Resource Allocation in Computer Networks: Fundamental Principles and Practical Strategies

A Thesis

Submitted to the Faculty

of

Drexel University

by

Yunkai Zhou

in partial fulfillment of the

requirements for the degree

of

Doctor of Philosophy

May 2003

Dedications

This thesis is dedicated to my family, especially to my wife, Dr. Shan Cheng, and my parents, Mr. Guanghua Zhou and Mrs. Honggen Shi, for their sincere support, encouragement and love.

Acknowledgments

In retrospect as I approach the completion of my doctorate, I realize that I am deeply indebted to many people's assistance and support. I would like to express my genuine gratitude to each of them, although it would be impossible for me to name all.

First of all, I would like to sincerely thank my advisor, Dr. Harish Sethu, for his tremendous time and effort spent in leading, supporting and encouraging me during the last five years. His passion for challenges has given me inspiration; his knowledge of engineering has given me guidance; his perseverance in research has given me confidence. Without his help and effort, it would be impossible for me to even get close to this point. I am also grateful to him for being not only a mentor in my professional life but also a friend in my personal life.

I want to express my gratitude to all committee members in my candidacy examination and/or my dissertation defense, Dr. Maja Bystrom, Dr. Athina Petropulu, Dr. William Regli, Dr. Warren Rosen and Dr. Oleh Tretiak, for their time and valuable suggestions.

Many thanks are due to my friends in the Department of Electrical and Computer Engineering, who make my life here memorable. I thank Haiguang Cheng and Xueshi Yang for discussions on traffic modeling. I thank Salil Kanhere, Alpa Parekh, Hongyuan Shi, Madhusudan Hosaagrahara, Harpreet Arora, Adam O'Donnell and Kunal Shah from our lab for their collaboration, discussions and help during these years. I also want to thank all the professors, staff and students in ECE for making the Department such a joyful working and studying environment.

I am greatly grateful to my parents, Mr. Guanghua Zhou and Mrs. Honggen Shi, for their continuous encouragement and support. Finally, my special gratitude is due to my wife, Dr. Shan Cheng, who has always provided me unconditional love and endless support. She is the person who is the happiest for me when I make progress, little or great;

she is the person who believes in me and encourages me when I am frustrated; she is the person who keeps me focused when I am in doubt; she is the person who makes me realize that life is so beautiful.

Table of Contents

List of Tables

List of Figures

Abstract

Resource Allocation in Computer Networks: Fundamental Principles and Practical Strategies

Yunkai Zhou

Harish Sethu, Ph.D.

Fairness in the allocation of resources in a network shared among multiple flows of traffic is an intuitively desirable property with many practical benefits. Fairness in traffic management can improve the isolation between traffic streams, offer a more predictable performance, eliminate certain kinds of transient bottlenecks and may serve as a critical component of a strategy to achieve certain guaranteed services such as delay bounds and minimum bandwidths. Fairness in bandwidth allocation over a shared link has been extensively researched over the last decade. However, as flows of traffic traverse the computer network, they share not only bandwidth resources, but also multiple other types of resources such as processor, buffer, and power in mobile systems. If the network is not fair in allocating any of the shared resources, denial of service attacks based on an excessive use of this resource becomes possible. Therefore, the desired eventual goal is overall fairness in the use of all the resources in the network.

This dissertation is concerned with achieving fairness in the joint allocation of multiple heterogeneous resources. We consider resources as either prioritized (such as bandwidth and buffer resources) or essential (such as processing and bandwidth resources). For each type of these systems, we present a simple but powerful general principle for defining fairness in such systems based on any of the classic notions of fairness such as max-min fairness, proportional fairness and utility max-min fairness defined for a single resource. Using max-min fairness as an example, we apply the principles to a system with a shared buffer and a shared link, and a system with a shared processor and a shared link, and propose practical and provably fair algorithms for the joint allocation of buffer and bandwidth

resources, and the joint allocation of processing and bandwidth resources. We demonstrate the fairness achieved by our algorithms through simulation results using both synthetic traffic and real traffic traces. The principles and the algorithms detailed in this dissertation may also be applied in a variety of other contexts involving resource sharing.

Chapter 1. Introduction

1.1 Motivation: Resource Allocation and Fairness

Fairness is an intuitively desirable property in the allocation of shared resources in a variety of contexts. In sociology and economics, fairness metrics and strategies have been exhaustively studied in the distribution of wealth and welfare [1]; in operating systems, they have been studied in task scheduling and the allocation of access to resources such as memory, bus and I/O [2,3]; in computer networks, they have been extensively studied in the allocation of available bandwidth among competing flows [4–10]. Besides being intuitively desirable, fairness in the allocation of shared resources has many practical benefits. In this section, we discuss the motivation behind the research presented in this dissertation within the context of computer networks.

In a computer network, flows of traffic share multiple types of resources such as bandwidth, buffers, and router processors. Congestion occurs when the available capacity of any resource is insufficient to satisfy the requirements of all competing flows. For example, when the total bandwidth demand of all flows headed to an output link is greater than the peak bandwidth rate of the link, packets have to be either buffered or dropped causing either delays or packet losses and consequently, a degradation in performance. A proper and fair management of each congested resource tends to improve the overall system performance, just as a police officer directing traffic at a congested intersection typically results in smoother traffic, shorter delays and faster congestion relief. The following are among the most important advantages of fair allocation of a shared resource in networks:

- *Fair resource allocation improves performance by eliminating some transient bottlenecks.* In a network with multiple hops, fair allocation policies in early hops can create balanced traffic loads and thus reduce the probability of traffic flows causing

congestion in later hops.

- *Fair resource allocation enables QoS guarantees.* A scheduler that allocates resources fairly generates output traffic that has a more predictable pattern than the input traffic. In fact, fair allocation algorithms have been shown to be able to provide certain quality-of-service guarantees. For example, a fair bandwidth scheduling policy can be used to provide minimum bandwidths and guaranteed delay bounds [5]. Multimedia sources such as video and audio streaming applications generate traffic requiring not only a minimum bandwidth for stable transmission, but a guaranteed delay and delay jitter for smooth playback as well. Fair scheduling algorithms in network devices, therefore, are often a critical component of Quality-of-Service (QoS) mechanisms proposed to satisfy the requirements of such applications.

- *Fair resource allocation provides isolation between traffic streams.* Since fair allocation policies ensure certain QoS guarantees, well-behaved flows can be protected from other misbehaving flows triggered by malicious users, malfunctioning software, or just heavy users. Such isolation is essential in a large-scale public network such as the Internet even if the resources are not likely to be congested most of the time.

- *Fair resource allocation enhances system security by countering certain kinds of denial-of-service attacks.* If any given resource is not allocated fairly, a denial-of-service (DoS) attack based on an excessive use of the given resource becomes possible. For example, if the router processor cycles are not allocated fairly, the router may be vulnerable to a DoS attack based on the processing resource (such as by using unnecessary or malicious optional headers in packets).

For all of these reasons, fair schedulers for the allocation of available *bandwidth* have now found widespread implementation in switches and Internet routers [11, 12]. However, bandwidth on a link is only one among several kinds of resources shared by multiple flows

in a typical network. As flows of traffic traverse through a network, they share with other flows a variety of resources that include the following:

- *Bandwidth*. Fair allocation of bandwidth on a link has been studied extensively in the literature. In the edge networks of the Internet or in wireless networks, bandwidth tends to be one of the most critical resources that should ideally be allocated fair amongst all the competing users.

- *Buffer*. Buffers in networks are used to improve throughput at output links. In the absence of a buffer, packets that arrive at a busy link have to be discarded, leading to packet losses and therefore, a loss in throughput. With a buffer resource available, an arriving packet can be temporarily stored in the buffer while the output link is busy transmitting another packet. When the link later becomes idle and available to transmit the next packet, the temporarily stored traffic can be transmitted onto the link, thus avoiding packet losses and improving the overall throughput.

- *Processor*. In packet-switched computer networks, for each arriving packet, switches and routers have to retrieve necessary information from the packet header, determine the destination of the packet and the forwarding interface corresponding to the destination, maintain and update certain information such as average arriving rate for the flow to which the packet belongs, and in many cases, modify the packet itself for the purpose of network or traffic management. Each of these tasks requires a certain number of processing cycles from the CPU, and therefore, the processing resource is another important resource shared by all traffic flows. With the current pervasiveness of high-bandwidth long-haul optical links in the Internet backbone, and with the occasional trend toward using over-provisioning as the solution to congestion in the edge networks, a router's processor is often also a critical resource to which, ideally speaking, all competing flows should have fair access.

- *Power*. Normally powered by batteries, mobile devices such as laptops, PDA's, and cellular phones have a finite power lifetime. A fair allocation of power amongst competing users becomes especially critical in certain kinds of networks such as mobile ad hoc networks and sensor networks, where power is the bottleneck resource given today's technological constraints [13].

Given these various types of resources, more than one of which could be congested at any given time, the allocation policy with respect to any one of these resources can have a significant impact on the overall performance and QoS achieved by flows. Even though fair scheduling of bandwidth over a link has received the most attention, the most desirable goal is overall fairness in the *joint* allocation of all resources shared by the flows of traffic and not just one specific kind of resource such as the link bandwidth. However, a rigorous theoretical framework that may be universally employed as a guide in the design of practical algorithmic strategies for the joint allocation of such heterogeneous sets of resources does not exist. This dissertation tries to establish such a theoretical foundation, and based on this foundation, develop practical strategies for achieving fairness in the joint allocation of such resources.

1.2 Fairness Criteria and Notions of Fairness

In attempting the design of a fair resource allocation policy, one has to first define a notion of fairness that determines the criteria by which one can judge the fairness achieved by an allocation policy. To solve the problem of what is fair if multiple entities compete for a single shared resource, many fairness criteria have been proposed in the literature. The most popular ones among them are max-min fairness, proportional fairness, and utility max-min fairness, which are described below. Without loss of generality, throughout this dissertation, we assume that the competing entities are network flows.

1.2.1 Max-Min Fairness

The max-min fair share policy of allocating a shared resource among multiple flows with equal rights to the resource but unequal demands, follows the following principles [14, 15]:

- The shared resource is allocated in order of increasing demand.
- No flow receives a share of the resource larger than its demand.
- Flows with unsatisfied demands receive equal shares of the resource.

The notion of max-min fairness can also be defined in the following equivalent way: no flow can increase its allocation without reducing the allocation of another flow with less or equal demand. Under max-min fairness, given no additional resources, an unsatisfied flow cannot increase its allocation by merely demanding more.

When there are weights associated with the entities demanding a share of the resource, the max-min fair procedure is based on resource allocations that are normalized by the corresponding weight. Consider N sources, labeled $1, 2, \ldots, N$, and with a weight w_i associated with source i. Let d_i be the demand corresponding to source i. Given R as the size of the resource shared among these N sources, the max-min fair share algorithm uniquely defines a fair allocation. For the sake of convenience, throughout this dissertation we use vectors to indicate values belonging to a set of sources. We denote a vector by the indexed value in a pair of square brackets, and when the context requires it, with a specification of the boundaries of the index values. For instance, we denote the weight vector as $[w_i : 1 \leq i \leq N]$ or just $[w_i]$, and the demand vector as $[d_i : 1 \leq i \leq N]$ or just $[d_i]$. Given the demand vector, the weight vector and the total available resource amount, R, the max-min fair share allocation is given by

$$[a_i] = \mathcal{F}_{\mathrm{MMF}}(R, [d_i], [w_i]) \tag{1.1}$$

```
1   F_MMF(R, d, w):
2       N ← Length(d);
3       Source ← {1, 2, ..., N};
4       Resource ← R;
5       Weight ← Σ_{i=1}^{N} w(i);
6       for each i, 1 ≤ i ≤ N
7           d_n(i) ← d(i)/w(i);
8       end for;
9       while (Source ≠ φ)
10          Find source i with minimum d_n in Source;
11          if (d_n(i) < Resource/Weight) then
12              a(i) ← d(i);
13              Remove source i from Source;
14              Resource ← Resource − a(i);
15              Weight ← Weight − w(i);
16          else
17              for each j, 1 ≤ j ≤ N
18                  if (j ∈ Source) then
19                      a(j) ← w(j)× Resource/Weight;
20                  end if;
21              end for;
22              Source ← φ;
23          end if;
24      end while;
25      return a;
```

Notation:

R	total resource amount	input variable
d	source demand	input vector
w	source weight	input vector
N	total number of sources	constant value
Source	set of unallocated sources	variable
Resource	amount of unallocated resource	variable
Weight	total weight of unallocated sources	variable
d_n	normalized demand	vector
a	allocation	output vector

Figure 1.1: Pseudo-code of max-min fair share.

where a_i is the max-min fair allocation for flow i, and $\mathcal{F}_{\text{MMF}}$, the max-min fair procedure, is a function of the available amount of the resource, the demand vector and the weight vector. For the sake of completeness in our definitions of fairness, Fig. 1.1 provides a pseudo-code of the $\mathcal{F}_{\text{MMF}}$ procedure, which returns an allocation vector $[a_i : 1 \leq i \leq N]$. In defining this function formally, we assume, of course, that vector elements in the same positions in their respective vectors always correspond to the same source.

1.2.2 Utility Max-Min Fairness

The notion of utility max-min fairness is similar to that of max-min fairness, except that under utility max-min fairness each flow is associated with a utility function [16] and it is the utility achieved by each flow that needs to be allocated fairly. The utility function of a flow under an allocated amount of resource indicates the level of performance (or "satisfaction") this flow achieves when given this amount of resource, and the utility functions for different flows can be different [17]. The following are a few examples for some common types of traffic.

- Elastic traffic such as E-mail, Telnet and FTP applications may have a convex utility. In other words, the utility increases rapidly with the allocated amount of resource when the amount is relatively small, and it saturates above a certain point. Fig. 1.2(a) illustrates this type of utility functions.

- Real-time traffic such as video and audio streams has a minimum requirement on the amount of allocated resources. The utility achieved by this type of traffic becomes substantial only when this minimum requirement has been satisfied. On the other side, overprovisioning would not further increase the utility significantly. As a summary, the utility function of real-time traffic normally has a threshold point corresponding to the minimum requirement, as shown in Fig. 1.2(b).

- Rate-adaptive traffic has a similar utility function as real-time traffic, except that the

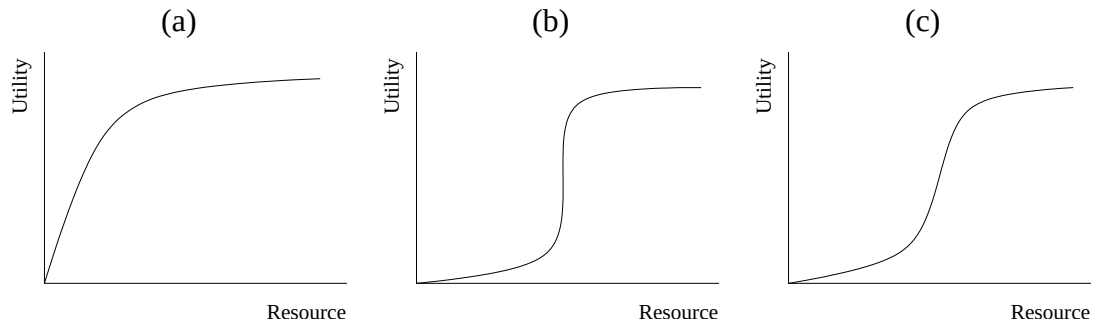

Figure 1.2: Examples of utility functions for: (a) elastic traffic; (b) real-time traffic; (c) rate-adaptive traffic.

utility differentiation at the point of minimum requirement is more smooth. In other words, the utility function of rate-adaptive traffic has a "knee point", as illustrated in Fig. 1.2(c).

Note that for a set of flows i, given the total amount of resource R, the demand vector $[d_i]$, the weight vector $[w_i]$ and the utility function of each flow, the allocation vector $[a_i]$ under utility max-min fairness is given by

$$[a_i] = \mathcal{F}_{\mathrm{UMMF}}(R, [d_i], [w_i]) \tag{1.2}$$

where the utility function of each flow is implicitly included in the function $\mathcal{F}_{\mathrm{UMMF}}$.

Max-min fairness can be considered as a special case of utility max-min fairness in the sense that in max-min fairness, all flows have the same linear utility function.

1.2.3 Proportional Fairness

Some researchers argue that max-min fairness gives higher priority to flows with small demands [18]. As an alternative, the notion of proportional fairness emphasizes less on flows with small demands. An allocation vector $[a_i]$ is said to be proportionally fair if and only if for any other feasible allocation $[a'_i]$, the aggregate of proportional changes is

non-positive, i.e.,

$$\sum_i \frac{a_i' - a_i}{a_i} \leq 0.$$

Here we refer to an allocation as a feasible one, if and only if, the shared resource is not overallocated under this allocation. This fairness criterion implies a logarithmic utility function.

Similar to the cases of max-min fairness and utility max-min fairness, proportional fairness also determines how to allocate the shared resource, given the resource amount R, the demand vector $[d_i]$ and the weight vector $[w_i]$. In other words, proportional fairness can be also represented as follows:

$$[a_i] = \mathcal{F}_{\mathrm{PF}}(R, [d_i], [w_i]). \tag{1.3}$$

1.2.4 General Notion of Fairness

Note that each of the above mentioned fairness criteria determines, in its own distinctive fashion, how a single resource should be allocated to competing flows with respect to their demands. Therefore, for the sake of convenience, we introduce a notation that allows a representation of any of these notions of fairness.

Consider a set of N flows, $1 \leq i \leq N$, competing for a single shared resource of amount R.[1] Denote by w_i the weight of flow i, indicating the flow's relative rightful share of the resources. For a flow under a Differentiated Services framework [19], its weight is determined by its traffic class among the 64 possible classes; for a flow in a best-effort network, its weight is typically the same as that of all other flows. Let d_i be the demand corresponding to flow i. Therefore, given the demand vector $[d_i]$, the weight vector $[w_i]$,

---

[1]Depending upon the property of the shared resource, the amount R may be measured either by the total capacity or the peak consumption rate. For example, in the case of buffer resource, the resource amount is measured in terms of the total capacity (i.e., in bytes); while in the case of link resource, the amount is measured in terms of the peak bandwidth rate (i.e., in bps). Similar situations occur for resource demands and allocations. Therefore, in the rest of this dissertation, we will not explicitly distinguish how the resources are measured, and generally refer to the term "resource amount".

and the total available resource amount R, any given notion of fairness may be represented as

$$[a_i] = \mathcal{F}(R, [d_i], [w_i]) \tag{1.4}$$

where a_i is the allocation for flow i based on the notion of fairness defined by the function $\mathcal{F}$. The function $\mathcal{F}$ is different for different notions of fairness such as max-min fairness, proportional fairness or utility max-min fairness.

One may notice that the utility functions are not explicitly presented in (1.4). However, the notion of fairness in (1.4) represents a general notation to describe how, given a certain vector of demands, one may determine the allocation of the resource for each flow, in order that the utilities corresponding to the allocations satisfy the given fairness notion with respect to the demands for utility. In other words, the notation of (1.4) implicitly incorporates utility functions into the notion of fairness. For example, max-min fairness implies a linear utility function, proportional fairness uses a logarithmic utility function, and in utility max-min, each flow determines its own utility function. The only constraint is that the utility functions are non-decreasing functions with respect to quantity of the allocated resource.

An equivalent representation of any notion of fairness uses normalized demands and allocations. Define the normalized demand of flow i, $\tilde{d}_i$, for the resource as follows:

$$\tilde{d}_i = \frac{d_i}{R}.$$

The normalized demand of flow i indicates the fractional share of the resource demanded by the flow. Define the normalized allocation of flow i, $\tilde{a}_i$, as follows:

$$\tilde{a}_i = \frac{a_i}{R}.$$

The normalized allocation of flow i indicates the fractional share of the resource allocated to flow i.

Therefore, given the normalized demand vector $[\tilde{d}_i]$ and the weight vector $[w_i]$, any given notion of fairness may be represented as a function as follows:

$$[\tilde{a}_i] = \mathcal{F}(C, [\tilde{d}_i], [w_i]). \tag{1.5}$$

Here C is the constraint, described later in greater detail, imposed on the system. Note that the notion of fairness in (1.5) imposes no dimension on any variable, thus making it applicable to systems with multiple heterogeneous resources.

The constraint C is used as a parameter in the function $\mathcal{F}$ because, given the same demand and weight vector, the fair allocation is different under different constraints imposed on the system. The constraint C can be used to indicate the performance level achieved by the allocation. For example, an allocation of no resource to any flow may also be considered a fair allocation by the max-min fair criterion albeit one that leads to very poor performance. In general, this parameter allows us to define the fairness of non-work-conserving allocation strategies by not imposing a specific level of performance achieved by the allocation in the definition of fairness. As a simple example, the constraint C can be just the sum of the utilities achieved by all flows.

Note that the resource amount R in (1.4) can be also considered as a simple constraint on the allocation policy: the total amount of allocated resources cannot exceed the resource amount R, i.e., $\sum_i a_i \leq R$. Therefore, the notion of fairness in (1.4) is just a special case of the notion of fairness in (1.5). Based on the context, we may use any one of these two notions of fairness. For example, in this dissertation, (1.4) is used when normalization is not necessary such as in considering buffer resources, while (1.5) is used when normalization is necessary such as in considering processing resources.

Given a system S and a notion of fairness $\mathcal{F}$, an ideal scheduling policy, denoted by $G_{\mathcal{F}}(S)$, is one that exactly achieves this notion of fairness in system S. For example, if $\mathcal{F}$ represents the function corresponding to the max-min fair policy with respect to the bandwidth [14, 15], and L represents a work-conserving system with a single shared link,

$G_{\mathcal{F}}(L)$ will denote the *Generalized Processor Sharing (GPS)* policy [6], the ideally fair scheduling policy for max-min fairness. Next we further discuss scheduling policies in bandwidth allocation under the notion of max-min fairness.

1.3 Fairness in Scheduling

1.3.1 Generalized Processor Sharing

The GPS scheduler, in the scheduling of bandwidth over a link, is an unimplementable but ideally fair scheduler that exactly achieves the max-min fair distribution of the bandwidths among the various flows [6, 15]. During each infinitesimal interval of time, the GPS scheduler visits each backlogged flow once and schedules an equal and infinitesimal amount of data for transmission over the output link. If different weights are imposed on network flows, the amount of data transmitted from each flow by the GPS scheduler is proportional to its weight during each infinitesimal interval of time. By this means, the GPS scheduler achieves max-min fairness.

It is apparent that the GPS scheduler is an ideal policy and cannot be implemented in real systems, where network traffic is packetized and flow packets have different sizes. Many practical scheduling algorithms have been proposed to approximate the ideal GPS scheduler, such as Weighted Fair Queueing (WFQ) [5], Self-Clocked Fair Queueing (SCFQ) [7], Worst-case Fair Weighted Fair Queueing (WF^{2}Q) [9], Deficit Round Robin (DRR) [8] and Elastic Round Robin (ERR) [10]. Before describing these practical scheduling algorithms, we first present two measures of fairness based on GPS.

1.3.2 Measures of Fairness

Since GPS is the ideally fair scheduling algorithm which achieves max-min fairness, for any other scheduling policy q approximating GPS, the discrepancy between the service achieved by GPS and that by the practical policy q can be used as a measure of fairness. The

absolute fairness bound (AFB) follows this concept [7]. Consider a flow i which is always backlogged. Denote by $S_i^G(t_1, t_2)$ and $S_i^q(t_1, t_2)$ the service received by flow i during time interval $[t_1, t_2)$ under the GPS policy and under policy q, respectively. The *absolute fairness (AF)* of policy q with respect to flow i during time interval $[t_1, t_2)$ is defined as

$$\mathrm{AF}_i^q(t_1, t_2) = \left| \frac{S_i^G(t_1, t_2)}{w_i} - \frac{S_i^q(t_1, t_2)}{w_i} \right| \tag{1.6}$$

where w_i is the weight of flow i. The absolute fairness bound of policy q is the maximum absolute fairness for all possible flows and all possible time intervals, i.e.,

$$\mathrm{AFB}^q = \max_{\forall i, t_1, t_2} \mathrm{AF}_i^q(t_1, t_2). \tag{1.7}$$

The absolute fairness bound may be difficult to determine for some scheduling algorithms such as SCFQ, since it requires the emulation of GPS. Another measure of fairness, relative fairness bound (RFB), simplifies the computation by comparing the service received by different flows. The relative fairness (RF) of policy q with respect to a pair of flows (i, j) during time interval $[t_1, t_2)$ is defined as

$$\mathrm{RF}_{(i,j)}^q(t_1, t_2) = \left| \frac{S_i^q(t_1, t_2)}{w_i} - \frac{S_j^q(t_1, t_2)}{w_j} \right|. \tag{1.8}$$

Similar to the definition of AFB, the relative fairness bound of policy q is the maximum relative fairness for all possible pairs of flows and all possible time intervals, i.e.,

$$\mathrm{RFB}^q = \max_{\forall i, j, t_1, t_2} \mathrm{RF}_{(i,j)}^q(t_1, t_2). \tag{1.9}$$

We have proved in [20] that for any work-conserving scheduling policy q, the absolute and relative fairness bounds have the following relationship:

$$\frac{1}{2}\,\mathrm{RFB}^q \leq \mathrm{AFB}^q \leq \left(1 - \frac{w_m}{W}\right) \mathrm{RFB}^q.$$

In this relationship, w_m is the smallest weight of all flows, and W is the sum of all weights. In addition, for many scheduling algorithms, the upper bound on the absolute fairness

bound is actually the true bound. Since these two fairness bounds are closely related, we can use one of them if the other is difficult to determine. The fairness bounds of some scheduling policies described below are determined using this relationship, and therefore, the proof of this relationship is presented in Appendix A.

1.3.3 Weighted Fair Queueing

The Weighted Fair Queueing (WFQ) policy approximates the GPS scheduler in the sense that WFQ tends to serve packets in the order of their finishing time under GPS [5]. The system maintains a variable referred as *round number*, indicating the time under GPS by increasing inversely as the number of active flows. Each packet upon arrival is associated with a tag referred as *finish number*. Assuming the k-th packet of flow i, p_i^k, arrives at time τ, its finish number $F(p_i^k)$ is computed as

$$F(p_i^k) = \max\{F(p_i^{k-1}), RN(\tau)\} + \frac{L(p_i^k)}{w_i}$$

where $RN(\tau)$ is the round number at time τ, $L(p_i^k)$ is the size of packet p_i^k, and w_i is the weight of flow i. In other words, the finish number of packet p_i^k depends on the round number at its arrival time if the previous packet from flow i, p_i^{k-1}, has finished service when p_i^k arrives, or on the finish number of p_i^{k-1} otherwise. When a packet finishes service, the WFQ scheduler selects the packet with the smallest finish number in the system as the packet to be served next. It has been shown that the WFQ discipline can lag GPS by a finite constant [21], i.e.,

$$\frac{S_i^G(t,\tau)}{w_i} - \frac{S_i^{\mathrm{WFQ}}(t,\tau)}{w_i} \leq \frac{L}{w_i}$$

where L is the maximum packet size. However it is possible that a flow can receive a substantial service lead under WFQ than under GPS. The WFQ has the following RFB:

$$\mathrm{RFB}^{\mathrm{WFQ}} = 3\frac{L}{w_m}.$$

It is not trivial to accurately track the round number since it is related to the number

of active flows under the emulated GPS system. The round number is updated when a packet finishes service under the WFQ scheduler. By this time, it is possible that some flow has finished service under GPS, and it needs to be removed from the list of active flows. This removal leads to an increase in the incrementing rate of the round number, and thus a larger round number. Again it is possible that by this larger round number, another flow has finished service under GPS and needs to be removed from the list of active flows. This is known as *iterated deletion* [22].

1.3.4 Self-Clocked Fair Queueing

Self-Clocked Fair Queueing (SCFQ) relaxes the WFQ scheduling policy by not strictly emulating the GPS system and not accurately maintaining the round number [7]. Instead, the finish number of the packet currently under service is used as the current round number, i.e.,

$$F(p_i^k) = \max\{F(p_i^{k-1}), CF(\tau)\} + \frac{L(p_i^k)}{w_i}$$

where $CF(\tau)$ is the finish number of the packet being served at time τ. SCFQ has a finite relative fairness as follows:

$$\mathrm{RFB}^{\mathrm{SCFQ}} = 2\frac{L}{w_m}.$$

1.3.5 Worst-case Fair Weighted Fair Queueing

It is shown in [9] that under WFQ, a flow can receive a substantial lead than under GPS. For example, a flow i with a large weight has a sequence of packets awaiting service while the weights of other flows are significantly smaller. In this case, the sequence of packets from flow i may each have a smaller finish number than the head-of-line packet of any other flow, due to the relatively large weight of flow i. Therefore, under WFQ, this entire sequence of packets from flow i may be scheduled before any packet from other flows can begin service, thus leading to significantly bursty service for flow i over short time periods.

Worst-case Fair Weighted Fair Queueing (WF^{2}Q) solves this problem by selecting the packet to be served only from among the *eligible* ones, which means they have begun service under GPS [9]. Specifically, under WF^{2}Q, each packet is associated with not only a finish number but a *start number* as well. The start number of a packet from flow i is either the finish number of the previous packet from flow i, or the round number upon its arrival, whichever is larger, i.e.,

$$S(p_i^k) = \max\{S(p_i^{k-1}), RN(\tau)\}$$

where $S(p_i^k)$ is the start number of the k-th packet from flow i and τ is its arrival time. When the WF^{2}Q scheduler finishes the service of a packet, it selects the packet with the smallest finish number among all the packets that have a start time smaller than the current round number. The absolute fairness of WF^{2}Q is

$$\mathrm{AFB}^{\mathrm{WF^2Q}} = \left(2 - \frac{w_m}{W}\right)\frac{L}{w_m}$$

where W is the sum of the weights of all flows.

1.3.6 Deficit Round Robin

When a packet finishes service, WFQ and all its variants need to search for the packet with the smallest finish number among all flows. Therefore, the per-packet computational complexity of these algorithms is $O(\log N)$ where N is the number of flows. This may impose significant overhead in large scale networks.

Deficit Round Robin (DRR) is a fair scheduling algorithm with a per-packet work complexity of $O(1)$ [8]. For each flow, two variables are maintained in DRR, a *deficit counter* and a *quantum*. The DRR scheduler serves each *backlogged* flow in a round robin fashion. When visiting a flow in each round, the scheduler first increments the deficit counter of this flow by its quantum, and transmits a maximum possible sequence of packets from this flow with the total length less than the deficit counter. The deficit counter is then decremented by the total size of all packets scheduled.

In each round, the service received by each backlogged flow under DRR is, on average, equal to its quantum. Therefore, if the quantum value of a flow is assigned proportional to its weight, the DRR scheduler can allocate to each flow the bandwidth resource with an amount, on average, proportional to its weight. It is shown that the DRR algorithm provides a finite RFB as follows:

$$\mathrm{RFB}^{\mathrm{DRR}} = 3\frac{L}{w_m}.$$

1.4 Allocation of Multiple Resources

As summarized in the previous section, the research on fairness in resource allocation over the last decade or two has primarily focused on the allocation of the bandwidth resource on a link [4–10]. It has also been shown that concepts and algorithms for achieving fairness in the allocation of a single resource can be extended to the case with multiple resources *of the same kind* [23]. However, as previously discussed, bandwidth is only one among several kinds of resources shared by multiple flows in a typical network. As flows of traffic traverse a computer network, they share many different kinds of resources such as link bandwidth, buffer space, time on the router processors and also electrical power, a critical resource in mobile systems. The ultimate goal, therefore, should be the overall fairness in the *joint* allocation of all resources shared by the flows of traffic and not just one specific kind of resource such as the link bandwidth.

