, according the RIAS fairness concept, the available bandwidth at link 4 will be divided among the participating nodes and the per-source fair rate is equal to 25Mbps. Thus, flow (4,5) and flow(4,6) transmit at rate equal to 12.5Mbps. Fairness and bandwidth allocation is the topic of the next three chapters.

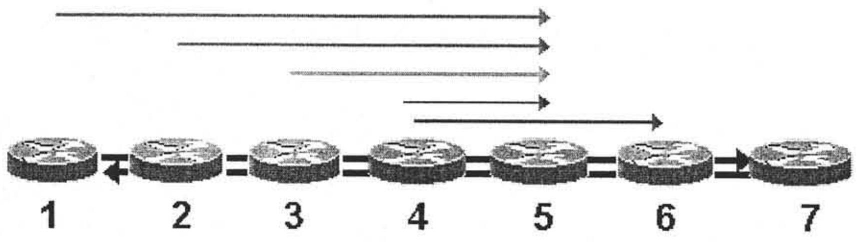


Figure 2.10 Fairness scenario

2.6.2 Protection

One of the RPR goals is to detect and repair a node or a link failure within 50 ms. RPR achieves this goal by supporting two protection mechanisms: steering and warping. Steering is supported by all stations. In the presence of a fault, the first node to detect the failure informs other nodes in the ring. Each node will update its topology table accordingly and chooses the ringlet that still has connectivity to the destination of the frame. The disadvantage of this mechanism is that all packets transmitted from the time of the failure till the time of topology update are lost.

Warping, on the other hand, reduces the number of lost packets where all packets destined to a node beyond the point of failure are looped through the other opposite ringlet. Warping continues till each node updates its topology table and switches to the steering mechanism. Warping introduces new problems such as packet reordering and duplication. Figure 2.11 shows a link failure scenario.

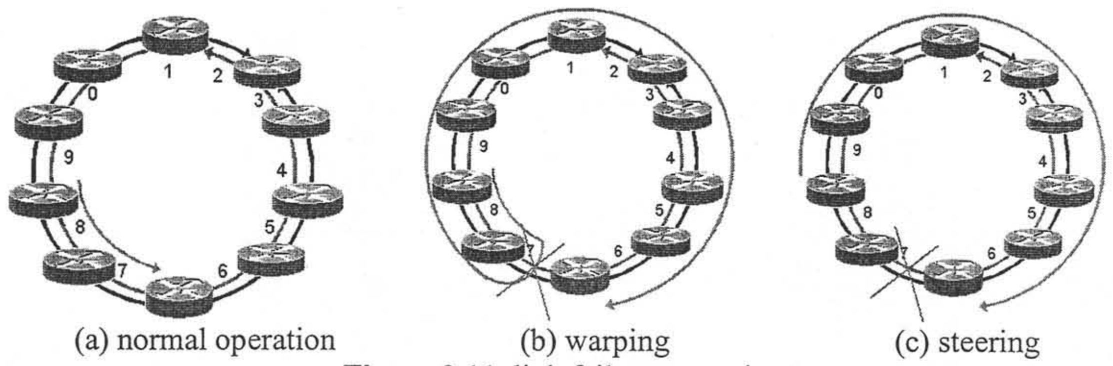


Figure 2.11 link failure scenario

2.6.3 Topology Discovery

The topology discovery mechanism provides each node with information about the number of nodes in the ring, their positions, and capabilities. Each node broadcasts a topology and protection (TP) frame containing information about the originating node making up the current topology image of that node. These frames are generated when the node becomes active on the ring, periodically, and on detection of a change in node or ring status.

2.7 Summary

This chapter provides an overview of the RPR technology. RPR is a buffer insertion ring which uses both rings for transmitting data packets. As a result, the spatial reuse is enabled and the ring utilization is maximized. Moreover, RPR has several important properties such as protection and fairness.

CHAPTER 3

FAIRNESS AND FLOW CONTROL IN RPR

The flow control in RPR is achieved by enabling a backlogged node to send the fairness message according to its measurements to the upstream nodes to throttle ingress data rates in order to eliminate the state of congestion and apply fairness among all the participating nodes.

3.1 Node Architecture in RPR

The RPR node architecture is illustrated in Figure 3.1. First, each node uses rate controllers to throttle the traffic entering the ring. Second, each node uses byte counters to measure transit traffic and node traffic. These measurements are used by the fairness algorithm to compute the fair rate which is fed-back to the upstream nodes in the form of a control message. Nodes that receive the control message will use the control message information with their local information to throttle their rates accordingly.

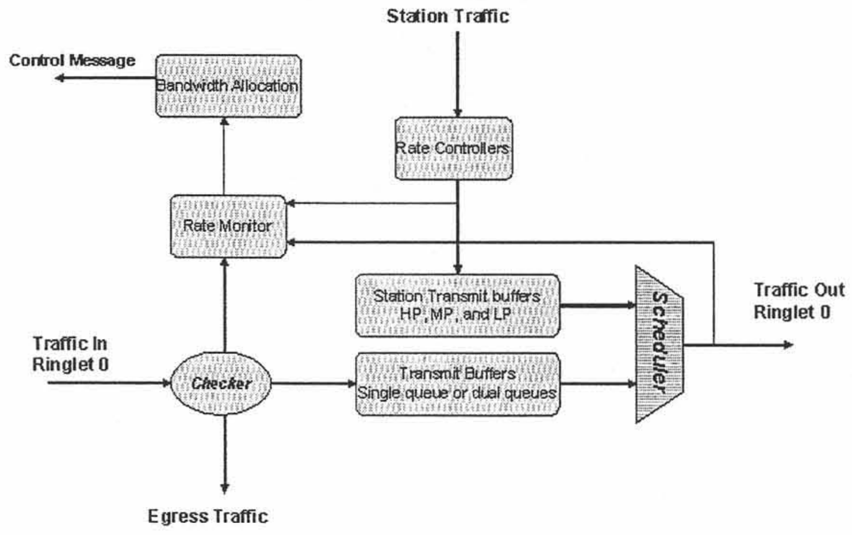


Figure 3.1 Generic RPR Node Architecture

3.2 The RPR Fairness Algorithm

The RPR fairness algorithm [7], [8], [9], operates in two modes: the aggressive mode and the conservative mode. In both modes, each node measures at the output of the scheduler the byte count of all serviced transit traffic named *forwardrate* and the byte count of all serviced station traffic named *myrate*. These measurements will be taken over a fixed *aging-interval.*

Both measurements are low-pass-filtered using exponential averaging as follows:

$$lp\_myrate = lp\_myrate\frac{lpcoef-1}{lpcoef} + myrate\frac{1}{lpcoef}. \tag{3.1}$$

$$lp\_forwardrate = lp\_forwardrate\frac{lpcoef-1}{lpcoef} + forwardrate\frac{1}{lpcoef}. \tag{3.2}$$

where $lpcoef \in [16, 32, 64, 128, 256, 512]$.

Both modes have the same measurements, but use them differently in detecting congestion and computing fair rates.

3.2.1 The Aggressive Mode

In the aggressive mode (RPR-AM), the transit path has a dual-queue. Node *k* is considered to be congested when either

STQdepth[k] > *low_threshold*, where the *low_threshold* is equal to 1/8 of the STQ size,

or

myrate[k] + *forwardrate[k]* > *unreserved_rate*, where the *unreserved_rate* is equal to the link capacity minus the reserved rate for the high priority class traffic.

When node *k* is congested, it calculates its *local_fairrate* as the normalized value of its own *lp_myrate* value, and then sends a fairness control message to upstream nodes containing *local_fairrate*. On the other hand, if the node is not congested, it sends a *NULL* value as the fairness control message to inform the upstream nodes to increase their rates.

When node *k*-1 receives the control message from node *k*, it will set its rate limiter value, namely, the *allowed-rate* based on the control message value, and then send a control message to the other upstream nodes with the value according to the following:

(a) Minimum of *lp_myrate [k-*1*]* and the received control message value *if* node *k*-1 is congested.

(b) A *Null* value *if* node *k*-1 is not congested.

(c) A *Null* value *if* node *k*-1 is congested, but *lp_myrate [k-*1*]* > *lp_forwardrate [k-*1*]* because node *k*-1 is the cause for congestion.

When a node receives a control message with a *NULL* value, it will increase its *allowed_rate* to reclaim the unused bandwidth as follows:

$$allowed\_rate = allowed\_rate + \frac{unreserved\_rate - allowed\_rate}{rampcoef}. \qquad (3.3)$$

where $rampcoef \in [16, 32, 64, 128, 256, 512]$.

Now consider the simple parking lot scenario [9] in Figure 3.2, where the flow from node 1 to node 3 is greedy while the flow from node 2 to node 3 is a low rate 50 Mbps, and both links have a capacity of 622Mbps. In the case of using the aggressive mode (see Figure 3.3), node 2 will be congested when the sum of its rate and the rate of flow (1, 3) is greater than the link capacity; then, it sends the fairness control message with its *lp_myrate* of 50 Mbps to node 1; accordingly, node 1 throttles its *allowed_rate* to 50 Mbps. When the congestion is resolved, node 2 sends the fairness control message with a *NULL* value, and so node 1 can increase its *allowed_rate* to claim the unused bandwidth until congestion occurs again starting a new cycle of oscillation.

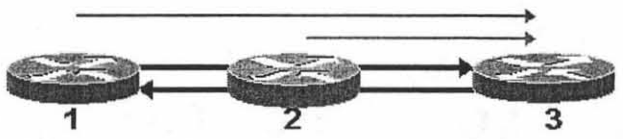


Figure 3.2 Simple parking lot (Scenario setup)

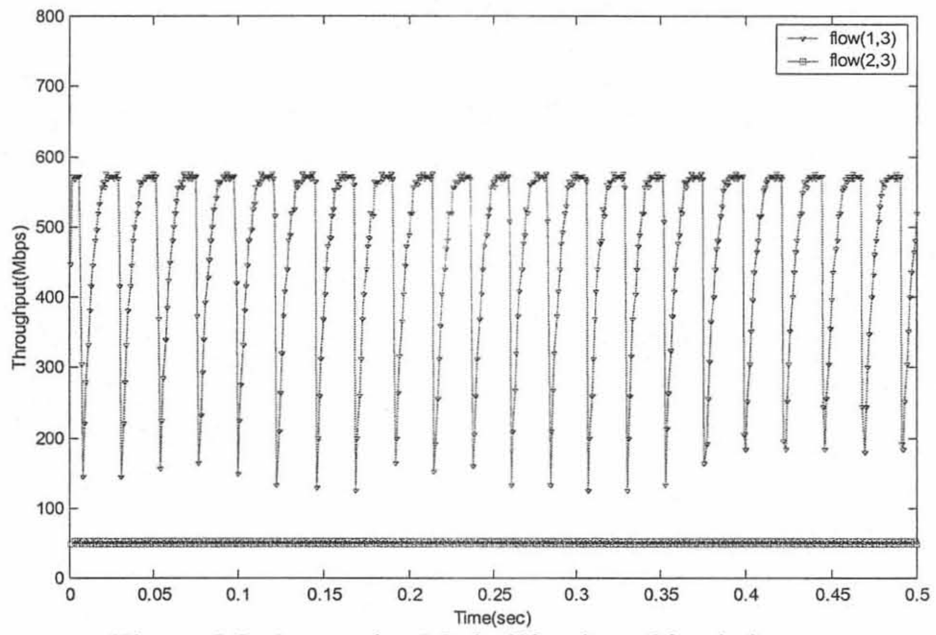


Figure 3.3 Aggressive Mode (Simple parking lot)

3.2.2 The Conservative Mode

In the conservative mode (RPR-CM), the transit path has a single queue and each node has an access timer to measure the time between its transmitted packets. Here, the node is considered to be congested when either the access time expires, or

Lp_myrate[k] + *lp_forwardrate[k]* > *low_threshold*, where the *low_threshold* is equal to 0.8 of the link capacity.

In the conservative mode, each node not only measures *myrate* and *forwardrate*, but also measures the number of active nodes where a node i will be counted active if at least a single packet was received from node i during the aging interval. If node k is congested in this aging interval, but was not congested in the previous interval, it will send a fairness control message containing the fair rate equal to the unreserved bandwidth divided by the number of active nodes.

$$local\_fairrate = \frac{unreserved\_rate}{number\_active\_nodes}. \tag{3.4}$$

If node k continues to be congested, then it sends a normalized *local_fairrate* depending on the value of the sum of *lp_myrate[k]* and *lp_forwardrate[k]*. If this value is less than the *low_threshold*, the *local_fairrate* will ramp up. On the other hand, the *local_fairrate* will ramp down when the sum is greater than the *high_threshold*, which is 0.95 of the *unreserved_rate*.

$$local\_fairrate = local\_fairrate + \frac{unreserved\_rate - local\_fairrate}{rampcoef}. \tag{3.5}$$

$$local\_fairrate = \frac{rampcoef - 1}{rampcoef} local\_fairrate. \tag{3.6}$$

Again consider the simple parking lot scenario in Figure 3.2. In the case of using the conservative mode (see Figure 3.4), node 2 will send the fairness control message with the fair rate equal to the link capacity divided by the number of active nodes (in this case, 2). When the congestion is resolved, node 1 can increase its rate to claim the unused bandwidth until the congestion occurs again.

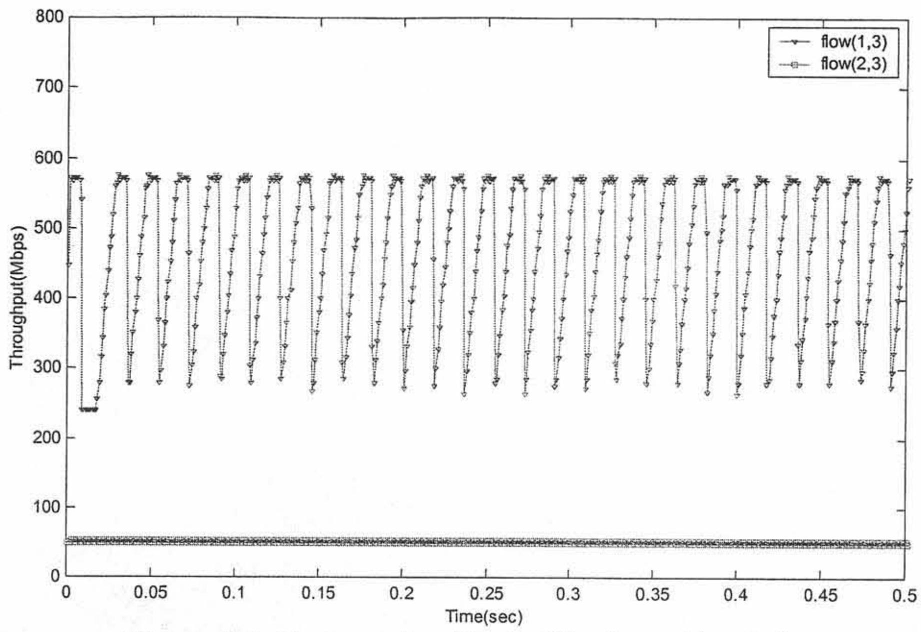


Figure 3.4 Conservative Mode (Simple parking lot)

3.3 The DVSR Algorithm

E. Knightly *et. al* [10], [11], proposed an algorithm called "Distributed Virtual Time Scheduling in Ring" (DVSR) to overcome the problems encountered in the RPR fairness algorithm. Unlike the RPR fairness algorithm, in DVSR, each node uses per ingress byte counters (measurement modules) to estimate the demand from every ingress node including itself during the measurement interval *T*. These measurements $r_1, r_2, ..., r_N$ are used by the fairness algorithm to compute the fair rate as follows:

First, the byte counts are ordered such that

$l_1 \le l_2 \le ... \le l_N$.

Then, the node fair rate F is computed using the *max-min* operation [18, p. 527].

$$F = \text{max-min}\ (C, l_1, l_2, ..., l_N). \tag{3.7}$$

where C is the $unreserved\_rate$. The pseudo code of the DVSR algorithm is given in Figure 3.5.

The performance of the DVSR algorithm for the simple parking lot scenario (Figure 3.2) is shown in Figure 3.6. DVSR solves the problem of bandwidth allocation at the price of complexity. DVSR requires maintaining per-source information. Moreover, the ordering operation has a computational complexity of *O(N log N)* where *N* stands for the number of nodes. Thus, the DVSR algorithm is not scalable for the ring network

```
$C$                  //the $unreserved - rate$
$M=N$                //the number of flows
$F=\frac{C}{M}$      //the fair share
$for$ i=1 $to$ $N-1$    //for all flows traversing the current link
 $if(l_i \le F)$     //if this flows does not require more than the fair rate then
 $C=C-l_i$           // eliminate this flow from the bottelnecked flows list
 $M=M-1$             // decrease the number of bottelnecked flows
 $F=\frac{C}{M}$     // recalculate the fair rate
 $end\ if$
$end\ for$
```

Figure 3.5 DVSR pseudo code

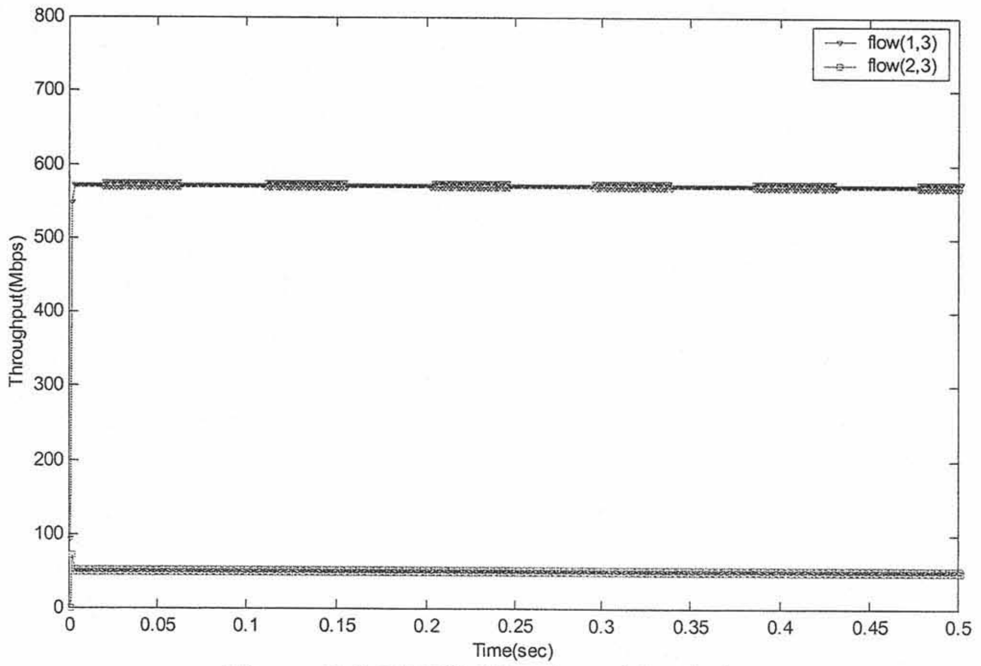


Figure 3.6 DVSR (Simple parking lot)

3.4 Summary

This chapter describes the existing algorithms for bandwidth allocation in RPR. The current RPR fairness algorithm operates in two modes, the Aggressive Mode (RPR-AM) and the Conservative Mode (RPR-CM). Both modes incur oscillations in the allocated bandwidth, resulting in a bandwidth loss and increased delay jitter. The reason of this performance limitation is that the congestion signals do not accurately reflect the exact fair rate. On the other hand, the DVSR algorithm achieves the fairness objective at the expense of a very high complexity. In following chapters, we will propose new bandwidth allocation algorithms to achieve fairness and maximize the utilization with a very low computational complexity of O(1).

CHAPTER 4

THE RATE –BASED FAIRNESS ALGORITHM

The RPR fairness algorithm is comparatively simple, but it poses some critical limitations. One of the major problems is that the amount of bandwidth allocated by the algorithm oscillates severely under unbalanced traffic scenarios. These oscillations are barrier to achieving spatial reuse and high bandwidth utilization. DVSR was another algorithm proposed to solve the fairness issue with no oscillation at the steady state, but at the expense of a high computational complexity $O(N\log N)$, where N is the number of nodes in the ring.

In this chapter, we propose a distributed bandwidth allocation algorithm to allocate bandwidth fairly to RPR nodes with a very low computational complexity O(1) that will converge to the optimal fair rate in a few measurement intervals with no oscillation at the steady state.

4.1 The Fairness Algorithm

In this section, we introduce a new bandwidth allocation algorithm referred to as the Distributed Bandwidth Allocation (DBA) algorithm [22]. This algorithm adopts the Ring Ingress Aggregated with Spatial Reuse (RIAS) fairness concept [9], [10], where the level of traffic granularity at a link is defined as an ingress-aggregated (IA) flow, i.e., the aggregate of all flows originated from the same node but destined to different nodes. At the state of congestion, all nodes should be able to send the same amount of data on the congested link relative to the other nodes.

Without loss of generality, throughout the analysis we consider only one of the two rings, the outer ring with N nodes numbered from *0 to N-1* along the ring direction.

Definition 4.1: The available bandwidth for the best effort traffic (Class C) is defined as

$$C = Link\_Capacity - reserved\_BW, \tag{4.1}$$

where $reserved\_BW$ is the bandwidth reserved for the higher priority class traffic.

Definition 4.2: At node k, the ingress aggregated traffic demand of node i during a measurement interval T is defined as follows:

$$R_i = \sum_{j=(k+1)\bmod N}^{(i-1)\bmod N} r_{i,j} \quad . \tag{4.2}$$

That is, R_i is equal to the sum of all flows $r_{i,j}$ originated from node i, traversing through node k, and destined to node j.

4.1.1 Derivation of The Algorithm

Define M as the number of flows traversing link k, and the arrival rate $\tilde{A}(n)$ at link k can be expressed as

$$\tilde{A}(n) = \sum_{i=1}^{M} R_i \tag{4.3}$$

Recall that each node i will send through link k at a rate according to the received fair rate from node k. Thus, the rate of source i through link k at time n is:

$$R_i(n) = \rho_i F_k(n), \tag{4.4}$$

where ρ_i is the activity level of source i with respect to the fair rate, $F_k(n)$, of the current interval [12], [14], [16]. The activity level is equal to one for flows bottlenecked at link k, and less than one for flows bottlenecked elsewhere.

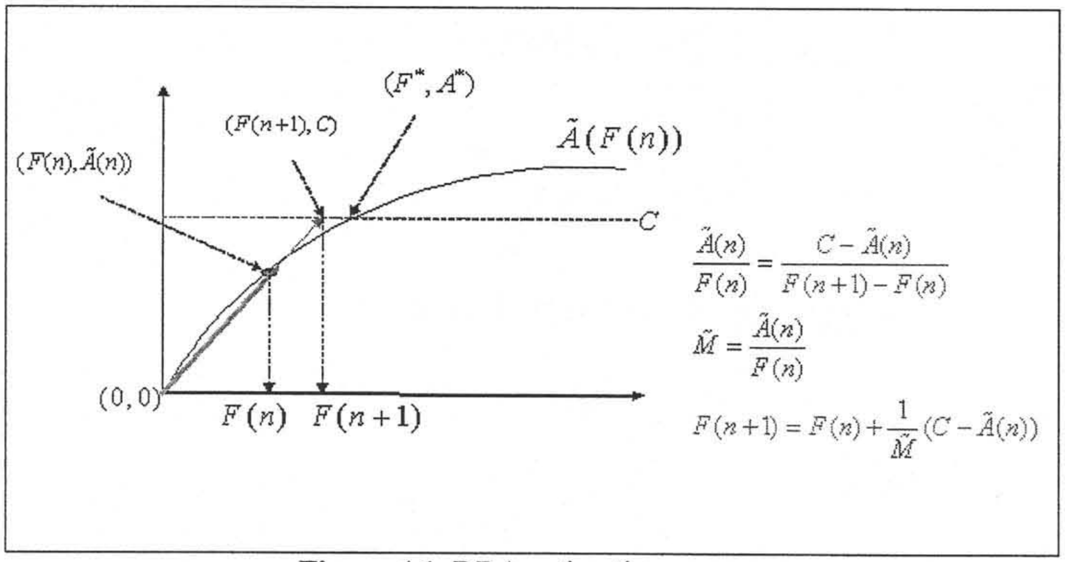
Now, the arrival rate $\tilde{A}(n)$ at link k can be expressed as a function of the link fair rate $F_k(n)$ as follows:

$$\tilde{A}(n)=F_k(n)\sum_{i=1}^{M}\rho_i \quad . \tag{4.5}$$

From Eq. (4.5) we see that the arrival rate $\tilde{A}(n)$ is continuous, non-decreasing and concave function of the link fair rate $F_k(n)$.

In [5], [6], and [7], we exploit the relation between the arrival rate $\tilde{A}(n)$ and the fair rate $F_k(n)$ to estimate the next fair rate $F_k(n+1)$ at the end of every time interval T.

