ROUTING, FLOW CONTROL AND FAIRNESS IN COMPUTER NETWORKS

Hak-Wai Chan

October 1986
CSD-860067

ROUTING, FLOW CONTROL AND FAIRNESS IN COMPUTER NETWORKS

by

Hak-Wai Chan

Computer Science Department
School of Engineering and Applied Science
University of California
Los Angeles

TABLE OF CONTENTS

LIST OF FIGURES

page

LIST OF TABLES

page

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my doctoral committee consisting of Professors Mario Gerla, Jack Carlyle, D. Stott Parker, Kirby Baker and Harold Borko. I am particularly grateful to the committee chair and my advisor, Professor Mario Gerla for his encouragement, support and enlightening discussions throughout the course of this research.

Special thanks are also expressed to friends in Modeling and Analysis Group at UCLA for their insightful comments and encouragement. The help from Edmundo Silva of his knowledge in queueing network analysis is greatly appreciated. I am also indebted to Baron Grey in sharing his knowledge in the difficult task of editing this dissertation.

Finally I express my deepest appreciation to my parents whose everlasting encouragement, understanding and love I shall forever owe.

ABSTRACT OF THE DISSERTATION

Routing, Flow Control and Fairness

in Computer Networks

by

Hak-Wai Chan

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 1986

Professor Mario Gerla, Chair

The success of computer networks is based on the efficient sharing of resources among users. Routing and flow control have been designed to enhance the sharing of common resources and to optimize overall system performance. A key performance criterion in this optimization is fairness i.e., the ability to provide equal satisfaction to all users. We propose a measure of fairness based on balanced interference among users for wide-area, packet-switched computer network. This measure guarantees that the throughput of the individual user cannot be zero and the actual throughput achieved by a user tends to be proportional to his traffic demand.

Assuming input rate flow control, an integrated routing and flow control algorithm is developed to optimize this fairness measure, and yet satisfy an overall average delay constraint. Open network model is used in the optimization.

We then develop an approximate algorithm to optimize the same fairness measure and satisfy an overall average network delay constraint using this time window flow control. Closed network model and fixed routing are used in the optimization. The algorithm is computationally very efficient and provides nearly optimal solution. It provides guidelines for window selection so that for a given delay constraint, the optimal set of windows is readily available.

Finally, the performance characteristics (delay, throughput and fairness) of a passive flow control known as backpressure are studied. Network performance under a combination of this control mechanism and window control is compared to that using window flow control alone. It is found that the combination of backpressure with window control is beneficial in that it provides a fair allocation of resources at heavy load.

CHAPTER 1

INTRODUCTION

A computer communication network has a limited amount of resources such as nodal buffers, channel bandwidth, logical channels, nodal processing power etc. These resources must be utilized in an efficient and careful way otherwise loss of information, performance degradation and deadlocks may happen.

Consider the computer network shown in Figure 1.1 where a collection of sources and destinations communicate with each other via a set of communication channels. Packets travel through this network over the data transmission lines. Suppose source S_1 sends messages using link 1 and 2 to destination D_1 at a rate faster than D_1 can consume. A queue of messages will soon build up at the destination D_1. If the activity of source S_1 does not slow down then either a large number of buffer in the network will be occupied by its messages (denying service to other users) or the destination must discard some messages. Of course, if no links other than 1 and 2 are used, the flow of traffic is limited by the maximum capacity of link 1 and 2. Thus, the bottleneck may then be represented by these channels. Some forms of control is necessary to utilize the resources available (such as link 3,5,6,7 and 9) without at the same time overloading the system.

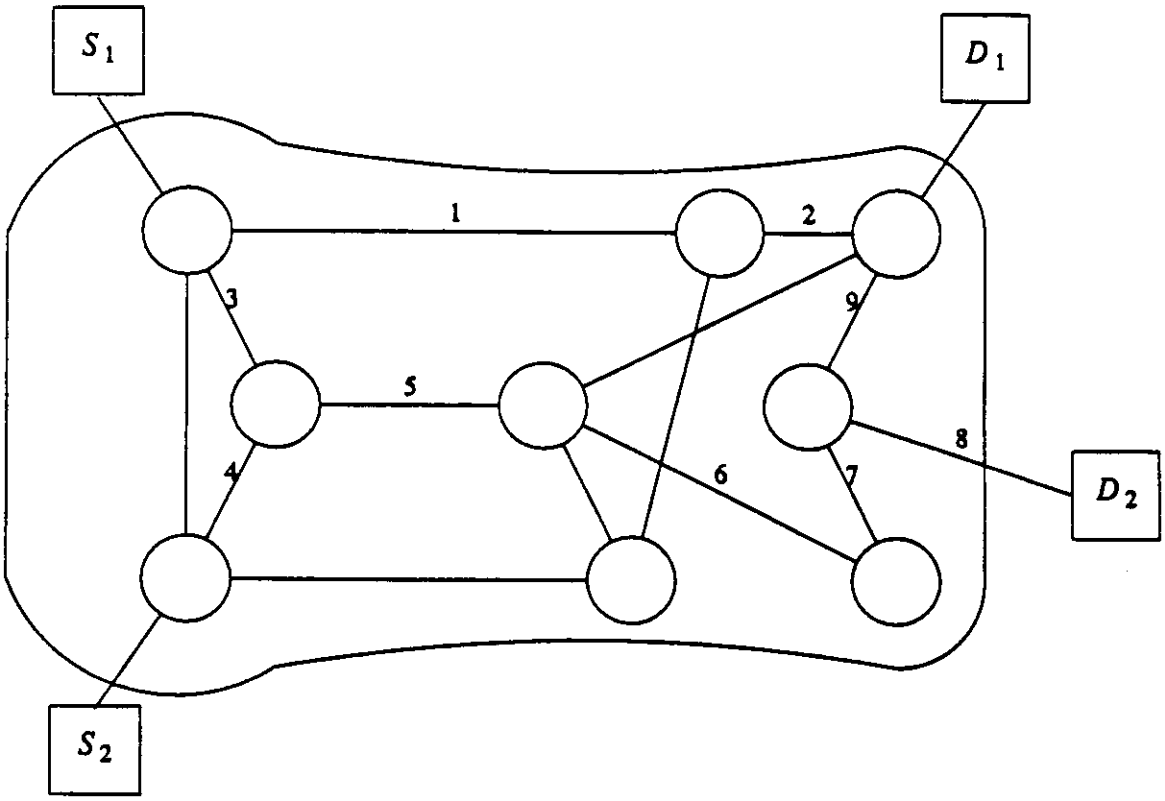

Fig. 1.1 A computer communication network.