The need for fairness in the joint allocation of multiple heterogeneous resources has also been recognized in other contexts. For example, it has been recognized that fair allocation of both the channel bandwidth and the power consumed needs to be achieved simultaneously in mobile networks where power and bandwidth are both critically important and scarce resources [24]. In addition, others have also recognized the importance of joint allocation of buffer and bandwidth resources [25–28].

1.4.1 Prioritized and Essential Resources

Network systems with multiple types of shared resources can be generally categorized into two different groups: those with *prioritized resources* and those with *essential resources*. In this dissertation we investigate the joint allocation in both types of systems, establish fundamental principles for defining fairness in such systems, and propose practical algorithms for realizing these principles. We first describe these types of resources in detail.

In many situations, the flows competing for a set of multiple heterogeneous resources have a preference toward using one of these resources over another. For example, at a switch or a router with a shared buffer and a shared output link, flows prefer to be allocated the output link resource, and only when the output link is not available do they choose to use the buffer resources. Even in the allocation of multiple resources of the same type, a certain resource may be preferred over another such as when, under certain conditions, terrestrial links are preferred over satellite links, for the sake of shorter transmission delay. We refer to the resources in these types of systems as *prioritized resources*, i.e., a shared set of resources of the same or different types, but ordered by preference. In this dissertation, we use systems with shared buffer and link resources as an example to establish the principles of fairness in such systems.

The other type of systems includes those with *essential resources*. We define an *essential* resource as one for which a flow's demand does not reduce with an increase in the allocation of other resources to the flow. A number of resources such as the link bandwidth, processor and power, in most contexts, are essential resources. In this dissertation we investigate a system with a shared processor and a shared link as an example of systems with essential resources.

1.5 Contributions

The primary contributions of this dissertation are general theoretical frameworks for defining and measuring fairness when a set of traffic flows share multiple resources in the network, either prioritized or essential. We make no assumptions on the notion of fairness defined for the allocation of a single shared resource; in fact, our frameworks may be applied to any of several notions of fairness such as max-min fairness, proportional fairness or utility max-min.

With respect to systems with prioritized resources, we introduce two concepts—the *Cumulative Resource Dividend* (CRDIV) of a flow under a certain resource allocation policy represents the benefit accrued to the flow due to the portion of the shared set of resources allocated to it under the policy; the *Cumulative Resource Demand* (CRDEM) of a flow is the benefit accrued to the flow when all of the shared set of resources is exclusively allocated to the flow. These two concepts are generic in the sense that we make no assumptions on what is the shared set of resources and how one may compute the desired benefit to a flow. One may now use one's favorite notion of fairness in the distribution of a single shared resource, and state that a fair allocation policy among a set of competing flows is one that achieves a fair distribution of the cumulative resource dividends with respect to the cumulative resource demands of the flows. This is referred to as the *Principle of Fair Prioritized Resource Allocation* or the *FPRA principle*. However, just as the notions of fairness in the allocation of a single shared resource can be applied only over certain specific intervals of time (intervals during which the number of actively competing flows stays constant) [29], this generalized principle also applies only over certain specific intervals of time depending on the properties of the traffic flows. A significant contribution of this dissertation is the formal definition of these intervals of time in the context of multiple prioritized resources shared by competing flows.

The concepts of cumulative resource dividend and demand cannot be readily extended

to systems with essential resources, since it may not be straightforward to determine the ultimate benefit flows receive from an allocation of multiple non-prioritized resources. Through illustrative examples, we claim that in these systems, at each instant of time, it is the maximum of a flow's normalized demand for the various essential resources that should count in the decisions made by a fair resource allocation algorithm. We then develop the fundamental principles of fairness for systems with multiple essential heterogeneous resources and propose the *Principle of Fair Essential Resource Allocation* or the *FERA principle*, expressed within a rigorous theoretical framework. We also prove that, under certain generic conditions, there exists a unique, fair, and work-conserving resource allocation policy which satisfies the FERA principle.

Given these principles of fairness, we proceed to apply them to a system with a shared buffer and a shared link, and a system with a shared processor and a shared link, both using max-min fairness as the notion of fairness. For the system with the shared buffer and link, we propose an ideally fair policy, called the *Fluid-flow Fair Buffering (FFB)* algorithm, for the joint allocation of buffer and bandwidth resources. We then present the practical *Packet-by-packet Fair Buffering (PFB)*, an approximation of the ideal FFB algorithm. We analytically prove that PFB achieves a close bounded approximation to the FFB algorithm. We use real gateway traffic and video traffic traces to compare the fairness of PFB against several combinations of popular entry and exit policies and demonstrate the improved fairness with PFB. Our results show that the entry policy used in PFB can significantly improve fairness even in combination with an unfair exit policy such as First-Come-First-Served (FCFS). Our results reveal that when buffer resources are constrained, a fair entry policy is more critical than a fair exit policy to the overall fairness goal.

For the system with a shared processor and a shared link, we also propose an ideally fair policy, called the *Fluid-flow Processor and Link Sharing (FPLS)* algorithm, for the joint allocation of processing and bandwidth resources. We then develop a practical and *provably* fair packet-by-packet approximation of the FPLS algorithm, called *Packet-by-*

packet Processor and Link Sharing (PPLS). The PPLS algorithm, based on an extension of the Deficit Round Robin algorithm [8], has a per-packet work complexity of $O(1)$. We illustrate the fairness of the PPLS algorithm using both synthetic traffic and real gateway traffic traces.

Another major contribution is the method proposed in this dissertation for the extension of these principles to the systems with multiple output links. Consider the allocation of a shared buffer in a multiple output link system as an example. The approach used in this extension is to decompose the multiple output link system with H output links into two classes of subsystems. The first subsystem, referred to as the *unshared link subsystem*, consists of H integrated flows, or *sessions*. Each session consists of all the flows headed to a particular output link. In this unshared link subsystem, only the buffer is shared among all sessions, thus allowing the application of the FPRA principle with a common set of shared resources. The decomposition of the system also creates H of the other class of subsystems, referred to as the *shared link subsystems*, each corresponding to one session. Flows within each of these subsystems share both the link and the buffer, and thus each shared link subsystem is identical to the single output link system considered in the study of multiple prioritized resource allocation. After investigating the question of what is fair in the unshared link subsystem, we define the fairness in the entire system based on the definition of fairness in each subsystem. We then present a measure of fairness for multiple output link systems, an extension of the absolute fairness bound in bandwidth scheduling on a link. The analysis of this fairness measure shows that achieving fairness in the unshared link subsystem is critically important to achieving fairness in the overall system. This method of decomposition can also be applied in other context of resource allocation in multiple output link systems such as the processing allocation in such systems.

1.6 Organization

The rest of this dissertation is organized as follows. Chapter 2 describes how to define fairness in systems with prioritized resources, using a system of a shared buffer and a shared link as an example. Chapter 3, on the other hand, investigates fairness in a system with a shared processor and a shared link, an example of systems with essential resources. A detailed discussion on the extension of these results to multiple link systems is presented in Chapter 4. Finally, Chapter 5 concludes this dissertation.

Chapter 2. The Joint Allocation of Buffer and Bandwidth Resources

2.1 Introduction

2.1.1 Motivation

As discussed in Chapter 1, buffers in switches and routers are among the most important kinds of resources shared by flows of traffic in a typical network. The primary purpose of a buffer is to improve bandwidth utilization through a reduced rate of packet loss. In the absence of a buffer, when packets from more than one flow contend for the same output link in a switch, one of the packets succeeds while the others are dropped. At a later time, if no new packets arrive for transmission through the output link, the link is idle and bandwidth is wasted. Using a buffer at the output link, however, packets could be saved instead of dropped in the presence of contention, and then transmitted later when there is no further contention for the output link. On the other hand, the buffer is less preferred by traffic flows as compared to the output link for the same reason. In other words, flows prefer the link resource if the output link is available and only when the output link is congested, do they require the buffer resource. Therefore, a system with a shared buffer and a shared link is one with prioritized resources. In this chapter, we investigate the problem of achieving fairness in the joint allocation of such resources, using the joint allocation of buffer and bandwidth resources as an example.

Even though researchers have primarily focused on the fair allocation of bandwidth resources during the last two decades, some have already recognized the importance of joint allocation of buffer and bandwidth resources [25–28]. Buffer allocation policies in switches and routers are directly related to congestion avoidance and flow control policies with a direct impact on end-user applications. Fair allocation of buffer resources in routers and switches takes on additional significance with the increasing prevalence of multime-

dia applications that use UDP instead of TCP and choose to avoid end-to-end congestion avoidance policies.

This chapter is concerned with achieving fairness in the joint allocation of buffer and bandwidth resources in a network. A management policy for a shared buffer consists of two components. The *entry scheduler* determines which data from which flows are permitted into the buffer and which are not. The entry scheduler is also responsible for pushout, i.e., the discarding of data from the shared buffer in order to accommodate new arriving traffic. The *exit scheduler* dequeues traffic from the shared buffer and transmits them onto the output link. It is the combination of both the entry and the exit schedulers that determines the overall fairness in the allocation of the buffer and bandwidth resources. Since the exit schedulers have already been described in Chapter 1, next we present a brief introduction to popular entry policies.

2.1.2 Buffer Allocation Algorithms

The majority of buffer allocation strategies proposed and analyzed over the last couple of decades have focused on maximizing performance as measured by the throughput achieved or the cell loss rate [30–40]. The simplest allocation policy, still common in many hardware switching devices, is *Complete Sharing (CS)* [30], in which the shared buffer accepts every packet as long as there is space available, and drops them otherwise. Another alternative, typically used when the number of flows is small, is the *Complete Partitioning (CP)* [30, 31] or the *Static Threshold (ST)* policy, which statically partitions the buffer among the multiple flows. It is shown in [30] that the CS strategy achieves better aggregate throughput than the ST strategy. However, the CS strategy is biased in favor of flows with heavy or bursty traffic, since such flows can easily fill up the shared buffer and leave little or no buffer space to flows with light traffic. This can cause an overall degradation in throughput with an unacceptable rate of packet loss for the low-traffic flows. The ST

strategy with equal partitions can protect the performance of low-traffic flows, but only at the expense of potentially wasting buffer capacity.

An improvement in performance can be obtained by combining the properties of CS and ST, such as in *Sharing with Maximum Queue Length (SMXQ), Sharing with Minimum Allocation (SMA)* and *Sharing with Maximum Queue and Minimum Allocation (SMQMA)* [30]. In SMXQ, each flow has a maximum queue length, i.e., a threshold, but unlike ST, the sum of all the thresholds is larger than the entire shared buffer space, in order to improve the utilization of the shared buffer. In SMA, a small portion of the shared buffer is reserved for each flow and the rest is completely shared by all flows. Finally SMQMA combines SMXQ and SMA, by reserving some buffer space for each flow and sharing the rest with a maximum queue length constraint for each flow. Yet another variant is *Buffer Admission Control (BAC)* algorithm [32], which dynamically groups flows into underloaded and overloaded categories, and uses different thresholds to accept packets, based on the flow's current categorization.

Note that none of the above buffer allocation policies implement *pushout*, whereby a packet in the buffer is discarded and its place in the buffer is taken by a newly arrived packet. Pushouts are necessary to correct past unfairness in the allocation, and can be accomplished using the *Virtual Partitioning (VP)* strategy [33, 34]. In this class of buffer sharing algorithms, the buffer is divided into virtual partitions among the flows. When the buffer is not full, all packets are accepted into the buffer as in the CS strategy. However, when the buffer is full, packets belonging to a flow with a buffer occupancy lower than its assigned virtual partition size may push out packets belonging to a flow with a buffer occupancy higher than its assigned virtual partition size. It is proved in [35–37] that VP strategies can achieve optimal performance measured in terms of the packet loss rates.

Note that an equal distribution of the buffer space can be achieved using the VP strategy by dropping packets, whenever necessary, from the longest queue. Another algorithm that seeks to achieve fairness and protection through apportioning the buffer space approx-

imately equally among the flows was proposed in [41, 42]. In this algorithm, an incoming packet is dropped if its flow's queue is greater than a certain minimum length and is also greater than a certain quantity which is a pre-defined function of its own queue length and the average among the queue lengths of all the flows. A review of this algorithm and its variants may be found in [43].

A majority of buffer allocation algorithms have attempted to be fair through apportioning the buffer space approximately equally among all the flows. Note that when all the flows have the same demand and the same weight assigned to them, the max-min fair distribution of the buffer space is the same as an equal distribution of the buffer space. While attempting to apportion the buffer space evenly among all the flows has often been described as fair and is intuitively appealing, this ignores the fact that different flows require different amounts of buffer space at different times depending on the arrival pattern of each flow. Most such buffer allocation strategies determine a flow's share of the buffer space based on the number of active flows or traffic profile parameters, but have no monitoring and feedback mechanisms on traffic arrival patterns. This weakness is somewhat addressed by the *Dynamic Threshold (DT)* strategies, which dynamically assign the threshold values based on the past traffic arrival and buffer occupancy patterns [38–40]. The DT strategies achieve a better utilization of the buffer space, and therefore, a better overall throughput since they can consider the instantaneous needs of each flow. While various DT strategies have been proposed for bursty or Poisson traffic patterns to achieve optimal performance, no allocation strategy has been proposed that seeks to achieve fairness while also considering the traffic patterns and the instantaneous needs of each flow.

In some buffer management algorithms, especially in those intended for use in Internet routers with traffic from TCP sources, a probabilistic determination is made on whether or not to drop a packet [44–50]. These algorithms seek to achieve multiple goals of congestion avoidance, protection and fairness in addition to performance. In *Random Drop (RD)* [44], when the shared buffer is full, one packet in the buffer is randomly selected to be replaced

by the packet that just arrived. *Early Random Drop (ERD)* [44], on the other hand, specifies a fixed *drop probability* for each arriving packet whenever the aggregate length of the queue in the shared buffer exceeds a certain *drop level*. The intent of this algorithm is to punish misbehaving sources based on the assumption that dropped packets are more likely to be from misbehaving sources since they send more packets.

A smoother increase in the drop probability as a function of the queue size is provided by the *Random Early Detection (RED)* algorithm [45] and the *Fuzzy Threshold (FT)* approach [46]. In FT, a dropping probability distribution function is defined. Each possible buffer occupancy is associated with a certain dropping probability of a new arriving packet, and this dropping probability increases as the buffer occupancy until it reaches 1 at some certain threshold. Unlike FT which uses the exact queue lengths, RED monitors the queue lengths and maintains an exponential average value of the queue length. In addition, it maintains two thresholds on the queue length. The RED algorithm specifies that incoming packets be dropped with a certain probability which is a function of the average queue length and the two thresholds. When the average queue length is below the lower threshold, incoming packets are always accepted. When the average queue length is above the upper threshold, incoming packets are always dropped. When the average queue length is between the lower and the upper thresholds, the incoming packets are dropped with a certain probability that is proportional to the average queue length. Instead of dropping packets, the RED algorithm also allows a marking of the packets such as in the DECbit algorithm [51]. In this paper, we focus on the buffer management aspects of these algorithms and therefore, we only consider the version of RED that drops incoming packets.

The fraction of dropped packets belonging to a flow in the RED algorithm has been shown to be roughly proportional to that flow's share of the total bandwidth. The RED algorithm, however, does not achieve or attempt to achieve fairness in terms of equal throughput for all the best-effort flows. The RED algorithm also does not explicitly control misbehaving flows. To address this problem, several enhancements to RED have been

proposed [47–50]. In *Early Fair Drop (EFD)* [49], each flow is associated with a nominal allocated occupancy. The EFD algorithm uses the same method as RED to determine when to drop a packet, but instead of dropping the arriving packet, it searches, in a round-robin fashion, for a flow with a buffer occupancy larger than its nominal value and drops the packet at the head of the queue corresponding to this flow. *CHOKe* [50] randomly selects one packet from the queue when the average queue length is greater than the lower threshold. If the arriving packet and the selected packet are from the same flow, both are dropped. Otherwise, the algorithm's subsequent actions are similar to those in the original RED algorithm. *Fair-Buffering Random Early Detection (FB-RED)* [48], uses the same basic algorithm as RED but weighs the drop probabilities assigned to a packet in a flow by the bandwidth-delay product of the flow. To provide better flow isolation, *Fair Random Early Detection (FRED)* uses per-flow variables [47]. It guarantees a minimum buffer space to each flow, and every packet from a flow is accepted when the queue length of the flow is below this minimum. It also uses a maximum per-flow queue length, above which all packets from the flow are dropped. When the flow's queue length is between these minimum and maximum thresholds, the average queue length of the entire buffer is calculated, and the arriving packet is either dropped or accepted, based on the same algorithm as in the original RED algorithm.

None of these variants of the RED algorithm addresses the issue of the policy to employ when an incoming packet encounters a full buffer; they merely seek to avoid the situation altogether since their primary goal is congestion avoidance rather than fairness in buffer allocation. However, these algorithms do manage buffer space, and one can judge them for how fairly they allow various flows to use and gain benefit from the buffer and link resources in the router.

As summarized above, most of the existing buffer allocation algorithms have attempted to maximize performance or achieve congestion avoidance although several of them have also tried to be fair by one measure or another. A precise and formal notion of fairness in

buffer allocation, however, has not yet been developed. Thus, there is currently no theoretical framework around which one can design practical and fair buffer allocation algorithms, and there also are no formal means of evaluating the various buffer allocation policies already proposed. This chapter seeks to provide such a framework to define fairness in the joint allocation of buffer and bandwidth resources, and to facilitate the design of provably fair buffer management strategies.

2.1.3 Contributions

The primary contribution of this chapter is a general theoretical framework for defining fairness when a set of traffic flows share more than one resource, such as a buffer and a link. The framework developed in this chapter provides a simple but powerful generalization of any of several notions of fairness previously defined for the allocation of a single shared resource or a set of resources viewed as a single entity. In this chapter, we introduce two concepts: the *Cumulative Resource Dividend* (CRDIV) of a flow under a certain resource allocation policy represents the benefit accrued to the flow due to the portion of the shared set of resources allocated to it under the policy; the *Cumulative Resource Demand* (CRDEM) of a flow is the benefit accrued to the flow when all of the shared set of resources is exclusively allocated to the flow. These two concepts are generic in the sense that we make no assumptions on what is the shared set of resources and how one may compute the desired benefit to a flow. One may now use one's favorite notion of fairness in the distribution of a single shared resource, and state that a fair allocation policy among a set of competing flows is one that achieves a fair distribution of the cumulative resource dividends with respect to the cumulative resource demands of the flows. However, just as the notions of fairness in the allocation of a single shared resource can be applied only over certain specific intervals of time (intervals during which the set of flows competing actively does not change), this principle applies only over certain specific intervals of time depend-

ing on the properties of the traffic arrival pattern. The formal definition of these intervals of time, called *stationary intervals*, forms an important component of our contributions in this chapter.

We illustrate the above framework with max-min fairness as the notion of fairness and proceed to define an ideally fair strategy, *Fluid-flow Fair Buffering (FFB)*, for the joint allocation of buffer and bandwidth resources. FFB is an ideally fair but unimplementable resource allocation strategy, just as the Generalized Processor Sharing (GPS) [6] is an ideally fair but unimplementable scheduling discipline for allocating bandwidth among flows sharing a link. FFB is intended to serve research efforts in the design of practical and fair buffer allocation strategies in a manner analogous to the role served by GPS for almost a decade in the design, analysis and measurement of scheduling disciplines for allocating bandwidth on a shared link.

This chapter also presents *Packet-by-packet Fair Buffering (PFB)*, an implementable approximation of the FFB algorithm. We analytically prove that PFB achieves a close bounded approximation to the FFB algorithm. We use real gateway traffic and video traffic traces to compare the fairness of PFB against several combinations of popular entry and exit policies and demonstrate the improved fairness with PFB. Our results show that the entry policy used in PFB can significantly improve fairness even in combination with an unfair exit policy such as First-Come-First-Served (FCFS). Our results reveal that when buffer resources are constrained, a fair entry policy is more critical than a fair exit policy to the overall fairness goal.

2.1.4 Organization

This chapter is organized as follows. In Section 2.2, we introduce the system model considered in this chapter. In Section 2.3, we define the concepts of cumulative resource dividends and demands, and also the concept of stationary intervals of time over which

one can apply notions of fairness in a system with multiple resources. We conclude the section with the statement of the Principle of Fair Prioritized Resource Allocation (the FPRA principle) for use in systems with more than one shared resource. In Section 2.4, we illustrate the application of the FPRA principle and define what is fair in the joint allocation of buffer and bandwidth resources based on the max-min notion of fairness. In this section, we also present the ideally fair but unimplementable FFB strategy. In Section 2.5, we present the PFB strategy, a novel and practical buffer allocation strategy and prove that it closely approximates FFB. In Section 2.6, we present simulation results using real gateway traffic and video traffic to demonstrate the improved fairness of PFB in comparison to popular combinations of buffer and bandwidth management strategies.

2.2 System Model

In the system model considered here, a shared buffer is fed by N flows, labeled as $1, 2, \ldots, N$, all destined to the same output link. Let $R(t)$ be the maximum link speed at time instant t and let $C(t)$ be the capacity of the shared buffer at time instant t. Both values are defined to be functions of time in order to accommodate general situations, such as a higher-level allocation scheme that may change the available capacity of the link or the buffer. We assume that all flows belong to the same service priority class, and w_i is the weight associated with flow i. In reservation-based networks, the reserved rate of a flow may be used as its weight. Fig. 2.1 illustrates our system model and some of the notation used in this chapter.

An *entry scheduler* regulates the entry of traffic from the N flows into the shared buffer. The entry scheduler determines which data from which flows are permitted into the buffer and which are not. The entry scheduler is also responsible for pushout, i.e., the discarding of data from the shared buffer in order to accommodate new arriving traffic from another flow. An *exit scheduler* dequeues traffic from the shared buffer and transmits them onto the

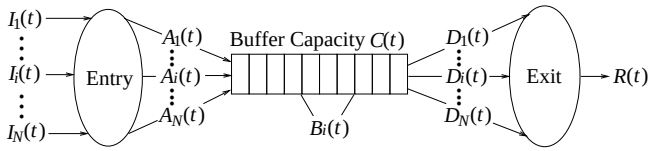

Figure 2.1: The system model.

output link. The exit scheduler, as in scheduling algorithms for the allocation of bandwidth on a link, determines the sequence in which traffic from various flows will exit through the output link.

Let S denote the system under consideration. Let $I_i(t)$ be the rate at which data arrives in flow i at time instant t seeking entry into the shared buffer. This is the only input into the system S. Consider a buffer allocation policy q, a combination of the entry and the exit scheduler's policies. Define the admission rate $A_i^{S,q}(t)$, at time instant t, as the rate at which data from flow i get accepted into the shared buffer of system S under the allocation policy q. Traffic that is not admitted into the shared buffer is dropped. Note that $A_i^{S,q}(t)$ can be negative, such as when the net rate of acceptance into the buffer is negative due to pushouts. $A_i^{S,q}(t) \leq I_i(t)$ holds for all i and t.

Define the departure rate, $D_i^{S,q}(t)$, as the actual rate at which traffic belonging to flow i departs the shared buffer through the output link of system S under the allocation policy q. At time instant t, let $B_i^{S,q}(t)$ be the queue length or the buffer occupancy of flow i in the shared buffer in system S under the allocation policy q. At any given time instant $t \geq t_0$,

$$B_i^{S,q}(t) = B_i^{S,q}(t_0) + \int_{t_0}^{t} \left(A_i^{S,q}(\tau) - D_i^{S,q}(\tau) \right) d\tau. \tag{2.1}$$

Throughout this chapter, the sum of a quantity over *all* flows is denoted by dropping the subscript for the flow in the notation. For example, $I(t)$ is the sum of the input rates of all of the N flows, i.e., $I(t) = \sum_{i=1}^{N} I_i(t)$. $A^{S,q}(t)$, $B^{S,q}(t)$, and $D^{S,q}(t)$ are also defined similarly. Of course, $D^{S,q}(t) \leq R(t)$, and $B^{S,q}(t) \leq C(t)$.

Note that, as mentioned before, the buffer allocation strategy is completely determined by the actions of the entry and the exit schedulers, which together determine $A_i^{S,q}(t)$ and $D_i^{S,q}(t)$. Also note that the queue length of a flow in the shared buffer is completely determined by the admission rate, the departure rate and the initial queue length, as given by (2.1). Defining what is fair in buffer allocation in system S over a certain interval of time $[t_1, t_2)$, therefore, is the same as defining the *conditions* on $A_i^{S,q}(t)$ and $D_i^{S,q}(t)$ for all t in $[t_1, t_2)$, such that q is fair.

2.3 The Principle of Fair Prioritized Resource Allocation

2.3.1 Resource Dividends and Demands

Consider a set of flows using a shared set of prioritized resources in a certain system S. Each flow in the system, depending upon the application that generates the flow, has a certain desired goal, which we generically refer to as the *utility* sought by the flow. Over any given interval of time, the *cumulative utility* is merely the utility considered over that interval of time. Note that the definitions of utility and cumulative utility may be very different in different contexts. For example, in the scheduling of bandwidth over a single shared link as accomplished by fair scheduling algorithms such as DRR [8], the utility may be defined as the bandwidth achieved by a flow; the cumulative utility achieved by a flow over an interval of time would be defined as the amount of its data transmitted through the shared link during the interval. For real-time applications with guaranteed delay requirements, one may define the cumulative utility over an interval as the fraction of packets that are successfully delivered within the specified guaranteed delay over this interval of time.

It is important to note that, in this chapter, we do not impose any particular notion of how cumulative utility over an interval should be defined. Our only assumption in this regard is that the cumulative utility over any interval achieved by a flow is always non-

negative and does not decrease with an increase in the amount of any resource allocated to it.

Consider a policy q for the allocation of the shared set of resources. Over time interval $[t_1, t_2)$, denote by $U_i^{S,q}(t_1, t_2)$ the cumulative utility achieved by flow i under allocation policy q in system S. Consider an allocation policy, $None(i)$, which grants none of the shared resources to flow i. By our notation, $U_i^{S,None(i)}(t_1, t_2)$ is the cumulative utility achieved by flow i during time interval $[_1, t_2)$ with the allocation policy $None(i)$. The difference in the cumulative utilities achieved by a flow with and without the use of the allocated portion of the shared set of resources, i.e., the difference between $U_i^{S,q}(t_1, t_2)$ and $U_i^{S,None(i)}(t_1, t_2)$, represents the benefit accrued to the flow due to this shared set of resources. The following formally defines this concept.

Definition 1 The *Cumulative Resource Dividend*, denoted by $\text{CRDIV}_i^{S,q}(t_1, t_2)$, of flow i in system S under the allocation policy q over an interval of time $[t_1, t_2)$ is defined as follows:

$$\text{CRDIV}_i^{S,q}(t_1, t_2) = U_i^{S,q}(t_1, t_2) - U_i^{S,None(i)}(t_1, t_2). \tag{2.2}$$

Now, a notion of fairness in the allocation of the shared resources should specify a distribution of these cumulative resource dividends among the flows. However, such a notion of fairness cannot be developed without also defining a notion of the demands placed on the shared set of resources by the flows. For example, it is only sensible that flows which have no need for the shared set of resources, i.e., with no demand for them, should not unnecessarily be allocated any of these resources. This principle is a trivial generalization of already existing notions of fairness in the allocation of a single resource.

The demand of a flow for the shared set of resources can be expressed in terms of the benefit or the cumulative resource dividend that the flow desires from an allocation of the shared set of resources. Any flow would like a biased allocation policy that grants all of the shared set of resources exclusively to it. Therefore, the demand of a flow is really the

benefit accrued to the flow, i.e., the cumulative resource dividend of the flow, when all of the shared set of resources is allocated exclusively to the flow. Let $All(i)$ be an allocation policy that allocates all of the shared resources, in entirety and exclusively, to flow i. The notion of the demand of a flow can now be formally defined as follows.

Definition 2 The *Cumulative Resource Demand*, denoted by $\text{CRDEM}_i^S(t_1, t_2)$, of flow i in system S over an interval of time $[t_1, t_2)$ is defined as follows:

$$\text{CRDEM}_i^S(t_1, t_2) = U_i^{S,All(i)}(t_1, t_2) - U_i^{S,None(i)}(t_1, t_2). \tag{2.3}$$

Note that the cumulative resource demand is independent of the allocation policy q. Note also that the cumulative resource demand of a flow is no less than the cumulative resource dividend of the flow under any allocation policy.

In scheduling of bandwidth over a single shared link, a flow gets no throughput at all with policy $None(i)$ since the link is the only resource contributing to the utility. Thus, over any time interval, all of the bandwidth allocated to a flow represents the benefit accrued to the flow from the shared resource. In this case, the cumulative resource dividend of a flow over a given interval of time with a scheduling policy is the same as the total amount of data from the flow scheduled for transmission by the policy during this interval. Similarly, the cumulative resource demand of a flow over a certain interval of time is just the total amount of data that the flow could transmit during the interval if it did not have to compete with any other flow.

2.3.2 The FPRA Principle

Based on the definition of the cumulative resource demand and the cumulative resource dividend over any given interval of time, the shared resources can now be allocated according to any given notion of fairness $\mathcal{F}$ applied to the cumulative resource dividends with respect to the cumulative resource demands. This would ensure that each flow receives, as per the notion of fairness $\mathcal{F}$, a fair share of the dividend from the shared set of resources.

However, one cannot apply such a notion of fairness over any *arbitrary* interval of time, and this significantly hinders a simple extension of the notion of fairness from the single-resource case to that for a system with a link and a buffer resource. For example, a notion of fairness such as the principle of max-min fairness also cannot be applied to any arbitrary interval of time in the allocation of bandwidth on a link among competing flows. In this case, a flow is considered *active* at any given instant of time if and only if it is backlogged [8]; and active over a given interval of time if and only if it is active at each instant of time during this interval. The principle of max-min fairness may only be applied over intervals of time during which no flow changes its state from being active to not being active, or vice-versa. In our study, we refer to such an interval of time over which one can apply a notion of fairness as a *stationary* interval. In extending a notion of fairness to the system model S discussed in Section 2.2, we will have to extend the concept of the active/inactive state of a flow and the concept of a stationary interval.

Consider a system with two distinct resources, one of which is the preferred resource. In the system under consideration, the link is the preferred resource; flows use the buffer resource only if the link resource is not available for immediate use. Our framework for the definition of fairness in the system under consideration is based on a simple common-sense and therefore, axiomatic approach whereby we allocate the preferred resource fairly, and then allocate the non-preferred resource fairly among the flows with unsatisfied demands. As per any given notion of fairness, a fair allocation in a system with two distinct resources is one which, firstly, fairly allocates the preferred resource among all the competing flows and then, fairly allocates the other resource among the flows that still have unsatisfied demands.

Denote by S^{-}, an identical system as the one under consideration, but without the buffer resource. This is a system with just a single shared output link. We assume that, as per any given notion of fairness $\mathcal{F}$, the ideally fair allocation strategy is known for system S^{-}. Based on the earlier discussion, only when the demand of a flow in system

S^- cannot be satisfied does it compete with other flows for the buffer resource. In other words, a flow should be considered in competition for the buffer resource, if and only if, in the absence of the buffer resource, the flow is not satisfied with a fair allocation of the link resource. Therefore, an active flow with respect to the buffer resource is one whose demand, in system S^-, would not be met under the ideally fair allocation policy, $G_{\mathcal{F}}(S^-)$. The following definitions formalize this thought.

Definition 3 With respect to the buffer resource, a flow i is *active* during an interval of time $[t_1, t_2)$ as per the notion of fairness $\mathcal{F}$, if and only if, over *each* subinterval of time $[\tau_1, \tau_2)$ such that $t_1 \leq \tau_1 < \tau_2 \leq t_2$, the cumulative resource demand of flow i in system S is greater than the cumulative resource dividend it would achieve in system S^- under the ideally fair allocation policy, $G_{\mathcal{F}}(S^-)$. In other words, flow i is active with respect to the buffer resource over time interval $[t_1, t_2)$, if and only if,

$$\text{CRDEM}_i^S(\tau_1, \tau_2) > \text{CRDIV}_i^{S^-, G_{\mathcal{F}}(S^-)}(\tau_1, \tau_2)$$

for all time intervals $[\tau_1, \tau_2)$ such that $t_1 \leq \tau_1 < \tau_2 \leq t_2$.