As shown in Figure 4.1, a line connects the current point ($F_k(n)$, $\tilde{A}(n)$) with the origin and intersects with the line representing the available bandwidth C at the new estimated fair rate $F_k(n+1)$.



(F^*, A^*)
$(F(n+1), C)$
$(F(n), \tilde{A}(n))$
$\tilde{A}(F(n))$
C
$(0,0)$
$F(n)$
$F(n+1)$
$\frac{\tilde{A}(n)}{F(n)} = \frac{C-\tilde{A}(n)}{F(n+1)-F(n)}$
$\tilde{M} = \frac{\tilde{A}(n)}{F(n)}$
$F(n+1) = F(n) + \frac{1}{\tilde{M}}(C-\tilde{A}(n))$

Figure 4.1 DBA estimation process

Define $\tilde{M}$ as the effective number of flows traversing link k. The effective number of flows is estimated by a linear function that connects the origin and the ($F_k(n)$, $\tilde{A}(n)$) point [12], [14].

$$\tilde{M} = \frac{\tilde{A}(n)}{F_k(n)}. \tag{4.6}$$

Substituting Eq. (4.5) into Eq. (4.6) yields

$$\tilde{M} = \sum_{i=1}^{M} \rho_i \quad . \tag{4.7}$$

The effective number of flows is the sum of the activity levels of flows traversing link *k* and is less than or equal to the number of flows *M*.

Now, we propose the following formula to estimate the link fair rate:

$$F_k(n+1) = F_k(n) + \frac{1}{\tilde{M}}(C - \tilde{A}(n)). \tag{4.8}$$

The goal of the fairness algorithm is to maximize the link fair rate $F_k(n)$ subject to the constraint:

$$\tilde{A}(n) \le C . \tag{4.9}$$

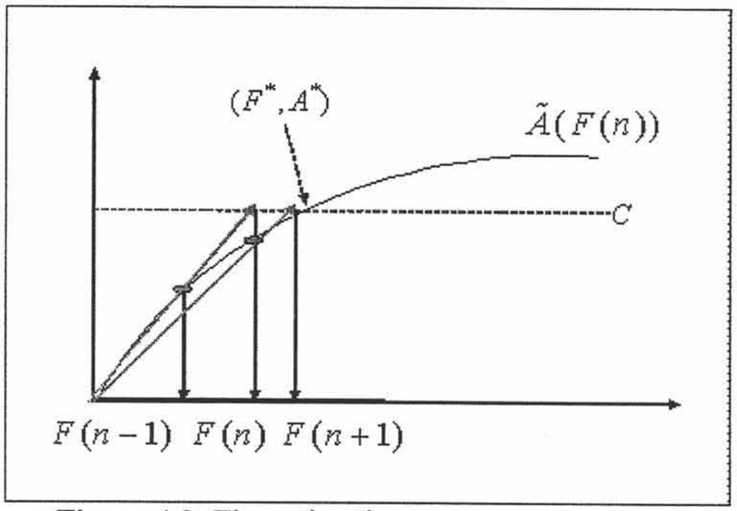


Figure 4.2 The estimation process convergence

Repeating the estimation process, Eq. (4.8), every *T* sec will generates a sequence which converges to the optimal fair rate F_k^* as illustrated in Figure 4.2.

DBA first estimates the effective number of flows, which can be estimated by the slope of the line connecting the origin and the current point ($F_k(n)$, $\tilde{A}(n)$). Then, it uses Eq. (4.8) to estimate the fair rate of the next interval. The goal is to adjust $F_k(n)$ so that the total arrival rate $\tilde{A}(n)$ matches the available bandwidth and $F_k(n)$ converges to the optimal fair rate F_k^*.

Note that one of the important features of the DBA algorithm is its low computation complexity of O(1), thus making DBA scalable for a ring network with any number of nodes, i.e., independent of *N*. Moreover, DBA dose not require per-source information as in DVSR and RPR-CM.

Figure 4.3 shows the pseudo code of DBA, where the total arrival rate $\tilde{A}(n)$ is updated at the arrival of every packet traversing link *k*. At the end of the measurement interval *T*, Eq. (4.6) and Eq. (4.8) are used to calculate the next advertised fair rate $F_k(n+1)$.

```
Initialization: at the beginning of every measurement interval T, reset $\tilde{A}(n)$
During the measurement interval
At the arrival of a packet from node i with length $L$ bytes
$\tilde{A}(n) = \tilde{A}(n) + L$
At the end of the measurement interval:
The effective number of active flows:    $\tilde{M} = \max(1, \frac{\tilde{A}(n)}{F_k(n)})$
The advertised fair rate:    $F_k(n+1) = F_k(n) + \frac{1}{\tilde{M}}(C - \tilde{A}(n))$
```

Figure 4.3 The DBA Pseudo code

4.1.2 The DBA Convergence

Let k be the bottlenecked link. The number of flows traversing link k is M, where M' is the number of flows bottlenecked elsewhere or at their ingress points, and $M''=M-M'$ is the number of flows bottlenecked at link k. Let $R_{b1}, R_{b2}, ..., R_{bM'}$ be the flows bottlenecked elsewhere, and $R_1, R_2, ..., R_{M''}$ be the flows bottlenecked at link k.

At the end of the *nth* measurement interval ($t=nT$), the effective number of flows is estimated as

$$\tilde{M} = \frac{\tilde{A}(n)}{F_k(n)} \quad , \tag{4.10}$$

where $\tilde{A}(n) = \sum_{i=1}^{M} R_i$ is the arrival rate at node k, and $F_k(n)$ is the advertised fair rate of the previous interval.

The next advertised fair rate is

$$F_k(n+1) = F_k(n) + \frac{1}{\tilde{M}}(C - \tilde{A}(n)). \tag{4.11}$$

Substituting Eq. (4.10) into Eq. (4.11) yields

$$F_k(n+1) = F_k(n)\frac{C}{\tilde{A}(n)} \quad . \tag{4.12}$$

Define $\alpha(n) = \frac{\tilde{A}(n)}{C}$ as the load factor, and rewrite Eq. (4.12) as

$$F_k(n+1) = \frac{F_k(n)}{\alpha(n)} \quad . \tag{4.13}$$

According to the load factor value, two cases are considered. First, consider the case where the load factor $\alpha(n)$ is less than one. In this case, the arrival rate is less than the available bandwidth and the link is under-loaded. According to Eq. (4.13), the advertised fair rate will increase. If all flows are bottlenecked elsewhere ($M''=0$), the fair rate have been achieved. On the other hand, if there are some flows bottlenecked at link k ($M''>0$), the bottlenecked flows will continue to increase their rates until the load factor becomes greater than or equal to one.

Second, consider the case where the load factor $\alpha(n)$ is greater than one. In this case, the arrival rate is greater than the available bandwidth and the link is over-loaded. According to Eq. (4.13), the advertised fair rate will decrease and the participating flows will decrease their rates. This will continue until the load factor becomes less than or equal to one.

It is obvious from the above two cases that the load factor oscillates around one and converges to one. Thus, in the following analysis, we assume that the load factor is close to one.

Next, we shall show that the iterative algorithm Eq. (4.11) will generate a sequence of $F_k(n)$ that
will converge to the optimal value of the advertised fair rate $F_k^* = \frac{C-\sum_{i=1}^{M'} R_{bi}}{M''}$.

Note that the iterative equation Eq. (4.11) is in the form of

$$F_k(n+1)=F_k(n)+\lambda\left[\nabla^2 D(F_k(n))\right]^{-1}\nabla D(F_k(n)) \quad . \tag{4.14}$$

That is, the link fair rate is adjusted in the direction of the gradient, where

$$\nabla D(F_k(n)) = C - \tilde{A}(F_k(n)) \quad . \tag{4.15}$$

Here, λ is a positive step size, and in our case is equal to one, and $\left[\nabla^2 D(F_k(n))\right]^{-1}$ is the inverse of the Hessian.

It is well known that the Newton method Eq. (4.11), where the gradient is scaled by the inverse of the Hessian typically converges faster than the gradient projection; see [17, pp.201].

The Hessian $\nabla^2 D(F_k(n)) = \tilde{M}$ is approximated by using two points, the current point of $(\tilde{A}(n)\ ,\ F_k(n))$ and the origin (0, 0).

Hence, the above iterative equation converges, and the stable value of the link advertised fair rate is detailed as follows:

First, assume that all the flows are bottlenecked at link k. In this case, $M' = 0$ and $M'' = M$. All flows are running at the fair rate $R_i(n) = F_k(n)$, and the total arrival rate at node k is

$$\tilde{A}(n) = F_k(n) \sum_{i=1}^{M} \rho_i \ . \tag{4.16}$$

Since all flows are bottlenecked at link k, the effective number of flows is $\sum_{i=1}^{M} \rho_i = M$.

Substituting the value of $\tilde{A}(n)$ into Eq. (4.12) with a load factor $\alpha(n)$ of one at the steady state yields

$$F_k(n+1) = \frac{C}{M} \ , \tag{4.17}$$

which is the desired value for F_k .

Finally, assume that some flows are bottlenecked elsewhere. These flows will have their rates $R_{b1}, R_{b2}, \ldots, R_{bM'}$ stabilized, and the allocated bandwidth for these flows is $B = \sum_{i=1}^{M'} R_{bi}$.

Since we have a load factor $\alpha(n)$ of one at the steady state, we have

$$\sum_{i=1}^{M''} R_i = C - \sum_{i=1}^{M'} R_{bi} \ , \tag{4.18}$$

and

$$\sum_{i=1}^{M'} R_i = M'' F_k(n) \ . \tag{4.19}$$

Substituting Eq. (4.19) into Eq. (4.18) yields

$$F_k(n) = \frac{C - \sum_{i=1}^{M'} R_{bi}}{M''} \ . \tag{4.20}$$

Substituting the value of $F_k(n)$ into Eq. (4.13) yields

$$F_k(n+1) = \frac{C - \sum_{i=1}^{M'} R_{bi}}{M''} \quad , \tag{4.21}$$

which is indeed the desired value for F_k and the proof is complete . ■

4.1.3 Fair Rate Advertisement

At the end of every measurement interval *T*, every node *k* will broadcast a control message containing the value of the last computed fair rate F_k. Thus, every node is aware of the supported fair rates at all links.

4.1.4 Rate Limiting and Per Flow Sub-allocation

Upon receiving all the last computed fair rates, the node itself will do sub-allocation for all the flows that are sharing the same link and are destined to different egress nodes. This will be the topic of Chapter 6.

4.2 The DBA algorithm performance

In this simulation, we have considered three scenarios. First, we consider the parking lot scenario to show the performance of our algorithm in achieving fairness. Second, we demonstrate convergence of our algorithm even in the unbalanced traffic scenario. Finally, we study the performance of the DBA algorithm in the presence of different traffic models. All simulation results are obtained by using the RPR simulator [19].

4.2.1 Parking Lot Scenario

Figure 4.4 shows the parking lot scenario [10], [11]. In this experiment, we compare the convergence time of the fairness algorithms. The links have the same capacity of 622 Mbps, and each link has a propagation delay of 0.1 ms. All flows are UDP flows, with a rate equal to 250 Mbps. The flows, flow (1,5), flow(2,5), flow(3,5) and flow(4,5) start at time 0, 0.1, 0.2 and 0.3 seconds, respectively. The measurement time interval was set to $T = 1$ ms.

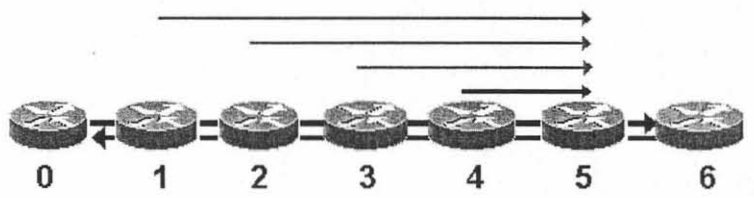


Figure 4.4 Parking Lot scenario

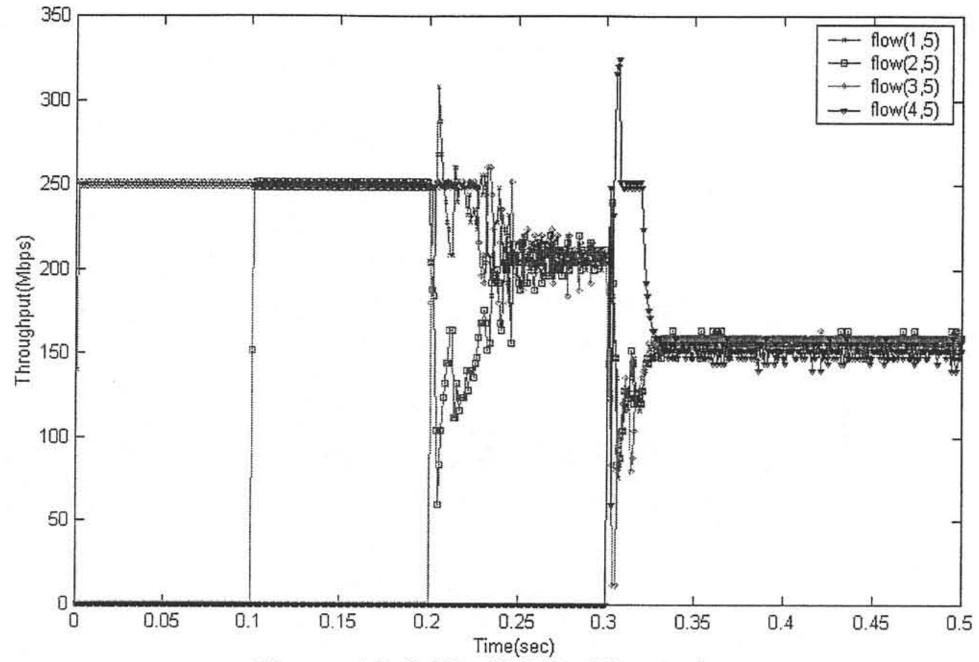


Figure 4.5 RPR-AM (Parking Lot)

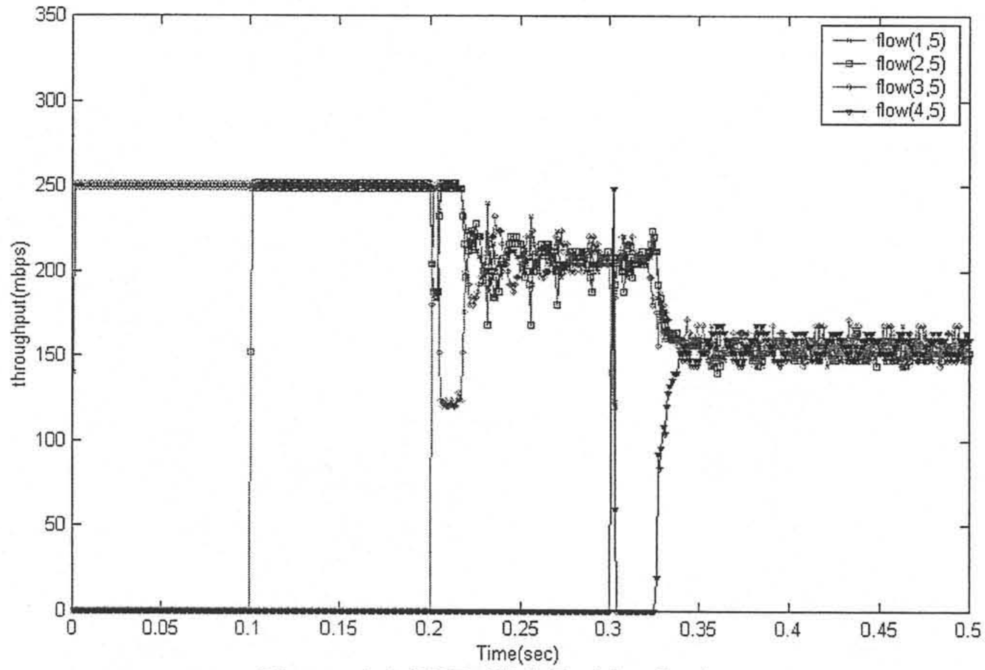


Figure 4.6 RPR-CM (Parking Lot)

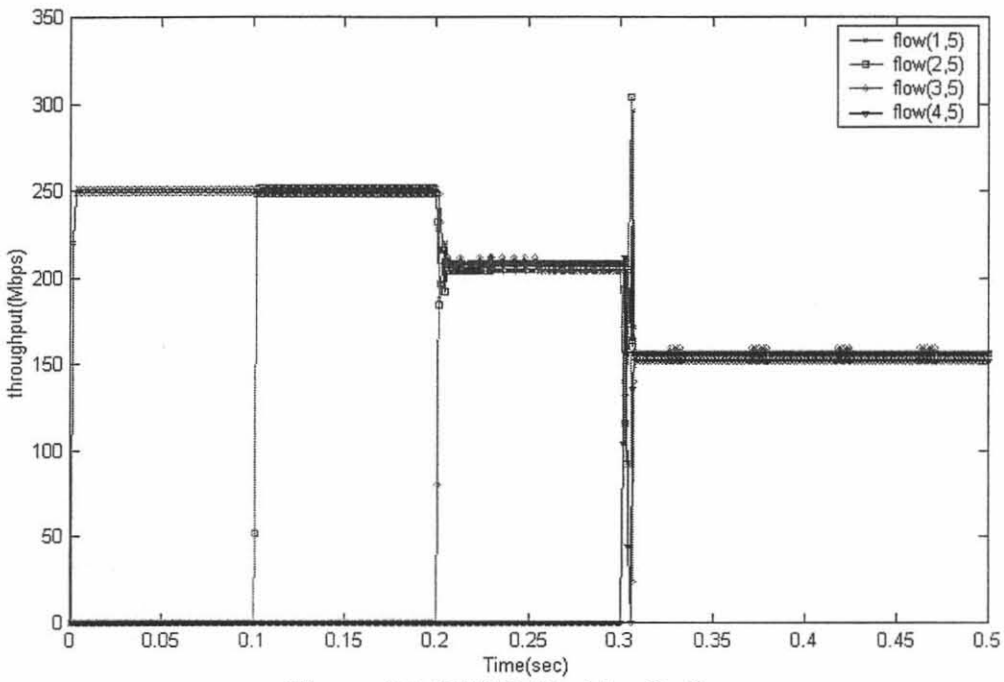


Figure 4.7 DVSR (Parking Lot)

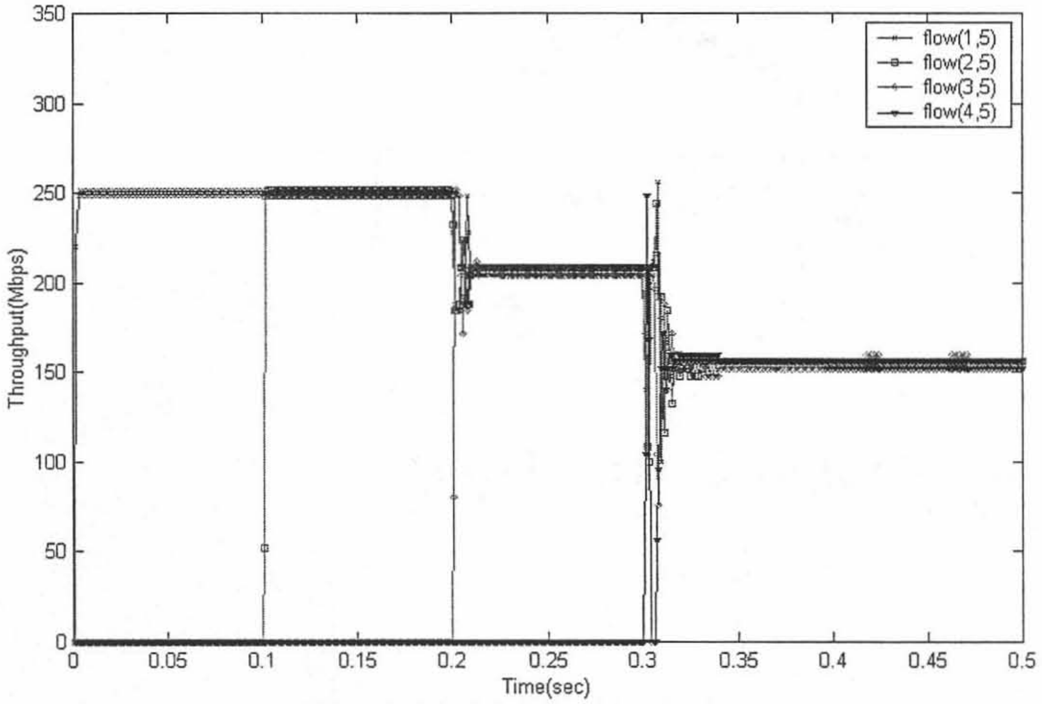


Figure 4.8 DBA (Parking Lot)

The performance of the RPR fairness algorithms RPR-AM and RPR-CM are shown in Figure 4.5 and 4.6, respectively. In both algorithms, flows oscillate for a significant period of time before converging to the fair rate. Moreover, the range of oscillation is large.

The results shown in Figure 4.7 exhibit that DVSR needs only a few milliseconds to converge to the RIAS faire rate, however, at the expense of $O(N \log N)$ computational complexity and requiring per-source information.

Results shown in Figure 4.8 have verified that DBA converges to the RIAS fair rate in a few measurement intervals with a very low computational complexity of $O(1)$, and it does not require per-source information as compared to DVSR [10]. The oscillation has also been significantly reduced.

4.2.2 Available Bandwidth Re-claim Scenario

In this experiment, we consider the scenario [10] illustrated in Figure 4.9, where all flows are greedy and start at time t=0.