Routing and flow control are two forms of resource control that should be used in this case. These are the key procedures which we will consider in our study of fairness in computer networks.

1.1 Routing

The choice of route in a store and forward packet switching network has a major effect on the efficient resource sharing in a network environment. Considering the network in Figure 1.1, if packets are sent from S_1 to D_1 and from S_1 to D_2, one possible efficient sharing of resources consists of sending traffic via link 1 and 2 for traffic from S_1 to D_1 and via link 3,5,6,7 and 8 for traffic from S_1 to D_2. (Note that the path 1,2 9,8 would be shorter for S_1 to D_2 traffic, but it would be exposed to interference with S_1 to D_1 traffic). This procedure, called routing, has the effect of distributing the resources (transmission bandwidth) among the users as efficiently as possible.

1.2 Flow Control

Figure 1.2 depicts a typical performance profile of an uncontrolled computer communication network. We observe that the throughput increases as the offered load increases until some threshold is reached. Beyond this point the throughput decreases rapidly to zero. This occurs because the traffic that has already been accepted into the network exceeds the capacity of the network. The effect of throughput degradation discourages us to operate the network in the region (B,C). On the other hand, operating near A is undesirable if the network could safely be operated around B. Safe operation can be obtained with a variety of flow control mechanisms. In this study, we will limit ourselves to two flow control procedures, namely input rate flow control and window flow control.

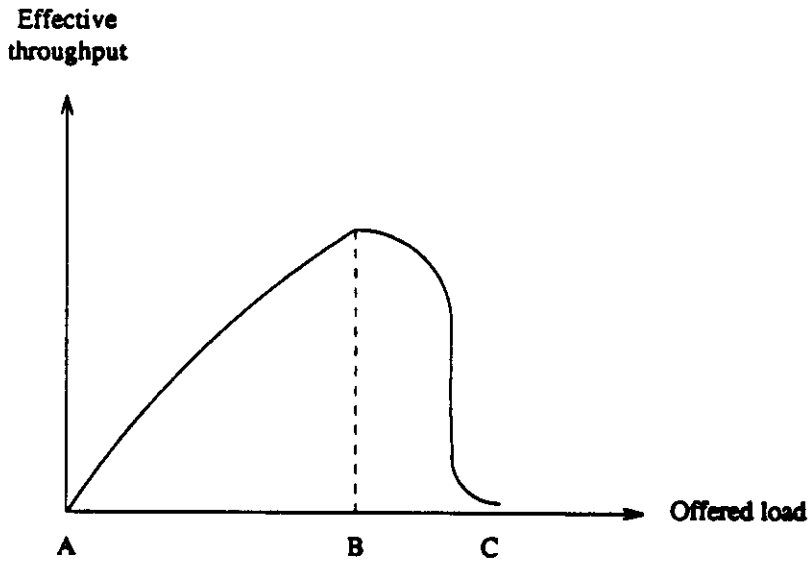

Fig. 1.2 Typical throughput vs. offered load.

The key idea of flow control is to keep the excess load out of the network in an coordinated way without compromising the principle of dynamic resource sharing in the network environment. Flow control prevents throughput degradation and loss of efficiency as a result of network overloading, i.e., prevents operating in region (B,C) in Figure 1.2. It also should attempts to allocate the resources fairly among competing users. This may include restriction on some of allocated buffer or limit on users transmission rates.

In summary, flow control is necessary to maintain traffic at a level compatible with the amount of available resource without overcommitting resources to a single user or a single user group.

1.3 Routing and Flow Control Integration

The efficiency of packet switching in transmitting bursty traffic rests on the ability to share resource. However, to produce an orderly sharing, it is necessary to exert some control. We discussed the need for two well known forms of control, namely routing and flow control These control procedures have traditionally been studied as separate problems; consequently routing procedures have been designed independently from flow control schemes. As a result, when a packet is submitted to the network, the flow control algorithm will first decide whether to accept it or not, generally based on buffer availability at entry node. If the packet is accepted, it is then the task of the routing algorithm to find a suitable path to transport the packet to its destination. It may happen that sometimes this path does not exist. The routing procedure does not consider the possible dynamic fluctuations in input traffic while the flow control procedure does not take advantage of existence of the optimal routing inside the network to alleviate heavy traffic load.

A closer look will show that these two forms of control are indeed related and must be jointly designed. Choice of routes should depend on the amount of traffic admitted into the network (routing procedure tends to direct traffic to less

congested area). On the other hand the effectiveness of the flow control mechanism depends on the choice of route after the admission of traffic in the network. Thus, a better approach would be to combine the routing and flow control decisions, i.e., to ascertain that a feasible path exists before accepting a packet into the network.

Recently a few contributions have been made towards achieving the integration of routing and flow control [Gerl80a]. This problem was formulated by Gallager as an optimization problem with a joint objective function of delay and penalty. The penalty function acts as a user dissatisfaction function which increases as the user's allowed traffic decreases. Thus the flow control mechanism was embedded in the objective function. On the other hand, to take full advantage of the sharing of expensive network resources, it is necessary to design and configure the network according to proper criteria. Different performance objectives have been addressed in the literature which will be presented in section 3.1 [Schw80, Gerl80b, Wong82b].

1.4 Fairness

One performance aspect which has received considerably less attention than other measures is fairness. Whenever a service relies on extensive resource sharing, it is likely to cause interference and competition among users. The level of this interference and the ability to fairly resolve the competition must be considered. The purpose of this research is to devise centralized algorithms for

integration routing and flow control with fairness taken into consideration.

Efficiency in sharing common resources does not always imply fairness. In other words, the network may achieve better efficiency by favoring some users over others. However, unfairness is undesirable in general; and especially in public networks where users pay the same tariff, supposedly for equal services. Therefore, the importance given here to fairness is justified. It is of interest to seek control algorithms which can fairly distribute resources among competing users.