Definition 4 With respect to the buffer resource, a flow i is *inactive* during an interval of time $[t_1, t_2)$ as per the notion of fairness $\mathcal{F}$, if and only if, over *each* subinterval of time $[\tau_1, \tau_2)$ such that $t_1 \leq \tau_1 < \tau_2 \leq t_2$, the cumulative resource demand of flow i in system S is equal to the cumulative resource dividend it would achieve in system S^- under the ideally fair allocation policy, $G_{\mathcal{F}}(S^-)$. In other words, flow i is active with respect to the buffer resource over a time interval $[t_1, t_2)$, if and only if,

$$\text{CRDEM}_i^S(\tau_1, \tau_2) = \text{CRDIV}_i^{S^-, G_{\mathcal{F}}(S^-)}(\tau_1, \tau_2)$$

for all time intervals $[\tau_1, \tau_2)$ such that $t_1 \leq \tau_1 < \tau_2 \leq t_2$.

Note that it is possible that a flow is neither active nor inactive with respect to the buffer resource over a certain interval of time, since the above definitions are based on

conditions that require to be satisfied in *each* subinterval of time within the given interval. For example, consider two contiguous intervals of time. In the first interval, assume that a certain flow is active with respect to the buffer resource while in the second interval the flow is inactive with respect to it. Then, in the combined interval of time consisting of both the above two intervals, the flow is neither active nor inactive with respect to the buffer resource.

Thus, during any given interval, a flow may be said to be in one of three states with respect to the buffer resource: active, inactive or neither. In our case of a system with more than one resource, if a flow does not need the less preferred resource, then it implies that the flow is satisfied and is not in active competition with other flows. Generalizing the concept used in the allocation of a single resource, one may define fairness with respect to a resource over an interval only when the set of flows competing for the resource stays constant during the interval. We are now ready to present the concept of a stationary interval in our system, and the Generalized Principle of Fairness.

Definition 5 In a system S with two distinct sets of resources, one of which is the preferred resource set, a certain interval of time is a *stationary interval*, if and only if, each flow is either active or inactive (but not neither) with respect to the non-preferred resource set over this entire interval.

Now we are ready to present the *Principle of Fair Prioritized Resource Allocation* or the FPRA principle as follows.

Principle 1 *Principle of Fair Prioritized Resource Allocation.* Consider a system S with two sets of prioritized resources and an allocation policy q. Policy q is fair as per a notion of fairness $\mathcal{F}$, if and only if, over all stationary intervals of time, the cumulative resource dividends achieved by the flows are distributed fairly, as per the notion of fairness $\mathcal{F}$, with respect to the cumulative resource demands requested by the flows.

Note that, if a flow i is neither active nor inactive over a certain time interval $[t_1, t_2)$,

this interval can be divided into a contiguous sequence of subintervals, during each of which flow i is either active or inactive. Thus, even though the FPRA principle defines fairness only over stationary intervals, any given interval of time may be broken down into a sequence of contiguous stationary intervals. Thus, the FPRA principle may be used to define a fair allocation over any given interval.

2.4 Application to Buffer-Link System Model

2.4.1 What is Fair?

In this section, we illustrate the application of the FPRA principle to the system model described in Section 2.2 under specific notions of fairness and the cumulative utility achieved by a flow. We will use max-min fairness as the notion of fairness. Throughout the rest of this chapter, we also use the total amount of data from a flow transmitted over the output link during any given interval of time as the cumulative utility achieved by the flow over this interval.

Thus, for any allocation policy q, the cumulative utility of a flow i in system S over any interval of time is given by

$$U_i^{S,q}(t_1, t_2) = \int_{t_1}^{t_2} D_i^{S,q}(\tau) d\tau. \tag{2.4}$$

Consider the allocation policy $None(i)$. In the absence of this set of shared resources (both the link and the buffer), the cumulative utility is obviously 0. With an allocation policy q, therefore, the cumulative resource dividend over an interval for each flow is exactly the cumulative utility achieved by the flow over the interval. The cumulative resource demand of flow i is the cumulative utility it gets using the allocation policy $All(i)$, which allocates the entire buffer and the output link exclusively to this flow. Thus, applying (2.4) into (2.2) and (2.3), we have,

$$\text{CRDIV}_i^{S,q}(t_1, t_2) \quad = \quad \int_{t_1}^{t_2} D_i^{S,q}(\tau) d\tau \tag{2.5}$$

$$\text{CRDEM}_i^S(t_1, t_2) \;=\; \int_{t_1}^{t_2} D_i^{S,All(i)}(\tau)d\tau \tag{2.6}$$

for any flow i and any allocation policy q.

Recall that to define the state of a flow as active or inactive with respect to the buffer resource, we need to consider the system without the buffer resource S^-. Note that the ideally fair allocation policy in system S^- defined earlier, given the max-min notion of fairness, is GPS [6]. Now, a flow i is said to be active with respect to the buffer resource, or simply active, over an interval of time $[t_1, t_2)$ if and only if, over each subinterval $[\tau_1, \tau_2)$ such that $t_1 \leq \tau_1 < \tau_2 \leq t_2$,

$$\int_{\tau_1}^{\tau_2} D_i^{S,All(i)}(\tau)d\tau > \int_{\tau_1}^{\tau_2} D_i^{S^-,GPS}(\tau)d\tau. \tag{2.7}$$

Similarly, a flow i is said to be inactive with respect to the buffer resource, or simply inactive, over an interval of time $[t_1, t_2)$ if and only if, during each subinterval $[\tau_1, \tau_2)$,

$$\int_{\tau_1}^{\tau_2} D_i^{S,All(i)}(\tau)d\tau = \int_{\tau_1}^{\tau_2} D_i^{S^-,GPS}(\tau)d\tau. \tag{2.8}$$

A stationary interval is one during which each flow is either active or inactive, and an allocation policy q is fair if and only if, over all stationary intervals $[t_1, t_2)$,

$$\left[\text{CRDIV}_i^{S,q}(t_1, t_2)\right] = \mathcal{F}_{\text{MMF}}\left(\text{CRDIV}^{S,q}(t_1, t_2), \left[\text{CRDEM}_i^S(t_1, t_2)\right], [w_i]\right) \tag{2.9}$$

where $\mathcal{F}_{\text{MMF}}$ represents the policy of max-min fairness.

2.4.2 An Ideally Fair Allocation Strategy

Based on the framework developed above and the notion of max-min fairness, we now discuss *Fluid-flow Fair Buffering (FFB)*, an ideally fair work-conserving strategy for the joint allocation of buffer and bandwidth resources. As in the ideally fair GPS scheduler, the FFB algorithm also assumes that traffic can be divided into infinitesimally small quantities and is schedulable at this granularity. With fluid flow traffic, a protocol where traffic may

be allowed to bypass the buffer if the buffer is empty is equivalent to one in which traffic has to always pass through the buffer. This is because traffic that is forced to pass through the buffer even though the buffer may be empty spends only an infinitesimal amount of time in the buffer in such a hypothetical system.

Recall that a buffer allocation policy contains two parts: an entry policy and an exit policy. It can be readily verified that the FFB has to use GPS as the exit policy in order for it to achieve max-min fairness. This is because FFB is the fair algorithm for the joint allocation of buffer and bandwidth resources, and therefore, it still needs to provide fairness in bandwidth allocation when buffer allocation is not an issue (such as if the buffer is of infinite capacity). With an infinite buffer, FFB is fair if and only if its exit policy is fair, implying that its exit should be GPS. We now proceed to discuss the entry policy in FFB.

In the FFB algorithm, we maintain for each flow i an *acceptance counter* (AC_i) which indicates the amount of data accepted into the shared buffer from flow i. When a packet from flow i is accepted into the buffer, AC_i is incremented by the size of the packet; when some data from flow i are pushed out from the shared buffer, AC_i is decremented by the amount of data pushed out. Note that AC_i is not decremented when some data from flow i exit the buffer and get transmitted through the shared output link; otherwise, AC_i would be just the buffer occupancy of flow i. Therefore, it is possible that a flow has a large value of the acceptance counter in comparison to other flows even while its buffer occupancy is relatively low.

Denote by $AC_i(t)$ the value of the acceptance counter of flow i at time instant t. If at time instant τ, the input rates for all flows become 0, then $AC_i(\tau)$ indicates the total amount of data transmitted from flow i after all of its data in the buffer at time τ are also transmitted. Thus, the acceptance counter of a flow represents its potential cumulative dividend and therefore, represents the quantity that the entry policy should attempt to be fair about in order to achieve fair distribution of the cumulative dividends with respect to the demands. By the max-min fair notion of fairness, therefore, the FFB strategy should ensure

that the acceptance counters of all flows conform to the weighted max-min fair allocation with respect to the demands.

In summary, the ideally fair FFB algorithm as per the max-min notion of fairness uses the GPS server as the exit policy and ensures a weighted max-min distribution of the acceptance counters as the entry policy during each stationary interval of time.

2.5 Packet-by-packet Fair Buffering

It is obvious that the FFB scheduler that assumes fluid flow behavior is not implementable with real traffic that is packetized. In this section, we present *Packet-by-packet Fair Buffering (PFB)*, a practical and implementable approximation to the ideally fair FFB scheduler.

2.5.1 The PFB Algorithm

The pseudo-code of the PFB algorithm is presented in Figs. 2.2 and 2.3. The PFB algorithm maintains a linked list, called *FlowList*, which consists of all the flows with packets waiting in the shared buffer. When a flow has no packets waiting in the shared buffer, it is removed from the *FlowList* (accomplished by lines 22–24 and 50–52) and other flows are not affected.

In the *Dequeue* procedure, an implementable fair scheduling algorithm such as SCFQ [7], SPFQ [52] or DRR [8] may be used that can achieve long-term fairness with a bounded value of the relative fairness bound as defined in [7, 15].

The *Enqueue* procedure (lines 3–16) is invoked whenever a packet arrives at an input port of the shared buffer. Assume that a packet p from flow i arrives. If flow i does not already exist in *FlowList*, it is appended to the tail of the *FlowList* and its normalized acceptance counter is set to the current minimum of the normalized acceptance counters of the active flows (lines 6–10). This is similar to the idea of the potential function in [52]

```
1   Initialize: /* Invoked when the system starts */
2     FlowList ← NULL;

3   Enqueue: /* Invoked whenever a packet arrives */
4     p ← ArrivingPacket;
5     i ← Flow(p); /* Flow of packet p */
6     if (ExistsInFlowList(i) = FALSE) then
7       Find flow k with minimum AC_j/w_j, ∀j ∈ FlowList;
8       AC_i ← w_i AC_k/w_k;
9       Append flow i to FlowList;
10    end if;
11    if (EmptySpaceInBuffer ≥ Size(p)) then
12      Accept p into buffer;
13      AC_i ← AC_i + Size(p);
14    else
15      Pushout(p); /* Pushout packets to accommodate p */
16    end if;

17  Dequeue: /* Always running */
18    Use any real approximation of GPS, except that
19      after each packet p is transmitted, invoke Transmit(p)

20  Transmit(p): /* Invoked whenever a packet departs */
21    i ← Flow(p); /* Flow of packet p */
22    if (QueueIsEmpty(i) = TRUE) then
23      Remove flow i from FlowList;
24    end if;
```

Figure 2.2: Pseudo-code of Packet-by-packet Fair Buffering.

where a newly backlogged flow has an initial potential equal to the system potential, which is no more than the minimum of the potentials of all existing flows.

The algorithm subsequently checks if there is enough empty space in the shared buffer to accommodate packet p (lines 11–16). If so, packet p is accepted into the shared buffer, and the acceptance counter of flow i is incremented by the size of packet p. If there does not exist enough buffer space for packet p, the *Pushout* procedure shown in Fig. 2.3 is invoked

```
25  Pushout(p):
26     i ← Flow(p); /* Flow of packet p */
27     SpaceNeeded ← Size(p) − EmptySpaceInBuffer;
28     Accepted ← FALSE;
29     while(Accepted ≠ TRUE)
30        Among all unmarked flows, find flow k with
31           the largest AC_j/w_j, ∀j ∈ FlowList;
32        if (AC_k/w_k = AC_i/w_i) then
33           Discard p;
34           Unmark all marked flows and data;
35           return;
36        end if;
37        if (Occupancy(k) < SpaceNeeded) then
38           Mark flow k and all data from flow k;
39           SpaceNeeded ← SpaceNeeded − Occupancy(k);
40        else
41           Mark flow k and at least SpaceNeeded worth
42              of data from flow k;
43           Accepted ← TRUE;
44        end if;
45     end while;
46     for each marked flow j
47        Unmark flow j;
48        Pushout all marked data from flow j;
49        Decrement AC_j by the amount of data pushed out;
50        if (QueueIsEmpty(j) = TRUE) then
51           Remove flow j from FlowList;
52        end if;
53     end for;
54     Accept p into buffer;
55     AC_i ← AC_i + Size(p);
```

Figure 2.3: Pseudo-code of the *Pushout* procedure in PFB algorithm.

to push out some packets from other flows to accommodate packet p.

In order to achieve a max-min distribution of the acceptance counters that is as close as possible to the ideal, based on the sizes of the packets involved, one may need to push out packets from more than one flow. In our approximation of the ideally fair algorithm,

however, for the sake of computational efficiency, we select exactly *one* flow and attempt to pushout as much data from this flow as necessary to accommodate the newly arrived packet. Pushout from another flow occurs only when the flow that is selected first does not have sufficient data that can be pushed out to accommodate the arriving packet.

When the *Pushout* procedure is invoked, it first finds a flow, say k, with the largest value of the normalized acceptance counter. If the normalized values of the acceptance counters are maintained in a sorted linked list, this can be done in $O(\log N)$ time with respect to the number of flows. If flows i and k have the same value of the normalized acceptance counter, i.e., $AC_i/w_i = AC_k/w_k$, packet p is discarded (lines 32–36). Otherwise, the *Pushout* procedure attempts to push out some data from flow k to accommodate packet p. If flow k has enough data, the *Pushout* procedure returns after packet p is accepted, and AC_i and AC_k are appropriately updated (lines 41–43). When the buffer occupancy of flow k is not large enough, all data from flow k are pushed out, and the *Pushout* procedure will continue with the next flow with the largest value of the normalized acceptance counter. Note that in PFB, data to be pushed out are only marked first (lines 38 and 41), and the true pushout is executed only if packet p can be accepted (lines 46–53), thus making PFB a work-conserving algorithm.

2.5.2 Fairness Analysis

With packetized traffic, as opposed to fluid-flow traffic, short-term fairness may be degraded but the algorithm will still achieve long-term fairness. The following theorem proves an upper bound on the lag of the PFB algorithm with respect to the ideally fair FFB algorithm.

Theorem 1 Consider a certain stationary interval $[t_1, t_2)$. Consider two identical systems with the same initial conditions and the same input traffic sequence, except that one uses the FFB entry policy and the other uses the PFB entry policy. For any flow i and for any

time instant $t \in [t_1, t_2)$, we have

$$AC_i^F(t) - AC_i^P(t) \leq M$$

where $AC_i^F(t)$ and $AC_i^P(t)$ are the acceptance counters corresponding to FFB and PFB respectively, and M is the maximum packet size.

Proof: Note that since $[t_1, t_2)$ is a stationary interval, no flow changes state during this interval, and therefore, during $[t_1, t_2)$ no flow becomes empty in the shared buffer due to pushout.

Note that a difference in the actions by the entry policies of PFB and FFB occurs only when the buffer is full. Therefore, we only need to consider the situation when a buffered packet is pushed out or an arriving packet is discarded. Consider a certain flow i. For the sake of convenience, denote by $\mathcal{T}^F(t)$ the set of flows with the largest normalized acceptance counter at time instant t under FFB, i.e., the set with the largest value of $AC_j^F(t)/w_j, \forall j$. The set $\mathcal{T}^P(t)$ is similarly defined for the PFB algorithm.

Assume time instant t_0 is the first time that $AC_i^F(t)$ and $AC_i^P(t)$ differ from each other, i.e., $AC_i^F(t) = AC_i^P(t)$, $\forall t < t_0$. Let t_0^+ be the instant of time that the execution of the pushout or the discard completes in response to an event at time t_0 (assume negligible length of time to complete such an execution). It can be verified that $AC_i^F(t_0^+)$ becomes larger than $AC_i^P(t_0^+)$ in only one of the two following situations:

- A packet p from flow i arrives at time t_0. At this instant, there exists a certain amount of space available in the shared buffer but it is not large enough to accommodate all of p. If flow i belongs to both $\mathcal{T}^F(t_0^-)$ and $\mathcal{T}^P(t_0^-)$, part of the packet p is accepted under FFB, while under PFB, the entire packet p is discarded. In this case, $0 \leq AC_i^F(t_0^+) - AC_i^P(t_0^+) \leq M$. Note that after FFB accepts part of packet p, flow i belongs to $\mathcal{T}^F(t_0^+)$.

- At time instant t_0, a packet p from a flow other than i arrives and the buffer does not have enough space. Assume both $\mathcal{T}^F(t_0^-)$ and $\mathcal{T}^P(t_0^-)$ contain more than one flow,

and flow i belongs to both sets. Thus, the amount pushed out from flow i under FFB is less than the size of packet p, since data from multiple flows are pushed out. In the case of PFB, however, only one flow is selected for pushout, and it is possible that flow i is chosen. In this case, $0 \leq AC_i^F(t_0^+) - AC^P(t_0^+) \leq M$. Note that, again we have $i \in \mathcal{T}^F(t_0^+)$, while in the case of PFB, there is another flow with a larger normalized acceptance counter, i.e., $i \notin \mathcal{T}^P(t_0^+)$.

Note that in both the above situations, we have $i \in \mathcal{T}^F(t_0^+)$. Next, we proceed to show that, for any time instant τ within a stationary interval, if flow i belongs to $\mathcal{T}^F(\tau^-)$,

$$AC_i^F(\tau^+) - AC_i^P(\tau^+) \leq AC_i^F(\tau^-) - AC_i^P(\tau^-). \tag{2.10}$$

It is sufficient to consider only the time instants when acceptance counters change, i.e., when new packets arrive.

Assume a packet p arrives at time τ and the flow under consideration $i \in \mathcal{T}^F(\tau^-)$.

- If packet p is from flow i, it will be discarded under FFB, and thus $AC_i^F(\tau^+) = AC_i^F(\tau^-)$. On the other hand, under PFB, packet p will either be accepted or discarded, i.e., the acceptance counter of flow i will not decrease. Therefore, (2.10) is satisfied.

- If packet p comes from another flow other than i, some data from flow i will be pushed out under FFB, since $i \in \mathcal{T}^F(\tau^+)$. While in PFB, $AC_i^P(\tau^+) = AC_i^P(\tau^-)$, since as mentioned above, $i \notin \mathcal{T}^P(\tau^+)$. Again, (2.10) is satisfied.

Note that in both cases, it is always true that $i \in \mathcal{T}^F(\tau^+)$. Therefore, by induction, we may conclude that once the difference $AC_i^F(t) - AC_i^P(t)$ becomes greater than 0, it only decreases with increasing time at least until it becomes negative. In addition, as shown above, when $AC_i^F(t) - AC_i^P(t)$ becomes positive, its maximum possible value is M, the size of the largest packet. Therefore, for any time instant $t \in [t_1, t_2)$, $AC_i^F(t) - AC_i^P(t) \leq M$, thus bounding the difference between the practical and the ideally fair schedulers. ■

2.5.3 Computational Efficiency

Theorem 2 The *Enqueue* procedure in PFB can be implemented with a computational complexity of $O(\log N)$, where N is the number of flows in the system.

Proof: Note that PFB can maintain the *FlowList* using a heap, based on the normalized value of the acceptance counter. To maintain such a heap, the work complexity is $O(\log N)$. In addition, to accommodate a packet, at most $\lceil M/m \rceil$ packets need to be marked and pushed out, where M and m are the maximum and minimum packet sizes, respectively. Therefore, the **while** loop (lines 29–45) and the **for** loop (lines 46–53) will be executed, in the worst case, $\lceil M/m \rceil$ times.

Furthermore, when marking a flow k (line 38), we can remove flow k from the *FlowList*, and insert it into another linked list, *MarkedFlowList*. This takes $O(\log N)$ of time since deletion from a heap takes $O(\log N)$ and insertion into a linked list takes $O(1)$. In addition, while traversing the *MarkedFlowList* as in the **for** loop (lines 46–53), unmarking a flow k can be implemented by removing flow k from the *MarkedFlowList* and inserting it back into the *FlowList*. Again, this takes $O(\log N)$ of time.

The complexity of the *Pushout* procedure, therefore, is $O(\lceil M/m \rceil \log N)$ or simply, $O(\log N)$. ■

Note that the computational complexity of the *Dequeue* procedure is simply the complexity of the fair scheduler used to implement the exit policy. For example, if DRR [8] is used, the per-packet dequeuing complexity will be $O(1)$ with respect to the number of flows.

2.6 Measure of Fairness and Simulation Results

In this section, we present a measure of fairness in the joint allocation of buffer and bandwidth resources, and using this measure we compare the fairness of PFB against some representative entry and exit policies using real gateway traffic traces and video traffic

traces.

2.6.1 Measure of Fairness

In measuring fairness in the joint allocation of buffer and bandwidth resources, we extend the basic premise of the Absolute Fairness Bound (AFB) defined in the context of allocating bandwidth on a link [15, 20]. The AFB captures the upper bound on the difference between the normalized service received by a flow under the policy being measured and that received by the same flow under the ideally fair policy.

Let $G_{\mathcal{F}}(S, q)$ be an ideally fair buffer allocation policy for system S due to the notion of fairness $\mathcal{F}$, such that its total cumulative utility is identical to that of q, i.e.,

$$\int_{t_1}^{t_2} D^{S,q}(\tau)d\tau = \int_{t_1}^{t_2} D^{S,G_{\mathcal{F}}(S,q)}(\tau)d\tau.$$

Note that in our study of fairness in buffer allocation, we make no assumption about whether or not the allocation policy being measured is work-conserving with respect to the shared set of resources. Therefore, a normalizing quantity based on performance is necessary in extending the notion of the fairness measure to our case of buffer allocation. This normalization should allow us to use our fairness measure in a valid comparison between various buffer allocation strategies. We now define the *normalized Absolute Fairness Measure* over an interval of time as follows:

Definition 6 In a system S with a shared buffer, a shared output link and a given input traffic arrival pattern, the *normalized Absolute Fairness Measure*, $\mathrm{nAFM}^{S,q}(t_1, t_2)$, of an allocation policy q over an interval of time $[t_1, t_2)$ is defined as follows:

$$\mathrm{nAFM}^{S,q}(t_1, t_2) = \frac{\max\limits_{\forall i} \left| \frac{\int_{t_1}^{t_2} D_i^{S,q}(\tau)d\tau}{w_i} - \frac{\int_{t_1}^{t_2} D_i^{S,G_{\mathcal{F}}(S,q)}(\tau)d\tau}{w_i} \right|}{\int_{t_1}^{t_2} D^{S,q}(\tau)d\tau}. \tag{2.11}$$

Note that the above measure depends on the input traffic arrival pattern, and therefore, an algorithm will naturally have different upper bounds, nAFB, for different input traffic

patterns. Also note that the fairness measured as above will approach 1.0 with any real algorithm when the size of the time interval, $t_2 - t_1$, is extremely small. At the same time, for most real buffer allocation strategies, the fairness measured as above will approach 0 when the size of the time interval considered is very large. Thus, a valid comparison between various allocation algorithms can be made using the above measure only if the sizes of the time intervals being considered are identical. Therefore, a meaningful measure of the fairness for a given input pattern is not a single number but a function of τ, the size of the time interval over which fairness is measured. In our simulation study, we use the observed maximum of $\mathrm{nAFM}^{S,q}(t, t+\tau)$ over all t to indicate the fairness of the allocation policy q for each interval of size τ.

2.6.2 Simulation Setup

Our simulation model consists of a shared buffer fed by 8 input traffic sources. Traffic from these 8 flows is headed to the same shared output link via the shared buffer. In our first set of simulation experiments, we use real gateway traffic traces. In our second set of simulation experiments, we use video traffic traces.

In our study, we have implemented five different entry policies including the PFB entry policy and three different exit policies, as summarized in Tables 2.1 and 2.2. Note that the PFB algorithm for buffer and bandwidth management uses the PFB entry policy and a fair packet scheduler such as DRR as the exit policy.

The entry policies we simulate are chosen to be representative and include the following: (i) *Drop From Longest Queue (DFLQ)*, which pushes out packets belonging to the flow with the longest queue in the buffer whenever the shared buffer is full, and accepts all packets, otherwise; (ii) *Static Threshold (ST)*, which assigns an equal fixed buffer occupancy threshold to each flow and no flow is allowed to occupy more than this threshold; (iii) *Random Early Detection (RED)* [45], which drops arriving packets with a probability

Table 2.1: Entry policies evaluated.

Policy	Description
DFLQ	Pushes out packets from the flow with the longest queue.
ST	Assigns each flow with an equal queue occupancy threshold.
RED	Drops arriving packets with a probability [45].
FB-RED	A variant of RED [48].
PFB	The fair entry policy proposed in this dissertation.

that is a dynamic function of the average buffer occupancy; (iv) *Fair Buffering Random Early Detection (FB-RED)* [48], which is a variant of RED that uses the bandwidth-delay product of a flow to determine the probability with which a packet from the flow is dropped; and finally, (v) the PFB entry policy. These five entry policies can be categorized into two groups: one including DFLQ, ST, and PFB, and the other including RED and FB-RED. This is because both RED and FB-RED are intended to be congestion avoidance algorithms, and therefore, assumed to work in situations where the shared buffer is never full (packets are dropped before the buffer gets full). In our simulation studies, all parameters of the RED algorithm follow the recommendations in [53].

Three exit policies are also implemented: (i) *First-Come First-Served (FCFS)*, which dequeues packets in the order of their arrival times; (ii) *Longest Queue First (LQF)*, which schedules packets from the flow with the longest queue in the shared buffer; and (iii) *Deficit Round-Robin (DRR)* [8], a simple and popular fair round-robin scheduler. In our implementation, the DRR quantum is set to be equal to the maximum packet size. In the scheduling of bandwidth over a link, both FCFS and LQF have an absolute fairness bound of infinity, i.e., both are unfair given the max-min fair notion of fairness. DRR, on the other hand, is

Table 2.2: Exit policies evaluated.

Policy	Description
FCFS	Dequeues packets in the order of arrival.
LQF	Schedules packets from the flow with the longest queue.
DRR	A fair round-robin scheduler [8].

the representative fair algorithm used here.

2.6.3 Gateway Traffic Traces

In this study, we use real traffic recorded at Internet gateways as the input traffic [54][2]. Fig. 2.4 plots the observed maximum value of $\mathrm{nAFM}^{S,q}(t, t+\tau)$ against τ for different pairs of entry and exit scheduling policies. Specifically, Fig. 2.4(a) plots the observed maximum value of $\mathrm{nAFM}^{S,q}(t, t+\tau)$ against τ when the entry scheduling policy is RED and FB-RED, while Fig. 2.4(b) plots the same for DFLQ, ST, and PFB entry policies. From Fig. 2.4(b) it is seen that among all examined combinations of entry and exit policies, five have a fairness measure approaching zero as τ increases. These five combinations that are able to provide long-term fairness are ST with DRR, DFLQ with DRR and all three combinations with PFB. To better illustrate the differences between these five combinations, a logarithmic plot is also presented in Fig. 2.4(c).

From Fig. 2.4(b), it is readily observed that the combinations with DRR as the exit scheduler are better in terms of fairness than those without. The fact that DRR is already

[2]The traces are obtained from the Passive Measurement and Analysis project at the National Laboratory for Applied Network Research (NLANR).

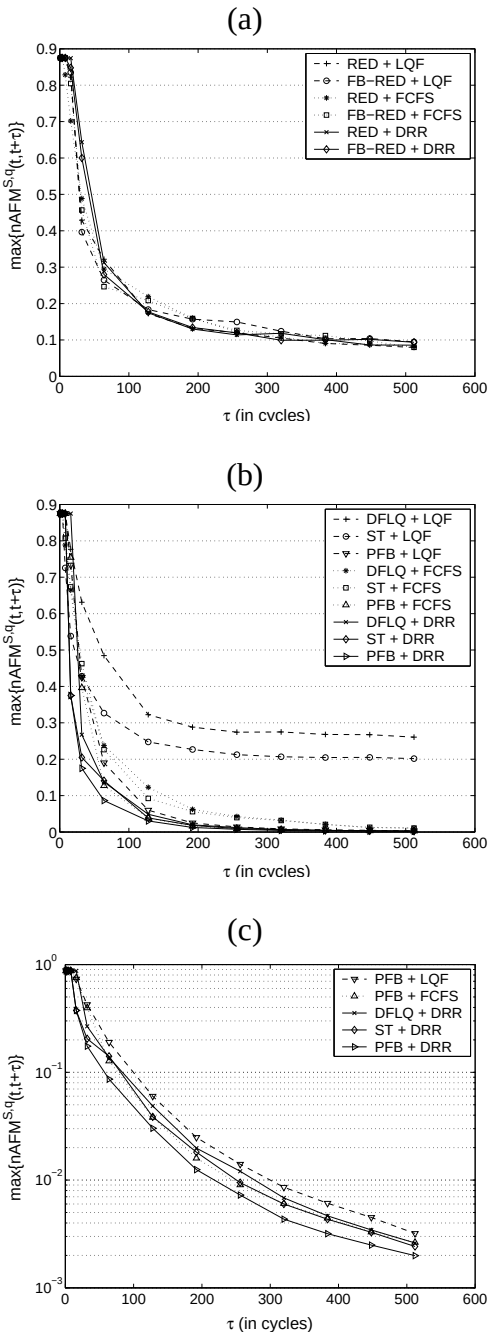

Figure 2.4: Observed maximum (over all t) of $\mathrm{nAFM}^{S,q}(t, t+\tau)$ vs. τ, with input traffic from a gateway trace: (a) the entry policy is RED or FB-RED, and the exit policy is LQF, FCFS, or DRR; (b) the entry policy is DFLQ, ST, or PFB, and the exit policy is LQF, FCFS, or DRR; (c) the logarithmic plot of several selected combinations.

fair with respect to bandwidth allocation on a link certainly helps in improving the overall fairness achieved with DRR as exit policy. However, having DRR as the exit policy does not by itself guarantee good fairness as is apparent in the case of RED and FB-RED. This is explained as follows. With PFB, DFLQ, or ST as an entry policy, whenever a flow is active, the shared buffer is guaranteed to contain packets belonging to this flow. Now, DRR serves all backlogged flows in a fair manner, and the situation becomes a close approximation to the case of scheduling bandwidth on a link. Therefore, DRR appears promising as an exit scheduler when an entry scheduler such as PFB, DFLQ, or ST is used. If, however, the entry scheduler is such that each active flow does not necessarily have packets waiting in the shared buffer (such as in the case of RED and FB-RED), using DRR as the exit scheduler is insufficient to guarantee good fairness, as shown exactly in Fig. 2.4(a). For example, with RED or FB-RED, an active flow may frequently end up without any data in the shared buffer when the size of the average queue length in the buffer is above the maximum threshold and all arriving packets are dropped.