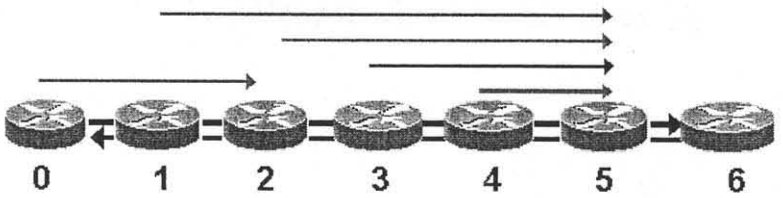


Figure 4.9 Available Bandwidth Re-claim Scenario

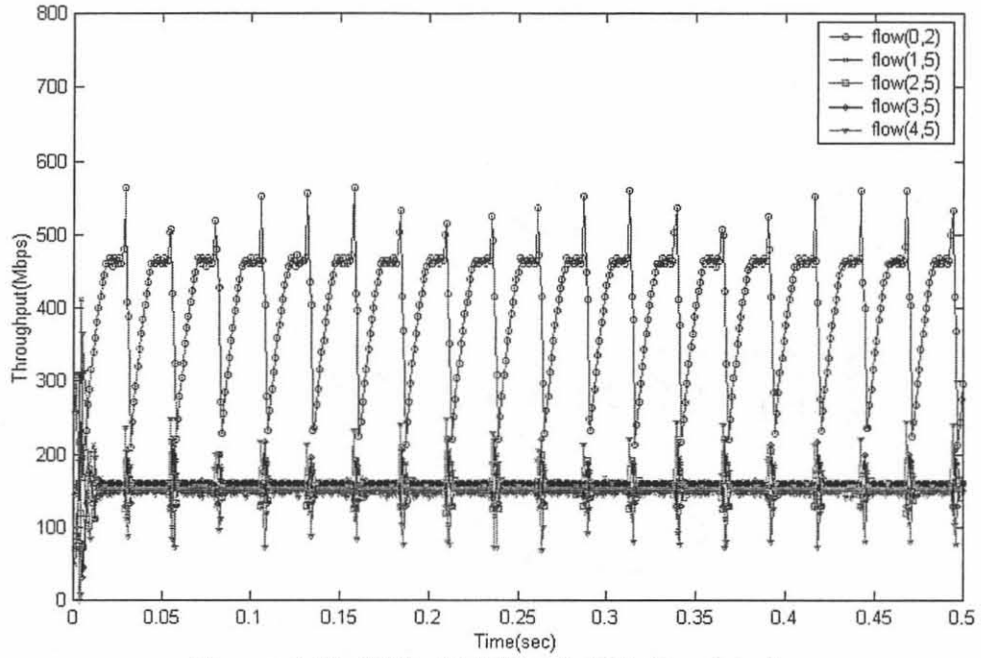


Figure 4.10 RPR-AM (Bandwidth Re-claim)

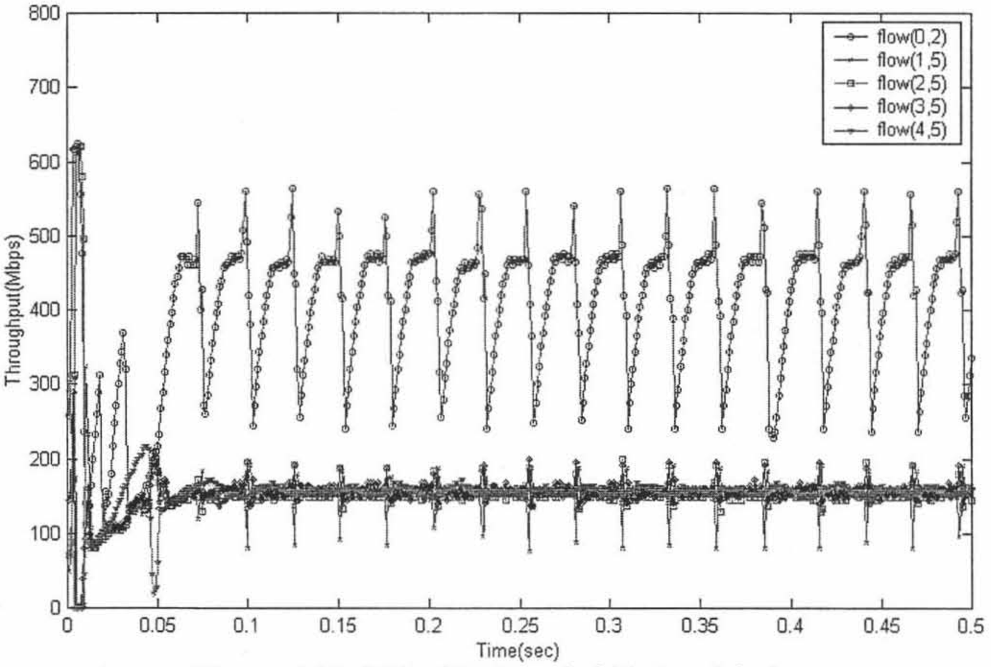


Figure 4.11 RPR-CM (Bandwidth Re-claim)

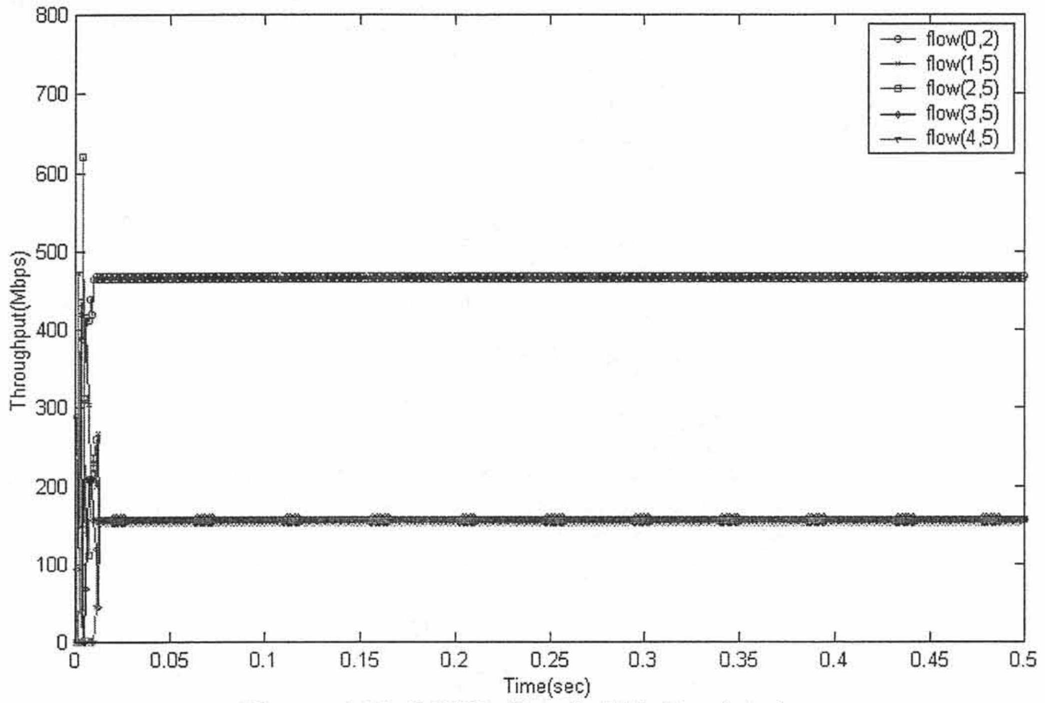


Figure 4.12 DVSR (Bandwidth Re-claim)

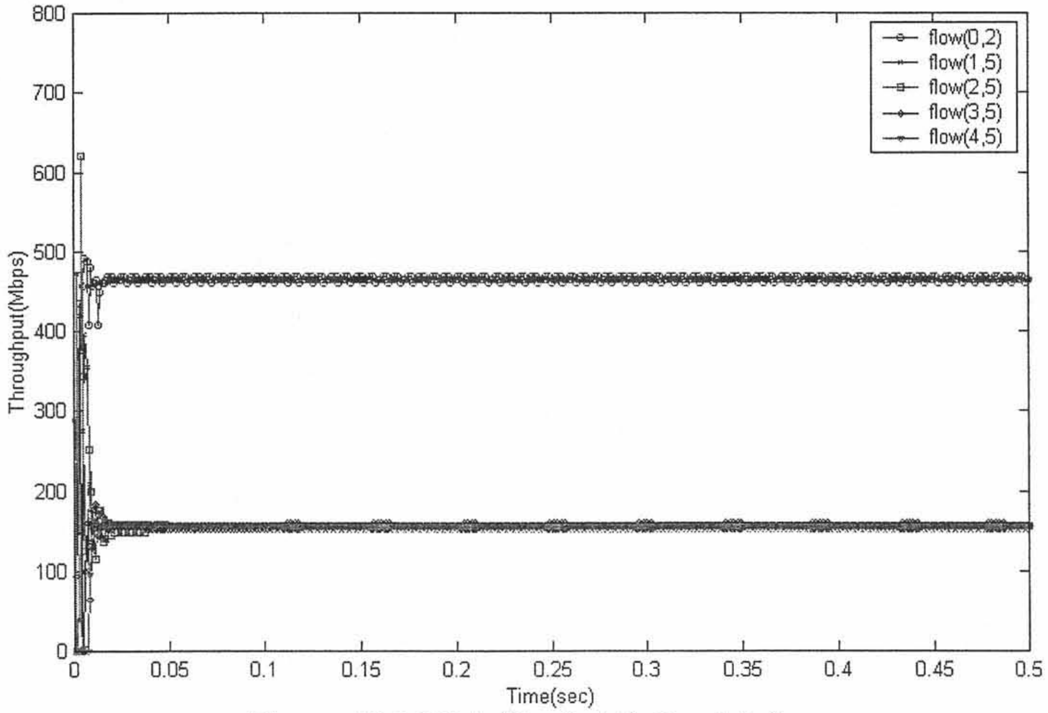


Figure 4.13 DBA (Bandwidth Re-claim)

Figure 4.10 shows the RPR-AM algorithm where all the flows (0,2), (1,5), (2,5), (3,5) and (4,5) start at time 0 seconds. After some time, due to the congestion experienced at link 4, flows (1,5), (2,5), (3,5) and (4,5) will converge to the fair rate (155.5 Mbps), meanwhile node 0 starts to reclaim the unused bandwidth at link 1. When node 1 becomes congested, it sends *my-rate* value of 155.5 Mbps to node 0, thus throttling flow (0,2) to 155.5 Mbps. When the congestion at node 1 is cleared, node 0 starts to increase its rate again starting another cycle of oscillation.

On the other hand, using the RPR-CM algorithm(Figure 4.11), node 1 will send *my-rate* value equal to the available bandwidth divided by the number of sources using link 1 (two in this case). Thus, flow (0,2) will be throttled to 311 Mbps. When the congestion at node 1 is cleared, node 0 starts to increase its rate again starting another cycle of oscillation.

Figure 4.12 shows that DVSR converges very fast to the RIAS faire rates at the expense of a high computational complexity and the need for per-source information.

Using the DBA algorithm, nodes converge very fast to the RIAS fair rates. Moreover, the oscillation is significantly damped, as shown in Figure 4.13.

4.2.3 Different Traffic Models

In this experiment (Figure 4.14), the congested link is shared by different traffic models. Flows (3,5) and (4,5) are greedy UDP flows and start at time 0.1 and 0.2 seconds, respectively. Flow (0,5) is an ON/OFF flow. During the ON period, flow (0,5) sends at a rate equal to 50 Mbps.

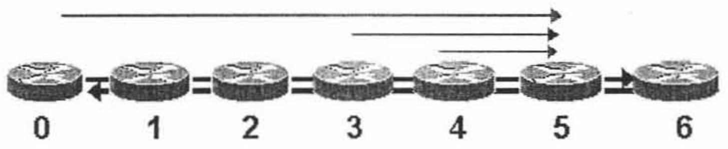


Figure 4.14 Different traffic models scenario

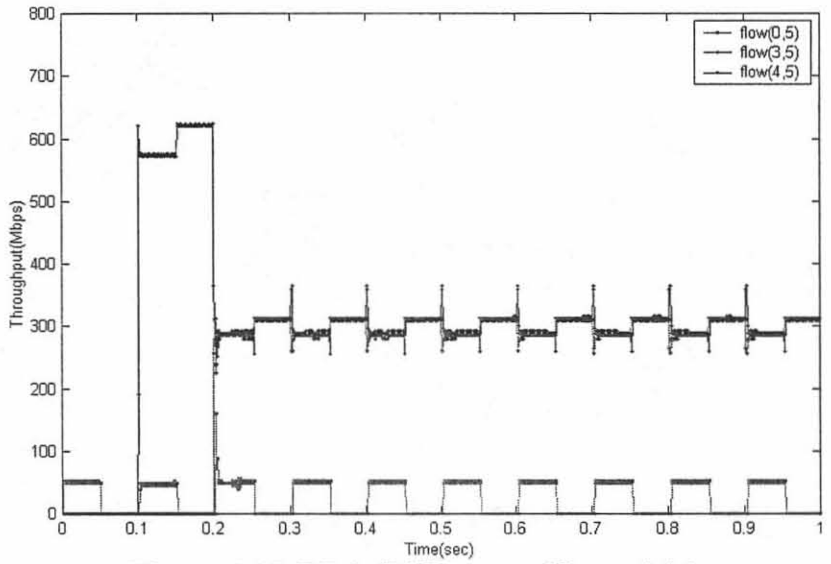


Figure 4.15 DBA (Different traffic models)

Figure 4.15 shows that our proposed algorithm reacts responsively to the presence of the ON/OFF flow, and converges very fast to the RIAS fair rates.

4.3 Achieving Faster Rate of Convergence

The DBA rate of convergence depends on the value of $\tilde{M}$. The value of $\tilde{M}$ is equal to the number of flows bottlenecked at link k plus the sum of fractions representing the flows bottlenecked elsewhere as follows:

$$\tilde{M} = \frac{\tilde{A}(n)}{F(n)} \quad . \tag{4.22}$$

The arrival rate $\tilde{A}(n)$ at link k is the sum of all flows traversing link k:

$$\tilde{A}(n) = \sum_{i \in l_u} R_i(n) + \sum_{i \in l_b} R_i(n) \quad . \tag{4.23}$$

where l_u and l_b are the set of flows bottlenecked at link k and the set of flows bottlenecked elsewhere, respectively.

$$\tilde{M} = \frac{\sum_{i \in l_u} R_i(n) + \sum_{i \in l_b} R_i(n)}{F(n)} \quad , \tag{4.24}$$

Each node i will transmit through link k at a rate according to the received fair rate from node k. Thus, the rate of source i through link k at time n is:

$$R_i(n) = \rho_i F_k(n), \tag{4.25}$$

where ρ_i is the activity level of source i with respect to the fair rate, $F_k(n)$, of the current interval. The activity level is equal to one for flows bottlenecked at link k, and less than one for flows bottlenecked elsewhere.

$$\tilde{M} = \frac{F(n)(\sum_{i \in l_u} 1 + \sum_{i \in l_b} \rho_i)}{F(n)} \quad , \tag{4.26}$$

$$= \sum_{i \in l_u} 1 + \sum_{i \in l_b} \rho_i \quad . \tag{4.27}$$

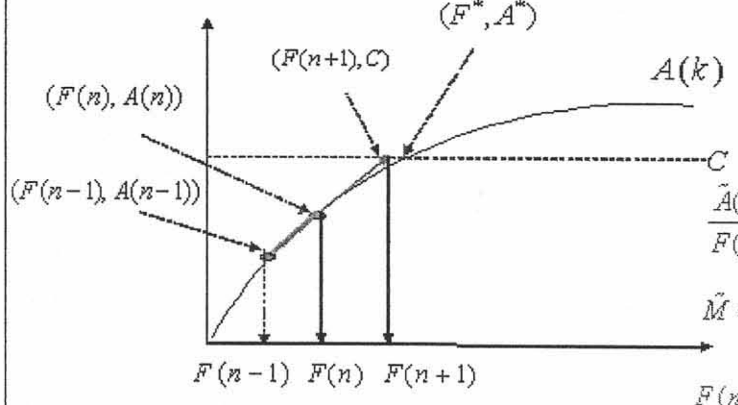
The link fair rate $F_k(n)$ is estimated by using the following formula:

$$F_k(n+1) = F_k(n) + \frac{C - \tilde{A}(n)}{\sum\limits_{i \in l_u} 1 + \sum\limits_{i \in l_b} \rho_i}. \tag{4.28}$$

It is clear that the rate of convergence will be faster if we divide the unused bandwidth $(C - \tilde{A}(n))$ by the number of flows bottlenecked at link k as follows:

$$F_k(n+1) = F_k(n) + \frac{C - \tilde{A}(n)}{\sum\limits_{i \in l_u} 1}. \tag{4.29}$$

In this section, we are proposing the estimation process illustrated in Figure 4.16. As shown in Figure 4.16, a line connects the current point ($F_k(n)$, $\tilde{A}(n)$) with the previous point ($F_k(n-1)$, $\tilde{A}(n-1)$) and intersects with the line representing the available bandwidth C at the new estimated fair rate $F_k(n+1)$.



$$\frac{\tilde{A}(n) - \tilde{A}(n-1)}{F(n) - F(n-1)} = \frac{C - \tilde{A}(n)}{F(n+1) - F(n)}$$

$$\tilde{M} = \frac{\tilde{A}(n) - \tilde{A}(n-1)}{F(n) - F(n-1)}$$

$$F(n+1) = F(n) + \frac{1}{\tilde{M}}(C - \tilde{A}(n))$$

Figure 4.16 The Improved DBA algorithm

The value of $\tilde{M}$ is equal to the number of flows bottlenecked at link k as follows:

$$\tilde{M} = \frac{\tilde{A}(n) - \tilde{A}(n-1)}{F_k(n) - F_k(n-1)} \quad . \tag{4.30}$$

Substituing Eq. (4.23) into Eq. (4.30) yields

$$\tilde{M} = \frac{\sum_{i \in l_u} R_i(n) + \sum_{i \in l_b} R_i(n) - (\sum_{i \in l_u} R_i(n-1) + \sum_{i \in l_b} R_i(n-1))}{F_k(n) - F_k(n-1)} \quad . \tag{4.30}$$

All flows bottlenecked else where will not change their rates, and thus $\tilde{M}$ is equal to

$$\tilde{M} = \frac{\sum_{i \in l_u} R_i(n) - \sum_{i \in l_u} R_i(n-1)}{F_k(n) - F_k(n-1)} \quad , \tag{4.31}$$

$$= \frac{\sum_{i \in l_u} R_i(n) - R_i(n-1)}{F_k(n) - F_k(n-1)} \quad . \tag{4.32}$$

All flows bottlenecked at link k will transmit at rate equal to the received fair rate $F(n)$ and the value of $\tilde{M}$ is

$$\tilde{M} = \frac{\sum_{i \in l_u} F_k(n) - F_k(n-1)}{F_k(n) - F_k(n-1)} = N_u \quad . \tag{4.33}$$

Here, $\tilde{M}$ is representing the number of flows bottlenecked at link k, N_u, and the next fair rate $F_k(n+1)$ is estimated as

$$F_k(n+1) = F_k(n) + \frac{C - \tilde{A}(n)}{N_u} . \tag{4.34}$$

The iterative formula Eq. (4.34) will generate a sequence that converges to the optimal fair rate faster than DBA.

4.4 Achieving Low Buffer Occupancy and Minimum End to End Delay

The goal of the fair rate estimation formula Eq. (4.35) is to adjust $F_k(n)$ so that the total arrival rate $\tilde{A}(n)$ matches the available bandwidth and $F_k(n)$ converges to the optimal fair rate F_k^*.

$$F_k(n+1) = F_k(n) + \frac{1}{\tilde{M}}(C - \tilde{A}(n)). \tag{4.35}$$

During the convergence time period, the transit queue length increases due to congestion, and thus the packet's end-to-end delay will increase if the packet traverses through several congestion points.

To clear the transit buffers and achieve minimum end-to-end delay, we add the transit queue length $B(n)$ in Eq. (4.35) as follows:

$$F_k(n+1) = F_k(n) + \frac{1}{\tilde{M}}(C - \tilde{A}(n) - \lambda B(n)). \tag{4.36}$$

The goal of the fair rate estimation formula Eq. (4.36) is to adjust $F_k(n)$ so that the transit queue is cleared and the total arrival rate $\tilde{A}(n)$ matches the available bandwidth.

We view the queue length as noise resulting from the initiation of new flows, and thus λ acts as a low-pass filter coefficient to avoid introducing oscillation at the steady state.

4.5 Performance Evaluation

In this experiment, we consider the simple parking lot scenario shown in Figure 4.17 to evaluate the existing fairness algorithms in achieving low buffer occupancy and minimum end-to-end delay. The flow from node 0 to node 5 is greedy while the flow from node 4 to node 5 is a low rate of 50 Mbps. All links have a capacity of 622Mbps and each link has a propagation delay of 0.1 ms. The transit buffers are 256KB and the size of each data packet is 500 Byte.

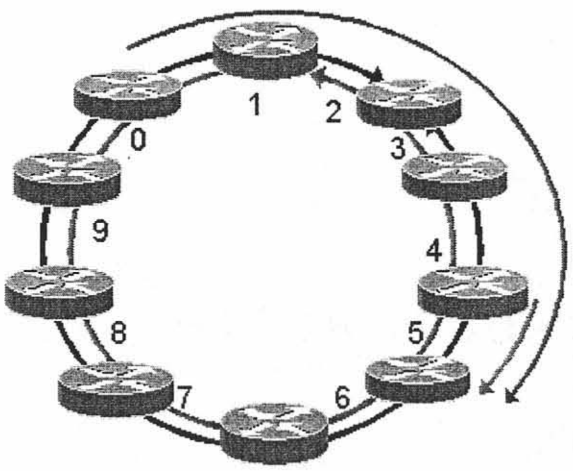


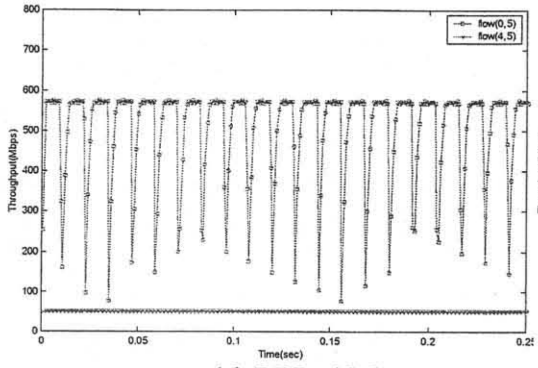
Figure 4.17 Low buffer occupancy scenario

The performance of all algorithms is shown in Figure 4.18. The RPR fairness algorithms suffer from severe oscillations due to the unbalanced traffic at link 4. On the other hand, DVSR, DBA and the Improved DBA converge very fast to the optimal fair rates.

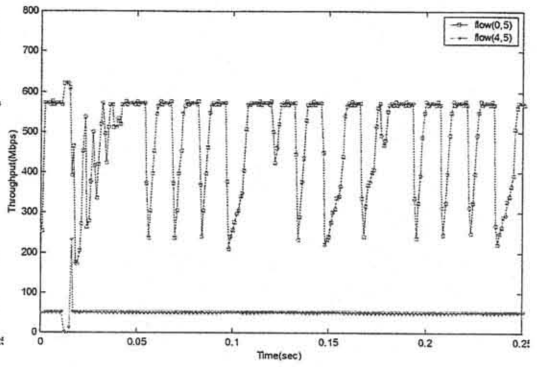
Figure 4.19 shows that using the RPR fairness algorithm result in an oscillation in the transit buffer length. This oscillation in the transit buffer length increases the end-to-end delay and delay jitter as shown in Figure 4.20.

DVSR and DBA are able to achieve a constant transit buffer length (near the High threshold) and accordingly packets have a constant end-to-end delay. The end-to-end delay for flow (0,5) packets is equal to the propagation delay(500 usec) plus the queuing time at link 4 as shown in Figure 4.20.

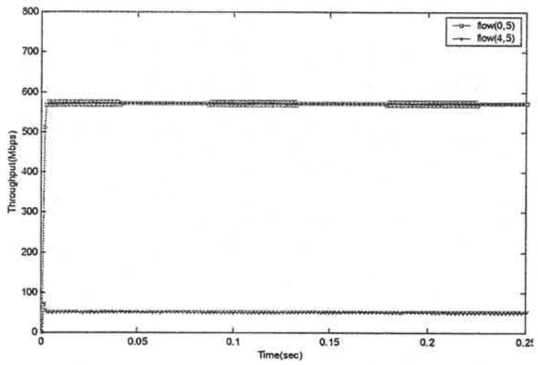
The Improved DBA algorithm converges very fast to the optimal fair rate and clears the transit buffer (Figure 4.19), and is thus able to achieve the minimum end-to-end delay. The end-to-end delay for flow (0,5) packets is equal to the propagation delay (500 usec) plus the processing time at each link as shown in Figure 4.20.