Designing a fair system is not simple. The first obstacle is a basic one: how to define fairness. In general, different users will have different conceptions of what is fair. Moreover, the service provider may have an approach to fairness which does not agree with the users point of view. It is therefore difficult to come up with a universal and generally accepted definition of fairness, let alone designing control algorithms to achieve it.

We will establish a quantitative measure of fairness in Chapter 3 so that a system optimization can be done using different models for different forms of flow control procedures. One important aspect of our proposed solution is that the fairness issue is included in the problem from the beginning, i.e., it is embedded in the objective function to be optimized.

1.5 Summary of Results

The fairness measure which is defined for our models has several pleasing properties. It can be used in both input rate control and window control. It allows routing to be incorporated in both controls easily. The measure also takes both the demands and throughputs of the users into consideration with the additional property that the virtual circuits that are not interfering with each other can send as much traffic as they need (up to channel capacity). This is in contrast to other fairness measures which do not include delay in the optimization nor are sensitive to initial user demands.

It can be shown that the proposed measure produces solution consistent with other definitions of fairness e.g., saturated cut condition in [Gerl82]. However, our measure is much easier to handle in system optimization involving both routing and flow control.

This definition of the fairness measure leads us directly to the design of an efficient algorithm for integrated routing and input rate flow control. This problem is difficult because of the interplay between routing and flow control. Improving routing will naturally allow more traffic to get through, which in turn imposes higher delay and may cause congestion. We show that the objective function in our formulation of the problem is convex, and so is the set of feasible solutions. Thus our algorithm finds the global minimum. The result shows that higher overall throughputs are achieved than with uniform scaling of user

requirements.

A more difficult problem is (and a more important one from the practical stand point) the window optimization for flow control and fairness. The problem is a difficult one because it requires multivariable discrete combinatorial optimization. The exact solution is computationally impractical because of the complexity of the combinatorial nature of the problem. We have developed a heuristic solution approach which is shown to be fast, stable and provides nearly optimal solutions.

In the above analytic investigation we have assumed infinite nodal buffers. However, nodal buffers in a real network are limited, and the allocation of these buffers will affect network performance characteristics such as delay, throughput and fairness. Therefore we have also studied a flow control technique, called backpressure which is based on finite buffer allocation. We have also studied the effect of the combination of this technique with window control on fairness. This evaluation was carried out via simulation. We found that combination of backpressure with window control is the best strategy for fair bandwidth allocation at heavy loads.

1.6 Organization of the thesis

In chapter 2 various performance measures are presented with particular emphasis on fairness which, is used as our objective function in later chapters. Some examples are given to illustrate the validity of the measure. The fairness

problem is formally stated at the end of the chapter.

In chapter 3 we review the network models which will be used for fairness optimization with input rate control and window flow control as variables respectively. We use the open network model for input rate control and closed network model for window control. Analytic solutions of the closed network model are discussed in some detail.

In chapter 4 our algorithm for fairness optimization in open networks is presented with input rates and routing as variables and with overall average delay as a constraint. The problem formulation is given and optimality conditions are established. Numerical examples are then presented to illustrate the network performance provided by the algorithm.

In chapter 5 our heuristic algorithm for fairness optimization in closed networks is presented again with window sizes as variables and with overall average delay as a constraint. Numerical results are provided to illustrate and validate the method and the approach.

In chapter 6 our investigation on the effect of flow control on fairness continues with the flow control mechanism known as backpressure. Three different schemes, namely window control, backpressure and combination of backpressure with window are investigated and compared under different traffic conditions. Numerical examples are provided to illustrate the effects of the schemes on fairness under different traffic load environments.

CHAPTER 2

DEFINITION OF PERFORMANCE MEASURES AND PROBLEM STATEMENT

2.1 Various Performance Measures

Several different criteria are commonly used to evaluate the performance and to optimize the design of communication networks. Different criteria should be used in different applications, for example, for interactive traffic short delay is important while in file transfer environment time delay is not that crucial. Availability is not so crucial for private networks as for military communication systems. In contrast, data security is critical in financial applications.

In our study fairness is the focus of our attention. Therefore we will concentrate much of our study on this measure. Here we list the criteria which are used directly or indirectly in our problem formulation.

2.1.1 Delay

Probably, delay is the single most important measure used in computer communication network. Below we present two variations of this measure.

2.1.1.1 Average Network Delay

The average network delay T was found by Kleinrock [Klei64] to be

$$T = \sum_{i=1}^{NL} \frac{\gamma_i}{\lambda} T_i \tag{2.1}$$

where

γ_i = the average rate of traffic flow on the ith link

λ = $\sum_{i}^{R} \lambda_i$ where λ_i is the average rate of traffic flow for virtual circuit i

NL = number of link in the network

T_i = average delay on link i

This is a very simple and general expression for the average network delay in terms of its single link components, namely the delay T_i.

If we account for the nodal processing time K and the propagation time P_i, the expression will become

$$T = \sum_{i=1}^{NL} \left[\frac{\gamma_i}{\lambda} T_i + P_i + K \right] \tag{2.2}$$

With the following assumptions: [Klei76]

- external Poisson arrivals

- exponential packet length
- infinite nodal storage
- error free links
- no nodal delay
- independence assumption (i.e., packets are assigned a new length using an exponential distribution at each intermediate node)

T_i can be computed using the M/M/1 model and is given by

$$T_i = \frac{1}{\mu C_i - \gamma_i} \tag{2.3}$$

where C_i = capacity of link i

and $\frac{1}{\mu}$ = average packet length

A more comprehensive expression for T will be given in section 4.3.1 for our fairness optimization in open network where we assume $k = 0$ and $P_i = 0$. Due to the simplicity of the expression for the average network delay, we will use this in our problem formulation in Chapter 4.

2.1.1.2 Maximum Average Delay

Maximum average delay is defined as follows:

$$D_{\max} = \max_i(D_i)$$

where D_i is the average delay for user i. It is a difficult criterion to handle mathematically. For minimizing this measure, the problem will be a min-max problem.