Note that, as shown in Fig 2.4(a), buffer allocation strategies with RED and FB-RED as the entry policy fail to achieve fairness, while those with PFB succeed, shown in Fig 2.4(b) and (c). One interesting observation is that, when PFB is used in combination with unfair exit schedulers such as LQF and FCFS, the fairness achieved is actually very close to that with DRR as the exit scheduler. FCFS and LQF have traditionally been understood to be unfair policies, since under both policies bursty flows can easily end up being rewarded at the expense of more steady flows. In the traditional scheduling of bandwidth over a shared link, both FCFS and LQF are considered to have an absolute fairness bound of infinity. However, this is strictly true only if the size of each burst that is accepted into the shared buffer is unlimited. The PFB entry scheduling policy guarantees to accept balanced amount of traffic from each flow into the shared buffer, thus leading to a similar level of overall fairness independent of the fairness characteristics of the exit scheduler.

In addition, a study of Figs. 2.4(a) reveals that there is very little difference between

RED and FB-RED in terms of fairness achieved. This tends to be true independent of the exit policy used. The RED algorithm was primarily proposed to avoid congestion in as fair a manner as possible, and not as a buffer management strategy. The unfairness in buffer management with the RED strategy arises since it does not distinguish between different flows and all flows will lose some bandwidth when one flow dominates the queue. A fair solution would have required that only the flow that is dominating the network resources should lose bandwidth. The FB-RED algorithm refines RED in the sense that, to differentiate between flows, it uses a per-flow quantity called *maxProbability*, which is determined by the bandwidth-delay product. This modification, however, does not help fairness in buffer allocation since the average queue length is still calculated for the aggregated set of flows sharing the buffer. For example, consider the case when one flow experiences a sudden burst and transmits a large amount of data, causing the average queue length to increase. After the average queue length is larger than *maxThreshold*, every packet is dropped, no matter which flow it belongs to. This continues until the average queue length falls back within the range between *minThreshold* and *maxThreshold*. Thus, the throughput achieved by a flow with a small demand can be adversely affected by another flow.

In summary, a fair entry policy in combination with a highly unfair exit policy leads to acceptable overall fairness; however, an unfair entry policy even with a fair exit policy cannot guarantee overall fairness. This conclusion that emerges from our simulation study is significant in that it stresses the importance of the entry policy as opposed to the exit policy when buffer resources are constrained, even though the exit policy has received almost all of the attention in the research literature.

2.6.4 Video Traffic Traces

Multimedia traffic is emerging as the dominant component of Internet traffic and such traffic typically does not use end-to-end congestion control. Therefore, it is important to

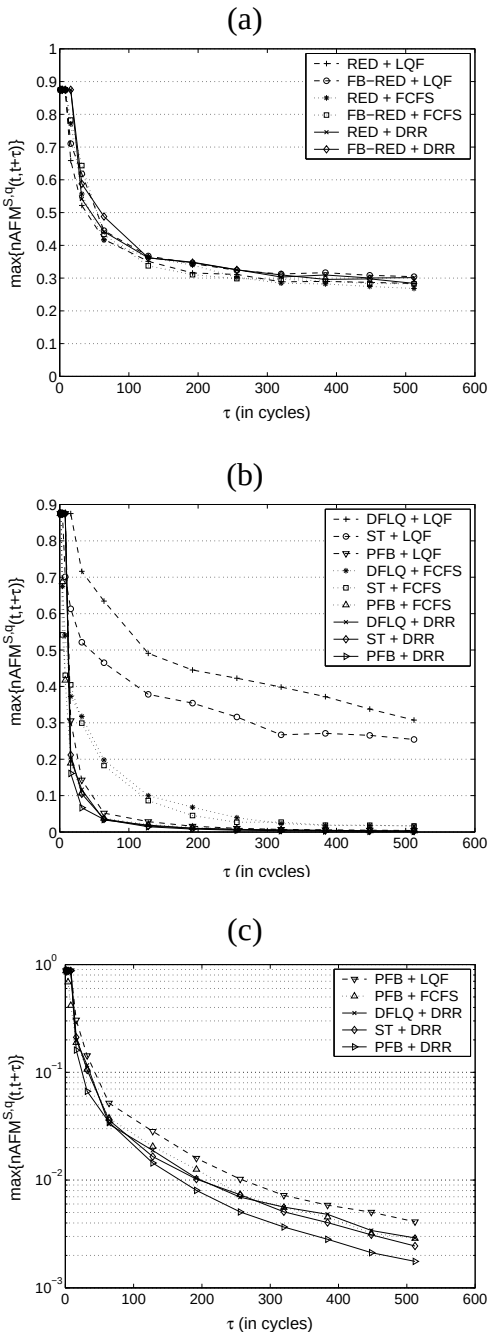

Figure 2.5: Observed maximum (over all t) of $\mathrm{nAFM}^{S,q}(t, t+\tau)$ vs. τ, with input traffic from video traces: (a) the entry policy is RED or FB-RED, and the exit policy is LQF, FCFS, or DRR; (b) the entry policy is DFLQ, ST, or PFB, and the exit policy is LQF, FCFS, or DRR; (c) the logarithmic plot of several selected combinations.

understand the fairness achieved by the various combinations of allocation policies with multimedia input traffic. We use video traces coded using MPEG-4 with high quality [55][3]. For each input, one distinct video stream is used, and the starting point within the video stream is randomly selected.

The results obtained and the conclusions drawn with the video traces are similar to those obtained with gateway traffic traces. Fig. 2.5 shows these results.

[3]The traces are collected from the Telecommunication Networks Group at Technical University of Berlin, Germany. The eight video streams selected are *Jurassic Park I*, *Silence of the Lambs*, *Star Wars IV*, *Mr. Bean*, *Star Trek*, *Die Hard III*, *Aladdin*, and *The Firm*. The categories covered are diverse, including drama, action and animation.

Chapter 3. The Joint Allocation of Processing and Bandwidth Resources

3.1 Introduction

3.1.1 Background and Motivation

As discussed in Section 1.2, to define fairness in the allocation of a single resource, various formal notions of fairness [5, 6, 14–16, 18] have been proposed in the literature. Based on these notions of fairness—most commonly, based on the notion of max-min fairness—much research over the last decade or two has focused on the allocation of the bandwidth resource on a link [4–10].

The significance of considering the fair allocation of more than just the link bandwidth is increasingly becoming apparent today, since the link bandwidth is often not the only critical resource. With the current pervasiveness of optical networking in the Internet backbone, and with the occasional trend toward using over-provisioning as the solution to congestion in the edge networks, a router's processor is often also a critical resource to which, ideally speaking, all competing flows should have fair access. If the network is not fair in allocating processing resources, denial of service attacks based on an excessive use of the router processor (such as by using unnecessary optional headers) become possible.

Given the fact that processing requirements of different packets vary widely, the issue of fairness in the allocation of the processing resources gains significance. In addition, besides the fact that packet lengths can vary widely, the presence of optional headers and the various kinds of control information carried by packets create a wide variation in the ratio of a packet's demand for bandwidth and its demand for processing cycles. Thus, packets of the same length cannot be guaranteed to have similar requirements for the processing resources on a router. In fact, the processing delay plotted as a function of the packet length shows that the processing requirements of packets vary across a wide range even for

packets of the same length [56].

Thus, one cannot achieve overall fairness merely with the fair allocation of link bandwidth alone, or merely through the fair allocation of the processing resources alone, since different flows—and different packets within the same flow—may have very different demands for these two kinds of resources. All of this begs the question that this chapter seeks to address: how does one achieve fairness in the *joint* allocation of the processing and bandwidth resources?

The need for fairness in the joint allocation of multiple heterogeneous resources has also been recognized in other contexts besides the one discussed here. For example, it has been recognized that fair allocation of both the channel bandwidth and the power consumed needs to be achieved simultaneously in mobile networks where power and bandwidth are both critically important and scarce resources [24]. However, a rigorous theoretical framework that may be universally employed as a guide in the design of practical algorithmic strategies for the joint allocation of such heterogeneous sets of resources does not exist.

In this chapter, we investigate the issue of fairness in such systems and develop a general principle that forms the foundation for the design of practical fair strategies for use in routers. We also present an evaluation of the practical strategies proposed in this chapter using both synthetic and real gateway traces.

3.1.2 Essential Resources

In the joint allocation of the processor and bandwidth resources, if a certain resource is never the bottleneck, then the fair allocation strategy degenerates to the fair allocation of just the other resource. For example, if the available bandwidth is large enough that no flow experiences congestion due to lack of bandwidth alone, one only needs to worry about the allocation of the processing resources. Fair allocation of a single bottleneck resource has been studied extensively in the literature and has led to a large number of practical

algorithms that are in use today in Internet routers, operating systems, and transport-level protocols. This chapter, on the other hand, answers the question of what is a fair allocation when more than one resource is congested and extends the notions of fairness applied to a single resource to systems with multiple heterogeneous resources.

We define an *essential* resource as one for which a flow's demand does not reduce with an increase in the allocation of other resources to the flow. A number of resources such as the link bandwidth, processor or power, in most contexts, are essential resources. On the other hand, buffer resources in a network are often non-essential resources; for example, in a system with a buffer and a link, a flow uses the buffer only if the link resource is currently unavailable to it, and thus a flow's demand for the buffer resource reduces as more of the link bandwidth is allocated to it. In the system model used in this chapter, we assume that the flows are in competition for resources that are all essential.

We define a pair of resources as *related* to each other if a flow's demand for one resource uniquely determines its demand for the other resource. Resources in a set are said to be *related* if each resource is related to every other resource in the set. Resources in real scenarios are almost always related since the demands of a flow for different individual resources are often related to each other. For example, since each packet is associated with certain processing and bandwidth requirements, a specific increase in a flow's demand for link bandwidth is typically associated with a specific increase in its demand for processing resources. A simpler example, involving multiple resources of the same kind, is a tandem network with multiple links where the demand of a flow for bandwidth is the same on all the links. In the system model used in this chapter, we assume multiple resources that are related, although we make no assumptions on the specific nature of the relationship between a flow's demand for different resources. The existence of a relationship between the demands of a flow for various resources calls for the *joint* allocation of these resources, as opposed to an independent and separate allocation of the resources.

3.1.3 Difference from Prioritized Resource Allocation

One may expect that the concepts of resource dividends and demands established in Chapter 2 for allocation of multiple prioritized resources can be readily applied in systems with multiple essential resources, and the fairness in such systems can be similarly defined. Unfortunately, this is not the case. In systems with multiple prioritized resources as discussed in Chapter 2, the resources can be ordered based on the preference of competing flows. In other words, the benefits different flows achieve by using the shared resources can be readily determined and directly compared. Therefore, for each flow it is straightforward to compute the utility of an allocation policy, and to define the concepts of resource dividends and demands. This feature, however, is no longer valid in systems with multiple essential resources. For example, in a system with a shared processor and a shared link, it is not straightforward to determine the benefit a flow receives by utilizing a portion of processing and bandwidth resources. It is also not easy to compare the benefit of a flow with a large amount of processing resource but a small amount of bandwidth resource, and that of another flow with the a small amount of processing resource but a large amount of bandwidth resource. Therefore, the premise in the framework for allocation of prioritized resources cannot be directly used in the case of multiple essential resources.

On the other hand, the allocation of essential resources also gives rise to some advantages over the allocation of prioritized resources. Recall that, to define fairness in the allocation of prioritized resources, we have to formally define the concept of stationary intervals, during which the notion of fairness can be applied, due to the state of each flow being active or inactive with respect to the resources. In systems with resources that are essential and related, any given flow will be always active with respect to any given resource. Otherwise the given flow will have no demand for the given resource, which implies that this resource is either not essential to this flow or not related to other resources, or both. Therefore, in this study, we do not need to worry about the validity of a time interval within

which the fairness principle can be applied.

3.1.4 Contributions

The primary contribution of this chapter is a theoretical framework based on which one can define fairness in the joint allocation of multiple heterogeneous resources that are essential and related. Similar to the framework in the joint allocation of multiple prioritized resources presented in Chapter 2, we make no assumptions on the notion of fairness; in fact, this framework may also be applied to any of several notions of fairness such as max-min fairness, proportional fairness or utility max-min. Through illustrative examples, we claim that, at each instant of time, it is the maximum of a flow's normalized demand for the various resources that should count in the decisions made by a fair resource allocation algorithm. We then develop the fundamental principles of fairness for systems with multiple essential and related heterogeneous resources and propose the *Principle of Fair Essential Resource Allocation* or the *FERA principle*, expressed within a rigorous theoretical framework. We also prove that, under certain conditions, there exists a unique, fair, and work-conserving resource allocation policy which satisfies the FERA principle.

Given the FERA principle, we proceed to apply it to a system with a shared processor and a shared link, using max-min fairness as the notion of fairness. We propose an ideally fair policy, called the *Fluid-flow Processor and Link Sharing (FPLS)* algorithm, for the joint allocation of processing and bandwidth resources. We then develop a practical and *provably* fair packet-by-packet approximation of the FPLS algorithm, called *Packet-by-packet Processor and Link Sharing (PPLS)*. The PPLS algorithm, based on an extension of the Deficit Round Robin algorithm [8], has a per-packet work complexity of $O(1)$. We illustrate the fairness of the PPLS algorithm using both synthetic traffic and real gateway traffic traces.

Even though this chapter primarily focuses on the joint allocation of the processing and

bandwidth resources, the FERA principle may be readily applied to a variety of contexts beyond those discussed in this chapter.

3.1.5 Organization

The rest of this chapter is organized as follows. Section 3.2 describes the general system model with multiple shared resources considered in this study, along with our notation. Section 3.3 presents the Principle of Fair Essential Resource Allocation, or the FERA principle, for the system model under consideration. Section 3.4 applies the FERA principle to a system with a shared processor and a shared link, and proposes a practical and fair scheduling algorithm for the joint allocation of the processing and bandwidth resources, called the Packet-by-packet Processor and Link Sharing (PPLS) policy. The fairness properties of the PPLS strategy are demonstrated by simulation experiments using real gateway traffic in Section 3.5. Finally, Section 3.6 concludes the chapter with further discussions on implementation issues of the PPLS algorithm.

3.2 System Model

In our system model, a set of N flows, $1 \leq i \leq N$, compete for a set of K related and essential resources, $1 \leq j \leq K$, as shown in Fig. 3.1. As also described in Section 3.1.2, we define an essential resource as one for which a flow's demand does not reduce with an increase in the allocation of other resources to it. Since a buffer is often not an essential resource, our assumption that flows only compete for essential resources implies that if there are buffers in the network shared by the flows, these buffers are of infinite capacity so that the flows never compete for the buffer resource. In developing our fundamental principles of fairness, we make no assumptions on the specific actions of the scheduler or the specific order in which the packets use the K resources.

Note that in this general model, we also make no assumptions on the internal architec-

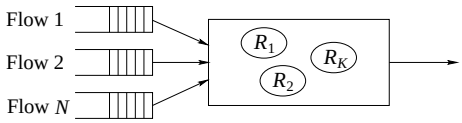

Figure 3.1: A general system model.

ture of the set of shared resources. It can be a simple sequence of resources such as in a tandem network with multiple links, a parallel structure such as the resources of electric power and bandwidth in a wireless sensor network, or a more complex hybrid.

Denote by R_j the peak rate at which resource j may be consumed. For example, in the case of a link resource L, R_L is the peak bandwidth available on the link. As before, denote by w_i the weight of flow i. Let $d_{i,j}$ be the consumption rate demanded by flow i for the shared resource j. Our assumption of related resources implies that, given $d_{i,k}$, one can determine $d_{i,j}$ for all $j \neq k$. Denote by $a^q_{i,j}$, the consumption rate of the shared resource j allocated to flow i under the allocation policy q.

3.3 The Principle of Fair Essential Resource Allocation

3.3.1 Notion of Fairness

As mentioned in Section 1.2.4, the normalized notion of fairness as in (1.5) is used in this chapter, unlike in Chapter 2 where (1.4) is used as the notion of fairness. This is because in a system with multiple essential resources, for each flow, the demands for different resources are typically in different dimensions, and so be the allocations. Therefore, normalization of the demands and allocations is needed in the notion of fairness used. In addition, in the presence of multiple essential resources, it is not straightforward to compute a unique resource dividend for each flow, based on which the benefits achieved by different flows can be readily compared, as in the case of prioritized resources shown in Chapter 2.

Note that, in the research literature, notions of fairness have not been defined for multiple heterogeneous resources[4]. We use the notation in (1.5) that specifies a notion of fairness for a single resource and extend the notion to multiple heterogeneous resources in subsequent sections.

3.3.2 The Concept of the Prime Resource

As mentioned above, the normalization of the demands and the allocations is needed in this study. Therefore, we begin with a few preliminary definitions.

Definition 7 Define the normalized demand of flow i for resource j, $\tilde{d}_{i,j}$, as follows:

$$\tilde{d}_{i,j} = \frac{d_{i,j}}{R_j}.$$

Define the *largest normalized demand* of flow i, $\mathcal{D}_i$, as the maximum amongst the normalized demands of flow i for all resources. That is,

$$\mathcal{D}_i = \max_j \tilde{d}_{i,j}.$$

Definition 8 Define $\tilde{a}^q_{i,j}$ as the normalized allocation of resource j to flow i under allocation policy q, i.e.,

$$\tilde{a}^q_{i,j} = \frac{a^q_{i,j}}{R_j}.$$

Definition 9 The *largest normalized allocation* of a flow i under allocation policy q, denoted by $\mathcal{A}^q_i$, is defined as the maximum amongst the normalized allocations to flow i of all resources. That is,

$$\mathcal{A}^q_i = \max_j \tilde{a}^q_{i,j}.$$

---

[4]Some notions of fairness such as max-min fairness and proportional fairness can be defined for multiple resources of the same kind (e.g., a network of links), under the assumption that, if a flow receives allocations of several resources, the allocations it receives of these resources are identical [14, 18]. However, it is not straightforward to extend these notions of fairness to systems with multiple heterogeneous resources. On the other hand, it can be readily verified that our framework is the same as these notions of fairness if the shared resources are of the same kind.

Definition 10 Under an allocation policy q, a resource is said to be a *prime resource* of flow i, denoted by $\mathcal{B}_i^q$, if and only if, the normalized allocation of this resource to flow i is its largest normalized allocation. In other words,

$$\mathcal{B}_i^q = \arg\max_j \tilde{a}_{i,j}^q = \arg\max_j \frac{a_{i,j}^q}{R_j}$$

where $\arg\max_x f(x)$ indicates the value of the argument x corresponding to the maximum value of function $f(x)$. In other words, we have

$$\tilde{a}_{i,\mathcal{B}_i^q}^q = \max_j \tilde{a}_{i,j}^q = \mathcal{A}_i^q.$$

In networking terminology, a bottleneck resource is one that is the most congested resource. It is critical to note that neither the resource for which a flow has the largest normalized demand nor its prime resource under an allocation policy is necessarily the same as the bottleneck resource in the system. In addition, a flow may have more than one prime resource. The prime resource is defined based on the actual allocations and not on the demand of the flows for the resources.

Note that for each flow, the extent to which the demand for resource share gets satisfied can be used as an indication of the extent to which the overall demand of this flow gets satisfied. In addition, the largest normalized allocation of a flow in fact represents the extent to which the demand of this flow for resource share gets satisfied. Therefore, we claim that the fairness of an allocation policy q is determined by the largest normalized demands, the prime resources under this policy, and the associated normalized allocations, which is further explored in the following.

3.3.3 The FERA Principle

We introduce our principle with a few illustrative examples shown in Table 3.1. In these examples, two flows with equal weights, labeled as 1 and 2, share two different resources: a processor P for packet processing, and a link L. The system model in these examples

Table 3.1: Examples illustrating what is a fair allocation in a system with a shared processor P and a shared link L. In all of these examples, the total amounts of the shared resources are, respectively, 100 MHz for P and 100 Mbps for L.

	Flow	Demand		Allocation	
	ID	P (MHz)	L (Mbps)	P (MHz)	L (Mbps)
A	1	75	25	75	25
	2	25	75	25	75
B	1	225	75	75	25
	2	50	150	25	75
C	1	100	20	50	10
	2	100	10	50	5

is the same as the one we will discuss later in Fig. 3.2. The peak processing rate is 100 million processor cycles per second, i.e., 100 MHz, and the peak link rate is 100 Mbps. Let us assume linear utility functions and max-min as the notion of fairness. In addition, for the sake of convenience, we also assume in these examples a proportional relationship between a flow's demands for these resources and therefore, a proportional relationship between the allocations. In other words, the ratio of a flow's demand for one resource and its demand for another resource is always a constant.

In example A, assume that packets in flow 1 are all small, and therefore, its demand for bandwidth is small relative to its demand for processing time. In contrast, assume that packets in flow 2 are large, and therefore, its demand for bandwidth is large relative to its demand for processing time. To better illustrate the concept, we exaggerate the difference between their demands as follows: Flow 1 has a demand of 75 MHz for processing time and 25 Mbps for bandwidth, while flow 2's demands are, respectively, 25 MHz and 75 Mbps.

If a work-conserving allocation policy is used, there is enough of both resources to satisfy the demands of both the flows and so the allocations are exactly the same as the demands for each of the resources. Note that for flow 1, the prime resource is P, while for flow 2, it is L.

Next, consider what happens when both flows proportionally increase their demands for both resources. In example B, in comparison to example A, flow 1 increases its demands by a factor of three while flow 2 doubles its demands. Specifically, the demands for flow 1 become 225 MHz for P and 75 Mbps for L, while those for flow 2 become 50 MHz and 150 Mbps, respectively. A fundamental principle behind the max-min notion of fairness is that, given no additional resources, a flow should not be able to increase its allocation by merely demanding more. Thus, the fair allocation should be as shown in example B. Again, the prime resource for either flow remains the same as in the previous example.

We discuss example B further. Obviously, in this case, neither flow is satisfied by the allocated resources. Is the allocation actually fair?

One might argue that both flows should get equal bandwidth from a fair allocation, since ultimately both flows will depart from this system and the processor is only an intermediate resource before the flow's packets reach the link resource. Denote by x the bandwidth rate allocated to each flow, in units of Mbps. Since the resources P and L are related for these flows, we know that the allocations of the processor resource are $3x$ for flow 1, and $x/3$ for flow 2, both in units of MHz. Note that the allocations have to be feasible, and therefore, we can compute the allocations as follows:

$$\begin{cases} 3x + x/3 & \leq & 100 \\ 2x & \leq & 100. \end{cases}$$

It can be readily verified that, under a work-conserving allocation policy, flow 1 gets 90 MHz of processing time and flow 2 gets only 10 MHz, while both flows get 30 Mbps of bandwidth. While this allocation underutilizes the link resources, that is not an argument against its fairness. The unfairness arises from the fact that it unnecessarily favors the flow

whose prime resource is the "intermediate resource", which turns out to be flow 1 in this case. Another argument against this notion is that, even though it is true that the processor in this case is positioned ahead of the link, it does not necessarily mean that the processing resource becomes less important, or less preferred, as compared to the link, which is positioned as the "final" resource.

Another allocation philosophy may be to allocate resources based on a fair allocation of the most congested resource as measured by the sum of the normalized demands for the resource. In this example, the processing resource P is the most congested resource. Denote by y as the processing resources allocated to either flow, in units of MHz. Again, to make the allocation feasible, one may allocate resource P fairly as follows:

$$\begin{cases} 2y & \leq & 100 \\ y/3 + 3y & \leq & 100. \end{cases}$$

Under a work-conserving allocation policy, flow 2 gets 90 Mbps of bandwidth and flow 1 gets only 10 Mbps, while both flows get 30 MHz of processing resources. Note that this allocation philosophy has a similar weakness as the one based on the fair allocation of the link resource. It unnecessarily favors the flow whose largest normalized demand is not for the most congested resource. A flow can trigger a change in the allocation policy by merely increasing its demand for a given resource, while that would actually be against the max-min notion of fairness.

Note that the previous analysis also implies that the fairness in the allocation of both processing and bandwidth resources should not be defined in terms of the fairness in the allocation of any single resource. This is because both resources are essential, and therefore, no resource should be treated as if it is more important than the other.

One may suggest the following slight modification to the allocation strategy: to fairly allocate the most congested resource as measured by the sum of the normalized *allocations* for the resource. However, it can be shown that such an allocation may not exist. Assume a certain resource r is the most congested resource. Let α denote the flow with the smaller

demand for resource r and let β denote the other flow. Assume that the normalized allocations of resource r are z_α and z_β for the two flows. It can be verified that the normalized allocations of the other resource are $3z_\alpha$ and $z_\beta/3$, independent of whether the resource r is the processing resource P or the bandwidth resource L. Since resource r is the most congested resource as measured by the sum of the normalized allocations, we have

$$3z_\alpha + z_\beta/3 \leq z_\alpha + z_\beta$$

which leads to $3z_\alpha \leq z_\beta$. Since both flows have a high demand, under the max-min notion, this condition cannot lead to a fair allocation except for the trivial case where $z_\alpha = z_\beta = 0$. Thus, it may not be possible to achieve a fair allocation of the most congested resource as measured by the sum of the normalized allocations of the resource.

Based on the discussions above, we claim that in a network where no explicit preference of one resource over another exists (i.e., each resource is essential), fairness should not be defined based only on a single resource, no matter how this single resource is determined and whether it is determined before allocation (i.e., based on demand) or after allocation (i.e., based on allocation). Instead, the fairness in such a system should be defined with overall consideration of various resources involved in the system and the relationships between the demands for the various resources.

Given this observation, one may propose yet another scheme to define fairness: the sum of the normalized allocations of the resources computed for each flow should be max-min fair. In the previous example B, this leads to an allocation of 75 MHz of processing time and 25 Mbps of bandwidth for flow 1, and 25 MHz of processing time and 75 Mbps of bandwidth for flow 2. In this case, for both flows, the sum of the normalized allocations of the two resources is $75/100 + 25/100 = 1$. While this appears to be a reasonable strategy for fair allocation, this scheme of fairness cannot, in fact, be extended to other situations. This is illustrated by example C described below.

Assume that both flows have a demand of 100 MHz for resource P, while the demands

for resource L are 20 Mbps and 10 Mbps for flows 1 and 2, respectively. Note that in this example, there is sufficient link bandwidth available for the demands of both flows, i.e., the flows are not in competition for resource L. In other words, the system regresses into an allocation of a single resource P. Applying the max-min notion of fairness on the single resource P, we know that the fair allocation would be 50 MHz of processing time for each flow, leading to 10 Mbps and 5 Mbps of bandwidth for flows 1 and 2, respectively. Thus, the ideally fair allocation leads to 0.6 and 0.55 as the sum of the normalized allocations. Clearly, if we were to be max-min fair in the sum of the normalized allocations of the resources to each flow, we would not get this result. This illustrates that the strategy of achieving max-min fair distribution in the sum of the normalized allocations fails to serve as the basis to define fairness in the allocation of multiple resources.

The fair allocation strategies in the three examples do have one property in common: the largest normalized allocations of the flows are distributed in a max-min fair manner among the flows. In our case with equal weights for the flows, the largest normalized allocations are equal for the two flows. In the first two examples in Table 3.1, resource P is the prime resource for flow 1, while the prime resource for flow 2 is resource L. In both examples, the largest normalized allocation equals 0.9. In the third example, the processor P is the prime resource for both flows, and this time the largest normalized allocation is 0.5 for both flows.

The observations from the above examples lead to the significance of incorporating the largest normalized allocation for each flow into a strategy for extending a notion of fairness to the allocation of multiple resources. In our examples, the fair allocation policy is to simply equalize the largest normalized allocations for different flows. In more general situations, different notions of fairness may be used and flows may have different weights and different largest normalized demands. We now present the *Principle of Fair Essential Resource Allocation* or the *FERA principle*.

Principle 2 *Principle of Fair Essential Resource Allocation.* In a system with multiple related and essential resources, an allocation policy q is said to be fair as per the notion of fairness $\mathcal{F}$, if and only if, the largest normalized allocations are distributed fairly, as per the notion of fairness $\mathcal{F}$, with respect to the largest normalized demands. In other words, allocation policy q is fair as per $\mathcal{F}$ if and only if,

$$[\mathcal{A}_i^q] = \mathcal{F}\left(C, [\mathcal{D}_i], [w_i]\right)$$

where C is some constraint imposed on the system.

3.3.4 Fair Work-Conserving Allocation Policy

Recall that we make no assumption on whether or not the allocation policy is work-conserving, and under different constraints, a single system can have more than one fair allocation policy as per the same normalized notion of fairness. Given a constraint, however, there exists a unique work-conserving fair allocation policy in most situations, as will be proved in this section.

First, we formally define a work-conserving policy in the allocation of multiple resources. Recall that in allocation of a single resource, an allocation policy is work-conserving if and only if one of the following two situations occurs.

1. All flows' demands are satisfied.

2. The shared resource is completely allocated.

In other words, no more of the resource can be further allocated to the flows. The same idea to allocation of multiple resources, except that now it is possible that only one resource is fully utilized.

Definition 11 In the allocation of multiple resources, an allocation policy is said to be *work-conserving*, if and only if, upon completion of the allocation, no more of any re-

source can be further allocated to a flow without also reducing the amount of some resource allocated to another flow.

Next we introduce two general classes of fairness notions which describe the conditions under which the uniqueness of the fair work-conserving allocation policy will hold.

Definition 12 A notion of fairness $\mathcal{F}$ is said to be *uniquely deterministic*, if and only if, given the constraint C, the normalized demand vector $[\tilde{d}_i]$ and the weight vector $[w_i]$, the normalized allocation vector $[\tilde{a}_i]$ as given in (1.5) can be uniquely determined.

Definition 13 A notion of fairness $\mathcal{F}$ is said to be *non-decreasing*, if and only if, given the normalized demand vector $[\tilde{d}_i]$ and the weight vector $[w_i]$, the normalized allocation $[\tilde{a}_i]$ is such that, for any two different constraints C_1 and C_2, one of the following holds true:

$$\begin{aligned} \mathcal{F}\left(C_1, [\tilde{d}_i], [w_i]\right) &\prec \mathcal{F}\left(C_2, [\tilde{d}_i], [w_i]\right) \\ \mathcal{F}\left(C_2, [\tilde{d}_i], [w_i]\right) &\prec \mathcal{F}\left(C_1, [\tilde{d}_i], [w_i]\right). \end{aligned}$$

Here $\prec$ is a relational operator between two vectors of identical dimensions, and $[u_i] \prec [v_i]$ implies $\forall i, u_i \leq v_i$. This definition of non-decreasing fairness notion can be also expressed as follows: when allocating a single resource under a non-decreasing fairness notion, no flow will get a lesser amount of the resource if the total amount of the shared resource increases.

These classes of fairness notions are actually very broad; it may be readily verified that many popular notions of fairness are both non-decreasing and uniquely deterministic. These include max-min fairness [5, 6, 15], proportional fairness [57], and utility max-min fairness [16] if the utility functions are non-decreasing.

Lemma 1 In a system with multiple essential and related resources, the normalized allocations received by a flow i are identical under two allocation policies q and s if $\mathcal{A}_i^q = \mathcal{A}_i^s$.

Proof: Let $\mathcal{B}_i^q$ and $\mathcal{B}_i^s$ be one of the prime resources of flow i under policies q and s, respectively. We have

$$\begin{aligned} \tilde{a}^q_{i,\mathcal{B}_i^s} &\leq \tilde{a}^q_{i,\mathcal{B}_i^q} \\ \tilde{a}^s_{i,\mathcal{B}_i^q} &\leq \tilde{a}^s_{i,\mathcal{B}_i^s}. \end{aligned}$$

In addition, we have $\tilde{a}^q_{i,\mathcal{B}_i^q} = \tilde{a}^s_{i,\mathcal{B}_i^s}$ since $\mathcal{A}_i^q = \mathcal{A}_i^s$. Therefore, $\tilde{a}^q_{i,\mathcal{B}_i^s} \leq \tilde{a}^s_{i,\mathcal{B}_i^s}$ which means flow i receives less allocation of resource $\mathcal{B}_i^s$ under policy q than under policy s. Also, since the resources are essential, flow i receives no more allocation of any other resource including $\mathcal{B}_i^q$ under policy q than under policy s, i.e.,

$$\tilde{a}^s_{i,\mathcal{B}_i^q} \geq \tilde{a}^q_{i,\mathcal{B}_i^q} = \tilde{a}^s_{i,\mathcal{B}_i^s} = \mathcal{A}_i^s.$$

This means $\mathcal{B}_i^q$ is also one of the prime resources of flow i under policy s. It may be similarly deduced that $\mathcal{B}_i^s$ is one of the prime resources of flow i under policy q. In summary, if $\mathcal{A}_i^q = \mathcal{A}_i^s$, the sets of prime resources of flow i are identical under policies q and s, and the allocations of these prime resources to flow i are also identical under policies q and s. Since the resources are related, flow i receives identical allocations of all resources from both policies q and s. ■

Theorem 3 If the applied notion of fairness is both non-decreasing and uniquely deterministic, there exists a unique fair work-conserving allocation policy that satisfies the FERA principle as stated in Section 3.3.3.