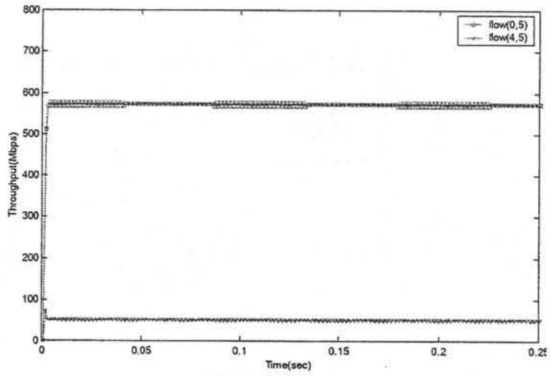
(a) RPR-AM



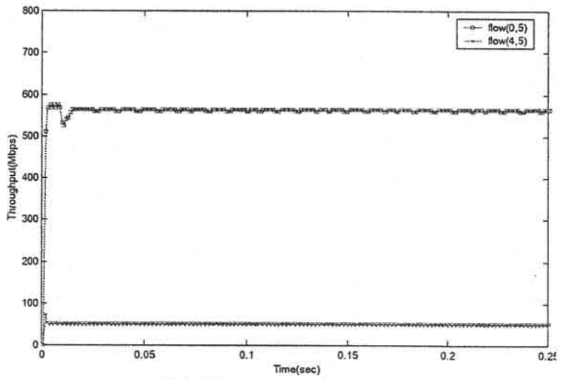
(b) RPR-CM



(c) DVSR

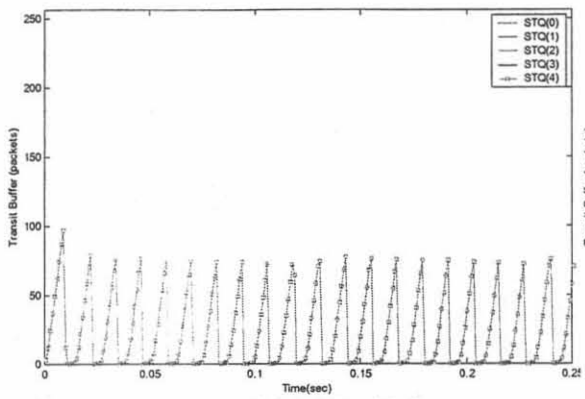


(d) DBA

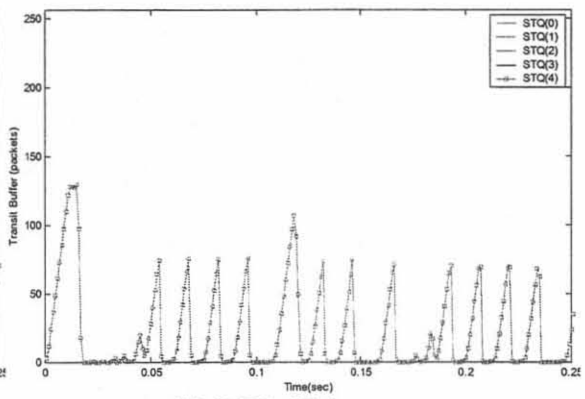


(e) Improved DBA

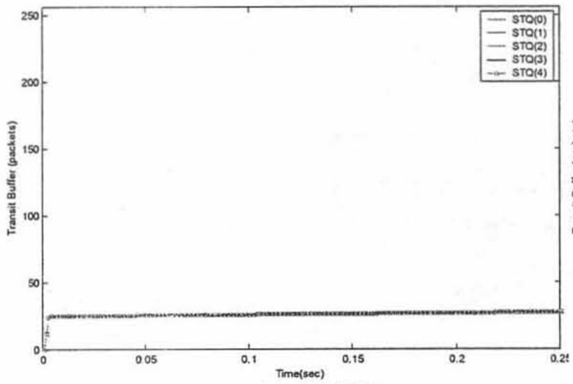
Figure 4.18 Low buffer occupancy scenario (throughput)



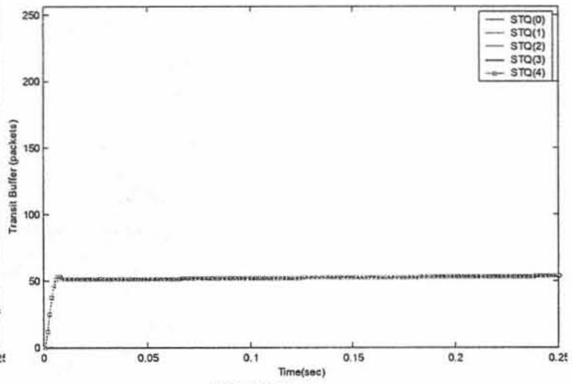
(a) RPR-AM



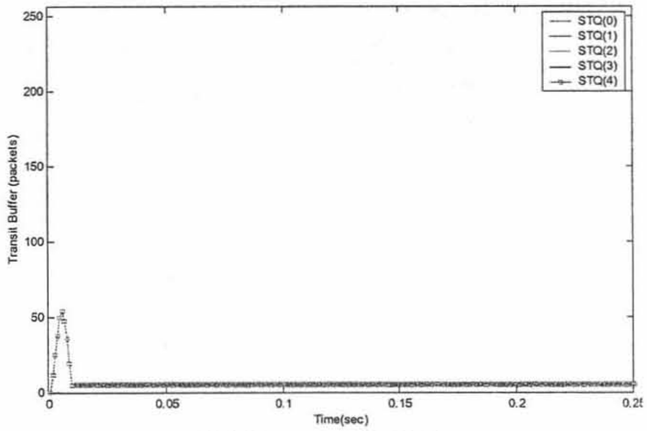
(b) RPR-CM



(c) DVSR

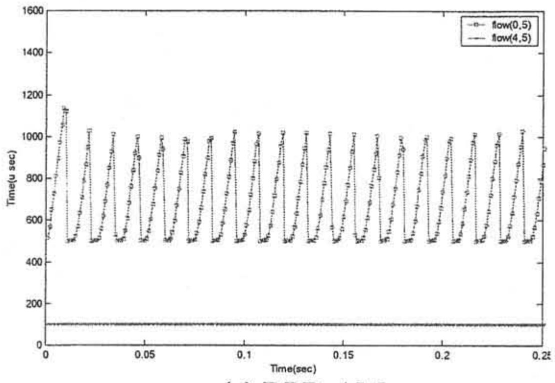


(d) DBA

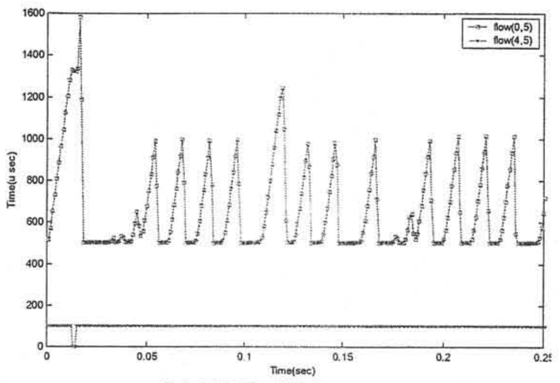


(e) Improved DBA

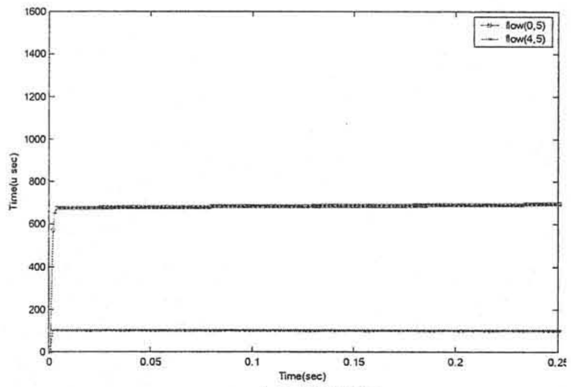
Figure 4.19 Low buffer occupancy scenario (buffer length)



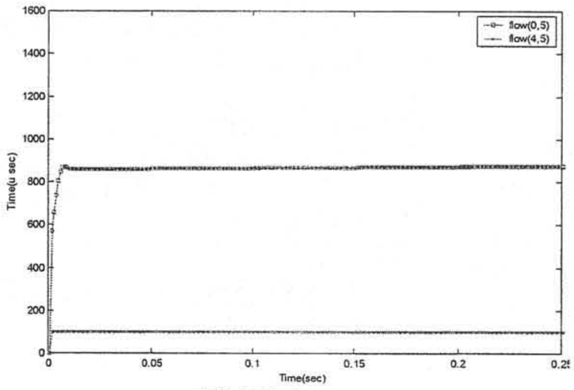
(a) RPR-AM



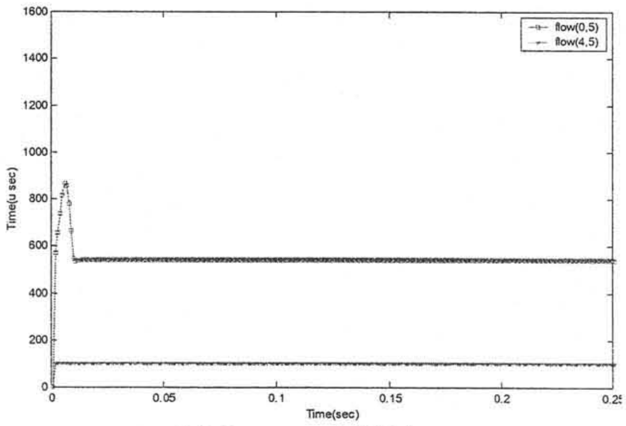
(b) RPR-CM



(c) DVSR



(d) DBA



(e) Improved DBA

Figure 4.20 Low buffer occupancy scenario (packet delay)

4.6 Summary

In this chapter, we have proposed the Distributed Bandwidth Allocation (DBA) algorithm for Resilient Packet Ring to achieve spatial reuse, high bandwidth utilization, and fairness. Unlike DVSR [10] and RPR-CM [7], DBA does not require per-source information and converges to the RIAS fair rates in a few measurement intervals with a very low computational complexity, i.e. $O(1)$, and is thus scalable.

Also, the Improved DBA algorithm is proposed to achieve low buffer occupancy and minimum end-to-end delay.

CHAPTER 5

ADAPTIVE BANDWIDTH ALLOCATION

In the current RPR fairness algorithms, RPR-AM and RPR-CM, nodes increase their fair rates to claim the unused bandwidth as follows:

$$local\_fairrate = local\_fairrate + \alpha(unreserved\_rate - local\_fairrate). \qquad (5.1)$$

where α is the rate adjusting factor and equal to $\frac{1}{rampcoef}$

The *rampcoef* value is a constant set at the configuration step and does not reflect the state of congestion. Thus, nodes increase their rates till the network reaches the state of congestion.

At the state of congestion, each node decrease its *local_fairrate* based on the mode of operation AM or CM. When the congestion is cleared, nodes will increase their rates again to claim the unused bandwidth, thus starting another cycle of oscillation.

In this chapter, we propose an Adaptive Bandwidth Allocation algorithm (ABA) [13] to dynamically set the value of the adjusting rate factor α .

5.1 Adaptive Bandwidth Controller

In this section, we develop an adaptive controller (Figure 5.1) which reads the instantaneous length of the transit queue at the end of every T sec interval. The controller uses the last two measurements to adjust the fair rate. For example, if the node is not congested and the queue length decreases rapidly, which implies that the node will soon be under utilized, the fair rate should thus be increased. On the other hand, if the node is near congestion and the queue length increases, which means that the node will be congested, the fair rate should thus be decreased.

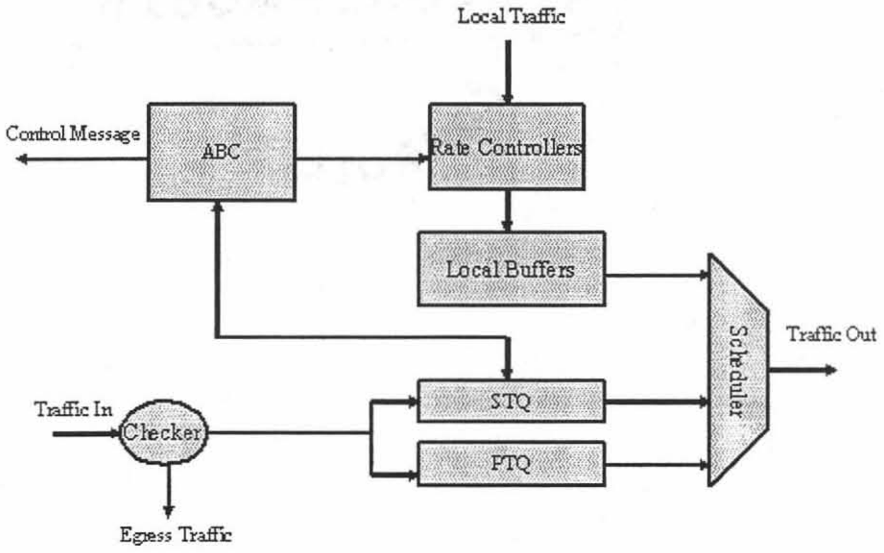


Figure 5.1 The RPR node architecture (ABC controller)

We propose to construct the Adaptive Bandwidth Controller (ABC) by the means of fuzzy logics. The generic form of the controller illustrated in Figure 5.2 is adopted from the functional fuzzy system described in [20].

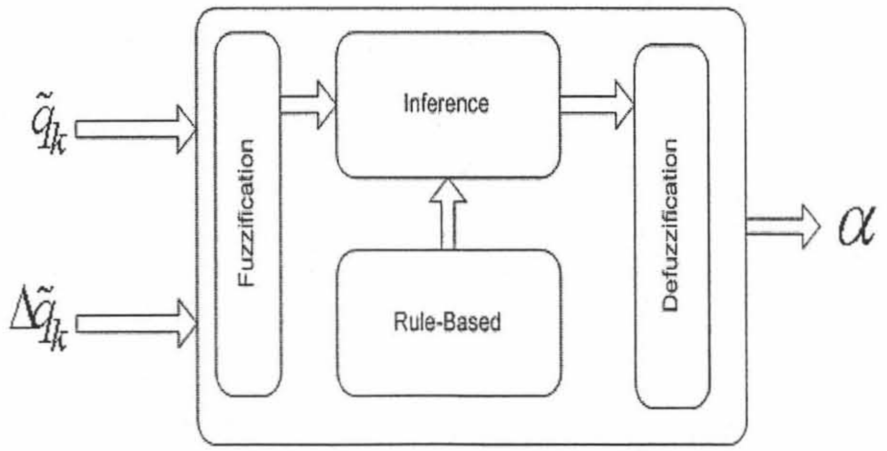


Figure 5.2 The fuzzy controller

The operation of ABC consists of the following.

Measurement: At the end of the kth controlling time period $(t = kT)$, the system will read the instantaneous queue length of the Transit Queue (q_k) .The first input to the fuzzy controller is the low pass filtered value of the queue length

$$\tilde{q}_k = (1-\beta)q_k + \beta\tilde{q}_{k-1}, \tag{5.3}$$

where the value of β is chosen to be small because the current state of the queue size is emphasized for faster reaction at the state of congestion.

The second input to the fuzzy controller is the queue growth rate

$$\Delta\tilde{q}_k = \tilde{q}_k - \tilde{q}_{k-1}. \tag{5.4}$$

These two input values will be normalized with respect to the Transit Queue size before being fed to the controller.

Fuzzification: This is the process of mapping the system input i to a set of linguistic values with the corresponding membership functions. First, the normalized queue length $\tilde{q}_k$ is mapped into three linguistic values $A^1_{\tilde{q}_k}$ (Low), $A^2_{\tilde{q}_k}$ (medium), and $A^3_{\tilde{q}_k}$ (High) through their corresponding membership functions $\mu_{A^1_{\tilde{q}k}}$, $\mu_{A^2_{\tilde{q}k}}$, and $\mu_{A^3_{\tilde{q}k}}$, respectively, as shown in Figure 5.3. Second, the normalized queue growth rate $\Delta\tilde{q}_k$ is mapped into five linguistic values $A^1_{\Delta\tilde{q}_k}$ (Large Decrease (LD)), $A^2_{\Delta\tilde{q}_k}$ (Slow Decrease (SD)), $A^3_{\Delta\tilde{q}_k}$ (No Change (NC)), $A^4_{\Delta\tilde{q}_k}$ (Slow Increase (SI)), $A^5_{\Delta\tilde{q}_k}$ (Large Increase (LI)) through their corresponding membership functions $\mu_{A^1_{\Delta\tilde{q}k}}$, $\mu_{A^2_{\tilde{q}k}}$, $\mu_{A^3_{\tilde{q}k}}$, $\mu_{A^4_{\tilde{q}k}}$, and $\mu_{A^5_{\tilde{q}k}}$, respectively, as shown in Figure 5.4.

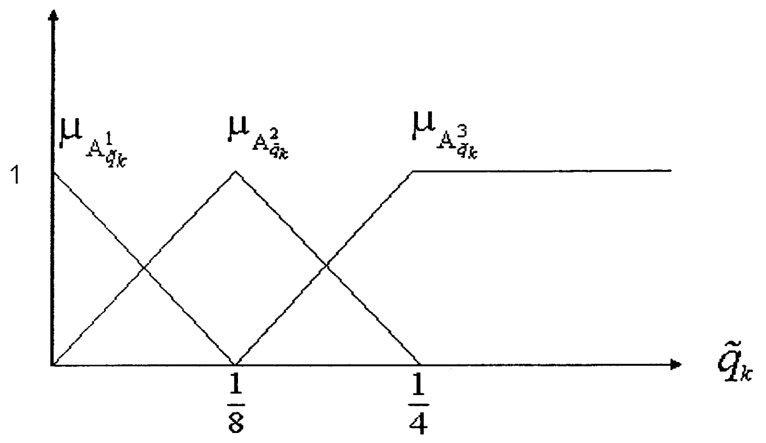


Figure 5.3 The membership functions of $\tilde{q}_k$

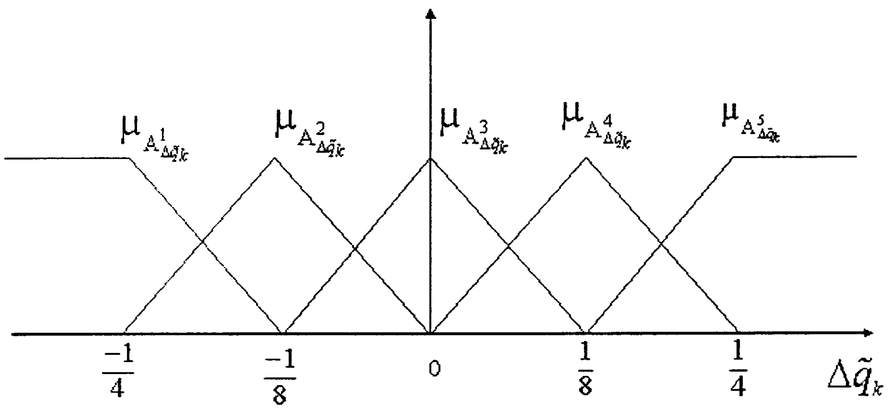


Figure 5.4 The membership functions of $\Delta\tilde{q}_k$

Inference: This is the process of applying the result of the fuzzification to a set of rules to produce a new fuzzy set that determines the controller decision. The rules are written in the form of

IF *premise* THEN *consequent*

where the fuzzy sets are only involved in the premises and the rule consequent is a crisp function. The controller has three linguistic values for the first input and five linguistic values for the second input, and thus the total number of rules is 15.

R^i : IF $\tilde{q}_k$ is $A^m_{\tilde{q}_k}$ AND IF $\Delta\tilde{q}_k$ is $A^j_{\Delta\tilde{q}_k}$ Then α is b_i

where R^i is the ith rule, $\tilde{q}_k$ and $\Delta\tilde{q}_k$ are the Controller's inputs with their linguistic values $A^m_{\tilde{q}_k}$ and $A^j_{\Delta\tilde{q}_k}$, respectively, and α is the output which takes a real numbers b_i instead of fuzzy numbers.

The values of b_i are tabulated in Table 5.1.

Rule #	*Premise* w_i	*Consequent* b_i
R^1	IF $\tilde{q}_k$ is Low AND IF $\Delta\tilde{q}_k$ is LD	$\frac{1}{4}$
R^2	IF $\tilde{q}_k$ is Low AND IF $\Delta\tilde{q}_k$ is SD	$\frac{1}{8}$
R^3	IF $\tilde{q}_k$ is Low AND IF $\Delta\tilde{q}_k$ is NC	$\frac{1}{64}$
R^4	IF $\tilde{q}_k$ is Low AND IF $\Delta\tilde{q}_k$ is SI	0
R^5	IF $\tilde{q}_k$ is Low AND IF $\Delta\tilde{q}_k$ is LI	0
R^6	IF $\tilde{q}_k$ is Medium AND IF $\Delta\tilde{q}_k$ is LD	$\frac{1}{16}$
R^7	IF $\tilde{q}_k$ is Medium AND IF $\Delta\tilde{q}_k$ is SD	0
R^8	IF $\tilde{q}_k$ is Medium AND IF $\Delta\tilde{q}_k$ is NC	$\frac{-1}{128}$
R^9	IF $\tilde{q}_k$ is Medium AND IF $\Delta\tilde{q}_k$ is SI	$\frac{-1}{64}$
R^{10}	IF $\tilde{q}_k$ is Medium AND IF $\Delta\tilde{q}_k$ is LI	$\frac{-1}{16}$
R^{11}	IF $\tilde{q}_k$ is High AND IF $\Delta\tilde{q}_k$ is LD	$\frac{-1}{32}$
R^{12}	IF $\tilde{q}_k$ is High AND IF $\Delta\tilde{q}_k$ is SD	$\frac{-1}{16}$
R^{13}	IF $\tilde{q}_k$ is High AND IF $\Delta\tilde{q}_k$ is NC	$\frac{-1}{8}$
R^{14}	IF $\tilde{q}_k$ is High AND IF $\Delta\tilde{q}_k$ is SI	$\frac{-1}{4}$
R^{15}	IF $\tilde{q}_k$ is High AND IF $\Delta\tilde{q}_k$ is LI	$\frac{-1}{4}$

Table 5.1 The Rules Table

Now, given a set of inputs and a set of rules, the controller will calculate the membership values w_i for the ith rule's premise that represents the certainty that each rule premise holds for the given inputs.

$$w_i = \mu_{A^m_{\tilde{q}_k}} * \mu_{A^j_{\Delta\tilde{q}_k}} . \tag{5.5}$$

Defuzzification: This is the process that produces a crisp control output from the result of the inference. We calculate the value of α by taking the weighted average of b_i with respect to w_i

$$\alpha = \frac{\sum_{i=1}^{15} w_i * b_i}{\sum_{i=1}^{15} w_i}. \tag{5.6}$$

The definition of the input membership functions results in the sum of the premises being always equal to one. Thus, the calculation of the rate adjusting factor α is further simplified.

$$\sum_{i=1}^{15} w_i = 1, \tag{5.7}$$

$$\alpha = \sum_{i=1}^{15} w_i * b_i. \tag{5.8}$$

Figure 5.5 shows the surface of the rate adjusting factor α as a function of the controller inputs.

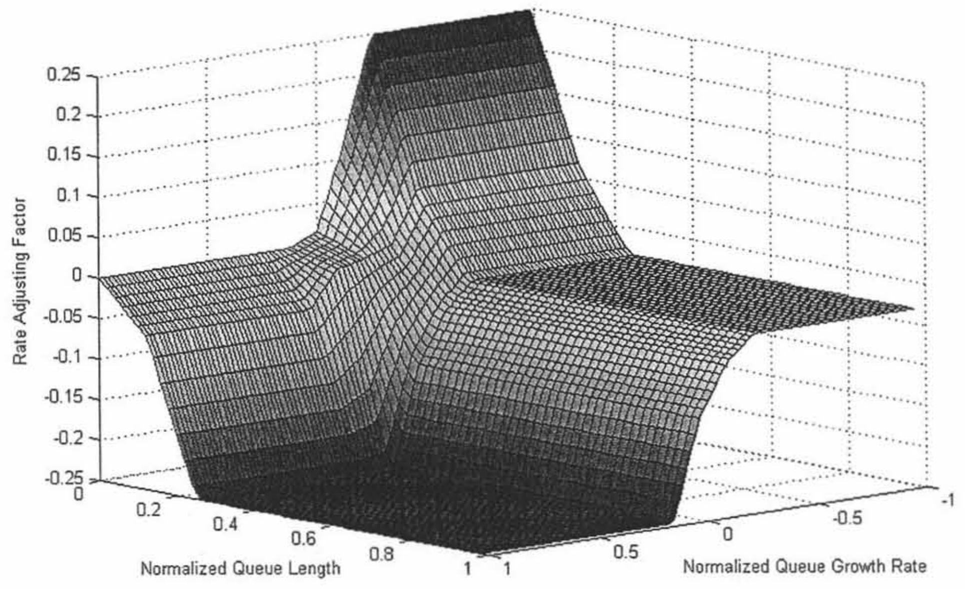


Figure 5.5 The rate adjusting factor α

5.2 Fair Rate Adjusting

The output of the fuzzy controller α will be used to adjust the next advertised fair rate F_k as follows

$$F_k = \begin{cases} (1+\alpha)F_{k-1} & \text{if } \alpha < 0 \\ F_{k-1} & \text{if } \alpha = 0 \\ F_{k-1} + \alpha(C - F_{k-1}) & \text{if } \alpha > 0 \end{cases} \tag{5.9}$$

where C is the available bandwidth and F_{k-1} is the fair rate of the previous interval .

5.3 Simulation Results

In this experiment, we have considered two scenarios. First, we consider the parking lot scenario to show the performance of our adaptive bandwidth allocation algorithm in achieving fairness. Finally, we demonstrate convergence of our algorithm even in the unbalanced traffic scenario. All simulation results are obtained by using the RPR simulator [19].

5.3.1 Parking Lot Scenario

In this experiment, we study the convergence time of our adaptive fairness algorithm using the same parking lot scenario illustrated in Figure 4.4.

Results shown in Figure 5.6 have verified that ABA converges to the RIAS fair rate in a few measurement intervals using the same information available to the current RPR fairness algorithm (RPR-AM).

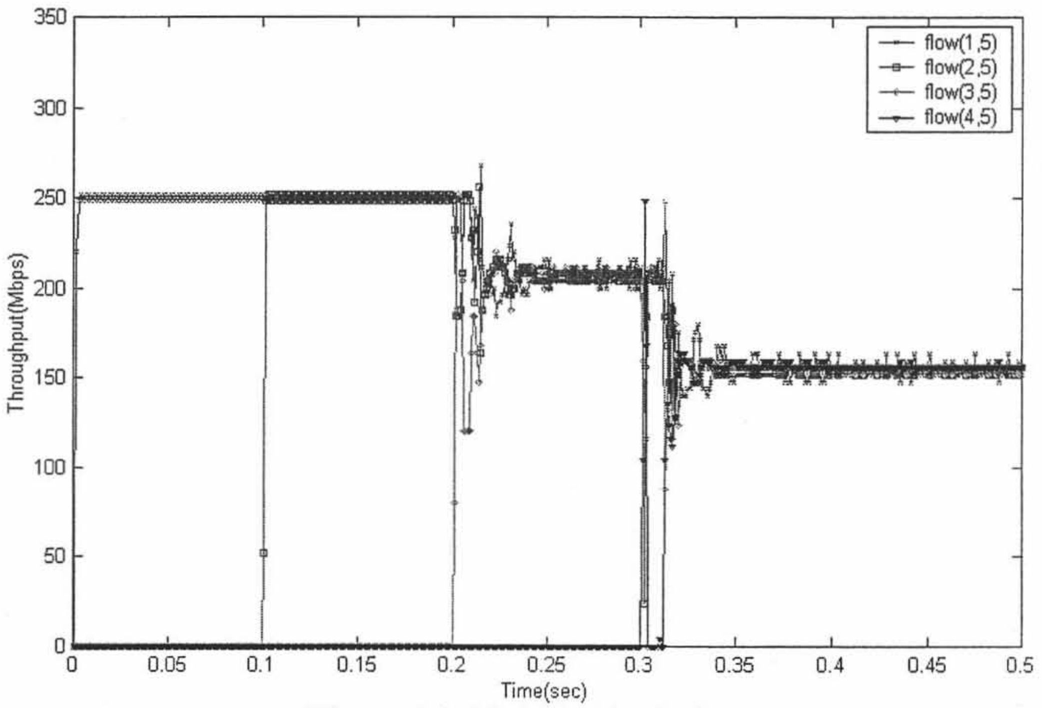


Figure 5.6 ABA (Parking Lot)

5.3.2 Available Bandwidth Re-claim Scenario

In this experiment, we use the same parallel parking lot scenario (Figure 4.9) to compare the convergence of our adaptive fairness algorithm and the current RPR fairness algorithm (RPR-AM). The controlling time interval for the ABA algorithm was set to $T = 1$ ms and the aging interval time for the RPR algorithm was set to 0.1ms. The value of *LpCoef* was set to its default value of 64 while *RampCoef* was set to 16, 64 and 512, respectively, for the RPR-AM algorithm.