2.1.2 Throughput

The other performance measure we most commonly use is the throughput. The throughput is defined as number of packets per second delivered out of the destinations. We would like to have as high a throughput as possible. Unfortunately, high throughput and low network delay are conflicting objectives because as the users' traffic rates increase, the queue lengths within the network grow and larger time is required to go from the source to the destination. Therefore a tradeoff measure including both the throughput and delay has been proposed and will be presented in the next section.

2.1.3 Power

This performance measure was first introduced in [Gies78]. It is defined as the ratio of mean throughput to average delay. This is an analogy to power defined in physics as energy over time. We mentioned in the last section that high throughput and low delay are two conflicting objectives. Therefore power is considered as a compromise measure. Maximizing power will be a desirable tradeoff between low delay and high throughput in computer networks. Some properties of power are reported in [Gail81, Jaff81a].

For single virtual circuit i, the above definition of power $P_i = \frac{\lambda_i}{T_i}$ or its generalization $P_i = \frac{\lambda_i^\beta}{T_i}$ can be maximized for the system M/M/1 yielding the "proper" operating point [Klei79]. For a multisource system, it is not quite clear how to define power. A couple of definitions were given by [Bhar81] and algorithms were proposed to optimize the network performance. The definitions are presented here.

(a)

$$P_N = \frac{\lambda^\beta}{T}$$

where $\lambda = \sum_{i=1}^{R} \lambda_i$ and

$$T = \frac{\sum_{i=1}^{R} \lambda_i T_i}{\lambda}$$

i.e., power = (total throughput)$^\beta$ / average delay

(b)

$$P_N = \sum_{i=1}^{R} P_i$$

where

$$P_i = \frac{\lambda_i^\beta}{T_i}$$

i.e., sum of powers of virtual circuits

(c)

$$P_N = \prod_{i=1}^{R} P_i$$

i.e., product of powers of virtual circuits

The parameter β can be adjusted to achieve different tradeoff points between throughput and delay. A large β emphasizes on throughput. A small β emphasizes on delay. It is shown that definitions given in (a) and (b) were unfair in the sense that certain users may get zero throughput. The definition in (c) is fair in a sense that no user will get zero throughput when maximizing power.

2.1.4 Fairness

This measure has been ignored in the past due to the difficulty in defining what is considered fair. Only recently it was addressed by some researchers [Jaff81b, Gerl82]. Here we will review and classify different fairness schemes proposed in the last few years.

The fairness schemes which have been proposed for application in computer networks can be broadly grouped into three classes, according to the criterion that they attempt to enforce, namely:

(i) fair resource utilization;

(ii) equal performance;

(iii) balanced interference among users.

In the first class, we identify some critical resources in the system (typically, the bottlenecks) and require that they be equally shared among the users competing for them. In the second class, we identify some critical user performance measures (e.g., delay, throughput, etc.) and attempt to equalize these performance measures across the entire user population. In the third class, we require that users get penalized to a degree proportional to the interference encountered while competing for a resource. Thus, for each user we define a penalty function related to the interference. We then attempt to equalize penalty over all users.

2.1.4.1 Fair Resource Utilization

Chronologically, the first contributions to the study of fairness were based on equal resource utilization. The schemes in this class attempt to guarantee that a given resource (e.g., a communication link), or a group of resources will be equally divided among the users competing for these facilities.

In 1980 Jaffe [Jaff81b] proposed the bottleneck sharing technique, which basically searches for a "uniform throughput" solution. In his model, users are assumed to have unrestricted demands which are accommodated through the network on pre-established, and fixed, single paths called virtual circuits. Individual input rates are then controlled by the network, in an attempt to provide a fair sharing of link capacities.

A set of input rates $\{\lambda\}$ is defined by Jaffe to be fair if, for each link k the following condition is satisfied by the rates λ_i, $i = 1,...,R_k$, of the virtual circuits traversing that link:

$$\lambda_i \leq X\ [C_k - f_k] \quad \forall\ i = 1,...,R_k \tag{2.4}$$

where:

R_k = number of virtual circuits traversing the k-th link;

C_k = capacity of the kth link;

f_k = $\frac{\gamma_k}{\mu}$, flow in the kth link, given by the sum of all contributions λ_i traversing the link;

X = a constant parameter which can be the same for the whole network or vary for each link.

Note that if link k is modeled as an M/M/1 queue, and unit packet length is assumed the average delay in this link is given by

$$t_k = \frac{1}{C_k - f_k} \tag{2.5}$$

Thus, X can be expressed as:

$$X = \lambda_i\ t_k = \bar{n}_{ik} \tag{2.6}$$

where $\bar{n}_{ik}$ is the average number of user packets queued for service at link k.

From Eq. (2.7) it can be seen that X acts as a flow control parameter, establishing also a delay-throughput trade-off. Large values of X will tend to produce large throughput for the users. On the other hand, for small X the user delays will be smaller.

If no other restrictions are present, a user will attempt to maximize its throughput on link k according to constraint (2.5) If N users are sharing link k, competitive equilibrium is reached when:

$$\lambda_i = \frac{XC_k}{1+NX} \tag{2.7}$$

Since a typical virtual circuit traverses several links, a given user is subject to a constraint of type (2.8) for each hop along the path. The most restrictive constraint will prevail, that is, the link corresponding to the strongest constraint will determine the throughput of the user. This link is called the bottleneck link for this user.

Jaffe's paper presents a decentralized algorithm for finding the throughputs λ_i of each virtual circuit, according to Eqs. (2.5) and (2.8). The solution produced by the algorithm has two important characteristics:

i. A user throughput is no smaller than the throughput of any other user sharing its bottleneck link.

ii. The bottleneck is the only factor preventing the user from obtaining a

higher throughput.

A few observations can be made about Jaffe's method:

a. it does not allow for adaptive routing;

b. it can only be applied to direct input rate control;

c. the method does not take into consideration user delays;

d. user demands are assumed to be unrestricted

e. although Jaffe defines a fairness condition, he fails to define a fairness measure. Such a measure could be helpful to a network designer to check how far off a given throughput assignment is from ideal fairness.