Proof: We will prove this theorem by contradiction.

Assume that in the considered system, when applying the FERA principle, two different policies q and s are both fair work-conserving allocation policies corresponding to the notion of fairness $\mathcal{F}$. Denote the constraints corresponding to these two allocation policies by C^q and C^s, respectively.

Note that for these two allocation policies q and s, the vectors of the largest normalized allocations, i.e., $[\mathcal{A}_i^q]$ and $[\mathcal{A}_i^s]$, cannot be equal. This is because if that is the case, from

Lemma 1, the allocated amount of each resource for each flow will be the same under policies q and s, and policies q and s will be identical.

Since the system under consideration remains the same, we know that both the vector of the largest normalized demands $[\mathcal{D}_i]$ and the vector of the flow weights $[w_i]$ are the same. From the definition of a uniquely deterministic notion of fairness, we know that $C^q \neq C^s$ since, otherwise, the two vectors of largest normalized allocations, $[\mathcal{A}_i^q]$ and $[\mathcal{A}_i^s]$, will be equal.

Since the notion of fairness $\mathcal{F}$ is non-decreasing, from the definition, we have either $[\mathcal{A}_i^q] \prec [\mathcal{A}_i^s]$ or $[\mathcal{A}_i^s] \prec [\mathcal{A}_i^q]$. Without loss of generality, we assume that $[\mathcal{A}_i^q] \prec [\mathcal{A}_i^s]$, i.e.,

$$\mathcal{A}_i^q \leq \mathcal{A}_i^s, \forall i. \tag{3.1}$$

Therefore, for all flows, the allocated amount of each resource under policy q is no more than that under policy s, since the resources are related and essential.

In addition, there must exist at least one flow, for which (3.1) is not an equality. In other words, there exists at least one flow, which gets more resources under policy s than under policy q.

Hence, under policy s, as opposed to policy q, no flow gets less allocation for any resource, and at least one flow is allocated more of some resources. This violates the assumption that policy q is work-conserving and completes the proof. ■

3.4 Fair Joint Allocation of Processing and Bandwidth Resources

In this section, we apply the framework established in the previous section into an important context of special interest: the fair joint allocation of a shared processor P and a shared link L under the max-min notion of fairness and linear utility functions.

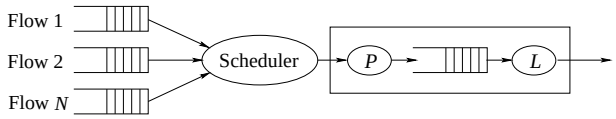

Figure 3.2: The system model with a shared processor P and a shared link L.

3.4.1 System Model

In this system model, a set of N flows share a processor P and a link L, as shown in Fig. 3.2. Packets from each flow are processed by processor P first and then transmitted onto the output link L. Denote by R_L the peak bandwidth rate of link L and by R_P the peak processing rate of processor P. Packets of each flow await processing by the processor in an input buffer of infinite capacity, and then upon completion of the processing, await transmission on the output link in another buffer of infinite capacity. The joint allocation of the processing and bandwidth resources is accomplished by the scheduler which acts on the packets in the input buffers and appropriately orders them for processing by the processor. No scheduling action takes place after the processing; processed packets are received in the buffer between the processor and the link and are transmitted in a first-come-first-served fashion.

Denote by w_i the weight of flow i, $1 \leq i \leq N$, indicating the flow's relative rightful share of the resources.

3.4.2 Fluid-flow Processor and Link Sharing

Denote by S the system illustrated in Fig. 3.2. We first consider fluid-flow traffic through system S, and describe an ideally fair allocation strategy called the *Fluid-flow Processor and Link Sharing (FPLS)* algorithm. FPLS is intended to serve the same purpose for system S as that served by GPS for a system with just a single shared link or a

single shared processor [6, 15].

In GPS, it is assumed that traffic from each flow can be divided into infinitesimally small chunks, and each chunk has its demand for access to the link L depending on the size of the chunk. The GPS scheduler visits each active flow's queue in a round-robin fashion, and serves an infinitesimally small amount of data from each queue in such a way that during any infinitesimal interval of time, it can visit each queue at least once. In our study, this assumption is still valid, and we further assume that each infinitesimal chunk also has its demand for the processing time on the shared processor P.

At each time instant τ, the prime resource for each flow, according to Definition 10, can be determined based on its instantaneous demands for processing time and bandwidth. In addition, we assume that during each infinitesimal interval of time, $[\tau, \tau + \Delta\tau)$, the prime resource for each flow does not change.

Note that in GPS, it is guaranteed that during each infinitesimal interval of time, the chunks of each flow are scheduled in such a way that, for each flow, the total demand for bandwidth corresponding to the chunks of the flow scheduled in this period is proportional to the weight of the flow. Extending GPS to our case leads to the following: Under the ideally fair allocation policy for system S, it is guaranteed that, during each infinitesimal interval of time, the chunks of each flow are scheduled in such a way that, for each flow, the total *normalized* demand for its *prime resource* corresponding to the chunks of the flow scheduled in this period is proportional to the weight of the flow. We refer to this as *Fluid-flow Processor and Link Sharing (FPLS)*. It can be readily verified that the FPLS strategy meets the FERA principle described in Section 3.3.3.

3.4.3 Packet-by-packet Processor and Link Sharing

It is apparent that FPLS is an ideally fair but unimplementable policy, in the same sense as GPS. In reality, network traffic is always packetized, and therefore, we next present

a practical approximation of FPLS, called *Packet-by-packet Processor and Link Sharing (PPLS)*. The PPLS algorithm extends one of the most practical and simple scheduling strategies, Deficit Round Robin (DRR) [8], used in the allocation of bandwidth on a link. Please refer to Section 1.3.6 for a brief description of DRR. The pseudo-code of PPLS is shown in Fig. 3.3.

The PPLS algorithm approximates the ideal FPLS in a very similar fashion as DRR achieves an approximation of GPS. The PPLS scheduler maintains a linear list of the backlogged flows, *FlowList*. When the scheduler is initialized, *FlowList* is set to an empty list (line 2). For each flow, two variables, instead of one as in DRR, are maintained in the PPLS algorithm: a *processor deficit counter (PDC)* and a *link deficit counter (LDC)*. The link deficit counter is exactly the same as the deficit counter in DRR, which represents the deviation of the bandwidth received by the flow from its ideally fair share. The processor deficit counter, on the other hand, represents the deviation of the processing time allocated to the flow from its ideally fair share. Thus, each flow in PPLS is assigned two quantum values, a *processor quantum (PQ)* and a *link quantum (LQ)*.

When a new packet arrives, the *Enqueue* procedure is invoked (lines 3-10). If this packet comes from a new flow, the *Enqueue* procedure appends this flow to the end of the *FlowList* (line 7) and initializes both of its deficit counters to 0 (lines 8-9).

The *Dequeue* procedure (lines 11-38) functions as follows. It serves all flows in the *FlowList* in a round-robin fashion. When the scheduler visits flow i, it first increments each of the two deficit counters of this flow by the value of the corresponding quantum (lines 16-17). It then verifies whether or not these two deficit counters exceed their upper bounds respectively, and if they do, it resets them to the maximum possible values (lines 18-23). The rationale behind this bounding process will be discussed later in detail. After the deficit counters of flow i are updated, a sequence of packets from flow i are scheduled as long as the total length of these packets is smaller than the link deficit counter, and the total processing cost is smaller than the processing deficit counter, as in the **while** loop in lines

```
1   Initialize:
2     FlowList ← NULL;

3   Enqueue: /* Invoked whenever a packet arrives */
4     p ← ArrivingPacket;
5     i ← Flow(p); /* Flow of packet p */
6     if (ExistsInFlowList(i) = FALSE) then
7       Append flow i to FlowList;
8       PDC_i ← 0;
9       LDC_i ← 0;
10    end if;

11  Dequeue: /* Always running */
12    while (TRUE) do
13      if (FlowList ≠ NULL) then
14        i ← HeadOfFlowList;
15        Remove i from FlowList;
16        PDC_i ← PDC_i + PQ_i;
17        LDC_i ← LDC_i + LQ_i;
18        if (PDC_i > maxPDC_i) then
19          PDC_i ← maxPDC_i;
20        end if;
21        if (LDC_i > maxLDC_i) then
22          LDC_i ← maxLDC_i;
23        end if;
24        while (QueueIsEmpty(i) = FALSE) do
25          p ← HeadOfLinePacketInQueue(i);
26          if (Size(p) > LDC_i OR
27                ProcessingCost(p) > PDC_i) then
28            break; /* escape from the inner while loop */
29          end if;
30          PDC_i ← PDC_i− ProcessingCost(p);
31          LDC_i ← LDC_i− Size(p);
32          Schedule p;
33        end while;
34        if (QueueIsEmpty(i) = FALSE) then
35          Append queue i to FlowList;
36        end if;
37      end if;
38    end while;
```

Figure 3.3: Pseudo-code of the Packet-by-packet Processor and Link Sharing (PPLS) algorithm.

24-33. In the meantime, when a packet is scheduled, both deficit counters are decremented by the corresponding cost of this packet (lines 30-31). Finally, when the scheduler finishes serving a flow and the flow still remains backlogged, the scheduler places the flow back at the end of the *FlowList* (lines 34-36).

Recall that in DRR, for each flow, the quantum is set to be proportional to its weight, therefore, each flow receives in each round, on average, a service with total amount proportional to its weight. In this chapter, the sum of a certain quantity over *all* flows is denoted by dropping the subscript for the flow in the notation. For example, w is the sum of the weights for all flows, i.e., $w = \sum_i w_i$. Therefore, in DRR we have

$$\frac{Q_i}{w_i} = \frac{Q}{w}, \forall i.$$

Similarly, in PPLS, the quantum values of each flow are also proportional to its weight, i.e., $\forall i$,

$$\frac{PQ_i}{w_i} = \frac{PQ}{w} \tag{3.2}$$

$$\frac{LQ_i}{w_i} = \frac{LQ}{w}. \tag{3.3}$$

Thus the amount of the shared resources each flow is entitled to utilize in each round is guaranteed to be, on average, proportional to its weight. In addition, the ratio of the sum of processing quanta for all flows, PQ, to the sum of link quanta for all flows, LQ, should also be equal to the ratio of the total amount of processing resource to the total amount of link resource in each round, i.e.,

$$\frac{PQ}{LQ} = \frac{R_P}{R_L}. \tag{3.4}$$

From (3.2), (3.3) and (3.4), it is apparent that,

$$\frac{PQ_i}{LQ_i} = \frac{R_P}{R_L}. \tag{3.5}$$

In other words, for each flow, the quantum value corresponding to a resource is proportional to the total amount of that resource.

Note that in PPLS, it is possible that the prime resource for flow i remains the same for a long period, and therefore, without the bounding procedure in lines 18-21, the deficit counter for the non-prime resource would reach a large value. For example, consider a case in which the prime resource for flow i has been the processing resource P for a long time and, as a result, the link deficit counter LDC_i is very large. Assume that at this point, the prime resource for flow i switches to the link resource L and, in addition, flow i now consumes almost no processing resource. In such a situation, flow i will be able to have a long sequence of packets scheduled because of its large link deficit counter LDC_i. This would significantly degrade the short-term fairness of the PPLS scheduler. For this reason, we choose to set a maximum threshold on the deficit counter for each resource, in case any specific resource has not been fully utilized for a long time. In cases where short-term fairness is not important, these thresholds may simply be set to infinity. A similar rationale may also be found in the context of fair scheduling in wireless networks where a maximum lag is applied when a flow has not fully utilized its share of the bandwidth [58].

It can be readily verified that if the processor resource P is sufficient for all flows, i.e., the processor resource P never becomes the prime resource for any flow, the PPLS strategy regresses into the DRR policy. It can also be readily verified that, like DRR, the per-packet computing complexity of the PPLS algorithm is $O(1)$, under the following conditions.

Theorem 4 The per-packet computing complexity of the PPLS algorithm is $O(1)$, if for each flow i,

$$\begin{cases} LQ_i \geq M_L \\ PQ_i \geq M_P \end{cases}$$

where M_L and M_P are the maximum packet size and the maximum packet processing cost, respectively.

Proof: The proof of this work complexity is simple and similar to that for DRR [8].

Note that when the PPLS scheduler visits a flow i, it first increments each of the two deficit counters of flow i by the value of the corresponding quantum (lines 16–17). There-

fore, it is guaranteed that when flow i gets served, its precessor deficit counter PDC_i is no less than its processor quantum PQ_i, and

$$PDC_i \geq PQ_i \geq M_P \geq ProcessingCost(p)$$

where p is the first packet of flow i at this moment. Similarly the link deficit counter LDC_i is no less than the maximum packet size M_L, and

$$LDC_i \geq LQ_i \geq M_L \geq Size(p).$$

Therefore, PPLS guarantees to transmit at least the first packet p of flow i. In addition, to transmit each packet, PPLS only updates the deficit counters and checks the resource costs of the packet. The execution time of all these tasks is $O(1)$, and therefore, PPLS has a per-packet work complexity of $O(1)$. ■

3.4.4 Fairness Analysis of PPLS

Our fairness analysis of PPLS is an extension of that in [8], and considers only the time intervals where all flows are backlogged.

The *cumulative processor allocation* of flow i during time interval (t_1, t_2), denoted by $\mathrm{CPA}_i(t_1, t_2)$, is defined as the total amount of the processing resource allocated to flow i during interval (t_1, t_2), i.e., the sum of the processing costs associated with the packets scheduled during (t_1, t_2). The *normalized cumulative processor allocation*, $\mathrm{nCPA}_i(t_1, t_2)$, is defined as the cumulative processor allocation $\mathrm{CPA}_i(t_1, t_2)$ normalized by the peak processing rate R_P, i.e.,

$$\mathrm{nCPA}_i(t_1, t_2) = \frac{\mathrm{CPA}_i(t_1, t_2)}{R_P}.$$

The *cumulative link allocation* and the *normalized cumulative link allocation* of flow i during time interval (t_1, t_2), denoted by $\mathrm{CLA}_i(t_1, t_2)$ and $\mathrm{nCLA}_i(t_1, t_2)$ respectively, are similarly defined, except that the resource considered is the link bandwidth.

Note that both the normalized cumulative link allocation and the normalized cumulative processor allocation are in units of time. Therefore, we are able to proceed to define the *normalized cumulative resource allocation* of flow i during time interval (t_1, t_2), denoted by $\mathrm{nCRA}_i(t_1, t_2)$, as the larger of the normalized cumulative processor and link allocations of flow i during (t_1, t_2). In other words,

$$\mathrm{nCRA}_i(t_1, t_2) = \max\{\mathrm{nCPA}_i(t_1, t_2), \mathrm{nCLA}_i(t_1, t_2)\}.$$

Now we can extend the definition of the fairness measure [7] as follows:

Definition 14 The *normalized fairness measure* $\mathrm{nFM}(t_1, t_2)$ is defined as the maximum value, amongst all pairs of flows (i, j) that are backlogged during time interval (t_1, t_2), of the normalized cumulative resource allocation $\mathrm{nCRA}_i(t_1, t_2)$. That is,

$$\mathrm{nFM}(t_1, t_2) = \max_{\forall(i,j)} \left| \frac{\mathrm{nCRA}_i(t_1, t_2)}{w_i} - \frac{\mathrm{nCRA}_j(t_1, t_2)}{w_j} \right|.$$

The *normalized fairness bound* nFB is defined as the maximum value of the normalized fairness measure $\mathrm{nFM}(t_1, t_2)$ over all possible intervals (t_1, t_2).

Analogous to the case of a single shared resource, if a scheduling algorithm for the joint allocation of processing and bandwidth resources leads to a finite normalized fairness bound, one can conclude that this algorithm approximates the ideally fair allocation and achieves long-term fairness. The following theorem states this about the PPLS algorithm.

Theorem 5 The normalized fairness bound of PPLS is a finite constant.

Proof: Without loss of generality, we assume that the flow weights are normalized in such a way that the smallest of the weights assigned to a flow is 1.

In the rest of this proof, we will consider the situations where lines 19 and 22 in Fig. 3.3 are never executed, i.e., the deficit counters of any flow are never above the thresholds. The reason of this assumption is similar to the one used in the design of IWFQ, where fairness in bandwidth cannot be guaranteed if any flow lags more than the maximum lag allowed by the wireless packet scheduler [59].

Lemma 2 Let

$$maxPDC = \max_{\forall i} maxPDC_i$$

$$maxLDC = \max_{\forall i} maxLDC_i$$

and denote by M_P and M_L, respectively, the maximum processing cost of a packet and the maximum link cost of a packet. In an execution of the PPLS strategy, at the end of each round k, for any flow i,

1. The following two statements are always satisfied:

$$0 \leq PDC_i(k) \leq maxPDC$$

$$0 \leq LDC_i(k) \leq maxLDC;$$

2. At least one of the following statements is always satisfied:

$$0 \leq PDC_i(k) \leq M_P$$

$$0 \leq LDC_i(k) \leq M_L.$$

Proof: First it can be readily verified that the deficit counters can never be negative. The first half of Lemma 2 can be directly derived from the assumption that lines 19 and 22 are never executed.

Next we prove the second half of Lemma 2 by contradiction. Assume that both statements are not true, then we have $PDC_i(k) > M_P$ and $LDC_i(k) > M_L$. Note that at this moment, flow i still has packets in the queue waiting to be scheduled. Otherwise both deficit counters of flow i should be reset to 0. Consider the head-of-line packet of flow i, say p. Apparently its processing cost is no more than M_P and its link cost is no more than M_L. In other words, its processing cost is less than $PDC_i(k)$ and its link cost is less than $LDC_i(k)$, and therefore, based on the PPLS algorithm, packet p should be scheduled in round k. This violates the assumption that packet p is the head-of-line packet from flow i at the end of round k, and completes the proof. ■

Lemma 2 readily leads to the following Corollary.

Corollary 1 In an execution of the PPLS strategy, at the end of each round k, for any flow i,

$$\max\left(\frac{PDC_i(k)}{R_P}, \frac{LDC_i(k)}{R_L}\right) \leq \alpha$$
$$\min\left(\frac{PDC_i(k)}{R_P}, \frac{LDC_i(k)}{R_L}\right) \leq \beta$$

where constants α and β are defined as follows:

$$\alpha = \max\left(\frac{\max_{\forall i} maxPDC_i}{R_P}, \frac{\max_{\forall i} maxLDC_i}{R_L}\right) \tag{3.6}$$

$$\beta = \min\left(\frac{M_P}{R_P}, \frac{L_P}{R_L}\right). \tag{3.7}$$

According to (3.5), we also define constant γ as follows:

$$\gamma = \frac{\min_{\forall i} PQ_i}{R_P} = \frac{\min_{\forall i} LQ_i}{R_L}. \tag{3.8}$$

Lemma 3 During an execution of the PPLS strategy over any m rounds, for any flow i,

$$mw_i\gamma - \beta \leq \mathrm{nCRA}_i(m) \leq mw_i\gamma + \alpha$$

where α, β, γ are constants defined in (3.6), (3.7) and (3.8), respectively.

Proof: Denote by $\mathrm{SCPA}_i(k)$ the cumulative processor allocation of flow i in a single round k. From the algorithm we have

$$\mathrm{SCPA}_i(k) = PQ_i + PDC_i(k-1) - PDC_i(k).$$

This leads to

$$\begin{aligned}\mathrm{CPA}_i(m) &= \sum_{k=1}^{m} \mathrm{SCPA}_i(k) \\ &= mPQ_i + PDC_i(0) - PDC_i(m)\end{aligned}$$

and

$$\begin{aligned} \mathrm{nCPA}_i(m) &= \frac{\mathrm{CPA}_i(m)}{R_P} \\ &= m\frac{PQ_i}{R_P} + \frac{PDC_i(0) - PDC_i(m)}{R_P}. \end{aligned}$$

From (3.2) we have

$$\frac{PQ_i}{R_P} = \frac{w_i}{\min_{\forall j} w_j}\frac{\min_{\forall j} PQ_j}{R_P} = w_i\gamma$$

and therefore,

$$\mathrm{nCPA}_i(m) = mw_i\gamma + \frac{PDC_i(0) - PDC_i(m)}{R_P}.$$

Since both $PDC_i(0)$ and $PDC_i(m)$ are non-negative,

$$mw_i\gamma - \frac{PDC_i(m)}{R_P} \leq \mathrm{nCPA}_i(m) \leq mw_i\gamma + \frac{PDC_i(0)}{R_P}.$$

Similarly we have

$$mw_i\gamma - \frac{LDC_i(m)}{R_L} \leq \mathrm{nCLA}_i(m) \leq mw_i\gamma + \frac{LDC_i(0)}{R_L}.$$

Applying into the definition of normalized cumulative resource allocation leads to the following:

$$\begin{aligned} \mathrm{nCRA}_i(m) &\leq mw_i\gamma + \max\left(\frac{PDC_i(0)}{R_P}, \frac{LDC_i(0)}{R_L}\right) \\ \mathrm{nCRA}_i(m) &\geq mw_i\gamma - \min\left(\frac{PDC_i(m)}{R_P}, \frac{LDC_i(m)}{R_L}\right). \end{aligned}$$

Applying Corollary 1 into the above inequalities completes the proof. ■

Consider a certain time interval (t_1, t_2) during which all flows remain backlogged. Consider any pair of flows i and j. Assume that during (t_1, t_2), flow i receives m_i rounds of service while flow j receives m_j rounds of service. Since both flows i and j are backlogged during time interval (t_1, t_2), and the scheduler serves the flows in a round-robin fashion, we have $|m_i - m_j| \leq 1$.

Applying Lemma 3 we have

$$\begin{aligned} \frac{\mathrm{nCRA}_i(t_1,t_2)}{w_i} &\leq m_i\gamma + \frac{\alpha}{w_i} \\ \frac{\mathrm{nCRA}_j(t_1,t_2)}{w_j} &\geq m_j\gamma - \frac{\beta}{w_j}. \end{aligned}$$

Therefore,

$$\begin{aligned} \frac{\mathrm{nCRA}_i(t_1,t_2)}{w_i} - \frac{\mathrm{nCRA}_j(t_1,t_2)}{w_j} &\leq (m_i - m_j)\gamma + \frac{\alpha}{w_i} + \frac{\beta}{w_j} \\ &\leq \alpha + \beta + \gamma. \end{aligned}$$

Similarly we can also derive that,

$$\frac{\mathrm{nCRA}_j(t_1,t_2)}{w_j} - \frac{\mathrm{nCRA}_i(t_1,t_2)}{w_i} \leq \alpha + \beta + \gamma.$$

Since flows i and j can be any pair of flows, based on the definition of the normalized fairness measure, we have

$$\mathrm{nFM}(t_1,t_2) \leq \alpha + \beta + \gamma.$$

Note that α, β and γ are all finite constants. Therefore, $\mathrm{nFM}(t_1,t_2)$ is bounded by a finite constant over any time interval during which all flows are backlogged, i.e., the fairness bound nFB exists for the PPLS strategy and it is finite. This proves the statement of Theorem 5. ■

3.5 Simulation Results and Analysis

Our simulation model consists of 8 flows with equal weights sharing a processor P and a link L, as shown in Fig. 3.2. Five different scheduling policies including the PPLS algorithm are implemented.

- FCFS (First-Come First-Served): A simple FCFS scheme is used. The scheduling order is only determined by the packet timestamps.

- PPLS: When the PPLS algorithm is implemented, a FCFS strategy is used on the buffer between the processor P and the link L, since the order of the packets has already been determined by the PPLS algorithm.

- LDRR (Link Deficit Round Robin): A DRR algorithm in the allocation of only the link bandwidth is implemented (i.e., the original DRR).

- PDRR (Processor Deficit Round Robin): A DRR algorithm in the allocation of only the processing resources is implemented.

- DDRR (Double Deficit Round Robin): Two DRR schedulers are used. PDRR is used before the processor P and LDRR is used before the link L. Note that this is the only scheme in which a scheduler is implemented between the processor and the link.

Two sets of simulation experiments have been tested. In the first set of experiments, a synthetic traffic sequence is used, while the second set uses real gateway traffic traces as the traffic sources.

3.5.1 Synthetic Traffic

In our first study, we use synthetic traffic to test the fairness properties of the PPLS algorithm under some extreme situations. In this study, all packets are randomly generated. For each flow, the ratio of the amount of the processing resource required to the amount of the bandwidth resource required is a fixed value. Note that in the definition of the normalized fairness measure $\mathrm{nFM}(t_1, t_2)$, if both the R_P and R_L are multiplied by the same value, the normalized fairness measure will also be multiplied by this value, in other words, the fact of whether or not the normalized fairness measure is bounded does not change except that the bound itself may vary. Therefore, for better illustration and easier comparison, in our study, we normalize the resource amount in such a way that the average processor cycles needed per packet (in units of cycles) is numerically equal to the average

Table 3.2: The ratio of the processing resource to the link resource required by each flow.

Flow ID	1	2	3	4	5	6	7	8
P/L Ratio (in cycle/byte)	1	2	3	4	1	1/2	1/3	1/4

size per packet (in units of bytes). Table 3.2 shows the ratios used. Note that flows 1 and 5 have equal normalized demand for both resources, while the prime resource for flows 2, 3 and 4 is the processor, and for flows 6, 7 and 8 is the link. For flows 1 to 4, the sizes of packets generated is uniformly distributed between 1 and 1,600 bytes, while for flows 5 to 8, the processing cost is uniformly distributed between 1 and 1,600 cycles. Therefore, the maximum packet size is 6,400 bytes and the maximum processing cost is 6,400 cycles, and these are also the quantum values assigned to each flow.

Fig. 3.4(a) shows the normalized cumulative resource allocation after a long run in the simulations. It is apparent that using the PPLS algorithm, in the time interval $(0, \tau)$ under consideration, the normalized cumulative resource allocation $\mathrm{nCRA}_i(0, \tau)$ for all flows i are very close, thus illustrating that fairness is achieved under the PPLS scheduling policy. Note that, as expected, the FCFS scheme is the worst among all in terms of fairness. Regarding LDRR and PDRR, each can achieve fair distribution of the normalized cumulative allocation with respect to a certain resource, but not the overall normalized cumulative resource allocation. Take LDRR as an example. It achieves fair distribution of the normalized cumulative link allocation for all flows. Therefore, those flows with the processor as the prime resource, namely flows 2 to 4 in this case, result in a large value of the normalized cumulative processor allocation, thus failing to achieve fairness. PDRR functions exactly in the opposite way: it fairly distributes the normalized cumulative processor allocation among all flows, but those flows with the link as the prime resource (flows 5 to 8) receive a

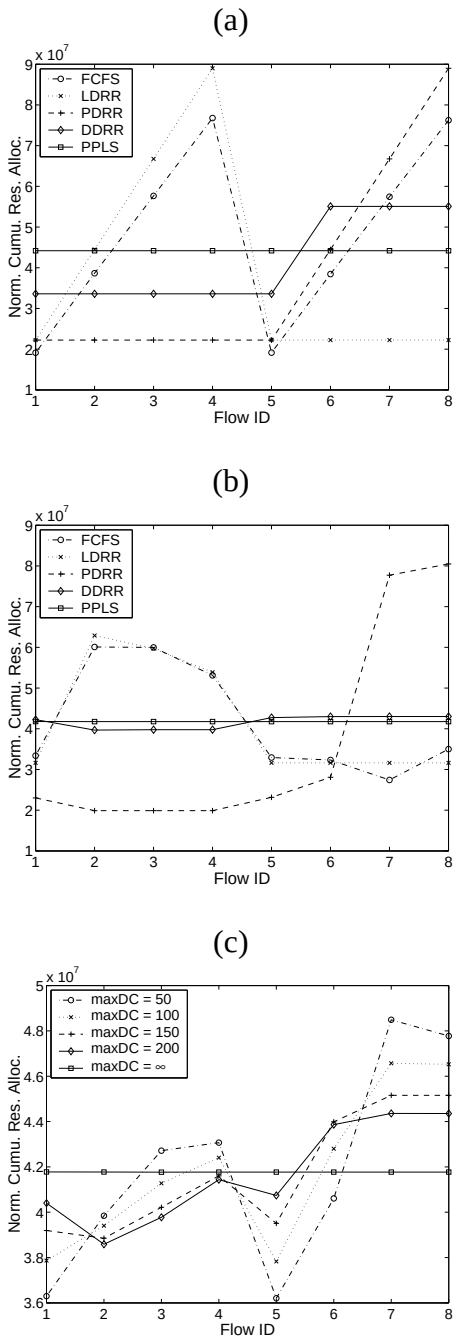

Figure 3.4: The simulation results, using (a) synthetic traffic, (b) gateway traffic traces. (c) The effect of *maxDC* in the PPLS algorithm. In these plots, a curve closer to a straight horizontal line implies a better fairness achieved by an allocation policy.

large normalized cumulative link allocation, also failing to achieve fairness.

One interesting observation is the DDRR scheme. Intuitively one may expect DDRR to serve as a fair scheduler for allocation of processing and bandwidth resources, since it has two schedulers, one fair with respect to the processor and the other fair with respect to the link. However, Fig. 3.4(a) shows that this is not the case. This is because the DDRR scheme implements the two fair schedulers in different stages. Note that the PDRR scheduler before the processor P is responsible for fairly allocating the normalized cumulative processor allocation to all flows. That means, at this point, more packets (in bytes) from those flows with the link as the prime resource (flows 6 to 8) will be scheduled from the processor P. On the other hand, those flows with the processor as the prime resource (flows 2 to 4) will not have enough packets to remain backlogged in the buffer before the link L. The LDRR scheduler then takes advantage of this from flows 2 to 4 and transmits more packets from flows 6 to 8, thus causing a higher normalized cumulative resource allocation for flows 6 to 8. In fact, the DDRR scheme allocates resources fairly to all flows with the same prime resource, but favors the flows with the "final" resource as the prime resource.

3.5.2 Gateway Traffic Traces

In this study, we use real traffic recorded at an Internet gateway as the input traffic [56, 60].[5] The traffic traces include the processing delay (in milliseconds) for each packet, along with the packet size (in bytes). For our experiments, we assume a fixed processing rate, and correspondingly convert the processing delay of each packet into processor cycles.

---

[5]Global Positioning System technology was used to precisely record the timestamp of each packet at each node. In the trace data, filtered IP headers were examined to track the same packet at different nodes. The difference between the timestamps of the same packet at adjacent nodes was computed as the delay. The link speed connecting these nodes was taken into consideration so that the transmission delay of each packet was removed from the recorded delay. Note that this delay was still the sum of the processing delay and the queueing delay. However, it was noticed that for traffic in a specific direction, the queue occupancy was never above 1 packet, and this eliminates the queueing delay and validates the use of this delay as the pure processing delay.

Again, we convert the processing delay of each packet in such a way that the average number of processor cycles needed per packet (in units of cycles) is numerically equal to the average size per packet (in units of bytes). For better comparison to the previous study, the flows have been ordered in such way that the overall prime resource for flows 1 to 4 is the processor, and the link for flows 5 to 8.