As shown in Figures 5.7-5.9, the RPR algorithm suffers from a permanent oscillation resulting in a bandwidth loss. The range of oscillation is sensitive to the value of *RampCoef*.

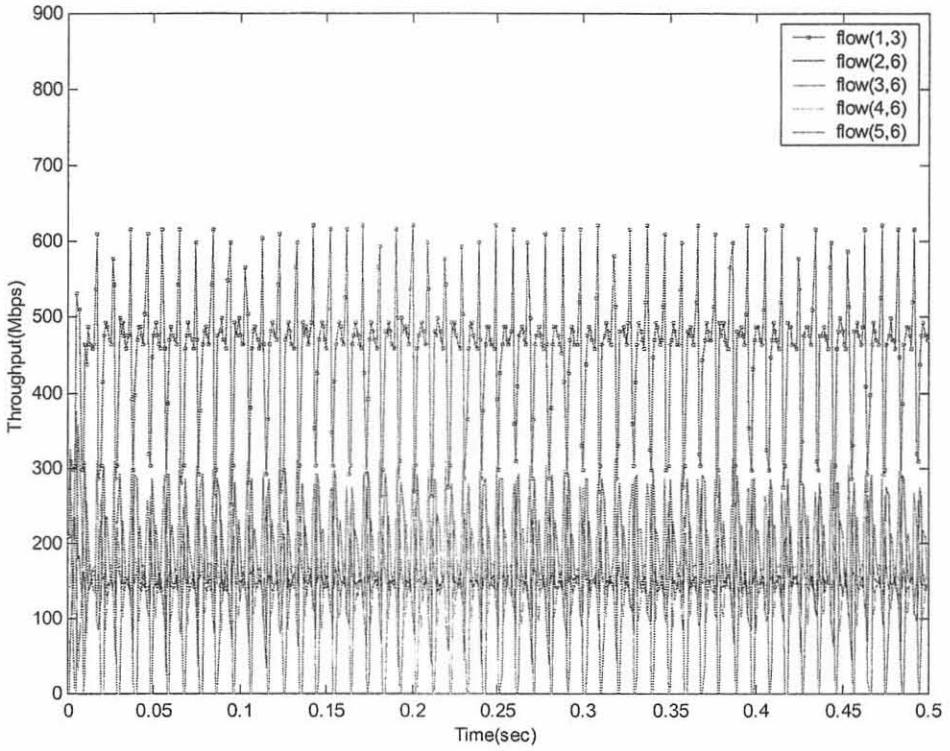


Figure 5.7 RPR-AM (*RampCoef=16)*

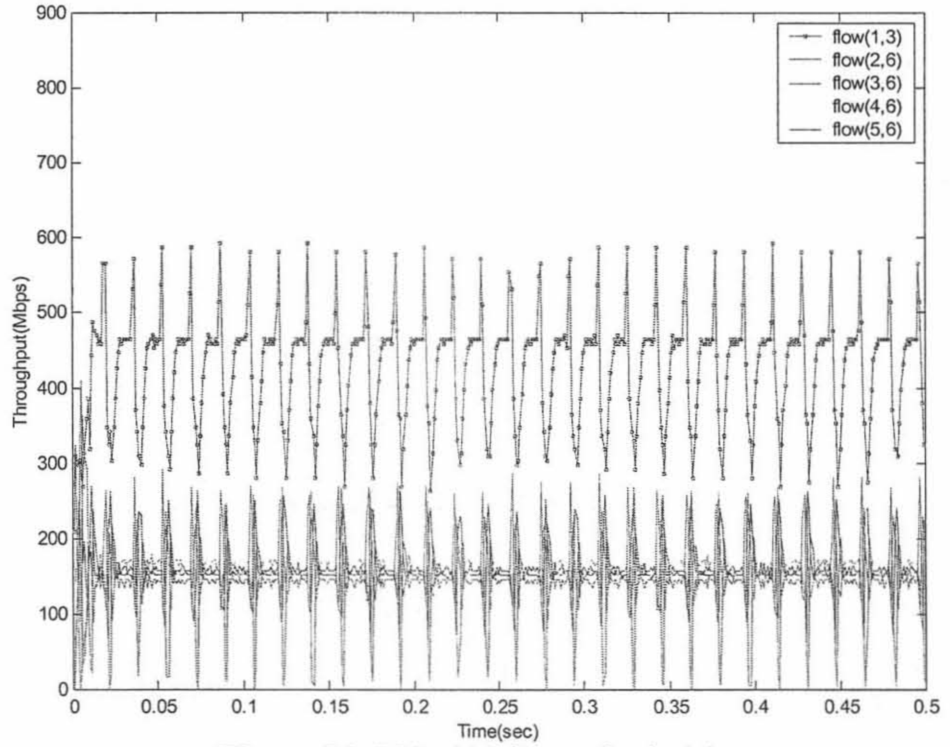


Figure 5.8 RPR-AM (*RampCoef=64)*

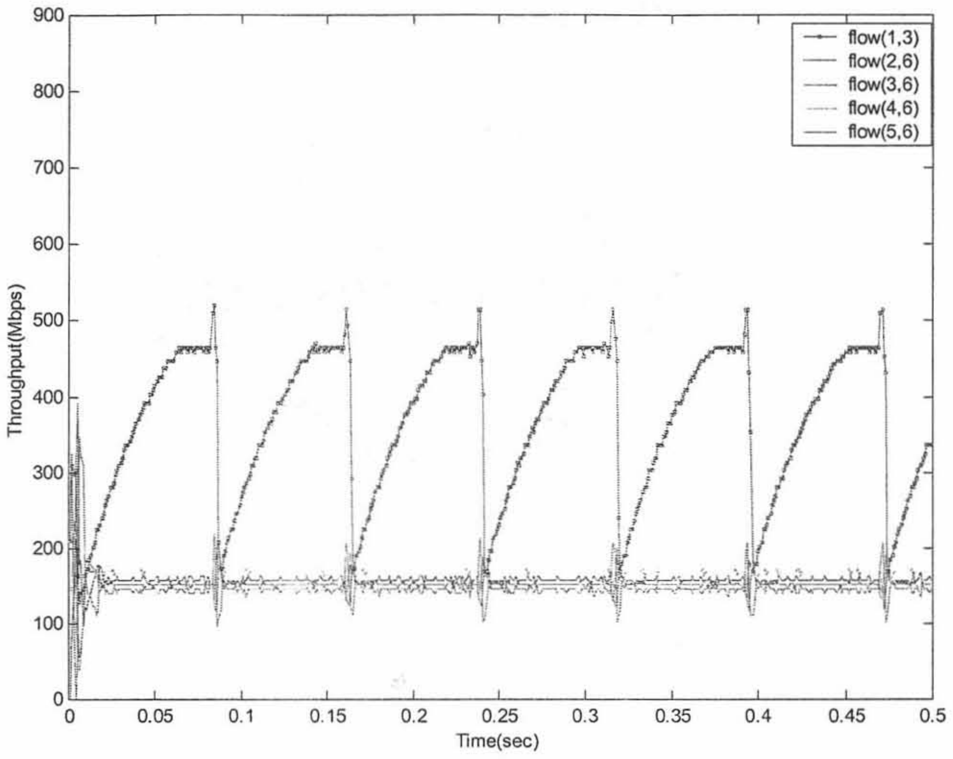


Figure 5.9 RPR-AM (*RampCoef=512)*

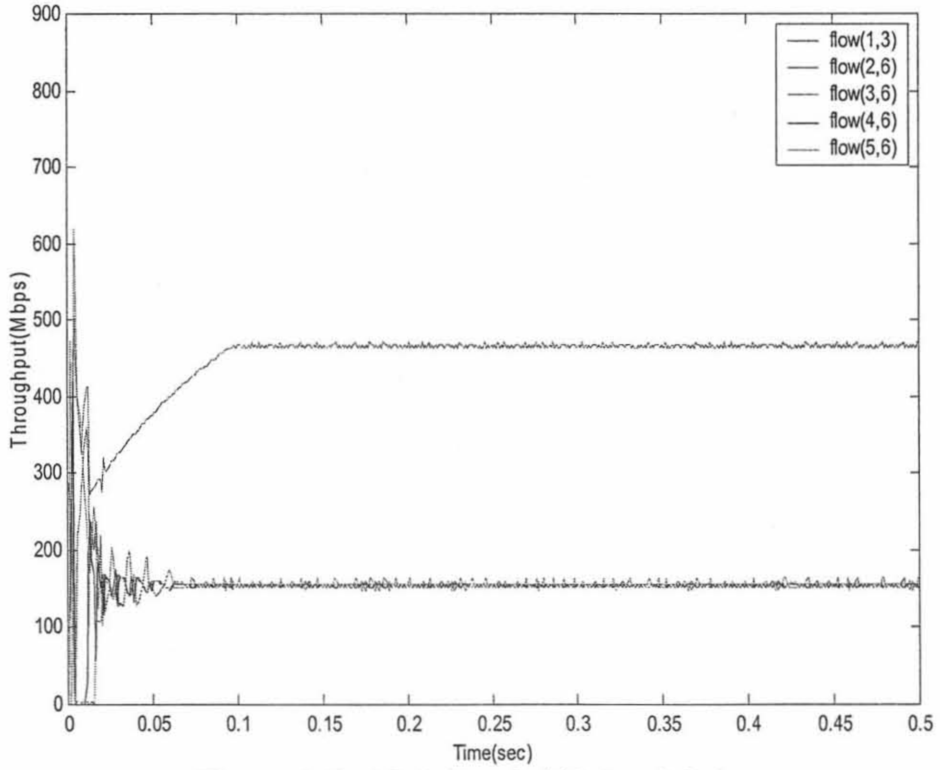


Figure 5.10 ABA (Bandwidth Re-claim)

The simulation results shown in Figure 5.10 have verified that ABA converges to the RIAS fair rate very fast using the same information available to the RPR algorithm. Moreover, the number of controlling messages exchanged between the nodes is less than those in the RPR fairness algorithm.

5.4 Summary

In this chapter, we have proposed the Adaptive Bandwidth Allocation algorithm (ABA) for Resilient Packet Ring to achieve spatial reuse, high bandwidth utilization, and fairness. The basic idea of the algorithm is to construct an Adaptive Bandwidth Controller by means of fuzzy logics. The controller uses the state of the transit queue and its growth rate as inputs to the controller and produces a decision factor to adjust the advertised fair rate.

The proposed ABA inherits the measurement simplicity of the RPR fairness algorithm (RPR-AM) and converges to the RIAS fair rates in a few measurement intervals, and is thus scalable.

CHAPTER 6

TRAFFIC SHAPING IN RPR

RPR defines three service classes for user traffics: Class A which provides guaranteed rate and jitter, Class B with a committed information rate (CIR) and bounded delay and jitter, and the best effort traffic (Class C). The current RPR standard uses a single FIFO for each class at the ingress point (Figure 6.1), and thus the head of line blocking is a potential problem. Optionally, the MAC may implement virtual destination queues (VDQs) to avoid the head of line blocking. In this chapter, we discuss the limitation of the per-class queue scheme. Also, we introduce the VDQ scheme, and show how to serve these VDQs using a unique and scalable bandwidth allocation algorithm.

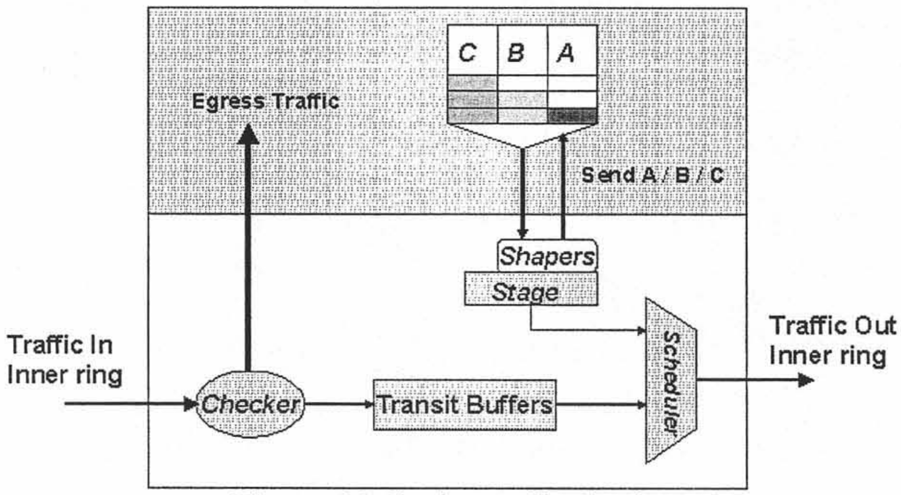


Figure 6.1 Station traffic shaping

The remaining of the chapter is organized as follows: Section 6.1 discusses the VDQ scheme and formulates the bandwidth allocation; Section 6.2 describes several bandwidth allocation policies and their limitations; Section 6.3 describes our proposed scheduling algorithm; Section 6.4 presents simulation results for different allocation policies; we finally conclude in Section 6.5 and emphasize that our proposed scheduling scheme has a very low complexity.

6.1 The VDQ Scheme

Class A traffic provides guaranteed rate and the unused Class A bandwidth cannot be reclaimed. Class B traffic is a Committed Information Rate (CIR). Thus, we omit the discussion of Class A and Class B traffics. Throughout the rest of the chapter, we consider Class C in which each node uses the unreserved bandwidth and reclaims Class B unused bandwidth.

The RPR node uses one queue per class for the station traffic. To show the limitation of this architecture, we consider the simple scheduling scenario (Figure 6.2) where all flows are Class C traffic. When virtual destination queues (VDQs) are not used, flow(1,2) is unnecessarily throttled and delayed due to the congestion that flow(1,5) is experienced at link 4. On the other hand, with VDQs (Figure 6.3), flow (1,2) will be able to reclaim the unused bandwidth at link 1.

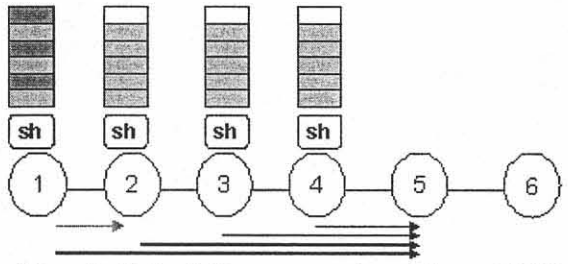


Figure 6.2 Simple Scheduling scenario without VDQ

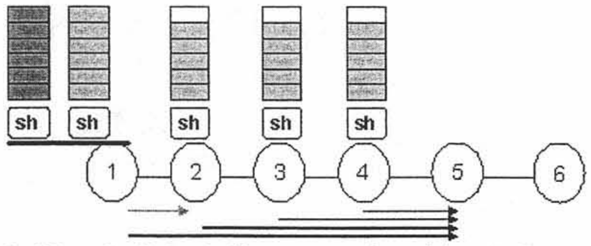


Figure 6.3 Simple Scheduling scenario with VDQ

The benefit of using the VDQ scheme is obvious, but the challenge is how to manage these queues to maximize the utilization and maintain fairness at the ring level.

To avoid the head of line blocking problem associated with the per class queue scheme, we introduce the scheme illustrated in Figure 6.4. Here, we only consider Class C, where each node uses per destination queue.

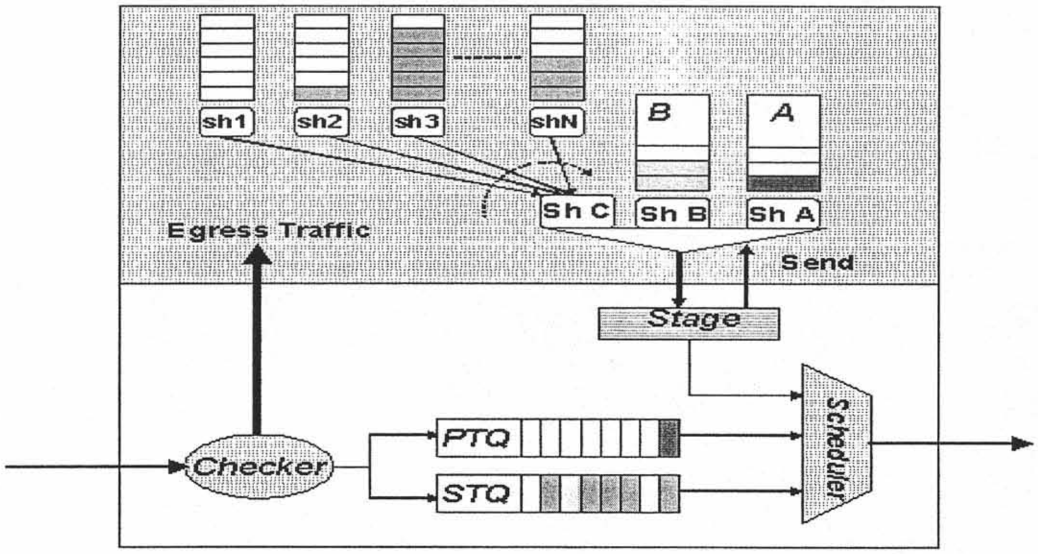


Figure 6.4 The proposed VDQ scheme

Now, we assume that each node is aware of the per source fair rates F_k at all the links. To make sure that a station does not exceed its fair rate at each link, each VDQ is controlled separately by its traffic shaper.

The above scheduling scheme would require a per source-destination allocation at the ingress point. For illustrative purpose, we consider the example shown in Figure 6.5 in which we consider traffic flows originated from node 1.

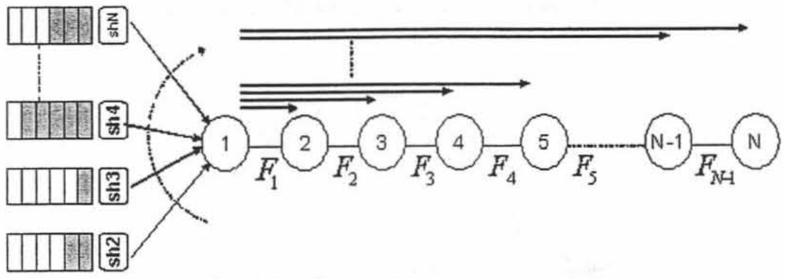


Figure 6.5 A Scheduling Scenario

Let $r_{1,j}$ be the amount of traffic in bytes transmitted from node 1 to node j over the period T sec. Node 1 will use these measurements along with the latest received fair rate from each link *k*, F_k, to adjust the rate of each VDQ shaper as follows.

Define $f_{1,k}$ as the rate allocated for every flow from node 1 to node j ($j = k+1, k+2, ..., N$) traversing link *k*.

The goal of the allocation policy is to maximize

$$f_{1,k} \quad \text{for} \quad k=1,2,3,...,N-1$$

subject to the constraint

$$\sum_{\substack{j=k+1,k+2,...,N \\ (i.e., \text{traversing link } k)}} r_{1,j} \leq F_k \quad . \tag{6.1}$$

The allocation policy has to make sure that the sum of all flows from the same ingress point, destined to different nodes, traversing link *k*, does not exceed the per-source fair rate at link *k*, F_k.

The MAC will set the VDQ shaper to the minimum $f_{1,k}$ along the path from the source to the destination.

6.2 Bandwidth Allocation

In this section, we consider three allocation policies. In the first policy [21], node 1 maps the received fair rate F_k into a counter, credit[k], which represents the number of bytes node 1 can transmit over link k during the next T sec. The virtual destination queues (VDQs) are served in a round robin fashion. When the VDQ[j] has a packet to be sent to destination j, the procedure illustrated in Figure 6.6 [21] will be executed. The procedure returns the link number in which node 1 has no more credit and is not allowed to transmit through. Thus, all destinations beyond the limited link are unreachable. On the other hand, if the destination is before the limited link the packet will be transmitted and all links traversed by the packet will have their credit[k] decreased by the packet size.

It is clear that this policy is very complex and not scalable. Moreover, it is not fair due to the fact that the packet size is not fixed.

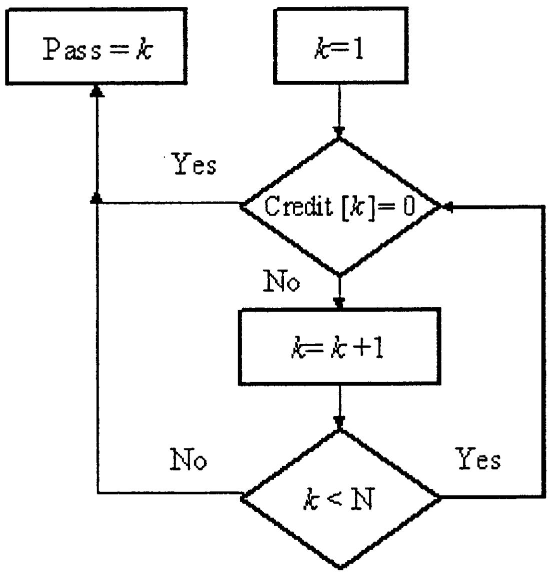


Figure 6.6 The credit based policy [21].

The second policy is the equal allocation policy. The link fair rate is divided by the number of flows traversing that link.

Define $m_{1,k}$ as the number of flows originated from node 1, traversing link *k*. Then, the per flow fair rate is

$$f_{1,k} = \frac{F_k}{m_{1,k}}. \tag{6.2}$$

The equal allocation (Figure 6.7) is simple and has a computation complexity of O(*N*) ,where *N* is the number of links in the ring.

```
$m = 1$
for $k=N-1$ to 1          //for all links
    $f_{1,k} = \frac{F_k}{m}$           // the fair rate
    if ($r_{1,k} > 0$){$m=m+1$}  //update the number of flows
end for
```

Figure 6.7 The equal allocation policy

Despite its simplicity, the equal allocation policy is not fair because it treats different flows equally regardless of their demands.

The third policy is the max-min allocation policy where the flows with demands less than or equal to the per-flow fair share will have their rate allocated first, and the left over bandwidth will be divided among the other flows which need more than their fair shares.

Let the rates of all flows traversing link *k* be ordered according to their demands such that

$$b_1 < b_2 < b_3 < \ldots < b_{m_{1,k}}$$

where $m_{1,k}$ is the number of flows traversing link *k*.

The max-min allocation policy (Figure 6.8) will achieve fairness among flows sharing the same link at the expense of a very high computation complexity.

To find the computation complexity of the max-min allocation policy, we consider the worst case where node 1 is sending traffic to all other stations. The per link fair rate calculation requires a sorting operation with complexity of O($m_{1,k}$ Log $m_{1,k}$). The total complexity can be calculated as follows:

$$\sum_{k=1}^{M} m_{1,k} \log(m_{1,k}) \quad , \qquad M = N-1 \quad . \tag{6.3}$$

The number of flows, $m_{1,k}$, is different for each link. For example, $m_{1,i} = M$ and $m_{1,N-1} = 1$.

Substituting the value of $m_{1,k}$ in Eq. (6.3) yields

$$\sum_{k=M}^{1} k \log(k) \geq \sum_{k=M}^{1} k \quad . \tag{6.4}$$

Hence, the max-min allocation policy has a computational complexity with a lower bound of $o(\frac{M(M-1)}{2})$ which is significantly complex and not scalable.

```
for $k=1$ to $N-1$          //for all links
$C=F_k$                     //the avilable bandwidth
$L=m_{1,k}$                 //the number of flows
$f_{1,k}=\frac{C}{L}$       //the fair share
for $j=1$ to $m_{1,k}$      //for all flows traversing link k
 if $(b_j \leq f_{1,k})$    //if this flow does not require more than the fair share
 $C=C-b_j$                  // eliminate this flow from the bottelnecked flows list
 $L=L-1$                    // decrease the number of bottelnecked flows
 $f_{1,k}=\frac{C}{L}$      // recalculate the fair share
 end if
end for
end for
```

Figure 6.8 The max-min allocation policy

6.3 Simple Scheduling Algorithm

In this section, we introduce a new allocation policy referred to as the Simple Scheduling Algorithm (SSA) [23]. At the end of the *nth* measurement interval, where $t=nT$, the algorithm first estimates the effective number of flows traversing link k as follows:

$$\tilde{m}_{1,k} = \frac{\tilde{A}_{1,k}(n)}{f_{1,k}(n)} \quad , \tag{6.5}$$

where $\tilde{A}_{1,k}(n)$ is the sum of flows in bytes transmitted from node 1 traversing link k during the previous interval T, $f_{1,k}(n)$ is the per flow fair rate of the previous interval, and $\tilde{m}_{1,k}$ is the effective number of flows traversing link k.