Regarding point (a) above, it may be observed that the degree of fairness of a computer network can be improved by allowing multipath routing. Optimal routing techniques yield a uniform distribution of traffic over the net, as opposed to fixed routing which may concentrate many users on a few paths. This property makes it possible to obtain a more uniform performance over all users. Perceiving this, Gerla and Staskauskas (G-S) [Gerl82] extended Jaffe's fairness notion to networks employing multipath routing. The fairness condition becomes:

1. each constrained user must traverse at least one saturated cut, which is the bottleneck;

2. each saturated cut must be fairly shared, i.e., it must satisfy the

following conditions:

$\lambda_i = a < d_i$ if the i^{th} user is constrained;

$\lambda_i = d_i < a$ if the i^{th} user is unconstrained;

$$\sum_{i=1}^{N} \lambda_i = C_s$$

where d_i is the demand of the i^{th} user, C_s is the total capacity of the cut and a is a constant for the cut.

Note that the definitions of Jaffe and G-S are quite similar. The main differences are the notion of saturated link, in Jaffe, and saturated cut, in G-S; and, the presence of finite user demands in G-S, as opposed to unrestricted demands in Jaffe.

In order to obtain a throughput assignment which satisfies the aforementioned fairness principle, the authors employed a measure which is a slight variation of an objective function earlier defined by Gallager and Golestaani [Gall80] for integrated routing and flow control optimization. This fairness measure F_{GS} is given by:

$$F_{GS} = T + A \sum_{i=1}^{R} \left[\frac{1}{\lambda_i} - \frac{1}{d_i} \right] \tag{2.8}$$

where T is the average network delay and the second term in the right-hand side of Eq. (2.9) is a sum of penalty functions, one for each user. These penalty functions measure the dissatisfaction of each user for not having all his demands

accepted. Note that the penalty function has the desired properties that,

$$\text{for } \lambda_i \to 0, \text{ penalty} \to \infty$$

and

$$\text{for } \lambda_i \to d_i, \text{ penalty} = 0.$$

If the constant A is very large, minimizing the function F_{GS} yields the G-S fairness conditions (i.e., fair sharing of the saturated cut) as shown later. For large A, the penalty term in Eq. (2.9) will dominate the delay term, implying that a significant contribution to F_{GS} will come only from the bottleneck links. Note also that all constrained users, in the optimal solution, must be traversing a saturated cut, or else the value of F_{GS} could be reduced by increasing the throughput of these users. Moreover, all constrained users sharing the same bottleneck must have the same throughput, otherwise the objective function could be reduced by equalizing the throughputs, while keeping the total throughput constant.

In the case of direct input rate control the optimization problem can be interpreted as an optimal routing problem with convex cost function and can thus be solved by standard techniques [Frat73]. The solution for the case of window flow control is much more difficult and a heuristic approach is proposed for it in [Gerl82].

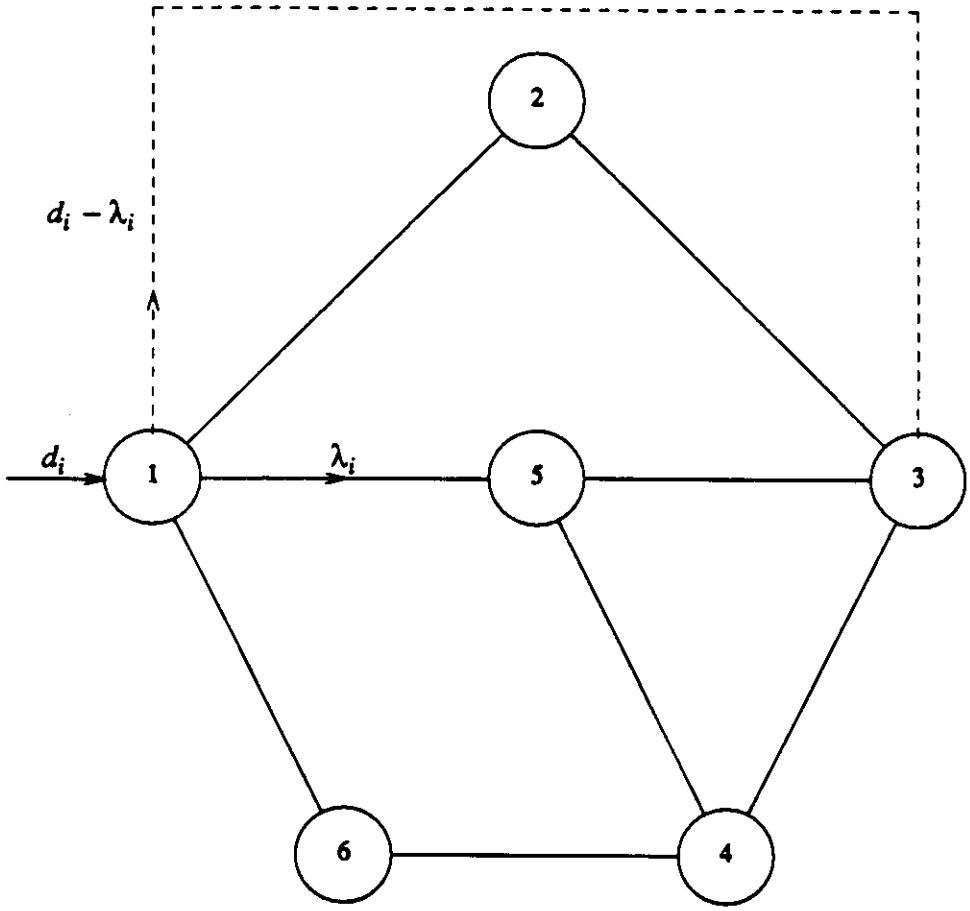

Fig. 2.1 Augmented network.

Considering Figure 2.1, the minimization of F_{GS} can be interpreted as a minimum cost routing problem in an augmented network in which a by-pass link is established between each source-destination pair. Thus by-pass link carries the overflow $d_i - \lambda_i$. The cost of the actual links is represented by their delay contri-

bution and is given by

$$l_i = \frac{\partial T}{\partial f_i} = \frac{1}{\lambda} \frac{C_i}{\left[C_i - \frac{\gamma_i}{\mu}\right]^2} \tag{2.9}$$

The cost of the by-pass links is represented by the penalty function and is given by

$$\frac{\partial Q}{\partial f_i} = \frac{A}{(d_i - \lambda_i)^2}$$

Table 2.1
Initial network requirements

Session	Source	Dest.	Demand
1	1	3	4
2	2	5	4

The new formulation of the problem can be easily solved with existing methods since both objective function and constraint set are convex. Thus, for each value of the parameter A we obtain a value of delay T and throughput $\lambda = \sum_{i=1}^{R} \lambda_i$. We give the result in Table 2.2 for the network configuration shown in Figure 2.1. The initial requirements are given in Table 2.1. Link capacity is C = 1 for all links and packet length is b = 1.