Fig. 3.4(b) illustrates the normalized cumulative resource allocation for the five scheduling schemes in this experiment. Again, the PPLS algorithm performs very well in terms of fairness. It is observed that all other conclusions drawn from study with synthetic traffic are still valid.

Note that in this study, the DDRR scheme performs closer to the PPLS algorithm than in the previous study. This can be attributed to the fact that in real traces, the demands of each flow for the processing and bandwidth resources are more balanced than those in the synthetic traffic. However, the PPLS algorithm only needs one scheduler in real implementation while the DDRR needs two.

3.5.3 Effect of Maximum Deficit Counter

Note that in the synthetic study, no flow changes its prime resource during the experiment. Therefore, the setting of the maximum deficit counters in the PPLS algorithm has no effect on the outcome of the simulations. Next, using the real gateway traces, we focus on the effect of maximum deficit counters on the PPLS algorithm. This is shown in Fig. 3.4(c).

It is apparent that the prime resource of a flow changes in this study, since the normalized cumulative resource allocation begins to show differences under the PPLS algorithm. However, it should be noticed that the normalized cumulative resource allocations for the flows are still reasonably close to each other, due to the long-term fairness achieved by the PPLS algorithm.

From Fig. 3.4(c), it is observed that, as expected, the long-term fairness among normal-

ized cumulative resource allocation degrades as the maximum deficit counter decreases. For example, when the maximum deficit counter is set to be 10 times as large as the quantum value, the normalized cumulative resource allocation exhibits a 10% variation from the ideal.

It is also observed from Fig. 3.4(c) that, flows with more balanced normalized cumulative allocations between the two resources over a long run, such as flows 1, 2 and 5, are likely to receive less normalized cumulative resource allocation. This may be attributed to the fact that these flows are more likely to temporarily change the prime resource, and therefore, setting the deficit counter for the current non-prime resource to the maximum value may reduce the future usage of this resource when it later becomes prime resource. On the other hand, the unbalanced flows are less likely to temporarily change the prime resource, and therefore, the effect on these flows of setting the deficit counter for the non-prime resource to the maximum value is limited. A similar scenario may also be found in other situations, such as bandwidth sharing. For example, in DRR, a flow that frequently changes its status of being backlogged or not will be sacrificed in a long run, since each time it becomes non-backlogged its unused deficit counter is reset to 0, thus causing it to lose bandwidth share.

Based on the above discussion, the maximum deficit counter can be used to tune the trade-off between the long-term and the short-term fairness of the PPLS algorithm. This is similar to the function of the maximum lag in wireless scheduling [58].

3.6 Discussions on Implementation of PPLS

In this chapter, we select DRR [8] as the starting point of the design of the fair allocation policy for a shared processor and a shared link, because of the relatively simple implementation of DRR. Other fair scheduling algorithms can be also used, such as Weighted Fair Queueing (WFQ) [5], Worst-case Fair Weighted Fair Queueing (WF^{2}Q) [9], Surplus Round

Robin (SRR) [61] and Elastic Round Robin (ERR) [10].

Note that in many situations, the processing cost of a packet cannot be determined before it is actually processed. If this is the case, one can have the following choices to modify the PPLS algorithm. The first way is to let the scheduler predict the processing cost, and make scheduling decisions. In the second choice, the scheduler serves the packet first, then updates the deficit counters accordingly. In this way, it is possible that after serving a packet, its processing deficit counter becomes negative, thus breaking the fairness property of the PPLS algorithm. Therefore, the scheduler needs an additional counter to record the minimum normalized deficit counter for all flows, and if this value becomes negative, at the beginning of next round, it needs to add a proper amount to the deficit counter of each flow to make it non-negative. Note that using prediction before scheduling still needs this protection from negative deficit counter, and therefore, one can combine these two approaches: predict first, and then correct if not accurate. If these changes are applied, the PPLS algorithm becomes closer to ERR [10] where packet sizes are unknown when scheduling.

In our study, it is assumed that each flow has a unique weight which determines its relative rightful share for each resource. One might claim that this assumption might not be true in all situations. If instead, for each flow, a different weight is associated with each individual resource, the premise of this work can still be applied. The only difference would be that when defining the prime resource for each flow, the weight for each individual resource needs to be taken into consideration, and an additional concept, prime weight, needs to be defined as the weight associated with the prime resource. Also, if that is the case, the quantum values for a flow in the PPLS algorithm need to be assigned proportional to its corresponding weight.

Chapter 4. A Discussion on Extensions to Multiple Output Link Systems

4.1 Introduction

4.1.1 Motivation and Challenges

In building the theoretical foundation for fair allocation of multiple resources, the studies in the previous two chapters consider the situations where only one output link is present in the system. In the investigated systems, all flows are headed to the same shared output link via a common shared resource (being either a buffer as in Chapter 2 or a processor as in Chapter 3). In a multiple output link system, however, several flows may share the same buffer or processor but different flows may be headed to different output links. Both the single and multiple output link systems arise in various real situations in switches and routers serving traffic from a multitude of flows [62–66]. In this chapter, we focus on the multiple output link systems, and using the principles developed in previous chapters, answer the question of what is fair in resource allocation in such systems. Note that the challenges imposed on multiple output link systems with either a shared buffer or a shared processor are very similar to each other, and the methods used to define fairness in one system can be readily applied to the other. Therefore, we will use the buffer allocation as an example in the rest of this chapter to develop the principle for fairness in such systems. Discussions on the processor allocation in multiple output link systems are presented at the end of this chapter.

The question of what is fair in the multiple output link systems offers a unique scenario that cannot be readily analyzed and understood by extrapolation from the single output link cases discussed in Chapter 2. Each link in the multiple output link system is a separate resource that is not necessarily shared by all the flows. For example, consider a flow that has an input rate into the buffer that is larger than the peak rate of its destination output

link. While this flow does not get its demand for throughput, unused bandwidth on other links may not be allocated to it. Thus, bandwidth wasted on one link is not necessarily available to other flows with unsatisfied demands for bandwidth. Therefore, one cannot treat the multiple output links as one, and operate an exit strategy from the buffer based on it. The exit policy has to recognize that various output links may have different peak rates, and that the buffer occupancies of the flows depend on it.

Exit schedulers at each output link can achieve a fair distribution of the throughput among all the flows headed to the output link, and may also achieve a fair allocation of the buffer space as in the shared link system shown in Chapter 2. However, unless the exit schedulers coordinate their actions, it is not trivial to achieve fairness or to even define what is fair among the various flows headed to different output links. If the size of the shared buffer is infinity and all arriving data is always accepted into the buffer, fair schedulers at each output link can readily lead to perfect fairness among the various flows. However, in real situations with finite buffers, fair schedulers at the output links cannot guarantee overall fairness since each scheduler only achieves fairness among the flows headed to its output link.

The following are two of the important properties that differentiate a multiple output link system from single output link systems such as those considered in Chapter 2. These differences illustrate the issues and the conceptual difficulties involved in defining fairness in a multiple output link system.

- *No common set of shared resources.* In single output link systems, all the flows share both the buffer and the output link. In multiple output link systems, however, flows headed to two different output links share only the buffer. Thus, some sets of flows share both the buffer and the link, while all others share only the buffer. This presents a significant conceptual difficulty in applying the Principle of Fair Prioritized Resource Allocation (the FPRA principle) among all the flows since the shared

set of resources is not common among all the flows. For example, the cumulative resource dividends and demands cannot be readily compared between flows since the shared set of resources is different for different flows.

- *Non-transferable resource dividends.* In single output link systems, or in the case of scheduling bandwidth on a link, when one flow temporarily has a small demand, other flows with larger demands can take advantage and achieve higher throughputs. Fair allocation, therefore, ends up being a matter of dividing up a certain total amount of the dividend from the shared resources in a fair manner among the flows. In a multiple output link system, however, there is no such thing as a fixed amount of the total dividend from a resource. When one flow has a small demand, other flows headed to a different output link may not be able to take advantage and increase their benefits from resource utilization. Thus, the total dividend available for distribution among the flows is itself a variable quantity, and poses a unique challenge to the task of defining fairness in such a system.

The above challenges, and the fact that many real switch and router architectures can be described using the multiple output link system model, explains the motivation for this study. In this chapter, we present a thorough study of fairness in such systems, and adapt the FPRA principle to exactly define what is fair.

4.1.2 Contributions

The approach used in this study is to decompose the multiple output link system with H output links into two classes of subsystems. The first subsystem, referred to as the *unshared link subsystem*, consists of H integrated flows, or *sessions*. Each session consists of all the flows headed to a particular output link. In this unshared link subsystem, only the buffer is shared among all sessions, thus allowing the application of the FPRA principle with a common set of shared resources. The decomposition of the system also creates H of the

other class of subsystems, referred as the *shared link subsystems*, each corresponding to one session. Flows within each of these subsystems share both the link and the buffer, and thus each shared link subsystem is identical to the system discussed in Chapter 2. The total buffer capacity in these H subsystems is determined by the allocation policy in the unshared link subsystem.

While the shared link subsystems are already studied in Chapter 2 for fairness, this chapter investigates the question of what is fair in the unshared link subsystem. We then base our definition of fairness in the entire system on the definitions of fairness in these subsystems. We subsequently present a method of measuring fairness in buffer allocation in multiple output link systems. Our measure of fairness is based on the same premise used in the definition of the absolute fairness bound (AFB) in the context of scheduling bandwidth on a link [15]. We present an analysis of this measure in the multiple output link system and its relationships with the corresponding measures in its subsystem components. This relationship suggests that achieving fairness in the unshared link subsystem is critically important to achieving fairness in the overall system.

While our presentation in this chapter is primarily based on buffer allocation, the method used and the majority of the concepts and principles developed in this study can also be applied in the allocation of other resources, such as the processing resource, within multiple output link systems.

4.1.3 Organization

The organization of this chapter is as follows. Section 4.2 describes the system model used in this chapter. This section also describes the decomposition of the system model into component subsystems. Section 4.3 discusses what is fair in buffer allocation in a multiple output link system. Section 4.3.1 presents an overview of those results obtained in Chapter 2 that are relevant to the question of what is fair in the shared link subsystems.

Section 4.3.2 tackles the question of what is fair in the unshared link subsystem, and Section 4.3.3 concludes the section with a definition of what is fair in the multiple output link system. Section 4.4 proposes a measure of fairness for use in the study of the fairness properties of real buffer allocation algorithms. Finally, Section 4.5 concludes the chapter with a discussion on the fairness in processor allocation with multiple output links.

4.2 Multiple Output Link System Model

Subsection 4.2.1 presents the multiple output link system model. Subsection 4.2.2 describes a decomposition of this system model into component models that can be more easily analyzed and understood for a study of fairness.

4.2.1 System Model

Our multiple output link system model consists of a shared buffer, a set of output links h, $1 \leq h \leq H$, and a set of flows i, $1 \leq i \leq N$. When $H = 1$, this system model reduces to that considered in Chapter 2. Fig. 4.1(a) illustrates the multiple output link system model.

Let w_i be the weight associated with flow i. Traffic from each flow is destined to one of the H output links, and several flows may share the same output link. Thus, the number of links, H, may be smaller than the number of flows, N. The set of flows headed to the same output link are said to belong to the same *session*. Note that each session corresponds to exactly one link and vice-versa. The session corresponding to the output link h is denoted by $\mathbf{F}_h$. Fig. 4.1(b) shows one session with flows 1 and 2 headed to the same output link H.

Similar to the the single output link system in Chapter 2, let $C(t)$ be the total capacity of the shared buffer at time instant t. The capacity of the buffer is a function of time since the buffer considered here may actually be a dynamically apportioned piece of a larger buffer. Let $R_h(t)$ be the maximum possible transmission rate on link h at time instant t. Note that this transmission rate is also a function of time, to accommodate flow control algorithms

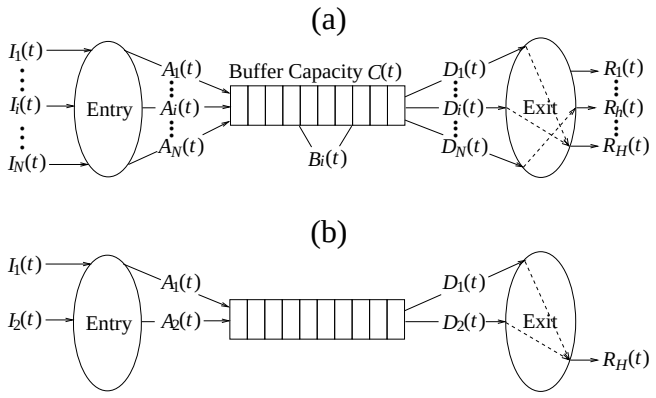

Figure 4.1: The multiple output link system model. (a) The entire system; (b) An example of one session with flows 1 and 2 headed to link H.

that may dynamically limit the rate on the link.

The buffer allocation policy is completely determined by the actions of the *entry* scheduler and the *exit* scheduler, just as in the single output link system model in Chapter 2. The entry scheduler chooses the traffic it will admit into the shared buffer. The entry scheduler, when necessary, also chooses which data to discard from the shared buffer in order to admit newly arriving traffic. The exit scheduler chooses the order in which it will transmit data from the shared buffer on to the output links.

Consider a multiple output link system, denoted S, as shown in Fig. 4.1(a). Let $I_i(t)$ be the input rate of flow i, i.e., the rate at which traffic from flow i seeks to enter the shared buffer at time instant t. Denote by $A_i^{S,q}(t)$ the admission rate of flow i under allocation policy q in system S at time instant t, i.e., the rate at which traffic from flow i is accepted into the shared buffer by the entry scheduler at time instant t. Traffic that is not admitted into the shared buffer is dropped. Note that $A_i^{S,q}(t)$ can be negative such as when some packets from flow i are pushed out from the shared buffer to make room for arriving packets. As in the single output link case, $A_i^{S,q}(t)$ is less than or equal to $I_i(t)$. Let $D_i^{S,q}(t)$ be the departure rate of flow i in system S under the allocation policy q at time instant t, i.e., the rate at

which the exit scheduler dequeues the packets from flow i on to the destined output link at time instant t. Denote by $B_i^{S,q}(t)$, the buffer occupancy or the queue length of flow i under policy q in system S at time instant t. Note that, while the admission rate, buffer occupancy and the departure rates are dependent upon the system and the allocation policy, the input rate, $I_i(t)$ is not. As in the case of a single output link system, the following relationship holds for $t \geq t_0$:

$$B_i^{S,q}(t) = B_i^{S,q}(t_0) + \int_{t_0}^{t} \Big(A_i^{S,q}(\tau) - D_i^{S,q}(\tau) \Big) d\tau. \tag{4.1}$$

Again in this chapter, the sum of a quantity over *all* flows is denoted by dropping the subscript. Thus, $I(t)$ is the sum of input rates of all the flows, and $A^{S,q}(t)$, $D^{S,q}(t)$ and $B^{S,q}(t)$ are defined similarly. Note that at any time instant t,

$$I(t) = \sum_{i=1}^{N} I_i(t).$$

Similar relationships hold for $A^{S,q}(t)$, $D^{S,q}(t)$ and $B^{S,q}(t)$. Also note that at any time instant t, the total buffer occupancy of all flows is less than the buffer capacity, i.e., $B^{S,q}(t) \leq C(t)$.

In all of this chapter, the sum of a per-flow quantity over all flows belonging to the same session is denoted by using the session label as the subscript. For example, $I_{\mathbf{F}_h}(t)$ denotes the aggregate input rate, at time instant t, of all flows that belong to the session $\mathbf{F}_h$. The quantities $A_{\mathbf{F}_h}^{S,q}(t)$, $D_{\mathbf{F}_h}^{S,q}(t)$ and $B_{\mathbf{F}_h}^{S,q}(t)$ are similarly defined. Note that $D_{\mathbf{F}_h}^{S,q}(t) \leq R_h(t)$, that is, the total departure rate of all flows headed to output link h is less than the maximum possible transmission rate on this link at time instant t.

Recall that the buffer allocation is completely determined by the actions of the entry and the exit schedulers, which together determine both the per-flow and per-session admission and departure rates, i.e., $A_i^{S,q}(t)$ and $D_i^{S,q}(t)$, $1 \leq i \leq N$, and $A_{\mathbf{F}_h}^{S,q}(t)$ and $D_{\mathbf{F}_h}^{S,q}(t)$, $1 \leq h \leq H$. Note that the queue length of a flow in the shared buffer is completely determined by the admission rate, the departure rate and the initial queue length, as given by (4.1).

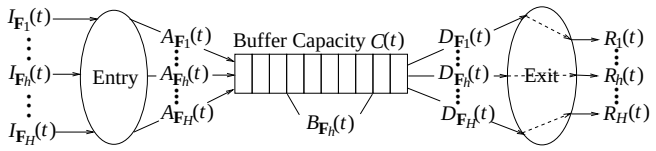

Figure 4.2: The unshared link subsystem, S^u. The number of sessions is equal to the number of output links.

Therefore, a buffer allocation policy over an interval of time is completely specified by the admission and departure rates at all instants during the interval.

4.2.2 System Decomposition

By definition, all flows which belong to the session $\mathbf{F}_h$, are headed to the same output link h. Now, consider this set of flows as one aggregate flow headed to output link h. This session, i.e., the aggregate flow, has an input rate of $I_{\mathbf{F}_h}(t) = \sum_{i \in \mathbf{F}_h} I_i(t)$ at time instant t. Under the allocation policy q, at time instant t, this aggregated flow has an arrival rate equal to $A^{S,q}_{\mathbf{F}_h}(t)$, a departure rate equal to $D^{S,q}_{\mathbf{F}_h}(t)$, and occupies space in the buffer equal to $B^{S,q}_{\mathbf{F}_h}(t)$. A subsystem model can now be defined where each of these H sessions or aggregated flows is treated as a single individual flow. The sessions do not share an output link, but they do share the buffer. We can define a component model of the system S, consisting of these H sessions each treated as a single separate flow. We call this the *unshared link subsystem* model, denoted by S^u, since there is exactly one session headed to each output link, and thus no link is shared among sessions. Fig. 4.2 shows this unshared link subsystem, S^u.

For each of the H links and the session corresponding to it in system S, one can also create a *shared link subsystem* model in which flows share both the buffer and the link. The size of the shared buffer in this shared link subsystem model for link h at any given instant is the buffer occupancy of the session or the aggregate flow $\mathbf{F}_h$ under system model

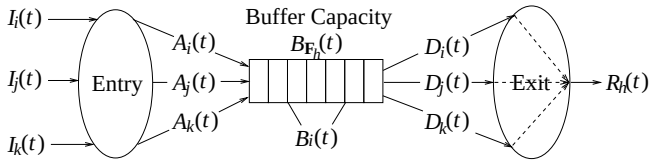

Figure 4.3: Shared link subsystem, S_h^s.

S at that instant. Note that this value of the shared buffer capacity under this component model is a function of time since the total space occupied by a session is always changing. This is part of the reason the capacity of the buffer in Chapter 2 is defined as a function of t. Each of these H shared link subsystems is an instance of the single output link system model presented in Chapter 2, and includes a shared link as well as a shared buffer. Denote the shared link subsystem corresponding to output link h as S_h^s, $1 \leq h \leq H$. Fig. 4.3 illustrates the shared link subsystem S_h^s where flows i, j and k belong to the session $\mathbf{F}_h$. The distribution of the available buffer capacity among various sessions changes with time, and therefore, the capacity allocated to session $\mathbf{F}_h$ is a function of time. This capacity is the total capacity available to all the flows in this shared link subsystem, as shown in the figure.

In subsystem S^u, the only shared resource among the sessions is the shared buffer. In subsystems S_h^s, $1 \leq h \leq H$, the set of shared resources includes both the buffer and the output link h. In this chapter, we proceed to define fairness in system S by defining what is fair in subsystem S^u and in subsystems S_h^s. The condition on the allocation policy, q, for fairness in subsystem S^u can be defined over the sessions, that is, we define what is fair with respect to each session. The condition on q for fairness in subsystems S_h^s can be defined for each of the flows that belong to the session $\mathbf{F}_h$, as in Chapter 2 for the single output link case. The condition on q for fairness in system S is completely defined when the conditions for fairness are defined for each of these component subsystems.

4.3 Fairness in Multiple Output Link Systems

In this section, we define the necessary and sufficient conditions under which a buffer allocation policy in the system model discussed in Section 4.2 can be said to be fair. Based on results derived in Chapter 2, Section 4.3.1 defines what is fair in the shared link subsystems, S_h^s, $1 \leq h \leq H$, followed by Section 4.3.2 which defines what is fair in the unshared link subsystem, S^u. Section 4.3.3 integrates the definitions of fairness in the component subsystems and answers the central question of what is fair in buffer allocation in the multiple output link case.

For the sake of convenience and clarity, in this study we assume that max-min fairness is the notion of fairness, and in addition, the cumulative utility of a flow achieved over an interval of time is just the cumulative throughput the flow receives over this time interval, similar to the single output link case in Section 2.4. The framework established in this chapter can be readily extended to incorporate any other notion of fairness and any generic way of computing flow utilities.

As defined in Chapter 2, the *cumulative resource dividend* of a flow over an interval of time under an allocation policy q represents the benefit the flow gains from the portion of shared resources allocated to it by q. More formally, the cumulative resource dividend of a flow i over a time interval $[t_1, t_2)$ under any system S' and allocation policy q is defined as the difference between the cumulative utilities achieved by flow i under policy q with and without the use of the allocated portion of the shared resource. That is,

$$\begin{aligned} \text{CRDIV}_i^{S',q}(t_1, t_2) &= U_i^{S',q}(t_1, t_2) - U_i^{S',None(i)}(t_1, t_2) \\ &= \int_{t_1}^{t_2} \left\{ D_i^{S',q}(\tau) - D_i^{S',None(i)}(\tau) \right\} d\tau \end{aligned} \tag{4.2}$$

where *None*(i) is an allocation policy that allocates none of the shared resources in system S' to flow i during the interval $[t_1, t_2)$. In this chapter, when the shared buffer is the only shared resource, we sometimes refer to the cumulative resource dividend as the *cumulative buffer dividend*.

The *cumulative resource demand* of a flow over an interval of time is the benefit the flow gains from having all of the shared set of resources exclusively allocated to it during this interval. More formally,

$$\begin{aligned} \text{CRDEM}_i^{S'}(t_1,t_2) &= U_i^{S',All(i)}(t_1,t_2) - U_i^{S',None(i)}(t_1,t_2) \\ &= \int_{t_1}^{t_2} \left\{ D_i^{S',All(i)}(\tau) - D_i^{S',None(i)}(\tau) \right\} d\tau \end{aligned} \tag{4.3}$$

where *All*(i) is an allocation policy that allocates *all* of the shared resources in the system S' exclusively to flow i during the time interval $[t_1, t_2)$.

4.3.1 Fairness in Shared Link Subsystems

Each of the H shared link subsystems, S_h^s, $1 \leq h \leq H$, is a single output link system such as the one considered in Chapter 2. Therefore, using the results in Chapter 2, we can define the cumulative resource demand and dividend for each of these subsystems as follows.

For each flow $i \in \mathbf{F}_h$,

$$\text{CRDIV}_i^{S_h^s,q}(t_1,t_2) = \int_{t_1}^{t_2} D_i^{S,q}(\tau) d\tau. \tag{4.4}$$

The total cumulative resource dividend of the session $\mathbf{F}_h$ is the sum of cumulative resource dividends of each of the flows in $\mathbf{F}_h$, i.e.,

$$\begin{aligned} \text{CRDIV}_{\mathbf{F}_h}^{S_h^s,q}(t_1,t_2) &= \sum_{i \in \mathbf{F}_h} \text{CRDIV}_i^{S_h^s,q}(t_1,t_2) \\ &= \int_{t_1}^{t_2} D_{\mathbf{F}_h}^{S,q}(\tau) d\tau. \end{aligned} \tag{4.5}$$

Recall that the size of the shared buffer in subsystem S_h^s at time instant t is equal to $B_{\mathbf{F}_h}^{S,q}(t)$. The cumulative resource demand of a flow $i \in \mathbf{F}_h$, over an interval of time $[t_1, t_2)$, is given by

$$\text{CRDEM}_i^{S_h^s}(t_1,t_2) = \int_{t_1}^{t_2} D^{S_h^s,All(i)}(\tau) d\tau \tag{4.6}$$

where *All*(i) is an allocation policy that, at all time instants t in the interval $[t_1, t_2)$, allocates the shared buffer with capacity $B^{S,q}_{\mathbf{F}_h}(t)$ and the output link h exclusively to flow i.

In Chapter 2, we define the concept of active and inactive flows for any given interval of time, based on whether or not the flow seeks some buffer space. In other words, an active flow is one that is in competition with other flows to gain some buffer space, or is already occupying some buffer space. An active flow within a shared link subsystem is also active in the overall multiple link system. We define a stationary interval as one during which each flow is either active or inactive, i.e., no flow is neither active nor inactive. Using the definition of the fairness in single output link systems, we can now define fairness in each of the subsystems S^s_h, $1 \leq h \leq H$.

In a shared link subsystem, S^s_h, a buffer allocation policy, q, is max-min fair if and only if, over all stationary intervals of time $[t_1, t_2)$,

$$\begin{aligned}&\left[\text{CRDIV}_i^{S^s_h,q}(t_1,t_2) : i \in \mathbf{F}_h\right] \\ &\quad = \quad \mathcal{F}_{\text{MMF}}\left(\text{CRDIV}_{\mathbf{F}_h}^{S^s_h,q}(t_1,t_2), \left[\text{CRDEM}_i^{S^s_h}(t_1,t_2) : i \in \mathbf{F}_h\right], [w_i : i \in \mathbf{F}_h]\right) \quad (4.7)\end{aligned}$$

where $\mathcal{F}_{\text{MMF}}$ is the notion of max-min fairness for the allocation of a single resource which, given the total resource amount, the demand vector and the weight vector, determines the allocation vector for each flow.

4.3.2 Fairness in an Unshared Link Subsystem

In seeking to apply the fairness principle to the unshared link subsystem model, S^u, we begin with defining the cumulative resource dividend and cumulative resource demand in this system.

Consider the sessions in subsystem S^u, traffic in each of which is headed to a separate output link. The only shared resource among these sessions is the buffer. Let us begin with computing the cumulative resource dividend for session $\mathbf{F}_h$, by first evaluating the dividend under a policy, $None(\mathbf{F}_h)$, that allocates no buffer space to this session. Recall

that a session does not share its output link with any other session, and therefore, all of the output link's bandwidth is available to it, which means the utility of session $\mathbf{F}_h$ under policy $None(\mathbf{F}_h)$ may not be 0. In fact, in the absence of the buffer, the departure rate of session $\mathbf{F}_h$ at time instant t is the minimum of the peak rate of the link h, $R_h(t)$, and the session input rate, $I_{\mathbf{F}_h}$. In other words,

$$D_{\mathbf{F}_h}^{S^u,None(\mathbf{F}_h)}(t) = \min\left\{I_{\mathbf{F}_h}(t), R_h(t)\right\}.$$

Therefore, using (4.2), under an allocation strategy q, the cumulative buffer dividend for the session $\mathbf{F}_h$ over an interval of time $[t_1, t_2)$, is given by

$$\begin{aligned} \text{CRDIV}_{\mathbf{F}_h}^{S^u,q}(t_1,t_2) &= \int_{t_1}^{t_2} \left(D_{\mathbf{F}_h}^{S^u,q}(\tau) - \min\{I_{\mathbf{F}_h}(\tau), R_h(\tau)\}\right) d\tau \\ &= \int_{t_1}^{t_2} \left(D_{\mathbf{F}_h}^{S,q}(\tau) - \min\{I_{\mathbf{F}_h}(\tau), R_h(\tau)\}\right) d\tau. \end{aligned} \tag{4.8}$$

The cumulative buffer dividend summed over all the sessions is given by

$$\begin{aligned} \text{CRDIV}^{S^u,q}(t_1,t_2) &= \sum_{h=1}^{H} \int_{t_1}^{t_2} \left(D_{\mathbf{F}_h}^{S,q}(\tau) - \min\{I_{\mathbf{F}_h}(\tau), R_h(\tau)\}\right) d\tau \\ &= \int_{t_1}^{t_2} \left(D^{S,q}(\tau) - \sum_{h=1}^{H} \min\{I_{\mathbf{F}_h}(\tau), R_h(\tau)\}\right) d\tau. \end{aligned} \tag{4.9}$$

From (4.3), the cumulative resource demand of a session, $\mathbf{F}_h$, over an interval of time $[t_1, t_2)$, is given by

$$\text{CRDEM}_{\mathbf{F}_h}^{S^u}(t_1,t_2) = \int_{t_1}^{t_2} \left(D_{\mathbf{F}_h}^{S^u,All(\mathbf{F}_h)}(\tau) - \min\{I_{\mathbf{F}_h}(\tau), R_h(\tau)\}\right) d\tau \tag{4.10}$$

where $D^{S^u,All(\mathbf{F}_h)}(t)$ is the session departure rate under an allocation strategy, $All(\mathbf{F}_h)$, which grants the entire shared buffer to the session $\mathbf{F}_h$.

A couple of examples below illustrate the concepts of the cumulative resource dividends and demands in the context of the multiple output link system under consideration.

Example 1: Consider a session $\mathbf{F}_h$ with an input rate of $I_{\mathbf{F}_h}(t)$, which is smaller than the peak link rate $R_h(t)$ at all time instants in the interval $[t_1, t_2)$. Also assume that the initial buffer occupancy of this session is 0, i.e., $B_{\mathbf{F}_h}^{S,q}(t_1) = 0$. Since the input rate is always

smaller than the maximum possible link rate, the capacity of the output link h is never completely used up during the time interval $[t_1, t_2)$. In this situation, even under allocation policy $All(\mathbf{F}_h)$, the session $\mathbf{F}_h$ gains no benefit at all from the presence of the buffer. Therefore, the allocation policies $All(\mathbf{F}_h)$ and $None(\mathbf{F}_h)$ result in the same departure rate of session $\mathbf{F}_h$ at all time instants in $[t_1, t_2)$. This yields

$$D_{\mathbf{F}_h}^{S^u, All(\mathbf{F}_h)}(\tau) = D_{\mathbf{F}_h}^{S^u, None(\mathbf{F}_h)}(\tau) = I_{\mathbf{F}_h}(\tau)$$

for all possible τ satisfying $t_1 \leq \tau < t_2$. From the above and from (4.10), it is readily derived that the cumulative resource demand of session $\mathbf{F}_h$ over this interval is equal to 0.

Note that, in this case, it is always true that no allocation policy can accept or transfer more packets than $I_{\mathbf{F}_h}(t)$ at time instant t, and therefore, the cumulative resource dividend cannot be greater than 0. Actually, by definition, the cumulative resource dividend is no more than the cumulative resource demand over any time interval. In fact, the cumulative resource dividend of this session may be negative. This could occur if the allocation policy, q, discards some packets from session $\mathbf{F}_h$ and the resulting throughput under allocation policy q is smaller than that under allocation policy $None(\mathbf{F}_h)$.