Here, we propose the following formula to estimate the per-flow fair rate:

$$f_{1,k}(n+1)=f_{1,k}(n)+\frac{1}{\tilde{m}_{1,k}}(\ F_k - \tilde{A}_{1,k}(n)) \quad . \tag{6.6}$$

The goal is to adjust $f_{1,k}(n)$ so that $\tilde{A}_{1,k}(n)$ matches the per-source fair rate F_k of link k and $f_{1,k}(n)$ converges to the optimal fair rate $f^{*}_{1,k}$.

Note that one of the important features of the SSA algorithm (Figure 6.9) is its low computation complexity of O(N).

```
$\tilde{A}=0$                                         // the amount of traffic transmitted through the link
for $k=N-1$ to 1                                      //for all links
    $\tilde{A}=\tilde{A}+r_{1,k+1}$                   //update the amount of traffic transmitted through the link
    $m_{1,k}=\max(1,\frac{\tilde{A}}{f_{1,k}})$       //the estimated number of flows
    $f_{1,k}=f_{1,k}+\frac{1}{m_{1,k}}(F_k-\tilde{A})$  // the fair rate
end for
```

Figure 6.9 The SSA allocation policy

6.3.1 The SSA Convergence

Theorem: The SSA algorithm generates a sequence that converges to the max-min fair rate.

Proof: The proof is similar to the proof given in Section 4.1.2.

Let k be the bottlenecked link. The number of flows transmitted from node i traversing link k is $m_{i,k}$, where $m_{i,k}^{b}$ is the number of flows bottlenecked elsewhere, and $m_{i,k}^{u} = m_{i,k} - m_{i,k}^{b}$ is the number of flows bottlenecked at link k. Let $r_1^b, r_2^b, \ldots, r_{m_{i,k}^b}^b$ be the rates of the corresponding flows bottlenecked elsewhere, and $r_1^u, r_2^u, \ldots, r_{m_{i,k}^u}^u$ be the rates of the corresponding flows bottlenecked at link k.

At the end of the *nth* measurement interval ($t=nT$), the algorithm estimates the effective number of flows traversing link k as

$$\tilde{m}_{i,k} = \frac{\tilde{A}_{i,k}(n)}{f_{i,k}(n)} \quad , \tag{6.7}$$

where $\tilde{A}_{i,k}(n)$ is the sum of flows transmitted from node i and traversed link k during the previous interval, and $f_{i,k}(n)$ is the per flow fair rate of the previous interval.

The next per-flow fair rate is

$$f_{i,k}(n+1) = f_{i,k}(n) + \frac{1}{\tilde{m}_{i,k}}(F_k - \tilde{A}_{i,k}(n)) . \tag{6.8}$$

Substituting Eq. (6.8) into Eq. (6.7) yields

$$f_{i,k}(n+1) = f_{i,k}(n) \frac{F_k}{\tilde{A}_{i,k}(n)} \quad . \tag{6.9}$$

Define $\alpha(n) = \frac{\tilde{A}_{i,k}(n)}{F_k}$ as the load factor, and rewrite Eq. (6.9) as

$$f_{i,k}(n+1) = \frac{f_{i,k}(n)}{\alpha(n)} . \tag{6.10}$$

According to the load factor value, two cases are considered. First, consider the case where the load factor $\alpha(n)$ is less than one. In this case, the sum of rates of flows traversing link k is less than the link fair rate F_k. According to Eq. (6.10), the per-flow fair rate $f_{i,k}(n)$ will increase. If all flows are bottlenecked elsewhere ($m^u_{i,k} = 0$), the fair rate has been achieved. On the other hand, if there are some flows bottlenecked at link k ($m^u_{i,k} > 0$), the bottlenecked flows will continue to increase their rates until the load factor becomes greater than or equal to one.

Second, consider the case where the load factor $\alpha(n)$ is greater than one. In this case, the sum of rates of flows traversing link k is greater than the link fair rate F_k. According to Eq. (6.10), the per-flow fair rate $f_{i,k}(n)$ will decrease and the participating flows will decrease their rates. This will continue until the load factor becomes less than or equal to one.

It is obvious from the above two cases that the load factor oscillates around one and converges to one. Thus, in the following analysis, we assume that the load factor is close to one.

Next, we shall show that the iterative algorithm Eq. (6.8) will generate a sequence of $f_{i,k}(n)$ that will converge to the optimal value of the per-flow fair rate $f_{i,k}(n) = \frac{F_k - \sum_{j=1}^{m^b_{i,k}} r^b_j}{m^u_{i,k}}$.

Note that the iterative equation Eq. (6.8) is in the form of

$$f_{i,k}(n+1) = f_{i,k}(n) + \lambda\left[\nabla^2 D(f_{i,k}(n))\right]^{-1} \nabla D(f_{i,k}(n)) \ . \qquad (6.11)$$

That is, the per-flow fair rate is adjusted in the direction of the gradient, where

$$\nabla D(f_{i,k}(n)) = F_k - \tilde{A}_{i,k}(f_{i,k}(n)) \ . \qquad (6.12)$$

Here, λ is a positive step size, and in our case is equal to one, and $\left[\nabla^2 D(f_{i,k}(n))\right]^{-1}$ is the inverse of the Hessian.

It is well known that the Newton method Eq. (6.8), where the gradient is scaled by the inverse of the Hessian typically converges faster than the gradient projection; see [17, pp.201].

The Hessian $\nabla^2 D(f_{i,k}(n)) = \tilde{m}_{i,k}$ is approximated by using two points, the current point of $(\tilde{A}_{i,k}(n), f_{i,k}(n))$ and the origin (0, 0).

Hence, the above iterative equation converges, and the stable value of the per-flow fair rate is detailed as follows:

First, assume that all the flows are bottlenecked at link k. In this case, $m^b_{i,k} = 0$ and $m^u_{i,k} = m_{i,k}$. All flows are running at the per-flow fair rate $f_{i,k}(n)$, and the sum of flows traversing link k is

$$\tilde{A}_{i,k}(n) = m_{i,k} f_{i,k}(n) \ . \qquad (6.13)$$

Substituting the value of $\tilde{A}_{i,k}(n)$ into Eq. (6.9) with a load factor $\alpha(n)$ of one at the steady state yields

$$f_{i,k}(n+1) = \frac{F_k}{m_{i,k}}, \qquad (6.14)$$

which is the desired value for $f_{i,k}(n)$.

Finally, assume that some flows are bottlenecked elsewhere. These flows will have their rates $r_1^b, r_2^b, \ldots, r_{m_{i,k}^b}^b$ stabilized and the allocated bandwidth for these flows is $B = \sum_{j=1}^{m_{i,k}^b} r_j^b$.

Since we have a load factor $\alpha(n)$ of one at the steady state, we have

$$\sum_{j=1}^{m_{i,k}^u} r_j^u = F_k - \sum_{j=1}^{m_{i,k}^b} r_j^b \ , \qquad (6.15)$$

and

$$\sum_{j=1}^{m_{i,k}^u} r_j^u = m_{i,k}^u f_{i,k}(n) \ . \qquad (6.16)$$

Substituting Eq. (6.16) into Eq. (6.15) yields

$$f_{i,k}(n) = \frac{F_k - \sum_{j=1}^{m_{i,k}^b} r_j^b}{m_{i,k}^u} \ . \qquad (6.17)$$

Substituting the value of $f_{i,k}(n)$ into Eq. (6.10) yields

$$f_{i,k}(n+1) = \frac{F_k - \sum_{j=1}^{m_{i,k}^b} r_j^b}{m_{i,k}^u} \ . \qquad (6.18)$$

which is indeed the desired value for $f_{i,k}(n)$ and the proof is complete . ■

6.4 Simulation Results

Here, we consider two scenarios. First, we consider the scenario illustrated in Figure 6.10. The links have the same capacity of 622 Mbps, and each link has a propagation delay of 0.1 ms. All flows are UDP flows and start at *t=0,* flows (1,5), (2,3), (3,5), (4,5) and (2,7) are greedy while flows (2,4), (2,5) and (2,6) are running at a rate equal to 10Mbps. The measurement time interval was set to $T = 1$ ms.

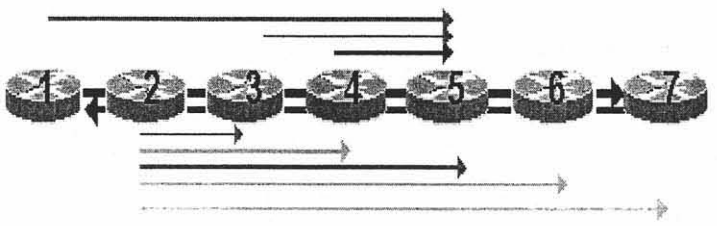


Figure 6.10 Scheduling Scenario I

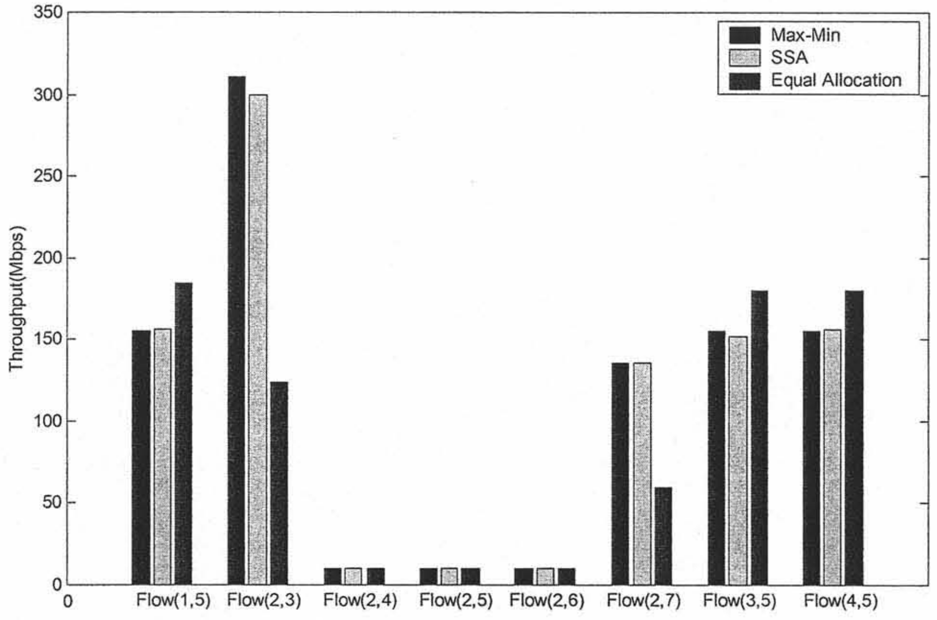


Figure 6.11 Policies comparison (scenario I)

Using the equal allocation policy, node 2 divides link 4 per-source fair rate (F_4=155.5Mbps) by the number of its flows traversing link 4 (in this case 3) that results in a per flow fair rate equal to 51.67Mbps. Flows (2,6) and (2,7) are running at a rate of 10Mbps. Thus, the unused bandwidth at link 4 due to the equal allocation policy would be reclaimed by other sources. This continues until it is stabilized at link 4 with a fair rate of 180Mbps and flow (2,5) is only able to get the fair rate equal to 60Mbps. The same is applied to link 2 where flow (2,3) is only able to get the fair rate equal to 120Mbps.

On the other hand, using the SSA policy, flows are able to achieve their max-min fair rates.

Figure 6.11 shows the comparison of the allocation policies where the SSA policy is able to achieve the max-min allocation at the same complexity of the equal allocation policy.

Second, we consider the scenario illustrated in Figure 6.12. Flows (1,3) and (2,3) are greedy and flows (2,4), (2,5), (2,6) and (2,7) are running at a rate equal to 10Mbps. The flows, flow (1,3) and flow(2,3), start at t=0 while flow (2,4), flow(2,5), flow(2,6) and flow(2,7) start at time 0.1, 0.2, 0.3 and 0.4 seconds, respectively.

Figure 6.13 shows the unfairness of the equal allocation policy. The rate of flow (2,3) decreases at the start of the other flows where the per-flow rate decreases as the number of flows increases despite their low rate demands.

Figure 6.14 shows the performance of the SSA policy where flows are able to achieve their max-min fair rates.

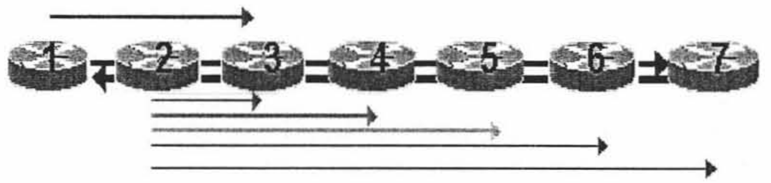


Figure 6.12 Scheduling Scenario II

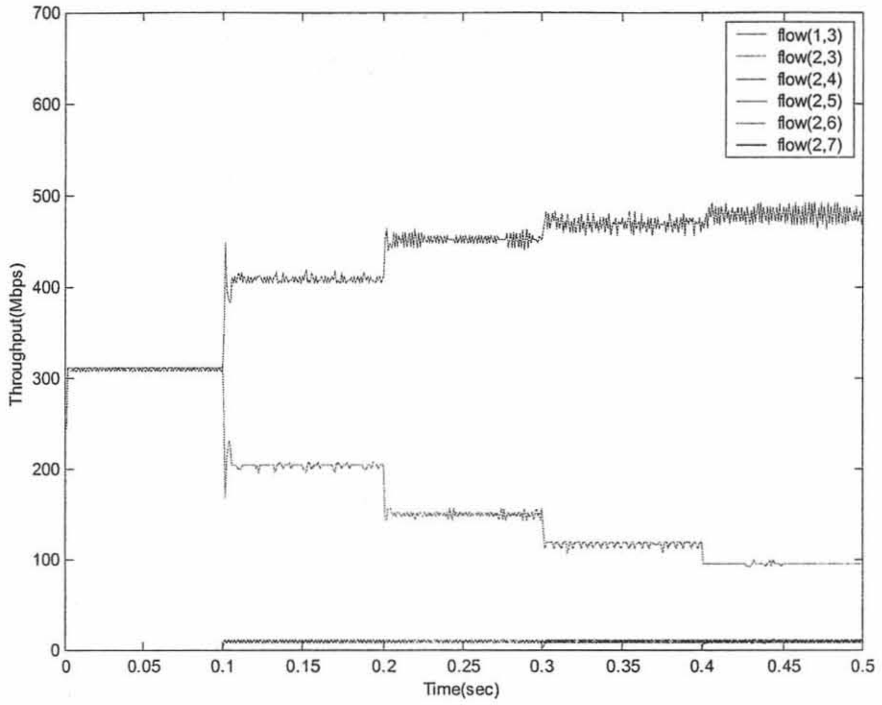


Figure 6.13 Equal Allocation (scenario II)

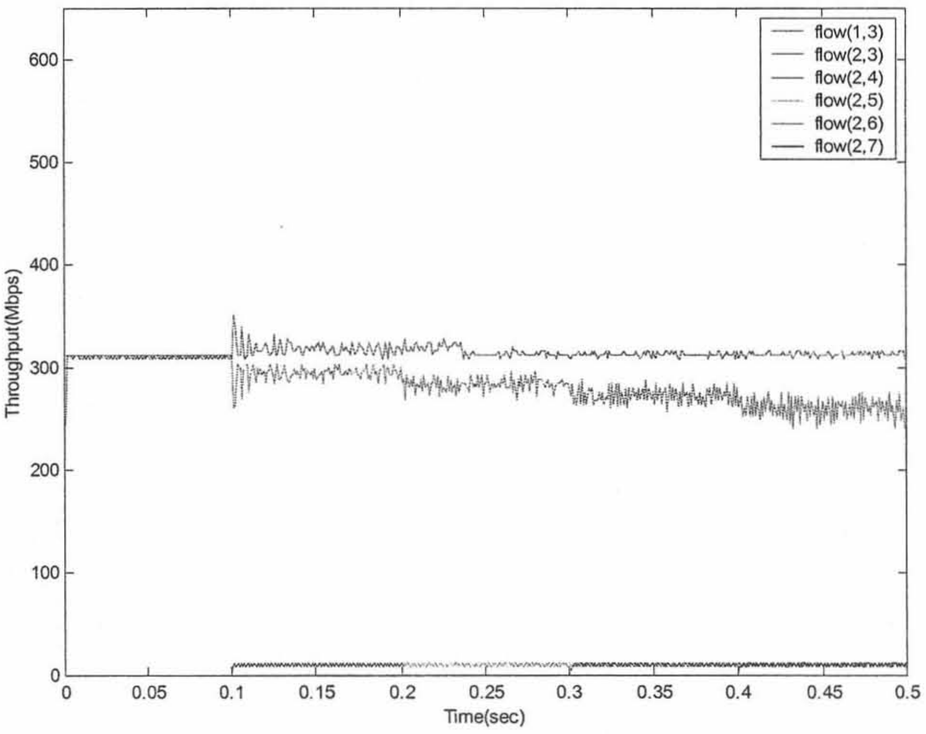


Figure 6.14 SSA Allocation (scenario II)

6.5 Summary

In this chapter, we have proposed a new traffic shaping scheme for the Resilient Packet Ring to maximize the utilization and avoid the head of line blocking associated with the current RPR traffic shaping scheme. The new scheme uses per-destination queues at the ingress point. Existing bandwidth allocation policies have been investigated and shown to be either inefficient or significantly complex.

An allocation policy, namely, Simple Scheduling Algorithm (SSA), has been proposed, and shown analytically and through simulations to be optimal where the flows achieve their max-min fair rates at a very low computation complexity.

CHAPTER 7

ROUTING IN RPR

The current RPR standard uses the shortest path routing policy. Based on the number of links between the source and destination, one of the rings will be selected as the shortest path. The shortest path routing is simple and results in the minimum end-to-end delay. On the other hand, the shortest path routing policy can be shown inefficient and unfair. For example, consider the scenario illustrated in Figure 7.1. All links have the same capacity of 622Mbps, flows (1,5), (2,5), (3,5) and (4,5) are greedy and flow(9,5) transmits at a rate of 311Mbps. According to the shortest path, flows (1,5), (2,5), (3,5) and (4,5) will be routed through the outer ring and flow (9,5) will use the inner ring. Owing to the routing policy, the outer ring is fully utilized while the inner ring is only used for flow (9,5), thus resulting in a poor utilization. Moreover, flow (1,5) and flow (9,5) have the same number of links to reach the same destination but they are treated differently, and thus the shortest path routing policy is unfair.

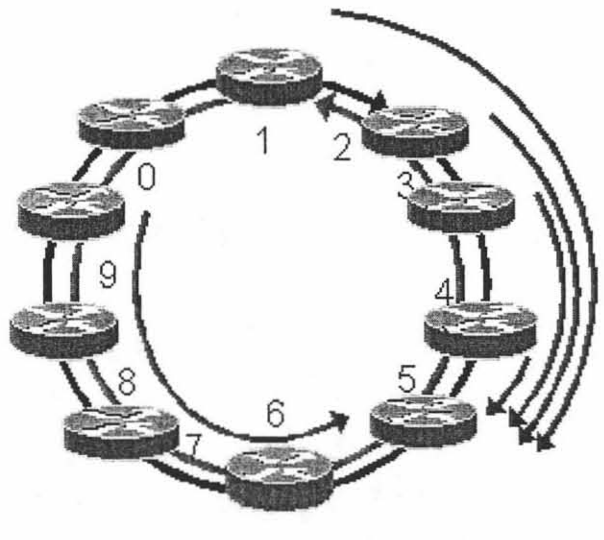


Figure 7.1 Routing scenario

In this chapter, we are proposing a routing scheme for RPR that will maximize the total ring utilization by allowing each node to forward packets in both rings in a weighted manner. In Figure 7.1, if each flow assigned a weight equal to one for its shortest path and a weight equal to 0.125 for its longest path, the total ring utilization will be maximized and all flows bottlenecked in the outer ring will be able to increase their rates using their weighted share at the inner ring.

In this chapter, we are proposing two routing models. Both have the same goal of maximizing the total ring utilization. They differ in allocating the available bandwidth; the first routing model uses the RIAS fairness concept and the second routing model uses the per-flow fairness concept (see Section 2.7.1).

7.1 Maximum Throughput Routing

In this section, we propose and analyze the first routing model namely the Routing Algorithm for RPR (RA-RPR) which can achieve maximum utilization in the dual-ring network. We first introduce the following definitions:

N : The number of nodes in the ring.

I : The ring id (zero for the outer ring and one for the inner ring).

C_k^0 : The available bandwidth at link k in the outer ring.

C_k^1 : The available bandwidth at link k in the inner ring.

F_k^0 : Per-source fair rate of link k in the outer ring (computed and broadcasted to other nodes every T sec).

F_k^1: Per-source fair rate of link k in the inner ring (computed and broadcasted to other nodes every T sec).

D_{ij}: The demand from source i to destination j.

l_{ij}^0: The set of links used by flow (i,j) in the outer ring.

l_{ij}^1: The set of links used by flow (i,j) in the inner ring.

$f_{i,k}^0$: The fair rate for flows from source i traversing link k at the outer ring (calculated locally every T sec).

$f_{i,k}^1$: The fair rate for flows from source i traversing link k at the inner ring (calculated locally every T sec).

ζ_{ij}^0: The minimum fair rate from source i to destination j at the outer ring $\zeta_{ij}^0 = \min_{k \in l_{ij}^0} f_{i,k}^0$

ζ_{ij}^1: The minimum fair rate from source i to destination j at the inner ring $\zeta_{ij}^1 = \min_{k \in l_{ij}^1} f_{i,k}^1$

w_{ij}^0: The weight associated with the path from source i to destination j in the outer ring.

w_{ij}^1: The weight associated with the path from source i to destination j in the inner ring.

If the shortest path along the outer ring is selected, then the amount of traffic to be routed to each ring is:

$$R_{ij}^0 = \min(w_{ij}^0 \zeta_{ij}^0, D_{ij}) \,, \tag{7.1}$$

$$R_{ij}^1 = \min(w_{ij}^1 \zeta_{ij}^1, D_{ij} - R_{ij}^0) \,. \tag{7.2}$$

On the other hand, if the shortest path along the inner ring is selected, then the amount of traffic to be routed to each ring is:

$$R_{ij}^1 = \min(w_{ij}^1 \zeta_{ij}^1, D_{ij}) \,, \tag{7.3}$$

$$R_{ij}^0 = \min(w_{ij}^0 \zeta_{ij}^0, D_{ij} - R_{ij}^1) \ . \qquad 7.3$$

Thus, the shortest path is always selected first.

The goal of the routing policy is to maximize the total routed traffic

$$\sum_{j=0}^{N-1} R_{ij}^0 + R_{ij}^1 \qquad 7.5$$

subject to the constraints:

$$\sum_{j=(k+1)\bmod N}^{(i-1)\bmod N} R_{i,j}^0 \le F_k^0 \ , \qquad 7.6$$

$$\sum_{j=(i+1)\bmod N}^{(k-1)\bmod N} R_{i,j}^1 \le F_k^1 \ , \qquad 7.7$$

for $i = 0,1,\ldots,N-1$ *and* $k = 0,1,\ldots,N-1$.

The Routing Algorithm has to make sure that the sum of all flows from node *i*, destined to different nodes, traversing link *k*, do not exceed the per-source fair rate at link *k*, F_k.

The solution to this problem is to find the set of weights that will jointly optimize the routing and congestion control to achieve the weighted fairness and maximum utilization. Here, we propose a solution, which can flexibly adapt any weighted routing policy.

	The shortest path is the Outer Ring	The shortest path is the Inner Ring
w_{ij}^0	1	$0 \le w_{ij}^0 \le 1$
w_{ij}^1	$0 \le w_{ij}^1 \le 1$	1

Table 7.1 The Routing Policies

A ring is considered to be the shortest path when the number of links between the source node *i* and the destination node *j* is less than or equal to $\left\lfloor \frac{N}{2} \right\rfloor$ along the ring direction.

According to Table 7.1, two routing polices are considered. First, when the shortest path routing policy (SP) is adopted, a node will be assigned a weight equal to one for the shortest path and zero for the longest path. Second, when the weighted fairness routing policy (WF) is adopted, a node will be assigned a weight equal to one for the shortest path and a weight less than or equal to one for the longest path.

7.1.1 Traffic Shaping and Ringlet Selection

In this section, we are proposing a new architecture for traffic shaping and ringlet selection (Figure 7.2).