Table 2.2
Results of the augmented network with different values of A

A	Session 1		Session 2		Total throughput	Network delay
	Thrup.	Delay	Thrup.	Delay		
0.5	0.38	0.26	0.37	0.25	0.74	0.26
1.0	0.49	0.38	0.47	0.35	0.96	0.37
5.0	0.81	0.87	0.76	0.78	1.57	0.82
25.0	1.13	1.96	1.06	1.68	2.18	1.82
50.0	1.23	2.73	1.16	2.32	2.39	2.53
75.0	1.28	3.30	1.21	2.81	2.48	3.06
100.0	1.31	3.78	1.24	3.23	2.55	3.51
500.0	1.43	7.89	1.35	7.08	2.79	7.50
1000.0	1.48	10.96	1.37	10.10	2.85	10.58

Note that for $A \rightarrow \infty$, the delay goes to infinity, while the throughput asymptotically approaches the optimally fair value. The asymptotical solution has little practical value since the delay is infinity. If fairness is desired and yet a given delay constraint T_{max} must be met, we simply identify in the table the value of the parameter A which yields T_{max}, and find the allocation and routing solution corresponding to such value. This procedure allows us to extend the definition of fairness to cases in which link utilization cannot be allowed to reach unity, rather, a constraint is set on overall average delay. We will later use this approach (T_{max} constraint) in our formulation of our fairness problem.

In summary, the methods based on fair resource sharing have the desirable property of allowing users which do not traverse congested areas to achieve

throughputs close to their demands. Furthermore, they guarantee that virtual circuits sharing the same saturated cut will have equal throughput. This is good if the users have similar demands. However, it may heavily penalize big users if the distribution of demands is non-uniform. It has also been shown that when Jaffe's method is employed to optimize user rates, the result has the tendency of being too conservative in terms of total throughput [Wong]. Another drawback of these schemes is the lack of consideration for user delays.

2.1.4.2 Performance Equalization

The methods in this category are inspired by the principle that users of a public network should get uniform quality of service. Arguing in this direction, and considering user delay to be the most relevant performance measure, Sauve, Wong and Field (SWF) [Wong82a] proposed the following fairness measure:

$$F_T = \frac{1}{T^2} \sum_{i=1}^{R} \frac{\lambda_i}{\lambda} [T_i - T_i^{id}]^2 \qquad (2.10)$$

with

$$\lambda = \sum_{i=1}^{R} \lambda_i$$

Eq. (2.11) basically measures the square deviation of the actual user delay (T_i) with respect to an "ideal" delay (T_i^{id}) for that same user. The value of T_i^{id} may be chosen to be a constant T; or it may be chosen to be proportional to the path length.

It is clear that optimal fairness is achieved when $F_T = 0$. To minimize F_T, Sauve et.al. chose to use time varying priorities. In this scheduling procedure, the instantaneous priority of a packet belonging to virtual circuit i, at time t, is given by $b_i\ (t-t_0)$ where t_0 is the packet arrival time to a given link and b_i is the priority level of customer i. Note that now F_T becomes indirectly a function of R continuous variables b_i, allowing for a fine adjustment of the delays.

This method seems to be quite flexible, although the choice of T_i^{id} is bound to be controversial. Another shortcoming of this method is the assumption that no flow control is being employed in the network; that is, the traffic demands posed to the network must be initially feasible. Thus, the throughput obtained by each virtual circuit does not change after optimization.

Recognizing that, in some situations, the users of a computer net may be more interested in the throughput as the main performance parameter, the same authors proposed the following alternative fairness measure: [Sauv82]

$$F_\lambda = \frac{1}{\lambda^2} \sum_{i=1}^{R} [\lambda_i - \lambda_i^{id}]^2 \tag{2.11}$$

where λ_i^{id} is the ideal throughput for user i and λ is the total throughput.

The authors described an approximate technique to optimize F_λ, as a function of the b_i's, for the more realistic case of nets with window control. When the channel is ready to transmit the next packet, the instantaneous of all packets in queue are evaluated, and the packet with the highest value is served.

Thus this favors chains with large values of b_i, and those packets that have been waiting in queue for a long time.

Although the two methods just described are very versatile, they fail to weigh appropriately the interference among the existing virtual circuits. This may cause, for example, certain users to be unnecessarily constrained by the values of λ_i^{id} or T_i^{id}, when these customers do not cross any bottleneck (i.e., they do not interfere with anyone else). In this case, they could be allowed a much better performance, without imposing any loss on others.

As was the case for the methods in Section 2.1.4.1, the schemes introduced by Sauve et.al. cannot combine delay and throughput in the fairness optimization.

In next section another family of procedures, which take into account both delay and throughput of each user, are described.

2.1.4.3 Balanced Interference among users

The basic idea behind the methods in this class can be described as follows: [Wong] since the traffic of a given user perturbs the network, in the sense that it causes delay increase and throughput degradation to several other users, it is just fair that this annoyance should be proportional to the discomfort the network causes to this user. Thus, if a user creates severe congestion in the network, it is reasonable that this user should suffer a stronger throughput limitation

than another user who, for example, does not interfere with anybody else.

In order to quantify this fairness concept, it is necessary to define measures for the congestion caused to the network by a virtual circuit and for the penalty inflicted by the network to a given user.

In [Wong] the authors propose the following measures:

a. Penalty suffered by virtual circuit i due to the presence of other network users:

$$P_y(i) = \frac{\lambda_i^{id} - \lambda_i}{\lambda_i} \tag{2.12}$$

where λ_i^{id} is the ideal throughput of user i, i.e., the throughput of V.C. i would have if there were no other users in the network. Note that, if window flow control is employed, λ_i^{id}, in most cases, will be smaller than d_i, the traffic offered by user i. Note also that, by using λ_i^{id} instead of d_i in the penalty definition, the effect of the flow control mechanism by itself is not contributing to the value of the penalty; only the interference from other users affects the measure.