Example 2. Consider a session, $\mathbf{F}_h$, with an input rate $I_{\mathbf{F}_h}(t)$ which is greater than or equal to the peak link rate $R_h(t)$ at all time instants in the interval $[t_1, t_2)$. In this case, the maximum possible departure rate is $R_h(t)$, and no allocation policy can achieve a higher departure rate. Thus, from (4.8), the cumulative resource dividend, $\mathrm{CRDIV}_{\mathbf{F}_h}^{S^u, q}(t_1, t_2)$, is 0. Similarly, from (4.10), the cumulative resource demand, $\mathrm{CRDEM}_{\mathbf{F}_h}^{S^u}(t_1, t_2)$ is also 0. This example illustrates that, over a certain interval of time, a session with a very high input rate can actually have a resource demand of 0. This appears to be against our common intuition which tends to presume that a session with an input rate higher than the maximum possible departure rate actually has a higher demand for buffer space. This apparent conflict is resolved by noting two points. Firstly, the cumulative resource demand as defined in this chapter, is not the demand for buffer space but rather a demand for a dividend from the

buffer space. Secondly, the cumulative resource demand is specified only over a certain interval of time. If one considers a larger interval of time than just $[t_1, t_2)$, and if after time t_2 the input rate reduces to 0, the session certainly has a cumulative resource demand greater than 0. This is because, the session's packets can now be stored in the buffer during the interval $[t_1, t_2)$, and transmitted after time t_2 to give it some benefit from the use of the buffer. In general, in the component subsystem S^u, the cumulative buffer demand over a time interval $[t_1, t_2)$ is positive, if and only if, under the allocation policy $All(\mathbf{F}_h)$, during any subinterval in the interval $[t_1, t_2)$, the input rate is smaller than the maximum possible output link rate, while the buffer occupancy is greater than 0. Then, during this subinterval, the allocation policy $All(\mathbf{F}_h)$ can use the packets in the shared buffer to better fill the capacity of the output link, and thus schedule more packets of the session than the allocation policy $None(\mathbf{F}_h)$.

Having determined the dividends and the demands in the context of the subsystem model, S^u, we now proceed to incorporate the timescale by defining the *active* and *inactive* flows and sessions over any given interval of time. As in the case of a single output link system, such a categorization of the flows during each interval is necessary to determine the applicable intervals over which we can apply the FPRA principle. In the case of the unshared link subsystem, S^u, the only shared resource is the shared buffer since the output links are not shared among sessions. Thus, whether or not a session should be considered active during an interval may be based on whether or not the session has a positive cumulative resource demand, which would imply that the session is in competition with other sessions for the shared buffer. In the following, we now present formal definitions of these flow categories.

Definition 15 A session $\mathbf{F}_h$ in an unshared link subsystem S^u is said to be *active* over time interval $[t_1, t_2)$, if and only if, over *each* possible subinterval of time $[\tau_1, \tau_2)$, $t_1 \leq \tau_1 < \tau_2 \leq t_2$, the cumulative resource demand of the session is positive, i.e.,

$$\text{CRDEM}_{\mathbf{F}_h}^{S^u}(\tau_1, \tau_2) > 0.$$

Definition 16 A session $\mathbf{F}_h$ in an unshared link subsystem S^u is said to be *inactive* over an interval of time $[t_1, t_2)$, if and only if, over *each* subinterval of time $[\tau_1, \tau_2)$, $t_1 \leq \tau_1 < \tau_2 \leq t_2$, the cumulative resource demand is zero, i.e.,

$$\text{CRDEM}_{\mathbf{F}_h}^{S^u}(\tau_1, \tau_2) = 0.$$

Note that the above flow categories are defined over an interval, and do not describe the properties of a session at any given instant of time. Thus, as in the definition of active and inactive flows in the case of a single output link system, a session may be neither active nor inactive over a certain interval of time. A session, however, may be defined to be active or inactive at any given instant of time t, based on the above definitions considered over an infinitesimal interval of time $[t, t + \delta t)$.

As described in Chapter 2, the notion of fairness cannot be extended to an interval of time unless none of the sessions changes its category from being active to inactive, or vice versa. This leads us to the following definition of a *stationary interval* only over which one can apply the FPRA principle.

Definition 17 In an unshared link subsystem, a certain interval of time is called a *stationary interval*, if and only if, each session is either active or inactive over this interval, i.e., no session is neither active nor inactive.

Note that any given interval can be broken down into a contiguous sequence of stationary intervals, the boundaries being the instants of time when some session changes from being active to inactive, or vice-versa. The stationary intervals are the *applicable* intervals over which we can apply the FPRA principle.

Note that only the active flows within each session are the ones competing for the shared buffer. Since an inactive flow gains no dividend from the buffer, and therefore, demands

no buffer space, only an active flow is to be considered in the issue of fairness in buffer allocation. Therefore, the weight of a session in the issue of buffer allocation should be the sum of only the active flows within the session. This makes intuitive sense, since a session with just one active flow and many inactive flows should not end up with a large weight, and therefore, an unfairly large share of the buffer space for the only active flow in it. The definition below of what is fair in the unshared link subsystem incorporates this thought.

Let the interval $[t_1, t_2)$ be a stationary interval. Let $\tilde{\mathbf{F}}_h^{S,q}(t_1, t_2)$ denote the set of flows in the session $\mathbf{F}_l$ which are all active over the interval (t_1, t_2) in subsystem S_h^s under policy q. Note that if a flow is active within subsystem S_h^s, it is also active within the overall system S. Therefore, $\tilde{\mathbf{F}}_h^{S,q}(t_1, t_2)$ is the same set as $\tilde{\mathbf{F}}_h^{S_h^s,q}(t_1, t_2)$. Let $w_{\tilde{\mathbf{F}}_h^{S,q}(t_1,t_2)}$, denote the sum of the weights of all the flows that belong to $\tilde{\mathbf{F}}_h^{S,q}(t_1, t_2)$, i.e.,

$$w_{\tilde{\mathbf{F}}_h^{S,q}(t_1,t_2)} = \sum_{i \in \tilde{\mathbf{F}}_h^{S,q}(t_1,t_2)} w_i. \tag{4.11}$$

Definition 18 Consider a buffer allocation policy, q, operating on an unshared link subsystem S^u. q is max-min fair, if and only if, over all stationary intervals of time $[t_1, t_2)$,

$$\left[\text{CRDIV}_{\mathbf{F}_h}^{S^u,q}(t_1, t_2)\right] = \mathcal{F}_{\text{MMF}}\left(\text{CRDIV}^{S^u,q}(t_1, t_2), \left[\text{CRDEM}_{\mathbf{F}_h}^{S^u}(t_1, t_2)\right], \left[w_{\tilde{\mathbf{F}}_h^{S,q}(t_1,t_2)}\right]\right) \tag{4.12}$$

where each of the three vectors, $\left[\text{CRDIV}_{\mathbf{F}_h}^{S^u,q}(t_1, t_2)\right]$, $\left[\text{CRDEM}_{\mathbf{F}_h}^{S^u}(t_1, t_2)\right]$ and $\left[w_{\tilde{\mathbf{F}}_h^{S,q}(t_1,t_2)}\right]$ is of length H with each vector element corresponding to the value of h in the range $1 \leq h \leq H$.

The above definition, as in the definition of fairness in a single output link system, seeks to distribute the cumulative resource dividends in a max-min fair manner with respect to the cumulative resource demands. Note that the above definition computes the cumulative resource demands and dividends for all the sessions and not just the active sessions over the stationary interval under consideration. However, since sessions that are inactive over a stationary interval have cumulative resource demands and dividends of 0 with respect to the buffer, their inclusion does not change the above definition of what is fair.

4.3.3 Fairness in Buffer Allocation

Given the definition of fairness in each of the component subsystems, we can now define fairness in buffer allocation in the multiple output link system, S. Firstly, we wish to be fair in the unshared link subsystem among each of the different sessions. Secondly, we wish to be fair in each of the shared link subsystems among the flows which make up each session. A fair policy is one that is fair in each of the component subsystems. Therefore, the definition of the stationary intervals over which we define fairness for the multiple output link system S, and the definition of the fairness itself, would have to integrate these definitions within the respective subsystems. The following defines the stationary interval in the entire multiple output link system S.

Definition 19 In the multiple output link system S, a certain interval of time is said to be stationary, if and only if, this interval is a stationary interval within each of the component subsystems, S^u and S_h^s, $1 \leq h \leq H$.

It is worthwhile to note that any given interval of time can be divided into a sequence of contiguous stationary intervals. Fairness in the system S can now be defined based on the concept of stationary intervals defined above.

Definition 20 In a multiple output link system S, a buffer allocation policy, q, is max-min fair, if and only if, over all stationary intervals of time $[t_1, t_2)$ in S, q is max-min fair among the sessions in the unshared link component subsystem S^u, and q is max-min fair among the flows in each of the shared link component subsystems S_h^s, $1 \leq h \leq H$.

4.4 A Measure of Fairness

4.4.1 Definitions

We base our measure of fairness on the same premise as in the definition of the absolute fairness bound (AFB) in the context of scheduling bandwidth on a link [15]. The AFB in the

scheduling of bandwidth over a link attempts to capture the maximum possible difference between the service received by a flow under the ideally fair policy, GPS, and that under the scheduling policy being measured. This is similar to the measure of fairness defined in the single output link case discussed in Chapter 2.

To define an absolute fairness bound to measure the fairness of buffer allocation policies, we need to consider a hypothetical scheduling policy that is exactly fair. Since the throughput achieved by a flow is the basis for our judgment on the service received by a flow in the system, we base our measure of fairness on the throughputs achieved by the flows. Now, an ideally fair allocation policy would be one that achieves the ideal distribution of throughputs through an ideal distribution of the dividends with respect to the demands. As also discussed in Section 2.6.1, we wish to compare the fairness of a policy only with an ideally fair policy at the same performance level. Thus, the ideal policy to use in our measure of fairness depends on the system model and the policy being measured. Let $G(S, q)$ be an ideally fair allocation policy in the multiple output link system S, and which delivers exactly the same performance as q. In other words, $G(S, q)$ is exactly fair and the sum of the cumulative throughputs achieved by the flows under $G(S, q)$ is the same as in q. That is,

$$\int_{t_1}^{t_2} D^{S,q}(\tau)d\tau = \int_{t_1}^{t_2} D^{S,G(S,q)}(\tau)d\tau. \tag{4.13}$$

Our measure of fairness can now be based on the difference in the cumulative throughputs achieved by a flow under the fair policy $G(S, q)$ and under the policy q being measured. However, since we wish to be able to compare the fairness characteristics of two different policies at different performance levels, as in Section 2.6.1, we normalize this difference by the total cumulative throughput achieved by the flows under the policy being measured. Our measure of fairness can now be defined as follows.

Definition 21 Given a system S, an allocation policy q and a certain input traffic arrival pattern, the *normalized Absolute Fairness Measure* over an interval of time $[t_1, t_2)$,

$\mathrm{nAFM}^{S,q}(t_1, t_2)$, is defined as follows:

$$\mathrm{nAFM}^{S,q}(t_1, t_2) = \frac{\max\limits_{\forall i} \left| \frac{\int_{t_1}^{t_2} D_i^{S,q}(\tau) d\tau}{w_i} - \frac{\int_{t_1}^{t_2} D_i^{S,G(S,q)}(\tau) d\tau}{w_i} \right|}{\int_{t_1}^{t_2} D^{S,q}(\tau) d\tau}. \tag{4.14}$$

For most real algorithms, the value of the above fairness measure ranges from 0 to 1 depending on the size of the time interval, $t_2 - t_1$, over which the measure is computed. A valid comparison between various allocation algorithms, therefore, can only be made using the above measure if the sizes of the time intervals being considered are identical. In addition, the more unfair an algorithm, the larger the timescales over which it apportions unfair amounts of throughput among the various flows. Thus, to compare two allocation policies, it is more convenient to compare the trend as the length of time interval over which the fairness measure is computed approaches infinity. Therefore, we now define a bound on the above measure of fairness as a function of the size of the time interval, as follows.

Definition 22 Define the *normalized Absolute Fairness Bound*, $\mathrm{nAFB}^{S,q}(\tau)$, of an allocation policy q in system S for time intervals of size τ as the upper bound on the normalized absolute fairness measure over an interval of size τ with any possible input traffic arrival pattern. In other words, $\mathrm{nAFB}^{S,q}(\tau)$ is the smallest value of $\Theta(\tau)$ such that, for any input traffic,

$$\Theta(\tau) \geq \max_{\forall t} \{\mathrm{nAFM}^{S,q}(t, t+\tau)\}.$$

The bound defined above is independent of the input traffic arrival pattern and is, therefore, a property of the allocation policy and the system. The normalized absolute fairness measures defined in this chapter are dependent on the input traffic pattern. We do not indicate this dependence in our notations since the input traffic pattern assumed is almost always obvious from the context. It is worthwhile to note that the above fairness measures and the related bounds are unique to each system, and two allocation policies can be compared using these measures only if they are both operating on the identical systems.

4.4.2 Relationship to Fairness within Component Subsystems

The relationships between the fairness measure in the multiple output link system and its component subsystems, offer unique insights into what is fair in such a system. We begin the definition of a fairness measure in the shared link subsystem. Let $G(S_h^s, q)$ be an ideal allocation policy that yields the same cumulative throughput for session $\mathbf{F}_h$ as the policy q being measured over the interval under consideration. In other words, over an interval of time $[t_1, t_2)$, let $G(S_h^s, q)$ be an ideally fair policy such that,

$$\int_{t_1}^{t_2} D_{\mathbf{F}_h}^{S,q}(\tau)d\tau = \int_{t_1}^{t_2} D_{\mathbf{F}_h}^{S,G(S_h^s,q)}(\tau)d\tau. \tag{4.15}$$

Now, we can define the fairness of the policy q within the shared link subsystem based on a comparison with the policy $G(S_h^s, q)$.

Definition 23 Given the shared link subsystem, S_h^s, an allocation policy q and a certain input traffic arrival pattern, the *normalized Absolute Fairness Measure* over an interval of time $[t_1, t_2)$, $\mathrm{nAFM}^{S_h^s,q}(t_1, t_2)$, is defined as follows:

$$\mathrm{nAFM}^{S_h^s,q}(t_1, t_2) = \frac{\max\limits_{\forall i \in \mathbf{F}_h} \left| \frac{\int_{t_1}^{t_2} D_i^{S,q}(\tau)d\tau}{w_i} - \frac{\int_{t_1}^{t_2} D_i^{S,G(S_h^s,q)}(\tau)d\tau}{w_i} \right|}{\int_{t_1}^{t_2} D_{\mathbf{F}_h}^{S,q}(\tau)d\tau}. \tag{4.16}$$

The *normalized absolute fairness bound* for an interval of length τ, $\mathrm{nAFB}^{S_h^s,q}(\tau)$, is defined as the upper bound on $\mathrm{nAFM}^{S_l^s,q}(t, t+\tau)$ over all t and input traffic arrival patterns.

We now define our fairness measure within the unshared link subsystem. The definition of the measure in the case of the unshared link subsystem is not exactly similar to that in the shared link subsystem or the overall multiple output link system, since the fairness is over sessions rather than over flows. However, our ultimate goal is fairness among flows, and therefore, in our definition of a fairness measure, each session needs to be weighted appropriately by the sum of the weights of all the active flows within it.

Recall that $\tilde{\mathbf{F}}_h^{S,q}(t_1, t_2)$ is the set of flows in the session $\mathbf{F}_h$ which are all active over the interval $[t_1, t_2)$ in system S under policy q. Thus, $w_{\tilde{\mathbf{F}}_h^{S,q}(t_1,t_2)}$ is the sum of the weights

of all the flows that belong to this set of flows. These quantities can be defined at any given instant of time as well, based on the actions of the policy and the system over an infinitesimal interval of time since the instant under consideration. Define $\tilde{\mathbf{F}}_h^{S,q}(t)$ as the set of flows in the session $\mathbf{F}_h$ which are all active over the infinitesimal interval $[t, t+\delta t)$, and $w_{\tilde{\mathbf{F}}_h^{S,q}(t)}$ is the sum of the weights of all the flows that belong to the set $\tilde{\mathbf{F}}_h^{S,q}(t)$. Only flows that are active at any given instant of time should be considered in the total weight of the session at that time instant. The measure of fairness in the unshared link subsystem can now be defined as follows.

Definition 24 Given the unshared link subsystem, S^u, an allocation policy q and a certain input traffic arrival pattern, the *normalized Absolute Fairness Measure* over an interval of time $[t_1, t_2)$, $\mathrm{nAFM}^{S^u,q}(t_1, t_2)$, is defined as follows:

$$\mathrm{nAFM}^{S^u,q}(t_1,t_2) = \frac{\max\limits_{1\le h\le H}\left|\int_{t_1}^{t_2}\frac{D_{\mathbf{F}_h}^{S,q}(\tau)}{w_{\tilde{\mathbf{F}}_h^{S,q}(\tau)}}d\tau - \int_{t_1}^{t_2}\frac{D_{\mathbf{F}_h}^{S,G(S^u,q)}(\tau)}{w_{\tilde{\mathbf{F}}_h^{S,q}(\tau)}}d\tau\right|}{\int_{t_1}^{t_2} D^{S,q}(\tau)d\tau}. \tag{4.17}$$

The *normalized absolute fairness bound* for an interval of length τ, $\mathrm{nAFB}^{S^u,q}(\tau)$, is defined as the upper bound on $\mathrm{nAFM}^{S^u,q}(t, t+\tau)$ over all t and input traffic arrival patterns.

The relationships between the ideal allocation policies used in the overall system and its component subsystems in the computation of the fairness measures within these systems, reveals insights into the relationships between the fairness measures of a policy in these systems. The ideal allocation policy in the shared link subsystem, $G(S_h^s, q)$, only fairly allocates the cumulative throughputs (same as cumulative resource dividends) of flows in each shared link subsystem; the ideal allocation policy in the unshared link subsystem, $G(S^u, q)$, fairly allocates the cumulative resource dividends with respect to the shared buffer; and the ideal allocation policy $G(S, q)$ in the multiple output link system has to be fair in both shared and unshared link subsystems. Let $\{(S, q)\}$, $\{(S^u, q)\}$ and $\{(S_h^s, q)\}$, respectively, denote the set of ideal allocation policies that may be used in the computation of the fairness measures in the overall system S, the unshared link subsystem S^u and the shared link

subsystem S_h^s for $1 \leq h \leq H$.

Let G be an ideally fair policy in the overall system, S, yielding the same overall performance as q. Now, by definition G is also fair within the unshared link subsystem, and in addition, the overall performance delivered by G in the unshared link subsystem is also the same as q. Thus, if $G \in \{(S, q)\}$, then $G \in \{(S^u, q)\}$. Thus,

$$\{(S, q)\} \subseteq \{(S^u, q)\}. \tag{4.18}$$

The converse, however, is not true since each policy which is fair in the unshared link subsystem cannot be guaranteed to be fair in the entire system.

It is not possible, however, to establish any particular relationship between $\{(S, q)\}$ and $\{(S_h^s, q)\}$. As before, let $G \in \{(S, q)\}$. Now, it is not necessarily true that G and q will yield the same throughputs for each of the sessions, although the sum of the cumulative throughputs will be the same by definition of G. Thus, policies in $\{(S_h^s, q)\}$ will not necessarily yield the same throughput for session $\mathbf{F}_h$ as policy G, although by definition, they will yield the same throughput for the session as policy q. Thus, there does not exist a trivial relationship such as in (4.18) between $\{(S, q)\}$ and $\{(S_h^s, q)\}$.

The above lack of a relationship between $\{(S, q)\}$ and $\{(S_h^s, q)\}$ suggests that the fairness measure of the overall system cannot be derived based on the fairness measures of the component subsystems. However, over any interval $[t_1, t_2)$ in which all flows are active, one can prove that the normalized absolute fairness measure under a policy in the multiple output link system is greater than the normalized absolute fairness measure of the policy in the unshared link subsystem. This is proved next, and it suggests that achieving fairness in the unshared link subsystem is critically important to achieving fairness in the overall system.

Theorem 6 If all flows are active over a time interval $[t_1, t_2)$, then the normalized absolute fairness measure of an allocation policy, q, in a multiple output link system, S, is no less

than the normalized absolute fairness measure in the unshared link subsystem S^u over the interval. That is, if all flows are active over $[t_1, t_2)$,

$$\mathrm{nAFM}^{S,q}(t_1, t_2) \geq \mathrm{nAFM}^{S^u,q}(t_1, t_2), \tag{4.19}$$

Proof: The relationship in (4.18) allows us to redefine the normalized absolute fairness measure in the unshared link subsystem S^u, $\mathrm{nAFM}^{S^u,q}(t_1, t_2)$, by substituting (S^u, q) by (S, q) in (4.17). Thus,

$$\mathrm{nAFM}^{S^u,q}(t_1, t_2) = \frac{\max\limits_{1\leq h\leq H} \left| \int_{t_1}^{t_2} \frac{D^{S,q}_{\mathbf{F}_h}(\tau)}{w_{\tilde{\mathbf{F}}_h^{S,q}(\tau)}} d\tau - \int_{t_1}^{t_2} \frac{D^{S,G(S,q)}_{\mathbf{F}_h}(\tau)}{w_{\tilde{\mathbf{F}}_h^{S,q}(\tau)}} d\tau \right|}{\int_{t_1}^{t_2} D^{S,q}(\tau) d\tau}. \tag{4.20}$$

Note that the denominators in (4.14) and (4.20) are identical and equal to the total sum of the cumulative throughputs of all the flows. Denote this quantity by Δ.

Without loss of generality, assume that the right hand side of (4.20) reaches its maximum value for some $h = \lambda$. Also, since all the flows are active over time interval $[t_1, t_2)$, the weight of session $\mathbf{F}_\lambda$, $w_{\tilde{\mathbf{F}}_\lambda^{S,q}(t)}$, is equal to the sum of weights of all flows belonging to the session $\mathbf{F}_\lambda$, i.e., a constant for all t during the entire interval $[t_1, t_2)$. Denote this constant by $w_{\mathbf{F}_\lambda}$, to indicate that it does not depend on t, i.e.,

$$w_{\mathbf{F}_\lambda} = \sum_{i\in\mathbf{F}_\lambda} w_i.$$

Now in (4.20), this constant can be taken outside the integration operators. Thus, we have

$$\begin{aligned}
\mathrm{nAFM}^{S^u,q}(t_1, t_2) &= \frac{1}{\Delta} \left| \frac{\int_{t_1}^{t_2} D^{S,q}_{\mathbf{F}_\lambda}(\tau) d\tau}{w_{\mathbf{F}_\lambda}} - \frac{\int_{t_1}^{t_2} D^{S,G(S,q)}_{\mathbf{F}_\lambda}(\tau) d\tau}{w_{\mathbf{F}_\lambda}} \right| \\
&= \frac{1}{\Delta} \left| \frac{\int_{t_1}^{t_2} \left(D^{S,q}_{\mathbf{F}_\lambda}(\tau) - D^{S,G(S,q)}_{\mathbf{F}_\lambda}(\tau) \right) d\tau}{w_{\mathbf{F}_\lambda}} \right| \\
&= \frac{1}{\Delta} \left| \frac{\int_{t_1}^{t_2} \sum_{i\in\mathbf{F}_\lambda} \left(D^{S,q}_{i}(\tau) - D^{S,G(S,q)}_{i}(\tau) \right) d\tau}{w_{\mathbf{F}_\lambda}} \right| \\
&= \frac{1}{\Delta} \left| \sum_{i\in\mathbf{F}_\lambda} \frac{\int_{t_1}^{t_2} \left(D^{S,q}_{i}(\tau) - D^{S,G(S,q)}_{i}(\tau) \right) d\tau}{w_{\mathbf{F}_\lambda}} \right|
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{\Delta}\left|\sum_{i\in\mathbf{F}_\lambda}\frac{w_i}{w_{\mathbf{F}_\lambda}}\frac{\int_{t_1}^{t_2}\left(D_i^{S,q}(\tau)-D_i^{S,G(S,q)}(\tau)\right)d\tau}{w_i}\right| \\
&\le \frac{1}{\Delta}\sum_{i\in\mathbf{F}_\lambda}\frac{w_i}{w_{\mathbf{F}_\lambda}}\left|\frac{\int_{t_1}^{t_2}\left(D_i^{S,q}(\tau)-D_i^{S,G(S,q)}(\tau)\right)d\tau}{w_i}\right| \\
&\le \frac{1}{\Delta}\sum_{i\in\mathbf{F}_\lambda}\frac{w_i}{w_{\mathbf{F}_\lambda}}\max_{\forall j}\left|\frac{\int_{t_1}^{t_2}\left(D_j^{S,q}(\tau)-D_j^{S,G(S,q)}(\tau)\right)d\tau}{w_j}\right| \\
&= \frac{1}{\Delta}\max_{\forall j}\left|\frac{\int_{t_1}^{t_2}\left(D_j^{S,q}(\tau)-D_j^{S,G(S,q)}(\tau)\right)d\tau}{w_j}\right| \\
&= \frac{1}{\Delta}\max_{\forall j}\left|\frac{\int_{t_1}^{t_2}D_j^{S,q}(\tau)d\tau}{w_j}-\frac{\int_{t_1}^{t_2}D_j^{S,G(S,q)}(\tau)d\tau}{w_j}\right| \\
&= \mathrm{nAFM}^{S,q}(t_1,t_2). \qquad (4.21)
\end{aligned}
$$

■

4.5 Allocation of Processing Resource

A system with a shared processor and multiple output links may be similarly decomposed into several shared link subsystems and an unshared link subsystem. Each shared link subsystem is also associated with an output link, and all flows headed to this link comprise a session. The set of shared resources among all flows in each session includes both the processor and the corresponding output link. On the other hand, the unshared link subsystem consists of all sessions, sharing only the processor.

First consider the fairness in the unshared link subsystem. In this system, there is only one shared resource, the processor, which is essential. In other words, one does not need to worry about the concept of stationary intervals since all sessions are always active with respect to the processing resource as long as they are backlogged. In addition, the fact that the processing resource is essential also simplifies the problem into the fair allocation of a single resource, which has been extensively studied. Let $\mathbf{F}_h$ be the session headed to output link h, and $w_{\mathbf{F}_h}$ be the aggregate weight of this session. Denote by $d_{\mathbf{F}_h,P}$ the demand for resource P from session $\mathbf{F}_h$, and by $a^q_{\mathbf{F}_h,P}$ the allocation of resource P from session $\mathbf{F}_h$

under policy q. The normalized demand and the normalized allocation in the unshared link subsystem can be defined as follows:

$$\tilde{d}_{\mathbf{F}_h,P} = \frac{d_{\mathbf{F}_h,P}}{R_P}$$

$$\tilde{a}^q_{\mathbf{F}_h,P} = \frac{a^q_{\mathbf{F}_h,P}}{R_P}.$$

Here R_P is the amount of resource P.

An allocation policy q is fair in the unshared link subsystem, if and only if,

$$\left[\tilde{a}^q_{\mathbf{F}_h,P}\right] = \mathcal{F}\left(C, \left[\tilde{d}_{\mathbf{F}_h,P}\right], [w_{\mathbf{F}_h}]\right)$$

where C is some constraint as described in Section 3.3.

Note that in each shared link subsystem, the total amount of processing resource is actually determined by the allocation in the unshared link subsystem, in the same sense that, in buffer allocation, the total amount of buffer resource in the shared link subsystem S^s_h is determined by the allocation policy in the unshared link subsystem S^u. Therefore, in the shared link subsystem S^s_h, the normalized demand for resource P of each flow i, $i \in \mathbf{F}_h$, is given by

$$\tilde{d}_{i,P} = \frac{d_{i,P}}{a^q_{\mathbf{F}_h,P}}$$

and similarly the corresponding normalized allocation is

$$\tilde{a}^q_{i,P} = \frac{a^q_{i,P}}{a^q_{\mathbf{F}_h,P}}.$$

On the other hand, the normalized demand and allocation for bandwidth resource of each flow in the shared link subsystem can be simply computed by normalization over the amount of link resource, i.e.,

$$\tilde{d}_{i,L_h} = \frac{d_{i,L_h}}{R_{L_h}}$$

$$\tilde{a}^q_{i,L_h} = \frac{a^q_{i,L_h}}{R_{L_h}}$$

where R_{L_h} is the amount of resource L on link h.

After these normalized allocations of each flow i in the shared link subsystem S_h^s have been determined, the prime resource of each flow i under policy q, $\mathcal{B}_i^q$, can be defined in the same way as described in Chapter 3. That is,

$$\mathcal{B}_i^q = \arg_{P,L_h} \max \left\{ \tilde{a}_{i,P}^q, \tilde{a}_{i,L_h}^q \right\}$$

or,

$$\tilde{a}_{i,\mathcal{B}_i^q}^q = \max \left\{ \tilde{a}_{i,P}^q, \tilde{a}_{i,L_h}^q \right\}.$$

Using the results from Chapter 3, the fairness in each shared link subsystem can now be readily defined.

Finally, as in the case of buffer allocation, a policy is said to be fair in the allocation of processing resource in the multiple output link system if and only if it is fair in the unshared link subsystem and also in each of the shared link subsystems.

Chapter 5. Conclusion

5.1 Summary

Fairness is an intuitively desirable property in the allocation of resources in a network shared among multiple flows of traffic from different users. During the last decade or two, research on achieving fairness in networks has primarily focused on the allocation of bandwidth. As flows of traffic traverse a network, however, they share various types of network resources such as buffer, processor and power as in mobile systems. A framework based on which one can define fairness in allocation of multiple resources has not yet been established.

In this dissertation, we investigate the challenge of achieving fairness in the joint allocation of multiple heterogeneous resources. We generally categorize the systems with multiple resources into two groups: those with *prioritized* resources such as the system with a shared buffer and a shared link, and those with *essential* resources such as the system with a shared processor and a shared link. For both types of systems, we have established fundamental principles to define and measure the fairness in the joint allocation of the shared resources within the system under consideration. These principles, namely the *Principle of Fair Prioritized Resource Allocation* or the FPRA principle and the *Principle of Fair Essential Resource Allocation* or the FERA principle, are simple but powerful generalizations of any given notion of fairness defined on a single shared resource, such as max-min fairness, proportional fairness and utility max-min fairness.

We further apply the FPRA principle to the system with a shared buffer and a shared link, and apply the FERA principle to the system with a shared processor and a shared link. Using the notion of max-min fairness as an example, we have developed ideally fair, though unimplementable, allocation strategies in both systems, i.e., the *Fluid-flow*

Fair Buffering (FFB) and the *Fluid-flow Processor and Link Sharing (FPLS)*, which may be used as benchmarks in the evaluation of the fairness of various practical and implementable allocation schemes in such systems. We anticipate that these algorithms will serve the same purpose as GPS does in research studies on the allocation strategies of a single shared resource. We have presented the *Packet-by-packet Fair Buffering (PFB)* and the *Packet-by-packet Processor and Link Sharing (PPLS)*, each of which is an implementable, computationally feasible and *provably fair* approximation of the corresponding ideally fair strategy. We have demonstrated the fairness of PFB and PPLS through extensive simulation experiments using real traffic traces.

Our study in the joint allocation of buffer and bandwidth resources shows that overall fairness is not determined by the exit scheduler alone, but instead by the combination of the entry and the exit policies. This work reveals that use of a fair exit policy such as DRR does not ensure overall fairness when buffer resources are constrained or when packet dropping is used as a congestion control policy. In fact, our study shows that, even though the fairness of exit policies has received far greater attention in the research literature, the entry policy is more critical to overall fairness than the exit policy.

Our study in the joint allocation of processing and bandwidth resources also leads to a similar conclusion. It is illustrated that, in a system with multiple essential resources, achieving fairness with respect to each resource alone does not guarantee the overall fairness in the entire system. In fact, neither the fair allocation of processing resource alone nor the fair allocation of bandwidth resource alone achieves the fair allocation in the overall system with a shared processor and a shared link. Only when these resources are allocated in a coordinated manner, does the allocation strategy such as PPLS provide overall fairness.

In addition, in order to extend our work to systems with multiple output links, an approach based on system decomposition is also presented in this dissertation. In this method, we decompose a multiple output link system with H output links into two types of subsystems: an *unshared link subsystem* and H *shared link subsystems*. Consider a system with

a shared buffer and a shared link as an example. The unshared link subsystem consists of *H sessions*, each of which contains all flows headed to the same output link, and corresponds to a shared link subsystem associated with the output link. In the unshared link subsystem, the only shared resource is the buffer; in each shared link subsystem, the set of shared resources includes both the buffer and the corresponding link. Therefore, the principles developed in the study of single output link systems can be applied into each of these subsystems, and the fairness in the entire system can be defined based on the fairness in each subsystem.