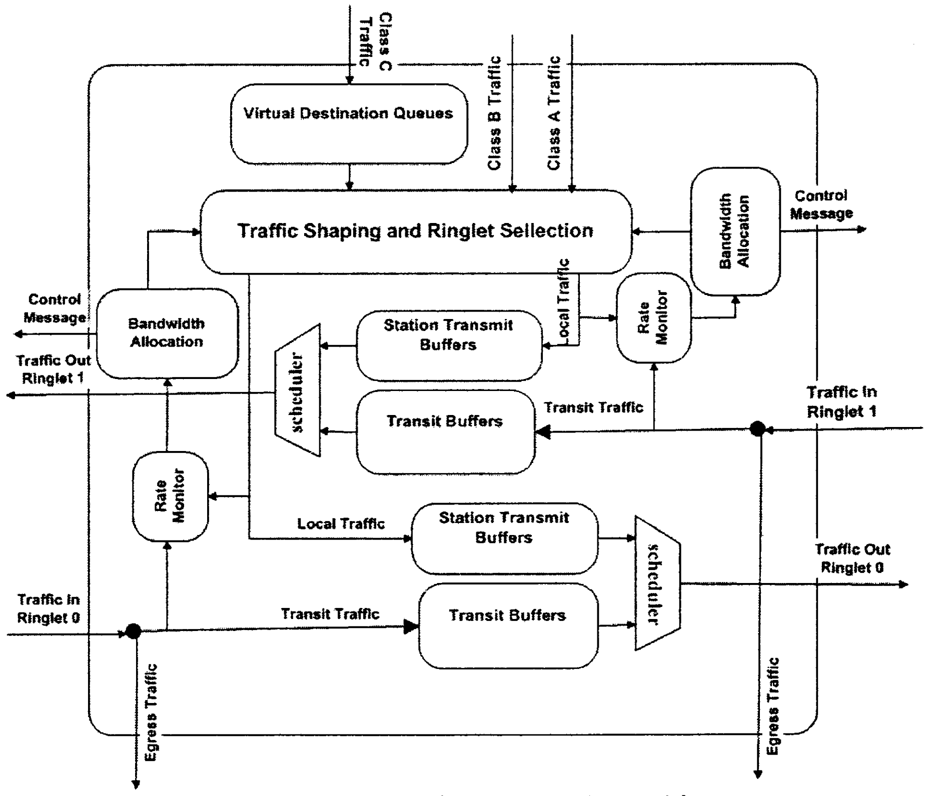


Figure 7.2 The generic RPR Node Architecture

To work according to the routing policy, the MAC uses one virtual queue for every destination to avoid the head of line blocking (HLB). The virtual destination queue is controlled by two token buckets, one for each ring. The token bucket has the maximum size of MTU tokens, where tokens are generated at a rate equal to the supported weighted fair rate to the destination in that ring $(w_{ij}^{I}\zeta_{ij}^{I})$. The virtual destination queues are served in a round-robin fashion. When a virtual destination queue has a packet to be served, if there is at least one token in the token bucket belonging to the shortest path, the MAC will forward the packet by the shortest path. Else, the packet will be forwarded by the longest path if there is at least one token available in the token bucket of the longest path. Otherwise, the next virtual destination queue will be served. Thus, the shortest path is always selected first to minimize the packet delay.

7.1.2 Bandwidth Allocation

In this section, we introduce the way to compute the fair rate from source i to destination j at ring $I\zeta_{ij}^{I}$. First, each node k calculates and broadcasts the per-source fair rate F_k^I every time interval T as detailed in Chapters 4 and Chapter 5.

Second, at the end of every time interval T, each node i uses the latest received per-source fair rates F_k^I from each node along with its routed traffic R_{ij}^{I} during the last interval to compute the per-link per-flow fair rate $f_{i,k}^{I}$ (the fair rate for flows from source i traversing link k at ring I) as detailed in Chapter 6.

Finally, the rate from source i to destination j at ring I, ζ_{ij}^{I}, is the minimum of $f_{i,k}^{I}$ along the path from source i to destination j, i.e., $\zeta_{ij}^{I} = \min_{k \in l_{ij}^{I}} f_{i,k}^{I}$.

7.2 Weighted Fairness Algorithm for Maximum Utilization

In this section, we propose and analyze the second routing model, namely, the Weighted Fairness Algorithm (WFA), which maximizes the dual-ring utilization. We first introduce the following definitions:

N : The number of nodes in the ring.

C_k^0 : The available bandwidth at link k in the outer ring.

C_k^1 : The available bandwidth at link k in the inner ring.

F_k^0 : Per-flow fair rate of link k in the outer ring (computed and broadcasted to other nodes every T sec).

F_k^1 : Per-flow fair rate of link k in the inner ring (computed and broadcasted to other nodes every T sec).

D_{ij} : The demand from source i to destination j.

l_{ij}^0 : The set of links used by flow (i,j) in the outer ring.

l_{ij}^1 : The set of links used by flow (i,j) in the inner ring.

ζ_{ij}^0 : The minimum fair rate from source i to destination j at the outer ring, $\zeta_{ij}^0 = \min_{k \in l_{ij}^0} F_k^0$

ζ_{ij}^1 : The minimum fair rate from source i to destination j at the inner ring, $\zeta_{ij}^1 = \min_{k \in l_{ij}^1} F_k^1$

w_{ij}^0 : The weight associated with the path from source i to destination j in the outer ring.

w_{ij}^1 : The weight associated with the path from source i to destination j in the inner ring.

These weights are setup parameters and known by all nodes.

If the shortest path along the outer ring is selected, then the amount of traffic to be routed to each ring is:

$$R_{ij}^0 = \min(w_{ij}^0 \zeta_{ij}^0, D_{ij}) \ , \tag{7.8}$$

$$R_{ij}^1 = \min(w_{ij}^1 \zeta_{ij}^1, D_{ij} - R_{ij}^0) \ . \tag{7.9}$$

On the other hand, if the shortest path along the inner ring is selected, then the amount of traffic to be routed to each ring is:

$$R_{ij}^1 = \min(w_{ij}^1 \zeta_{ij}^1, D_{ij}) \ , \tag{7.10}$$

$$R_{ij}^0 = \min(w_{ij}^0 \zeta_{ij}^0, D_{ij} - R_{ij}^1) \ . \tag{7.11}$$

Thus, the shortest path is always selected first.

The goal of the routing policy is to maximize the total routed traffic

$$\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (R_{ij}^0 + R_{ij}^1) \ , \tag{7.12}$$

subject to the constraints:

$$\sum_{k=1}^{N-1} R_{(i,i+k)\,\mathrm{mod}\,N}^0 + \sum_{j=2}^{N-1} \sum_{k=1}^{j-1} R_{(i+j,i+k)\,\mathrm{mod}\,N}^0 \le C_i^0 \ , \tag{7.13}$$

$$\sum_{k=1}^{N-1} R_{(i,i+k)\,\mathrm{mod}\,N}^1 + \sum_{j=1}^{N-2} \sum_{k=1}^{N-j-1} R_{(i+j,i+j+k)\,\mathrm{mod}\,N}^1 \le C_i^1 \ , \tag{7.14}$$

for $i = 0, 1, \ldots, N-1$,

where $\sum_{k=1}^{N-1} R_{(i,i+k)\,\mathrm{mod}\,N}^0$ and $\sum_{k=1}^{N-1} R_{(i,i+k)\,\mathrm{mod}\,N}^1$ are the transmitted traffic from node i in the outer and the inner ring, respectively.

On the other hand, $\sum_{j=2}^{N-1}\sum_{k=1}^{j-1} R^{0}_{(i+j,i+k)\bmod N}$ is the total traffic transmitted from upstream nodes traversing link i in the outer ring, and $\sum_{j=1}^{N-2}\sum_{k=1}^{N-j-1} R^{1}_{(i+j,i+j+k)\bmod N}$ is the total traffic transmitted from upstream nodes traversing link i in the inner ring.

Here, we use the same weights assignment scheme proposed in Section 7.1 that will jointly optimize the routing and congestion control to achieve the weighted fairness and maximum utilization. Also, we use the same traffic shaping and ring selection scheme proposed in Section 7.1 to schedule the virtual destination queues according to the weighted fairness algorithm.

7.2.1 The Weighted Fairness Algorithm

In this section, we introduce a new bandwidth allocation algorithm referred to as the Weighted Fairness Algorithm (WFA). Unlike the RPR fairness algorithm, where the level of traffic granularity at a link is defined as an ingress-aggregated (IA) flow, i.e., the aggregate of all flows originated from the same node but destined to different nodes, the WFA algorithm adopts a source-destination traffic granularity level to define a flow. Throughout the analysis, we consider only one of the two rings, the outer ring with N nodes numbered from 0 to N-1 along the ring direction.

Definition 7.1: At node k, the traffic demand from node i to node j during a measurement interval T is defined as

$$b^{0}_{ij} \text{ in Bytes}/T(\text{sec}). \qquad 7.15$$

Since T is fixed, we will interchangeably refer to a traffic demand in bytes from a node as a rate. At the end of the measurement interval, these rates are compared with the advertised fair rate F_k^0 in bytes of the previous interval as a means to measure the activity level of all flows [12], [16].

Definition 7.2: The flow *(i, j)* activity with respect to node *k* is defined as

$$\min(w_{ij}^0, \frac{b_{ij}^0}{F_k^0}). \tag{7.16}$$

From this definition, flow *(i,j)* is considered to be fully active if it is running at the weighted advertised fair rate, i.e., $b_{ij}^0 = w_{ij}^0 F_k^0$. Otherwise, it will be considered as partially active. When the activity level of flow *(i,j)* is 0, this implies that node *i* is not sending to node *j* through node *k*, and it is thus considered to be not active.

Definition 7.3: The effective sum of weights of active flows at node *k,* $\tilde{W}_k^0$, is defined as

$$\tilde{W}_k^0 = \sum_{i=1}^{N-1} \min\left(w_{(k,k+i)\bmod N}^0, \frac{b_{(k,k+i)\bmod N}^0}{F_k^0} \right) + \sum_{j=2}^{N-1} \sum_{i=1}^{j-1} \min\left(w_{(k+j,k+i)\bmod N}^0, \frac{b_{(k+j,k+i)\bmod N}^0}{F_k^0} \right), \tag{7.17}$$

Thus, the effective sum of weights of active flows at node *k,* $\tilde{W}_k^0$, is always less than or equal to the sum of weights of flows at node *k* as follows:

$$\tilde{W}_k^0 \le \sum_{i=1}^{N-1} w_{(k,k+i)\bmod N}^0 + \sum_{j=2}^{N-1} \sum_{i=1}^{j-1} w_{(k+j,k+i)\bmod N}^0 , \tag{7.18}$$

where N is the number of nodes in the ring.

Definition 7.4: The advertised fair rate at node *k* is

$$F_k^0 = \frac{C_k^0}{\max(1, \tilde{W}_k^0)} . \tag{7.19}$$

From Eq. (7.19), it is obvious that the denominator will exhibit three cases, thus resulting in the following three lemmas.

Lemma 7.1*:* When all flows are running at a rate equal to the weighted advertised fair rate F_k^0, the new calculated advertised fair rate is the weighted fair share

$$F_k^0 = \frac{C_k^0}{\sum_{i=1}^{N-1} w_{(k,k+i)\bmod N}^0 + \sum_{j=2}^{N-1}\sum_{i=1}^{j-1} w_{(k+j,k+i)\bmod N}^0} .$$

Proof: In this case, the denominator of Eq. (7.19) becomes

$$\tilde{W}_k^0 = \sum_{i=1}^{N-1} w_{(k,k+i)\bmod N}^0 + \sum_{j=2}^{N-1}\sum_{i=1}^{j-1} w_{(k+j,k+i)\bmod N}^0 .$$

Thus, we have the stated weighted fair rate $F_k^0 = \frac{C_k^0}{\sum_{i,j} w_{i,j}^0}$.

Lemma 7.2: When all the flows are not active, the new calculated advertised fair rate is

$$F_k^0 = \frac{C_k^0}{\max(1,\tilde{W}_k^0)} = C_k^0 .$$

Proof: In this case, the denominator of Eq. (7.19) becomes

$$\max(1,\tilde{W}_k^0) = 1 .$$

Here, the advertised fair rate F_k^0 of node k is not higher than the available capacity.

Lemma 7.3: When some flows are not active or running at rates less than their weighted fair shares, the new calculated advertised fair rate is

$$F_k^0 > \frac{C_k^0}{\sum_{i=1}^{N-1} w_{(k,k+i)\bmod N}^0 + \sum_{j=2}^{N-1}\sum_{i=1}^{j-1} w_{(k+j,k+i)\bmod N}^0} .$$

Proof: In this case, the denominator of Eq. (7.19) becomes

$$\tilde{W}_k^0 < \sum_{i=1}^{N-1} w_{(k,k+i)\bmod N}^0 + \sum_{j=2}^{N-1}\sum_{i=1}^{j-1} w_{(k+j,k+i)\bmod N}^0 .$$

Thus, flows that are bottlenecked at link *k,* but are not bottlenecked elsewhere will be able to increase their rates and claim the unused bandwidth left by the other flows that are bottlenecked elsewhere or at their ingress points.

The simple calculation in Eq. (7.19) requires per-flow information. To eliminate the overhead of storing per-flow information, the effective sum of weights of all flows traversing link *k*, $\tilde{W}_k^0$, is estimated as follows.

We first introduce a flow activity detection bit s_{ij}^0; this bit will be set to one at the arrival of the first packet of flow *(i, j)* during the measurement interval *T.* Thus, Eq. (7.16) and Eq. (7.17) are redefined as

$$\min\left(S_{ij}^0 * w_{ij}^0, \frac{b_{ij}^0}{F_k^0}\right), \qquad 7.20$$

$$\tilde{W}_k^0 = \sum_{i=1}^{N-1} \min\left(S^0_{(k,k+i)\bmod N} \cdot w^0_{(k,k+i)\bmod N}, \frac{b^0_{(k,k+i)\bmod N}}{F_k^0}\right) + \sum_{j=2}^{N-1}\sum_{i=1}^{j-1} \min\left(S^0_{(k+j,k+i)\bmod N} \cdot w^0_{(k+j,k+i)\bmod N}, \frac{b^0_{(k+j,k+i)\bmod N}}{F_k^0}\right), \qquad 7.21$$

with the assumption that the ingress points will throttle their rates according to the received feedback signal from the bottlenecked link, and transmit at rates less than or equal to their weighted fair rate, i.e., $b_{ij}^0 \le w_{ij}^0 F_k^0$.

The calculation of $\tilde{W}_k^0$ would be estimated as follows

$$\tilde{W}_k^0 = \min\left(\sum_{i=1}^{N-1} S^0_{(k,k+i)\bmod N} \cdot w^0_{(k,k+i)\bmod N} + \sum_{j=2}^{N-1}\sum_{i=1}^{j-1} S^0_{(k+j,k+i)\bmod N} \cdot w^0_{(k+j,k+i)\bmod N}, \frac{\sum_{i=1}^{N-1} b^0_{(k,k+i)\bmod N} + \sum_{j=2}^{N-1}\sum_{i=1}^{j-1} b^0_{(k+j,k+i)\bmod N}}{F_k^0}\right), \qquad 7.22$$

which can be simplified to

$$\tilde{W}_k^0 = \left[\frac{\tilde{A}_k^0}{F_k^0}\right]_1^{w_k^0}, \qquad 7.23$$

where $[y]_a^b = \max(a, \min(b, y))$.

The calculation in Eq. (7.23) only requires the total arrival rate at node k during the current interval $\tilde{A}_k^0 = \sum_{i,j} b_{ij}^0$, and the sum of weights of flows traversing link k, $W_k^0 = \sum_{i,j} w_{ij}^0$.

Figure 7.3 shows the pseudo code of the weighted fairness algorithm (WFA), where the computational complexity of the advertised fair rate is $O(1)$, thus making the algorithm scalable for a ring network with any number of nodes, i.e., independent of N.

```
Initialization: at the beginning of every measurement interval T, reset $\tilde{A}_k^0$, $W_k^0$, and $S_{ij}^0$
During the measurement interval
  At the arrival of a packet with length L bytes
  $\tilde{A}_k^0 = \tilde{A}_k^0 + L$
  $if(S_{ij}^0 = 0)\{$
  $W_k^0 = W_k^0 + w_{ij}^0$ ;
  $S_{ij}^0 = 1$ ;
  }
  At the end of the measurement interval

  $\tilde{W}_k^0 = \max(1, \min(W_k^0, \frac{\tilde{A}_k^0}{F_k^0}))$
  $F_k^0 = \frac{C_k^0}{\tilde{W}_k^0}$
```

Figure 7.3 The Weighted Fairness Algorithm

7.2.2 The WFA Convergence

Let k be the bottlenecked link. The number of flows traversing link k is N, where M is the number of flows bottlenecked elsewhere or at their ingress points, and $S=N-M$ is the number of flows bottlenecked at link k. Let $\hat{r}_1, \hat{r}_2, \ldots, \hat{r}_M$ be the rates of flows bottlenecked elsewhere, and $r_1, r_2, \ldots, r_S$ be the rates of flows bottlenecked at link k.

At the end of the *nth* measurement interval ($t=nT$), the effective sum of weights is estimated as

$$\tilde{W}_k(n) = \frac{\tilde{A}_k(n)}{F_k(n)} \quad , \qquad 7.24$$

where $\tilde{A}_k(n) = \sum_{i=1}^{N} r_i$ is the arrival rate at node k and $F_k(n)$ is the advertised fair rate of the previous interval.

The next advertised fair rate is

$$F_k(n+1) = \frac{C_k}{\tilde{W}_k(n)} \quad . \qquad 7.25$$

Substituting Eq. (7.24) into Eq. (7.25) yields

$$F_k(n+1) = F_k(n)\frac{C_k}{\tilde{A}_k(n)} \quad . \qquad 7.26$$

Define $\alpha(n) = \frac{\tilde{A}_k(n)}{C_k}$ as the load factor, and rewrite Eq. (7.26) as

$$F_k(n+1) = \frac{F_k(n)}{\alpha(n)} \quad . \qquad 7.27$$

According to the load factor value, two cases are considered. First, consider the case where the load factor $\alpha(n)$ is less than one. In this case, the arrival rate is less than the available bandwidth and the link is under loaded. According to Eq. (7.27), the advertised fair rate will increase. If all flows are bottlenecked elsewhere (S=0), the weighted fair rate has been achieved. On the other hand, if there are some flows bottlenecked at link k (S>0), the bottlenecked flows will continue to increase their rates until the load factor becomes greater than or equal to one.

Second, consider the case where the load factor $\alpha(n)$ is greater than one. In this case, the arrival rate is greater than the available bandwidth and the link is over loaded. According to Eq. (7.27), the advertised fair rate will decrease and the participating flows will decrease their rates. This will continue until the load factor becomes less than or equal to one.

It is obvious from the above two cases that the load factor oscillates around one and converges to one. Thus, in the following analysis, we assume that the load factor is close to one.

Next, we shall show that the iterative algorithm Eq. (7.25) will generate a sequence of $F_k(n)$ that will converge to the optimal value of the fair rate, $F_k^* = \frac{C_k - \sum_{i=1}^{M} \hat{r}_i}{\sum_{i=1}^{S} w_i}$.

Based on Eq. (7.24)-(7.25), the following differential equation is obtained.

$$F_k(n+1) = F_k(n) + \frac{1}{\tilde{W}_k(n)}[C_k - \tilde{A}_k(n)] \ . \qquad 7.28$$

This is in the form of

$$F_k(n+1) = F_k(n) + \lambda\left[\nabla^2 D(F_k(n))\right]^{-1} \nabla D(F_k(n)) \ . \qquad 7.29$$

That is, the link fair rate is adjusted in the direction of the gradient, where

$$\nabla D(F_k(n)) = C_k - \tilde{A}_k(F_k(n)) \ . \qquad 7.30$$

Here, λ is a positive step size, and in our case is equal to one, and $\left[\nabla^2 D(F_k(n))\right]^{-1}$ is the inverse of the Hessian.

Again, it is well known that the Newton method (Eq. (7.28)), where the gradient is scaled by the inverse of the Hessian, typically converges faster than the gradient projection; see [10, pp.201].

The Hessian $\nabla^2 D(F_k(n)) = \tilde{W}_k(n)$ is approximated by using two points, the current point of $(\tilde{A}_k(n)\ ,\ F_k)$ and the origin (0, 0).

Hence, the above iterative equation converges, and the stable value of the link advertised fair rate is detailed as follows:

First, assume that all the flows are bottlenecked at link *k*. In this case, *M=0, S=N*, and flows are running at their weighted fair rate $r_i(n) = w_i F_k(n)$, and the total arrival rate at node *k* is

$$\tilde{A}_k = F_k(n) \sum_{i=1}^{N} w_i \ . \qquad 7.31$$

Substituting the value of $\tilde{A}_k$ into Eq. (7.26) with a load factor $\alpha(n)$ of one at the steady state yields

$$F_k(n+1) = \frac{C_k}{\sum_{i=1}^{N} w_i} \ , \qquad 7.32$$

which is the desired value for F_k .

Finally, assume that some flows are bottlenecked elsewhere. These flows will have their rates $\hat{r}_1, \hat{r}_2, ..., \hat{r}_M$ stabilized and the allocated bandwidth for these flows is $B = \sum_{i=1}^{M} \hat{r}_i$.

Since we have a load factor $\alpha(n)$ of one at the steady state, we have

$$\sum_{i=1}^{S} r_i = C_k - \sum_{i=1}^{M} \hat{r}_i \ , \qquad 7.33$$

and

$$\sum_{i=1}^{S} r_i = F_k(n) \sum_{i=1}^{S} w_i \qquad . \qquad 7.34$$

Substituting Eq. (7.33) into Eq. (7.34) yields

$$F_k(n) = \frac{C_k - \sum_{i=1}^{M} \hat{r}_i}{\sum_{i=1}^{S} w_i} \qquad . \qquad 7.35$$

Substituting the value of $F_k(n)$ into Eq. (7.27) yields

$$F_k(n+1) = \frac{C_k - \sum_{i=1}^{M} \hat{r}_i}{\sum_{i=1}^{S} w_i} \qquad 7.26$$

which is indeed the desired value for F_k .

■

7.3 Simulation Results

Here, we have considered two scenarios. First, we consider the scenario illustrated in Figure 7.4, where all the links have the same capacity of 622 Mbps, and each link has a propagation delay of 0.1 ms. All flows are UDP flows where all flows start at time 0. Flow (1,5) is greedy, and flows (2,5) ,(3,5) and (0,6) have a rate equal to 150Mbps, 100Mbps, and 311Mbps, respectively. The measurement time interval was set to $T = 1$ ms.

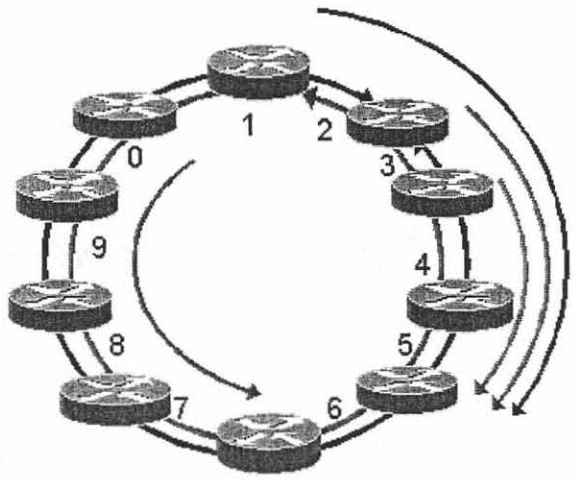


Figure 7. 4 Simulation set-up (scenario I)

Table 7.2 shows the weight values for the two different routing policies. The performance of these policies using the RA-RPR algorithm and the WFA algorithm, the limitation of the current RPR fairness algorithm, and the limitation of the shortest path routing policy are investigated.

Routing policy	SP		WF	
	w_{ij}^{1}	w_{ij}^{0}	w_{ij}^{1}	w_{ij}^{0}
Flow(0,6)	1	0	1	1
Flow(1,5)	0	1	1	1
Flow(2,5)	0	1	1	1
Flow(3,5)	0	1	1	1

Table 7.2 The Routing Policies (scenario I)

Figures 7.5 and 7.6 show that both the Aggressive Mode (AM) and the Conservative Mode (CM) of the current RPR fairness algorithm suffer from a severe oscillation due to the unbalanced traffic at link 3 in the outer ring, thus resulting in a bandwidth loss. Moreover, due to the shortest path routing policy, the inner ring is under-utilized.