The choice of penalty function in Eq. (2.13) was influenced in part by the previously cited work of Gallager and Golestaani.

b. Congestion caused in the network by virtual circuit i:

$$C_g(i) = \frac{1}{\gamma} \sum_{\substack{j=1 \\ j \neq i}}^{R} \lambda_j \left[\frac{T_j - T_j^i}{T_j^i} \right] \tag{2.13}$$

where T_j^i is the average delay of V.C. j if user i were removed from the network and

$$\gamma = \sum_{\substack{j=1 \\ j \neq i}}^{R} \lambda_j = \lambda - \lambda_i \tag{2.14}$$

Hence, $C_g(i)$ is the average delay increase of other users due to the presence of virtual circuit i.

This congestion measure is similar to the one presented by Pennotti and Schwartz [Penn75]. In that work, however, the value of delay used to evaluate the congestion was the delay of a given virtual circuit in a given node whereas, here, the delay is averaged over all virtual circuits and over all nodes.

The two measures defined in Eqs. (2.13) and (2.14) can be combined to produce a fairness condition, namely:

$$\frac{P_y(i)}{C_g(i)} = K \qquad \forall\ i = 1,\ldots,R \tag{2.15}$$

where K is a constant to be determined.

One pleasing property of this criterion is the fact that virtual circuits which are not interfering with anybody (small value of C_g) can send nearly as much traffic as they need. On the other hand, circuits involved in bottlenecks have a high value of C_g and are penalized more.

Several objective functions can be defined whose optimization satisfies the criterion in Eq. (2.16) In [Marc84] it was proposed to minimize the following objective:

$$F_1 = \frac{1}{R-1} \sum_{i=1}^{R} \frac{[P_y(i) - K\, C_g(i)]^2}{1+K^2} \tag{2.16}$$

It was shown that a better result is obtained if, instead of $C_g(i)$, a normalized measure of congestion per link is employed, i.e., the denominator of (2.16) is replaced by $C_g(i) / L_i$ where L_i is the length of the path of virtual circuit i.

This method was applied to optimize fairness in a variety of networks, and several properties were observed:

i. the method can be applied to both input rate control and window flow control optimization;

ii. the method explicitly attempts to avoid unfair situations both in delay and in throughput;

iii. the actual traffic demand of each user is accounted for in assigning throughputs;

iv. the global performance of the network, in terms of total throughput and average delay, compares favorably with other methods.

One disadvantage of this scheme is the difficulty in dealing with route optimization. Even for direct input rate control, the function F_I is not easy to handle in a joint optimization of routes and flow control.

The congestion caused by a given user can also be measured in terms of the penalties suffered by other users. This notion was explored in [Gerl84].

The fairness measure utilized is

$$F_{GCM} = \sum_{i=1}^{R} P_y(i)$$

and λ_i^{id} was replaced by d_i, the demand of user i in Eq. (2.13).

The problem is to optimize both routes and user throughputs which minimize F_{GCM}, subject to a constraint on the average network delay. Flow control is assumed to be direct input rate control. An efficient algorithm is proposed to find the global minimum. The algorithm was applied to several networks, leading to results that exhibit properties typical of this class of fairness schemes. This is fully explored in Chapter 4.

2.2 Constraints

For specific design problem and application, it is logical to impose restriction on some of the variables. This is done by using constraints in our problem and we will give a list of the constraints that apply to our study.

2.2.1 Flow Constraint

$$f_j \leq C_j \qquad j = 1,\ldots,NL$$

where f_j is the sum of flow from users traversing link j and C_j is the capacity of link j.

This is the capacity constraint stating that the total flow in link j cannot exceed the capacity of link j. This is a linear constraint with respect to the variable λ_i.

2.2.2 Delay Constraint

$$T \leq T_{max}$$

where T_{max} is the maximum admissible average delay.

This constraint put an upper bound on the network average delay. It can be shown that this is also a linear constraint with respect to the variable λ_i for our model.

2.2.3 Throughput Constraint

$$\lambda_i \leq d_i \qquad\qquad i = 1,\ldots,R$$

where d_i is the demand of user i.

This constraint states that users will not get more throughput than what they asked for and obviously this is linear with respect to λ_i.

2.3 Problem Statement

Our objective is to design control algorithms such that our definition of fairness can be achieved. Two forms of flow control, namely the input rate flow control and the window flow control are commonly used. In the input rate flow control, we assume we can limit the input rate directly and in the window flow control we use window sizes as an indirect way to limit amount of traffic allowed into network. Also two forms of routing are used, namely the bifurcated and non-bifurcated routing. In the bifurcated routing, the traffic is split up into several parts, each part being directed on a different path from the source to the destination. In the non-bifurcated routing, traffic is constrained to follow a single path.

In computer networks, individual input rates are variables of interest for input rate control procedure. Virtual circuit window sizes are the network design variables to be optimized for window flow control procedure. If these two forms of flow control are combined with routing as a design parameter, four sets of

problem can be formed with fairness as objective as follows:

Optimize measure of fairness over variables -

1. fixed routing and input rates

2. fixed routing and window sizes

3. multipath routing and input rates

4. multipath routing and window sizes

with or without delay and throughput constraints.

Users can be grouped and different emphasis with priorities can be put on the measure of fairness for different groups. In addition different delay constraints can also be imposed on individual users or on different groups of user, e.g., maximum delay constraint on interactive traffic and no delay or very large delay constraint on other traffic.

We may now define our fairness optimization problems that differ in the set of design variables. In each of these problems it is assumed that we are given the node locations, the external traffic flow requirements d_i, the channel capacities C_j and the maximum average delay T_{max}. We have the following list of problems.