5.2 Concluding Remarks and Future Work

The Random Early Detection (RED) algorithm has been widely employed in Internet routers to cooperate with TCP end users as a congestion avoidance strategy. The end-to-end congestion control algorithms, implemented in various versions of TCP protocol, depend not only on the bandwidth allocation in the network but also on the packet loss rate as the indication of congestion. Therefore, an unfair management of buffers can cause biased packet loss rates for different flows, and thus lead to a failure in providing end-to-end fairness. It has been shown in Chapter 2 that RED and its variants fail to provide a fair buffer management. Incorporating the principles developed in this dissertation in the design of enhancements to RED-like algorithms for congestion avoidance will likely lead to true fairness in resource allocation as well as an overall improvement in the utilization of the network resources.

Countering denial-of-service (DoS) attacks has become one of the most critical challenges in network security today. In packet flooding, the most common type of DoS attacks, innumerable malicious packets from attackers overwhelm the victim by causing congestion on its resources such as bandwidth, processors and TCP/UDP ports, and thus make the resources unavailable for normal flows. More secure operating systems may help avoid DoS

attacks, but only with limited effectiveness due to the considerable number of computers (i.e., potential attackers when compromised) in the current Internet. An alternative defense is to prevent DoS attacks from significantly impairing system performance, through fair resource allocation [67, 68]. Even in the presence of a DoS attack, fair resource allocation guarantees that malicious packets cannot occupy more system resources than a certain amount, thus making service to normal flows unaffected, or minimally affected. It has been shown in Chapter 3 that the PPLS algorithm can achieve a fair allocation of the processing and bandwidth resources, and therefore, minimize the impact of a DoS attack based on excessive use of either the bandwidth or the processing resource. Furthermore, PPLS-like algorithms based on other contexts of resources can be designed for defending DoS attacks on other resources.

This dissertation has primarily focused on the joint allocation of buffer and bandwidth resources, and that of processing and bandwidth resources. However the principles and algorithms developed in this dissertation can be applied to a variety of contexts in communication networks and operating systems design. A few examples are the resources of CPU, I/O, and memory access in operating systems, and the resources of bandwidth and power in mobile systems.

Take the joint allocation of processor, link and power resources in a wireless system as an example. It has already been recognized that these resources need to be fairly allocated, especially in wireless ad hoc networks and wireless sensor networks. Applying the FERA principle, one may implement a fair allocation policy similar to the PPLS algorithm. Specifically, for each flow, three quanta and three deficit counters are needed, each corresponding to one resource. The basis of the algorithm, however, remains the same, i.e., a packet from a flow can be scheduled only if all three deficit counters of this flow are large enough. By this means, it can be guaranteed that each flow receives a fair share of processing, bandwidth and power resources, if the quantum values are appropriately assigned.

The allocation of storage resources in operating systems is another example where our

principles can be applied. For example, first level cache, second level cache, main memory and disk comprise a storage system with ordered preference. Applying the FPRA principle, one may define the cumulative resource dividends and demands with respect to each resource, the stationary intervals, and then the fairness in resource allocation in such a system. A PFB-like algorithm can also be implemented in such systems as a practical strategy for fair allocation of these resources.

Quality-of-service is often an end-to-end issue, of which end-to-end fairness is a critically important piece. Fair bandwidth allocation algorithm such as Weighted Fair Queueing (WFQ) have frequently been used as a component of an overall mechanism that ensures end-to-end delay guarantees. Similarly, mechanisms to achieve end-to-end fairness in the use of all the resources in the network will play a significant role in achieving true end-to-end quality-of-service guarantees. This dissertation focuses on the principles and the design of such mechanisms, serving as the basis for achieving end-to-end fairness.

This dissertation is the first attempt to develop theoretical frameworks to define fairness in the allocation of multiple resources. While this work has established a foundation for achieving this goal, it also raises many other possibilities for further investigation.

In this dissertation, we have defined the fairness in systems with multiple output links by using system decomposition, as shown in Chapter 4. A fair allocation policy, which can achieve fairness in such systems according to the definition, is still unknown. Consider buffer allocation in multiple link systems. As one can observe from the fairness definition, the fair allocation policy needs to implement a PFB-like algorithm for each shared link subsystem. In addition, for the unshared link subsystem, the fair allocation policy has to be able to take into consideration the different peak rates of all output links, and achieve a fair distribution of resource dividends among sessions.

While we have defined in this dissertation the fairness in the joint allocation of multiple prioritized resources and in the joint allocation of multiple essential resources, a more complicated system may consist of both prioritized and essential resources. One example

is a system with a shared processor, a shared buffer, and a shared link, similar to the model used in Fig. 3.2, except that the buffer between the processor P and the link L has a finite capacity. Therefore, in this system, flows compete for all three shared resources, and both prioritized and essential resources exist. Specifically, the shared buffer and the shared link comprise a subsystem with two prioritized resources, while this subsystem and the processor P are both essential to all flows. A future research goal is to develop a definition of fairness for such a system, potentially using the principles proposed in this dissertation. Given that most switches and routers have both prioritized and essential resources, it will be worthwhile to also develop practical strategies for resource allocation in such systems.

It is our hope that this dissertation will facilitate future research in the design of *provably* fair strategies for achieving *overall* fairness in *joint* resource allocation.

Bibliography

[1] F. A. Cowell, *Measuring Inequality: Techniques for the Social Sciences*, John Wiley & Sons, New York, NY, 1977.

[2] W. Stallings, *Operating Systems: Internals and Design Principles*, Prentice Hall, Upper Saddle River, NJ, 3rd edition, 1995.

[3] A. Silberschatz and P. Galvin, *Operating System Concepts*, John Wiley & Sons, New York, NY, 5th edition, 1997.

[4] L. Kleinrock, *Queueing System*, vol. 2, Computer Applications, John Wiley & Sons, New York, NY, 1976.

[5] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *Proc. ACM SIGCOMM*, Austin, TX, Sep. 1989, pp. 1–12.

[6] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated service networks – the single node case," in *Proc. IEEE INFOCOM*, Florence, Italy, May 1992, pp. 915–924.

[7] S. J. Golestani, "A self-clocked fair queueing scheme for broadband application," in *Proc. IEEE INFOCOM*, Toronto, Canada, Jun. 1994, pp. 636–646.

[8] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round-robin," *IEEE/ACM Trans. Networking*, vol. 4, no. 3, pp. 375–385, Jun. 1996.

[9] J. C. R. Bennett and H. Zhang, "WF^2Q: Worst-case fair weighted fair queueing," in *Proc. IEEE INFOCOM*, San Francisco, CA, Mar. 1996, pp. 120–128.

[10] S. S. Kanhere, H. Sethu, and A. B. Parekh, "Fair and efficient packet scheduling using elastic round robin," *IEEE Trans. Parall. Distr. Syst.*, vol. 13, no. 3, pp. 324–336, Mar. 2002.

[11] D. C. Stephens, J. C. R. Bennett, and H. Zhang, "Implementing scheduling algorithms in high-speed networks," *IEEE J. Select. Areas Commun.*, vol. 17, no. 6, pp. 1145–1158, Jun. 1999.

[12] Cisco Systems Inc., "Cisco 12016 gigabit switch router: Application note," 1999.

[13] V. Rodoplu and T. H. Meng, "Minimum energy mobile wireless networks," *IEEE J. Select. Area Commun.*, vol. 17, no. 8, pp. 1333–1344, Aug. 1999.

[14] D. P. Bertsekas and R. Gallager, *Data Networks*, Prentice Hall, Upper Saddle River, NJ, 2nd edition, 1991.

[15] S. Keshav, *An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network*, Addison-Wesley, Reading, MA, 1997.

[16] Z. Cao and E. W. Zegura, "Utility max-min: An application-oriented bandwidth allocation scheme," in *Proc. IEEE INFOCOM*, New York, NY, Mar. 1999, pp. 793–801.

[17] S. Shenker, "Fundamental design issues for the future Internet," *IEEE J. Select. Areas Commun.*, vol. 13, no. 7, pp. 1176–1188, Sep. 1995.

[18] F. Kelly, "Charging and rate control for elastic traffic," *Europ. Trans. Telecom.*, vol. 8, no. 1, pp. 33–37, Jan. 1997.

[19] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," Dec. 1998, IETF RFC 2475, http://www.ietf.org/rfc/rfc2475.txt.

[20] Y. Zhou and H. Sethu, "On the relationship between absolute and relative fairness bounds," *IEEE Commun. Lett.*, vol. 6, no. 1, pp. 37–39, Jan. 2002.

[21] A. G. Greenberg and N. Madras, "How fair is fair queuing?," *J. ACM*, vol. 39, no. 3, pp. 568–598, Jul. 1992.

[22] S. Keshav, "On the efficient implementation of fair queueing," *J. Internetworking Research and Experience*, vol. 2, no. 3, pp. 157–173, Sep. 1991.

[23] J. M. Blanquer and B. Özden, "Fair queuing for aggregated multiple links," in *Proc. ACM SIGCOMM*, San Diego, CA, Aug. 2001, pp. 189–197.

[24] V. Raghunathan, S. Ganeriwal, C. Schurgers, and M. Srivastava, "E^2WFQ: An energy efficient fair scheduling policy for wireless systems," in *Proc. Int. Symp. Low Power Electr. Design*, Monterey, CA, Aug. 2002, pp. 30–35.

[25] A. Elwalid, D. Mitra, and R. H. Wentworth, "A new approach for allocating buffers and bandwidth to heterogeneous, regulated traffic in an ATM node," *IEEE J. Select. Areas Commun.*, vol. 13, no. 6, pp. 1115–1127, Aug. 1995.

[26] R. Guérin, S. Kamat, V. Peris, and R. Rajan, "Scalable QoS provision through buffer management," in *Proc. ACM SIGCOMM*, Vancouver, Canada, Aug. 1998, pp. 29–40.

[27] F. Lo Presti, Z.-L. Zhang, J. Kurose, and D. Towsley, "Source time scale and optimal buffer/bandwidth tradeoff for heterogeneous regulated traffic in a network node," *IEEE/ACM Trans. Networking*, vol. 7, no. 4, pp. 490–501, Aug. 1999.

[28] S. H. Low, "Equilibrium bandwidth and buffer allocations for elastic traffics," *IEEE/ACM Trans. Networking*, vol. 8, no. 3, pp. 373–383, Jun. 2000.

[29] G. Fayolle, A. de La Fortelle, J.-M. Lasgouttes, L. Massoulié, and J. Roberts, "Best-effort networks: Modeling and performance analysis via large networks asymptotics," in *Proc. IEEE INFOCOM*, Anchorage, AK, Apr. 2001, pp. 709–716.

[30] F. Kamoun and L. Kleinrock, "Analysis of shared finite storage in a computer network node environment under general traffic," *IEEE Trans. Commun.*, vol. COM-28, no. 7, pp. 992–1003, Jul. 1980.

[31] D. Tipper and M. K. Sundareshan, "Adaptive policies for optimal buffer management in dynamic load environments," in *Proc. IEEE INFOCOM*, New Orleans, LA, Mar. 1988, pp. 535–544.

[32] K. Kumaran and D. Mitra, "Performance and fluid simulations of a novel shared buffer management system," in *Proc. IEEE INFOCOM*, San Francisco, CA, Mar. 1998, pp. 1449–1461.

[33] A. Gupta and D. Ferrari, "Resource partitioning for real-time communication," *IEEE/ACM Trans. Networking*, vol. 3, no. 5, pp. 501–508, Aug. 1995.

[34] G.-L. Wu and J. W. Mark, "A buffer allocation scheme for ATM networks: Complete sharing based on virtual partition," *IEEE/ACM Trans. Networking*, vol. 3, no. 6, pp. 660–670, Dec. 1995.

[35] L. Tassiulas, Y. C. Hung, and S. S. Panwar, "Optimal buffer control during congestion in an ATM network node," *IEEE/ACM Trans. Networking*, vol. 2, no. 4, pp. 374–386, Aug. 1994.

[36] I. Cidon, L. Georgiadis, R. Guérin, and A. Khamisy, "Optimal buffer sharing," in *Proc. IEEE INFOCOM*, Boston, MA, Apr. 1995, pp. 24–31.

[37] S. Sharma and Y. Viniotis, "Optimal buffer management policies for shared-buffer ATM switches," *IEEE/ACM Trans. Networking*, vol. 7, no. 4, pp. 575–587, Aug. 1999.

[38] A. K. Choudhury and E. L. Hahne, "Dynamic queue length thresholds for shared-memory packet switches," *IEEE/ACM Trans. Networking*, vol. 6, no. 2, pp. 130–140, Apr. 1998.

[39] R. Fan, A. Ishii, B. Mark, G. Ramamurthy, and Q. Ren, "An optimal buffer management scheme with dynamic thresholds," in *Proc. IEEE GLOBECOM*, Rio de Janeiro, Brazil, Dec. 1999, pp. 631–637.

[40] S. Krishnan, A. K. Choudhury, and F. M. Chiussi, "Dynamic partitioning: A mechanism for shared memory management," in *Proc. IEEE INFOCOM*, New York, NY, Mar. 1999, pp. 144–152.

[41] J. Heinanen and K. Kilkki, "A fair buffer allocation scheme," *Comput. Commun.*, vol. 21, no. 3, pp. 220–226, Mar. 1998.

[42] K. Kilkki, *Differentiated Services for the Internet*, Addison-Wesley, Reading, MA, 1999.

[43] O. Bonaventure and J. Nelissen, "Guaranteed frame rate: A better service for TCP/IP in ATM networks," *IEEE Network*, vol. 15, no. 1, pp. 46–54, Jan./Feb. 2001.

[44] E. Hashem, "Analysis of random drop for gateway congestion control," Tech. Rep. LCS TR-465, Lab. for Computer Science, MIT, Cambridge, MA, 1989.

[45] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

[46] A. R. Bonde, Jr. and S. Ghosh, "A comparative study of fuzzy versus "fixed" thresholds for robust queue management in cell-switching networks," *IEEE/ACM Trans. Networking*, vol. 2, no. 4, pp. 337–344, Aug. 1994.

[47] D. Lin and R. Morris, "Dynamics of random early detection," in *Proc. ACM SIGCOMM*, Cannes, France, Sep. 1997, pp. 127–137.

[48] W.-J. Kim and B. G. Lee, "The FB-RED algorithm for TCP over ATM," in *Proc. IEEE GLOBECOM*, Sydney, Australia, Nov. 1998, pp. 551–555.

[49] J. Bruno, B. Özden, A. Silberschatz, and H. Saran, "Early fair drop: A new buffer management policy," in *Proc. SPIE: Multimedia Comput. & Networking*, San Jose, CA, Jan. 1999, pp. 148–161.

[50] R. Pan, B. Prabhakar, and K. Psounis, "CHOKe: A stateless active queue management scheme for approximating fair bandwidth allocation," in *Proc. IEEE INFOCOM*, Tel-Aviv, Israel, Mar. 2000, pp. 942–951.

[51] K. K. Ramakrishnan and R. Jain, "A binary feedback scheme for congestion avoidance in computer networks with a connectionless network layer," in *Proc. ACM SIGCOMM*, Stanford, CA, Aug. 1988, pp. 303–313.

[52] D. Stiliadis and A. Varma, "Efficient fair queueing algorithms for packet-switched networks," *IEEE/ACM Trans. Networking*, vol. 6, no. 2, pp. 175–185, Apr. 1998.

[53] S. Floyd, "RED: Discussions of setting parameters," Nov. 1997, http://www.icir.org/floyd/REDparameters.txt.

[54] NLANR, "NLANR network traffic packet header traces," http://pma.nlanr.net/Traces/Traces.

[55] Telecommunication Networks Group, "MPEG-4 and H.263 video traces for network performance evaluation," http://www-tkn.ee.tu-berlin.de/research/trace/trace.html.

[56] K. Mochalski, J. Micheel, and S. Donnelly, "Packet delay and loss at the Auckland Internet access path," in *Proc. Passive Active Measure. Workshop*, Fort Collins, CO, Mar. 2002.

[57] I. Stoica, H. Abdel-Wahab, K. Jeffay, S. K. Baruah, J. E. Gehrke, and C. G. Plaxton, "A proportional share resource allocation algorithm for real-time, time-shared systems," in *Proc. IEEE Real-Time Syst. Symp.*, Washington, DC, Dec. 1996, pp. 288–299.

[58] S. Lu, V. Bhargavan, and R. Srikant, "Fair scheduling in wireless packet networks," *IEEE/ACM Trans. Networking*, vol. 7, no. 4, pp. 473–489, Aug. 1999.

[59] S. Lu, V. Bharghavan, and R. Srikant, "Fair scheduling in wireless packet networks," in *Proc. ACM SIGCOMM*, Cannes, France, Sep. 1997, pp. 63–74.

[60] WAND Research Group, "Auckland-VI trace data," http://pma.nlanr.net/Traces/long.

[61] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM Trans. Networking*, vol. 3, no. 4, pp. 365–386, Aug. 1995.

[62] T. Kozaki, N. Endo, Y. Sakurai, O. Matsubara, M. Mizukami, and K. Asano, "32×32 shared buffer type ATM switch VLSI's for B-ISDN's," *IEEE J. Select. Areas Commun.*, vol. 9, no. 8, pp. 1239–1247, Oct. 1991.

[63] Y. Shobatake, M. Motoyama, E. Shobatake, T. Kamitake, S. Shimizu, M. Noda, and K. Sakaue, "A one-chip scalable 8×8 ATM switch LSI employing shared buffer architecture," *IEEE J. Select. Areas Commun.*, vol. 9, no. 8, pp. 1248–1254, Oct. 1991.

[64] N. Endo, T. Kozaki, T. Ohuchi, H. Kuwahara, and S. Gohara, "Shared buffer memory switch for an ATM exchange," *IEEE Trans. Commun.*, vol. 41, no. 1, pp. 237–245, Jan. 1993.

[65] W. Denzel, A. Engbersen, and I. Iliadis, "A flexible shared-buffer switch for ATM at Gb/s rates," *Comput. Networks & ISDN Syst.*, vol. 27, no. 4, pp. 611–624, January 1995.

[66] C. B. Stunkel, "The SP2 high-performance switch," *IBM Syst. J.*, vol. 34, no. 2, pp. 185–204, Feb. 1995.

[67] A. Miyoshi and R. Rajkumar, "Protecting resources with resource control lists," in *Proc. IEEE Real-Time Technol. Applic. Symp.*, Taipei, Taiwan, May 2001, pp. 85–94.

[68] D. Cohen and K. Narayanaswamy, "A fair service approach to defending against packet flooding attacks," http://www.cs3-inc.com/ddos.html.

Appendix A. Relationship between AFB and RFB

As described in Section 1.3.2, the absolute and the relative fairness bounds are two common measures of fairness in bandwidth allocation. Denote by $S_i^G(t_1, t_2)$ and $S_i^q(t_1, t_2)$ the service received by flow i during time interval $[t_1, t_2)$ under the GPS policy and under a given practical policy q, respectively. Denote by W the sum of the weights of all flows, and by w_m the smallest weight of all flows.

Under any given scheduling policy q, the absolute fairness with respect to flow i over time interval $[t_1, t_2)$, denoted by $\mathrm{AF}_i^q(t_1, t_2)$, is defined as,

$$\mathrm{AF}_i^q(t_1, t_2) = \left| \frac{S_i^q(t_1, t_2)}{w_i} - \frac{S_i^G(t_1, t_2)}{w_i} \right|. \tag{A.1}$$

The absolute fairness over time interval $[t_1, t_2)$, denoted by $\mathrm{AF}^q(t_1, t_2)$, and the absolute fairness bound, AFB^q, are defined as,

$$\mathrm{AF}^q(t_1, t_2) = \max_{\forall i} \mathrm{AF}_i^q(t_1, t_2) \tag{A.2}$$

$$\mathrm{AFB}^q = \max_{\forall (t_1, t_2)} \mathrm{AF}^q(t_1, t_2). \tag{A.3}$$

Under any given scheduling policy q, the relative fairness with respect to a pair of flows (i, j) over time interval $[t_1, t_2)$, denoted by $\mathrm{RF}_{(i,j)}^q(t_1, t_2)$ is defined as,

$$\mathrm{RF}_{(i,j)}^q(t_1, t_2) = \left| \frac{S_i^q(t_1, t_2)}{w_i} - \frac{S_j^q(t_1, t_2)}{w_j} \right|. \tag{A.4}$$

The relative fairness with respect to a flow i over time interval $[t_1, t_2)$, denoted by $\mathrm{RF}_i^q(t_1, t_2)$, is defined as,

$$\mathrm{RF}_i^q(t_1, t_2) = \max_{\forall j} \mathrm{RF}_{(i,j)}^q(t_1, t_2). \tag{A.5}$$

The relative fairness over time interval $[t_1, t_2)$, $\mathrm{RF}^q(t_1, t_2)$, and the relative fairness bound, RFB^q, can now be defined as,

$$\mathrm{RF}^q(t_1, t_2) = \max_{\forall i} \mathrm{RF}_i^q(t_1, t_2) \tag{A.6}$$

$$\mathrm{RFB}^q = \max_{\forall(t_1,t_2)} \mathrm{RF}^q(t_1,t_2). \tag{A.7}$$

Lemma 4 Under any work-conserving policy q, over any interval of time $[t_1,t_2)$,

$$\sum_{i=1}^{N} S_i^q(t_1,t_2) = \sum_{i=1}^{N} S_i^G(t_1,t_2)$$

where N is the number of flows. This obvious lemma is also stated in [21]. For the sake of brevity, we denote $\sum_{i=1}^{N} S_i^G(t_1,t_2)$ by $S(t_1,t_2)$.

Lemma 5 Over any interval of time $[t_1,t_2)$ and for any pair of flows (i,j),

$$\mathrm{RF}^q_{(i,j)}(t_1,t_2) \le \mathrm{AF}^q_i(t_1,t_2) + \mathrm{AF}^q_j(t_1,t_2).$$

Proof: Recall that under the GPS scheduler, the service received by each backlogged flow is exactly proportional to its weight, i.e.,

$$\frac{S_i^G(t_1,t_2)}{w_i} = \frac{\sum_{i=1}^{N} S_i^G(t_1,t_2)}{\sum_{i=1}^{N} w_i} = \frac{S(t_1,t_2)}{W}.$$

Thus from (A.1), we can express the absolute fairness of flow i as follows:

$$\mathrm{AF}^q_i(t_1,t_2) = \left| \frac{S_i^q(t_1,t_2)}{w_i} - \frac{S(t_1,t_2)}{W} \right|. \tag{A.8}$$

From (A.4) and using (A.8), we get,

$$\begin{aligned}
\mathrm{RF}^q_{(i,j)}(t_1,t_2) &= \left| \frac{S_i^q(t_1,t_2)}{w_i} - \frac{S_j^q(t_1,t_2)}{w_j} \right| \\
&= \left| \frac{S_i^q(t_1,t_2)}{w_i} - \frac{S(t_1,t_2)}{W} + \frac{S(t_1,t_2)}{W} - \frac{S_j^q(t_1,t_2)}{w_j} \right| \\
&\le \left| \frac{S_i^q(t_1,t_2)}{w_i} - \frac{S(t_1,t_2)}{W} \right| + \left| \frac{S(t_1,t_2)}{W} - \frac{S_j^q(t_1,t_2)}{w_j} \right| \\
&= \mathrm{AF}^q_i(t_1,t_2) + \mathrm{AF}^q_j(t_1,t_2).
\end{aligned}$$

■

Lemma 6 Over any interval of time $[t_1,t_2)$,

$$\mathrm{RF}^q(t_1,t_2) \le 2\mathrm{AF}^q(t_1,t_2).$$

Proof: Over the time interval $[t_1, t_2)$, assume that the maximum of the relative fairness with respect to any pair of flows occurs with flows i' and j'. Therefore,

$$\begin{aligned} \mathrm{RF}^q(t_1,t_2) &= \mathrm{RF}^q_{(i',j')}(t_1,t_2) \\ &\leq \mathrm{AF}^q_{i'}(t_1,t_2) + \mathrm{AF}^q_{j'}(t_1,t_2) \\ &\leq 2\mathrm{AF}^q(t_1,t_2). \end{aligned}$$

■

Lemma 7 Over any interval of time $[t_1, t_2)$, for any flow i,

$$\mathrm{AF}^q_i(t_1,t_2) \leq \left(1 - \frac{w_i}{W}\right) \mathrm{RF}^q_i(t_1,t_2).$$

Proof: Denote by $S^q_{i-}(t_1,t_2)$, the sum of the service received by all the flows except flow i during time interval $[t_1, t_2)$ under policy q. From (A.8), we have,

$$\begin{aligned} \mathrm{AF}^q_i(t_1,t_2) &= \left| \frac{S^q_i(t_1,t_2)}{w_i} - \frac{S(t_1,t_2)}{W} \right| \\ &= \left| \frac{S^q_i(t_1,t_2)}{w_i} - \frac{S^q_i(t_1,t_2) + S^q_{i-}(t_1,t_2)}{W} \right| \\ &= \left| \frac{(W-w_i)S^q_i(t_1,t_2)}{Ww_i} - \frac{S^q_{i-}(t_1,t_2)}{W} \right| \\ &= \frac{W-w_i}{W} \left| \frac{S^q_i(t_1,t_2)}{w_i} - \frac{S^q_{i-}(t_1,t_2)}{W-w_i} \right|. \end{aligned} \tag{A.9}$$

If we denote

$$\alpha_j = \frac{S^q_j(t_1,t_2)}{w_j}$$

then $S^q_j(t_1,t_2) = w_j\alpha_j$, and,

$$\frac{S^q_{i-}(t_1,t_2)}{W-w_i} = \frac{\sum_{j\neq i} w_j\alpha_j}{\sum_{j\neq i} w_j}$$

which means $S^q_{i-}(t_1,t_2)/(W-w_i)$ can be considered as the weighted average of $\alpha_j, j \neq i$. Therefore,

$$\min_{j\neq i} \frac{S^q_j(t_1,t_2)}{w_j} \leq \frac{S^q_{i-}(t_1,t_2)}{W-w_i} \leq \max_{j\neq i} \frac{S^q_j(t_1,t_2)}{w_j}. \tag{A.10}$$

Thus, we have,

$$\begin{aligned}\left|\frac{S_i^q(t_1,t_2)}{w_i} - \frac{S_{i-}^q(t_1,t_2)}{W-w_i}\right| &\leq \max_{\forall j}\left|\frac{S_i^q(t_1,t_2)}{w_i} - \frac{S_j^q(t_1,t_2)}{w_j}\right| \\ &= \mathrm{RF}_i^q(t_1,t_2). \end{aligned} \tag{A.11}$$

Applying the above in (A.9) completes the proof. ■

Note that $w_i \geq w_m$, and therefore, from Lemma 7,

$$\mathrm{AF}_i^q(t_1,t_2) \leq \left(1 - \frac{w_m}{W}\right)\mathrm{RF}_i^q(t_1,t_2). \tag{A.12}$$

In the inequality above, if the RHS is no less than the LHS for any given flow, then over all flows, the maximum possible value of the RHS is also no less than the maximum possible value of the LHS. This leads into Lemma 8 below.

Lemma 8 Over any interval of time $[t_1, t_2)$,

$$\mathrm{AF}^q(t_1,t_2) \leq \left(1 - \frac{w_m}{W}\right)\mathrm{RF}^q(t_1,t_2).$$

Combining Lemmas 6 and 8, we have,

$$\frac{1}{2}\,\mathrm{RF}^q(t_1,t_2) \leq \mathrm{AF}^q(t_1,t_2) \leq \left(1 - \frac{w_m}{W}\right)\mathrm{RF}^q(t_1,t_2). \tag{A.13}$$

We now proceed to prove that the above relationship also holds between the absolute and relative fairness bounds.

Theorem 7 For any work-conserving scheduling policy q,

$$\frac{1}{2}\,\mathrm{RFB}^q \leq \mathrm{AFB}^q \leq \left(1 - \frac{w_m}{W}\right)\mathrm{RFB}^q.$$

Proof: Without loss of generality, we assume that the maximum value of absolute fairness is achieved over the time interval $[t_1, t_2)$. Thus, from Lemma 8,

$$\begin{aligned}\mathrm{AFB}^q &= \mathrm{AF}^q(t_1,t_2) \\ &\leq \left(1 - \frac{w_m}{W}\right)\mathrm{RF}^q(t_1,t_2) \\ &\leq \left(1 - \frac{w_m}{W}\right)\mathrm{RFB}^q.\end{aligned}$$

Similarly, if we assume that the maximum of relative fairness is achieved over the time interval $[t_3, t_4)$, then from Lemma 6,

$$\begin{aligned} \mathrm{RFB}^q &= \mathrm{RF}^q(t_3, t_4) \\ &\leq 2\mathrm{AF}^q(t_3, t_4) \\ &\leq 2\mathrm{AFB}^q. \end{aligned}$$

■

The bounds stated in Theorem 7 can be shown to be tight. Consider a set of backlogged flows of equal weight managed by a scheduler that behaves exactly as GPS all of the time except during a certain short interval of time. During this interval, it gives all of the share of flow i's service to another flow j. All other flows receive service exactly equal to what they would have received under GPS. During this interval, flow i receives no service at all while flow j receives twice the service it would have received under GPS. One can readily verify that the absolute fairness bound of this scheduler is one half of its relative fairness bound, establishing that the lower bound in Theorem 7 is tight.

We use the Deficit Round Robin (DRR) scheduler [8] to show that the upper bound in Theorem 7 is also tight. Consider $N-1$ backlogged flows, each of unit weight, served by a DRR scheduler. Assume that a new N-th flow, also of unit weight, becomes active at time t, when all other flows are already backlogged and active. Denote by Q the quantum size associated with each flow, and by Δ the smallest unit of service provided by the DRR scheduler. Recall that the maximum possible value of the deficit counter is $Q - \Delta$ at the beginning of each round, and that an additional amount of Q may be served from each flow in each new round. Therefore, each of the first $N-1$ flows may receive service equal to $2Q - \Delta$ before flow N receives its service opportunity. If the first packet in flow N's queue is of size Δ and the second packet is of size Q, flow N will receive service equal to Δ in this first service opportunity. Before flow N receives its second service opportunity, each of the other flows may have received service equal to a maximum of Q. Now, let t'

be the time instant when flow n begins its second service opportunity. During the interval $[t, t')$, flow N receives a service equal to Δ while each of other flows receives service equal to $3Q - \Delta$. It can be shown that both absolute and relative fairness bounds of the DRR scheduler are achieved over this time interval $[t, t')$. Thus, the relative fairness bound is $3Q - 2\Delta$, while the absolute fairness bound is

$$\frac{(N-1)(3Q-\Delta)+\Delta}{N} - \Delta = \frac{N-1}{N}(3Q-2\Delta)$$

showing that the upper bound in Theorem 7 is tight.

Vita

Yunkai Zhou was born in Shanghai, China. He received his B.S. in Electrical Engineering from the Department of Automation, Tsinghua University, Beijing, China, in 1998, and M.S. in Electrical Engineering from the Department of Electrical and Computer Engineering, Drexel University, in 2002. Since September 1998, he has been affiliated with the Computer Communications Laboratory, Department of Electrical and Computer Engineering, Drexel University, under the supervision of Dr. Harish Sethu. His research has involved a variety of areas in computer networking, such as resource allocation, wireless and sensor networks, switching networks, system performance evaluation, system architecture design and traffic modeling. He has also been a teaching assistant in the Department of Electrical and Computer Engineering, Drexel University, from 1998 to 2003. His teaching responsibilities include lecture, recitation, help sessions, and quiz/homework assignment/grading.

His research work has been published in or is under review with various refereed journals and conferences. He has been awarded the ACM SIGCOMM Student Travel Award from the Association of Computing Machinery, the George Hill Jr. Fellowship from Drexel University and the ECE Graduate Travel Award from InterDigital Communications. He is a member of IEEE, IEEE Computer Society, IEEE Communications Society, ACM and ACM Special Interest Group in Data Communications.