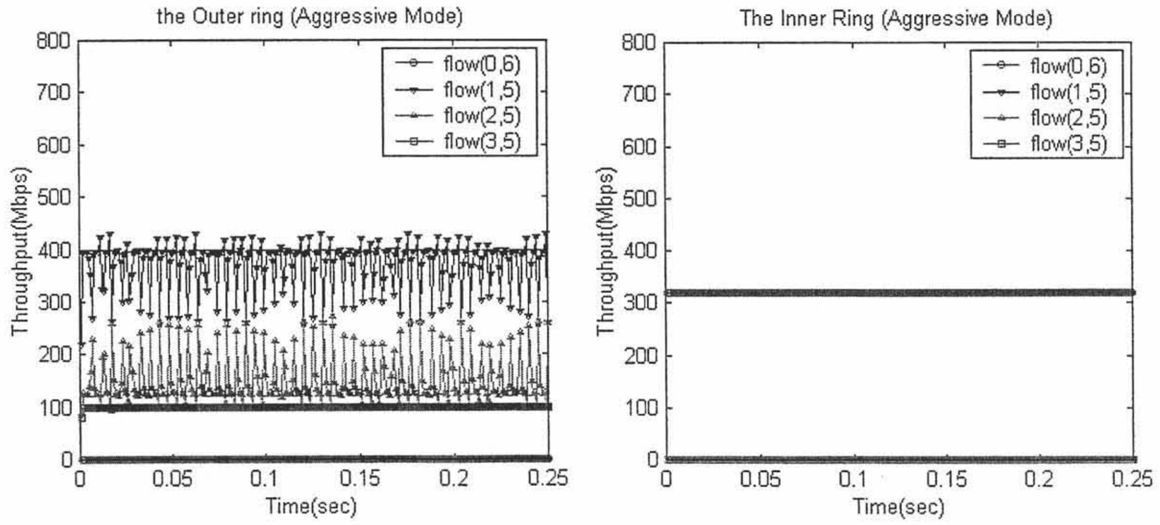


Figure 7. 5 RPR Aggressive Mode (scenario I)

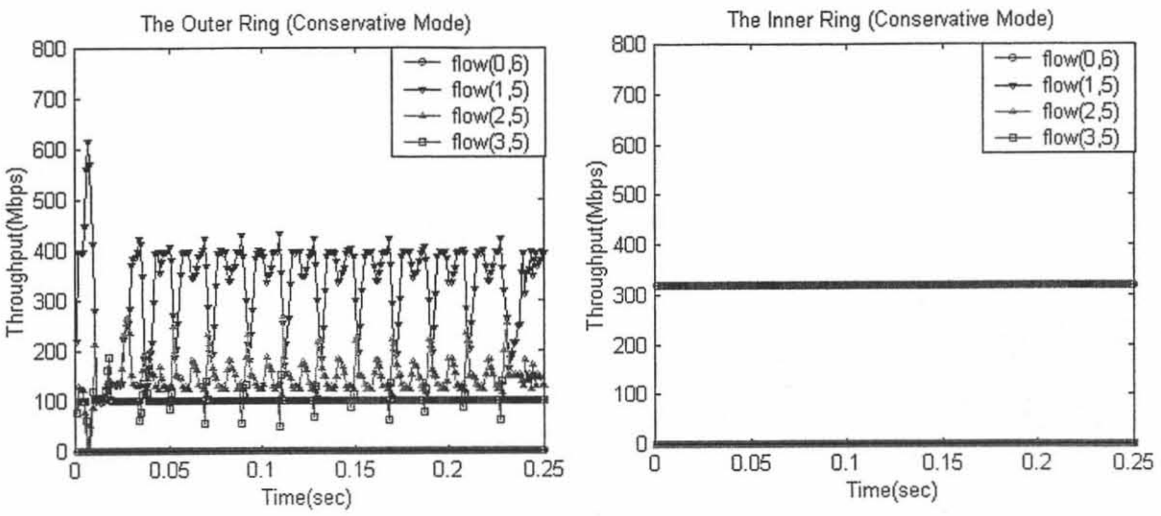


Figure 7.6 RPR Conservative Mode (scenario I)

Figure 7.7 shows the performance of the RA-RPR algorithm with the shortest path routing policy. The oscillation is significantly damped, but the inner ring is still under utilized due to the shortest path routing policy. Since each node is transmitting only one flow, the WFA algorithm performs similarly to the RA-RPR algorithm.

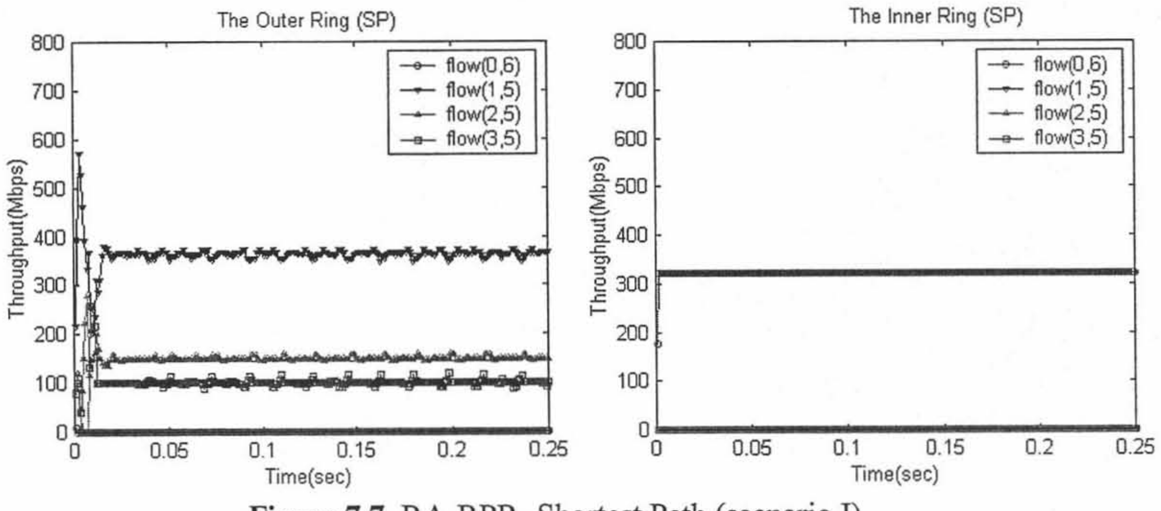


Figure 7.7 RA-RPR -Shortest Path (scenario I)

The performance of the RA-RPR algorithm with the weighted fairness routing policy is shown in Figure 7.8. Since flows (2, 5) and (3, 5) have their demands less than the weighted fair rate at the congested link (link 3 at the outer ring), these flows will select the outer ring for all their routed traffic. On the other hand, link 0 in the inner ring is now shared by flow (0, 6) and flow (1, 5). Flow (0, 6) is satisfied with the weighted fair rate (622Mbps/2), implying that flow (0,6) will only utilize the inner ring. Finally, flow (1, 3) would be able to claim its fair rate and the unused bandwidth at link 3 in the outer ring (370Mbps). Moreover, flow (1, 3) is able to use its fair rate in the inner ring to satisfy its demand. The result using the WFA algorithm is similar to that of RA-RPR, and both algorithms are able to maximize the total ring utilization by allowing flows to transmit in both rings in a weighted manner.

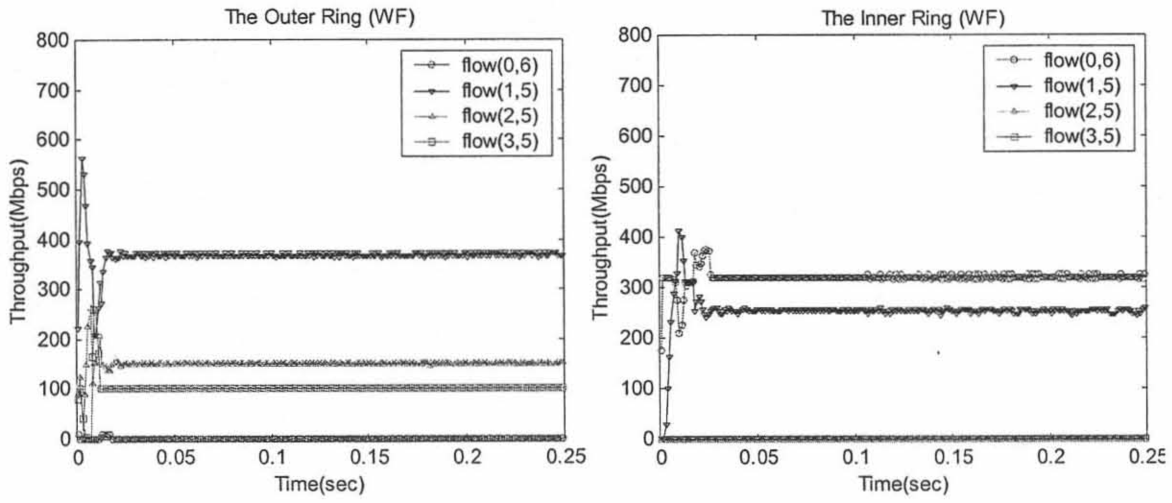


Figure 7.8 RA-RPR –Weighted Fairness (scenario I)

Finally, we consider the scenario illustrated in Figure 7.9. Flow (1,5) is greedy and start at time 0. Flows (0,6) and (2,5) have a rate equal to 311Mbps and start at time 0 and 0.2 seconds, respectively. Flow (1,3) has a rate equal to 155.5Mbps and starts at time 0.1 seconds.

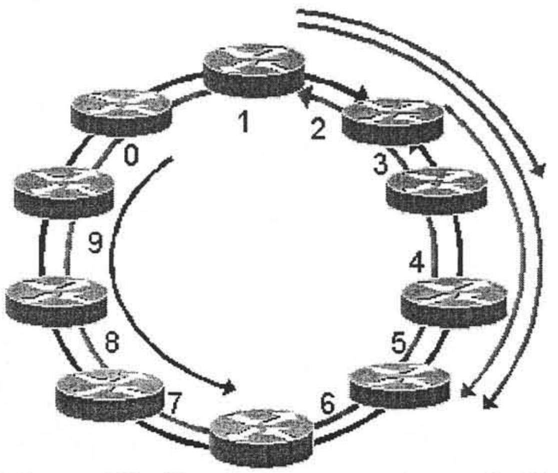


Figure 7.9 Simulation set-up (scenario II)

Table 3 shows the weight values for the two different routing policies. The performance of these policies using the RA-RPR algorithm and the WFA algorithm, and the limitation of the shortest path routing policy are investigated.

Routing policy	SP		WF	
	w_{ij}^{1}	w_{ij}^{0}	w_{ij}^{1}	w_{ij}^{0}
Flow(0,6)	1	0	1	1/6
Flow(1,3)	0	1	1/8	1
Flow(1,5)	0	1	1/6	1
Flow(2,5)	0	1	1/7	1

Table 7.4 The Routing Policies (scenario II)

Figure 7.10 shows the performance of the RA-RPR algorithm with the shortest path routing policy. The per-source fair rate at link 2 in the outer ring is 311Mbps. Thus, flow (1,5) shares the bandwidth allocated for source 1 with flow (1,3) and transmits at a rate equal to 155.5Mbps. Due to the shortest path routing policy, flow (1,5) is unnecessarily limited while the inner ring is under-utilized.

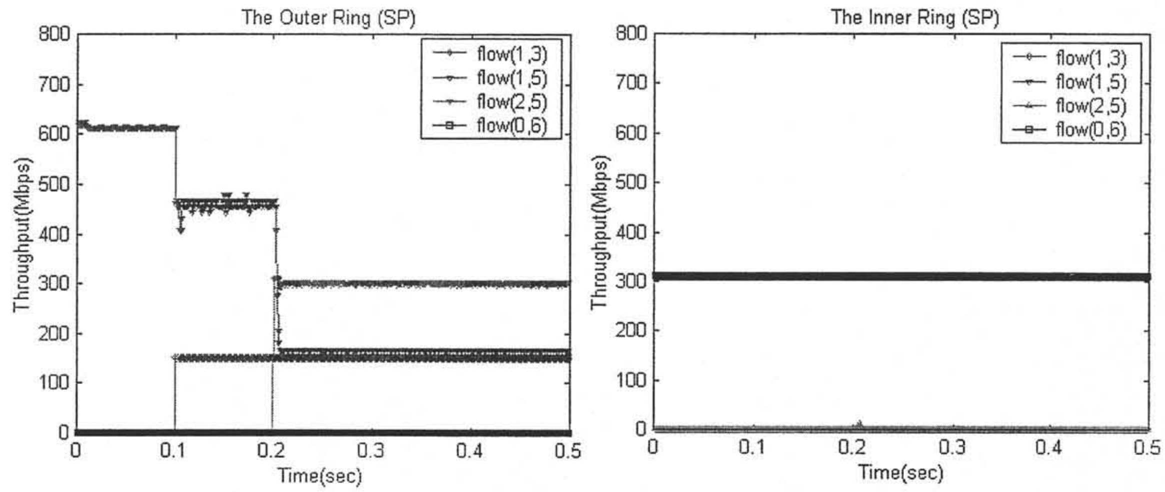


Figure 7.10 RA-RPR -Shortest Path (scenario II)

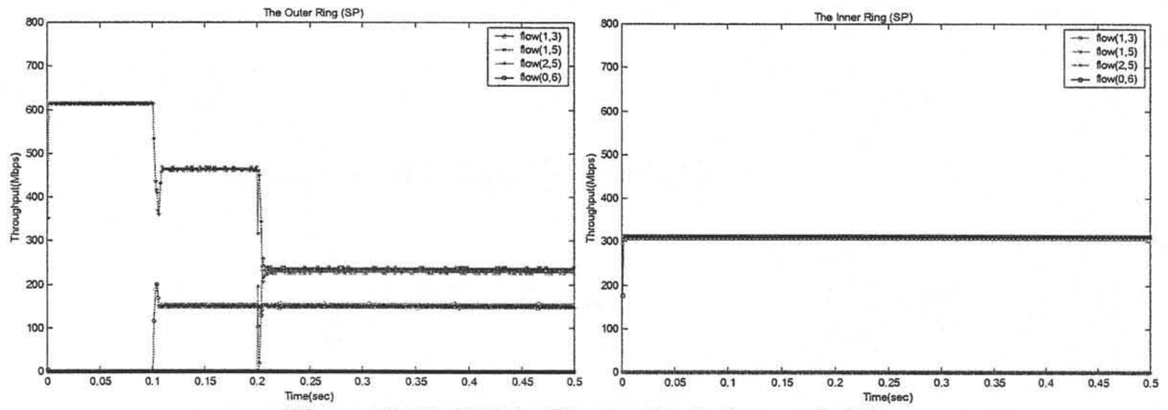


Figure 7.11 WFA -Shortest Path (scenario II)

The performance of the WFA algorithm with the shortest path routing policy is shown in Figure 7.11. Since flow(1,3) is running at a rate less than the per-flow fair share at link 2 in the outer ring , the unused bandwidth will be divided among flow(1,5) and flow (2,5).

The WFA algorithm achieves per-flow fairness where flow (1,5) is running at a rate equal to 233Mbps instead of 155.5Mbps using the RA-RPR algorithm. The inner ring is under utilized due to the shortest path routing policy.

The performance of the RA-RPR algorithm with the weighted fairness routing policy is shown in Figure 7.12. The total ring utilization is maximized by allowing flow (1, 5) to use its fair rate at link 2 in the outer ring (155.5Mbps) and its weighted fair rate in the inner ring, $(w^1_{1,5}\zeta^1_{1,5})$.

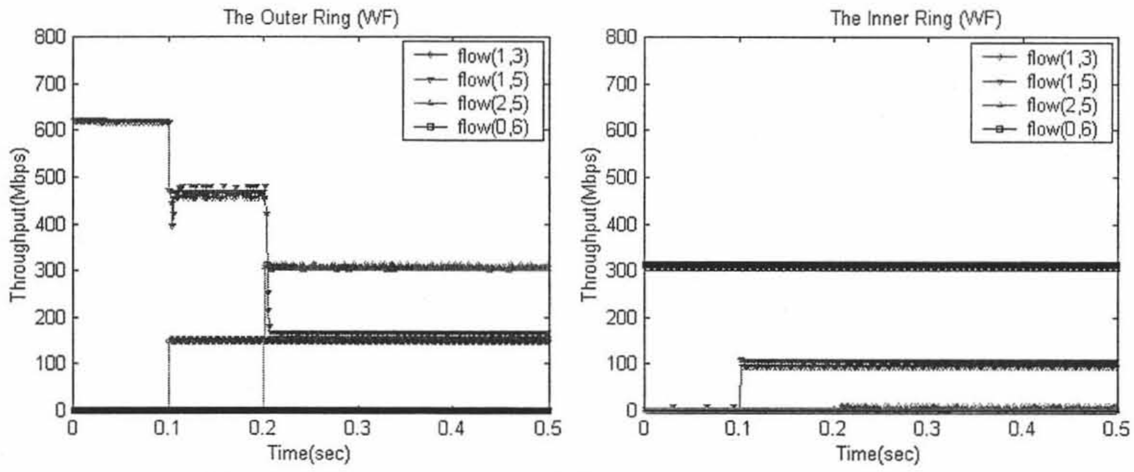


Figure 7.12 RA-RPR –Weighted Fairness (scenario II)

Figure 7.13 shows the performance of the RA-RPR algorithm with the weighted fairness routing policy. Flow (1, 5) and flow (2,5) use their fair shares at link 2 in the outer ring (233Mbps) plus their weighted fair rates in the inner ring $w^1_{1,5}\zeta^1_{1,5}$ and $w^1_{2,5}\zeta^1_{2,5}$, respectively.

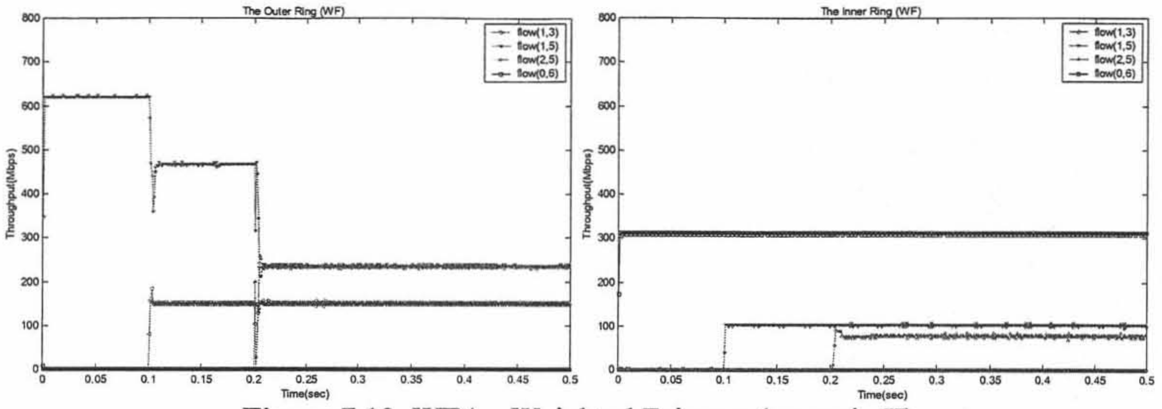


Figure 7.13 WFA - Weighted Fairness (scenario II)

7.4 Summary

The current RPR fairness algorithm uses the shortest path routing policy which is shown to be inefficient and unfair. In this chapter, we have proposed two Routing Algorithms. The first algorithm is the Routing Algorithm for RPR (RA-RPR) that adapts the RIAS fairness concept. The second algorithm is the Weighted Fairness Algorithm (WFA) which adapts the per-flow fairness concept. Both algorithms are demonstrated to be fair and able to maximize the total ring utilization by enabling RPR nodes to transmit in both rings in a weighted manner.

CHAPTER 8

CONCLUSIONS

In this dissertation, we have provided an overview of the Resilient Packet Ring (RPR) and its features. The limitations of the current RPR standard have been investigated and several solutions have been proposed. To allocate the bandwidth fairly among the nodes in the ring, two bandwidth allocation algorithms have been proposed. DBA is a rate based algorithm which achieves fairness at a very low complexity. On the other hand, ABA uses the same measurements as the RPR aggressive mode (RPR-AM) and improves the performance significantly. Moreover, we have proposed the VDQ scheme along with the Simple Scheduling Algorithm (SSA) to overcome the problem of the head of line blocking associated with the current RPR traffic shaping. Also, we have investigated the limitation of the shortest path routing policy adopted by the current RPR standard and proposed two routing algorithms to maximize the ring utilization.

In our future work, we will focus on two tasks. First, we will apply Control Theory to solve the congestion control issue. Our preliminary analysis has drawn two interesting observations. First, better convergence performance may be achieved by using the PID controller. Second, our proposed algorithms (the DBA and Improved DBA algorithm) in Chapter 4 can be categorized as a special case of the PID controller, i.e., the P- and PI-controller, respectively. Finally, we will perform delay analysis to the existing fairness algorithms and their effect on the access delay of Class A and Class B traffics at their ingress points.

REFERENCES

1. ANSI T1.105.01-2000: Synchronous Optical Network (SONET)- Automatic Protection.

2. I. Cidon and Y. Ofek. Metaring - a full-duplex ring with fairness and spatial reuse. IEEE Transactions on Communications, 41(1):110–120, January 1993.

3. I. Cidon, L. Georgiadis, R. Guerin, Y. Shavitt: Improved fairness algorithms for rings with spatial reuse. INFOCOM '94. Networking for Global Communications. IEEE, 1994

4. *IEEE Standard 802.5–1989*, "IEEE standard for token ring".

5. F.E. Ross, "Overview of FDDI: The Fiber Distributed Data Interface", IEEE J. on Selected Areas in Communications, Vol. 7, No. 7, September 1989.

6. *IEEE Standard 802.17:* Resilient Packet Ring http://ieee802.org/17

7. *IEEE Draft P802.17, draft 3.0*, Resilient Packet Ring, November 2003.

8. F. Davik, M. Yilmaz, S. Gjessing, N. Uzun, "IEEE 802.17 resilient packet ring tutorial," *IEEE Communications Magazine*, vol. 42, no. 3, March 2004, pp.112-118.

9. V. Gambiroza, P. Yuan, and E. Knightly, ``The IEEE 802.17 media access protocol for high-speed metropolitan area resilient packet rings," *IEEE Network,* 18(3):8 15, May 2004.

10. V. Gambiroza, Y. Liu, P. Yuan, and E. Knightly, "High-Performance Fair Bandwidth Allocation for Resilient Packet Rings," in *Proceedings of the 15th ITC Specialist Seminar on Internet Traffic Engineering and Traffic Management*, Wurzburg, Germany, July 2002.

11. V. Gambiroza, P. Yuan, L. Balzano, Y. Liu, S. Sheafor, and E. Knightly, ``Design, Analysis, and Implementation of DVSR: A Fair, High Performance Protocol for Packet Rings," *IEEE/ACM Transactions on Networking,* 12(1):85102, February 2004.

12. F.Alharbi and N. Ansari, "Low Complexity Distributed Bandwidth Allocation for Resilient Packet Ring Networks," *Proceedings of IEEE Workshop on High Performance Switching and Routing (HPSR2004),* pp.277-281, April 2004.

13. F.Alharbi and N. Ansari, "Adaptive Fairness Algorithm for Resilient Packet Ring Networks," *Proceedings of the first IFIP international conference on Wireless and Optical Communication Networks (WOCN2004), pp.86-89,* June 2004.

14. F.Alharbi and N. Ansari, "A Novel Fairness Algorithm for Resilient Packet ring Networks with Low Computational and Hardware Complexity," *Proceedings of 13th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN2004), pp 11-16,* Mill Valley, Ca, USA, April 24-27, 2004.

15. F. Davik and S. Gjessing, "The Stability of the Resilient Packet Ring Aggressive Fairness Algorithm," *Proceedings of 13th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN2004), pp 17- 22*, Mill Valley, Ca, USA, April 24-27, 2004.

16. S. Fahmy, R. Jain, S. Kalyanaraman. R. Goyal and B. Vandalore, "On determining the fair bandwidth share for ABR connections in ATM networks," *Proceedings of the IEEE International Conference on Communications (ICC98),* Volume 3, pp.1485-1491, June 1998.

17. D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation*, Prentice Hall, 1989.

18. D. Bertsekas and R. Gallager. Data Networks. Prentice Hall, 1992.

19. S. Gjessing, "The Simula RPR Simulator implemented in Java," Simula Research Laboratory Technical Report 2003-12, December 2003.

20. K.M. passion and S. Yurkovish, Fuzzy Control, Addison Wesley 1998

21. IEEE Draft P802.17, draft 1.0, Resilient Packet Ring, August 12, 2002.

22. F.Alharbi and N. Ansari, "Distributed Bandwidth Allocation for Resilient Packet Ring Networks," *accepted for publication in the Journal of Computer Network.*

23. F.Alharbi and N. Ansari, "SSA: Simple Scheduling Algorithm for Resilient Packet Ring Networks," *submitted to the IEE Proceeding on Communications.*

24. F.Alharbi and N. Ansari, "The Weighted Fairness Algorithm for Efficient Routing and Maximum Utilization in Resilient Packet Ring Networks," *submitted to the Journal of Lightwave Technology.*