1. Given: Network topology and flow requirements $\{d_i\}$
   Minimize: Measure of unfairness
   With respect to: $\{\lambda_i\}$ and multipath routing

Under constraints: $f_j \le C_j \quad j = 1,...,NL$

$T \le T_{max}$

$\lambda_i \le d_i \quad i = 1,...,R$

2. Given: Network topology and flow requirements $\{d_i\}$

Minimize: Measure of unfairness

With respect to: Window sizes $\{N_i\}$ and multipath routing

Under constraints: $f_j \le C_j \quad j = 1,...,NL$

$T \le T_{max}$

$\lambda_i \le d_i \quad i = 1,...,R$

3. Given: Network topology and flow requirements $\{d_i\}$

Minimize: Measure of unfairness

With respect to: $\{\lambda_i\}$ and multipath routing

Under constraints: $f_j \le C_j \quad j = 1,...,NL$

$T_I \le T_{maxI}$

$T_{II} \le T_{maxII}$

$\vdots \quad \vdots \quad \vdots$

$\lambda_i \le d_i \quad i = 1,...,R$

4. Given: Network topology and flow requirements $\{d_i\}$

Minimize: Measure of unfairness

With respect to: Window sizes $\{N_i\}$ and multipath routing

Under constraints: $f_j \le C_j \quad j = 1,...,NL$

$$T_I \leq T_{maxI}$$

$$T_{II} \leq T_{maxII}$$

$$\vdots \quad \vdots \quad \vdots$$

$$\lambda_i \leq d_i \qquad i = 1,\ldots,R$$

Note that fixed routing eliminates routing as one of the variables to be optimized. When routing is the variable to be optimized, bifurcated routing is implied. Also T_I for $0 \leq I \leq R$ can be either the individual user end-to-end delay or the group average delay.

We assume centralized routing and static assignments of window sizes at set up time. In out study, for input rate control we use open network model where input rates are the variables to be optimized while for window flow control we use closed network model where window sizes are the variables to be optimized. The description for the particular model will be given in the next chapter.

2.4 Summary

In this chapter we presented our overall problem with unfairness as a measure to be minimized. The fairness measure given in 2.1.4.3 appears to be most flexible of all, featuring delay-throughput tradeoffs, sensitivity to user demands, and unconstrainted performance for non-interfering users. This measure permits us to carry out an actual system optimization which is the subject of subsequent chapters.

CHAPTER 3

REVIEW OF NETWORK MODELS AND METHODS FOR SOLVING THE MODELS

3.1 Open Network Models

In this section we will describe the open Markovian model (open in the sense external arrivals and departures are allowed). We limit our scope of discussion to the class of queueing networks suitable for modeling store-and-forward networks. This class of network models is only a subset of network models that have a product form solution [Jack63, Bask75, Reis75]. Markovian network of queues will be presented in 3.1.1 and an input rate flow control model will be given in 3.1.2.

3.1.1 Network of Queues Model

Prior to 1970 much of the research on queueing models was focused on single resource models. The necessity of multiple resource models brought forth the queueing network models. In general, a queueing network contains an arbitrary number of service centers organized in such fashion that a customer, after receiving service from an service center, will move to another service center according to some transition probabilities. This allows multiple independent resources and the sequential use of these resources to be modeled.

We consider an arbitrary open Markovian network of queues. An example is given in Figure 3.1.

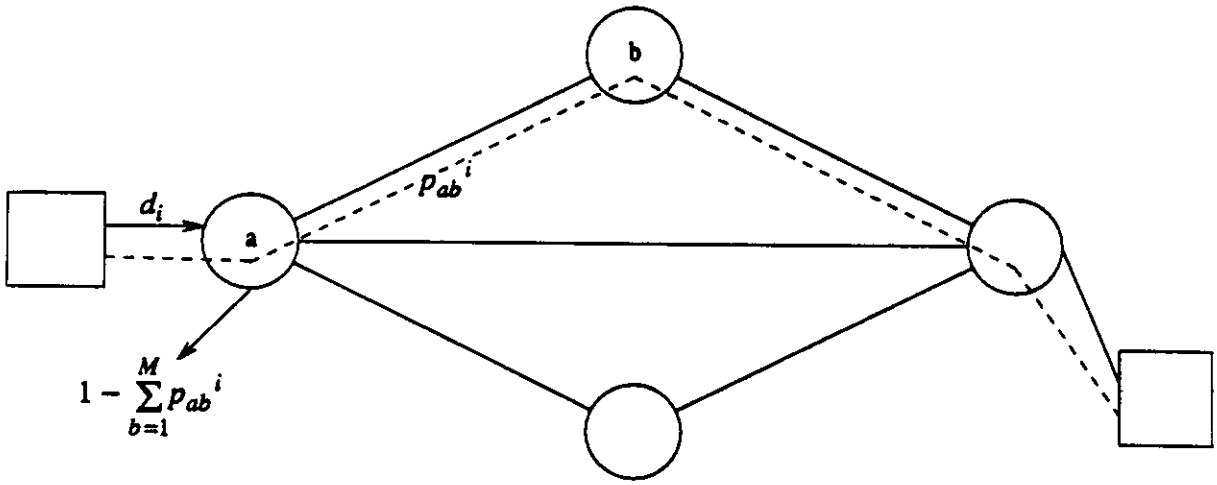

Fig. 3.1 Example of an open network.

The model consists of M nodes, each node contains servers with an exponential mean service time. Chain (i.e., source and destination pair) i customer arrives externally to the node with a Poisson arrival rate. After receiving service at node a the customer goes to next node b with some probability $p_{ab}{}^i$ or leaves the network with probability $1 - \sum_{b=1}^{M} p_{ab}{}^i$. Feedback is allowed in the model.

It can be shown that for networks with feedback the arrival processes to the nodes will not be Poisson in general. Jackson [Jack57] was able to show that the joint distribution can be factored into the product of each of the marginal distributions. This is referred to as the product form solution for Markovian queues in equilibrium.

We will use the Markovian network of queues as our store-and-forward packet switching network model in the next section.

3.1.2 Input Rate Flow Control Model

Product-form queueing network models [Bask75] can be applied to the performance analysis of store-and-forward networks. In packet switching network, a packet enters the network at the source node, visits several nodes along the communication channel, and leaves the network at the destination node (the dashed line in Figure 3.1). However queueing network models have a number of limitations. Adaptive routing cannot be modeled. Only deterministic or random routing can be modeled. Also control traffic is not accounted for in the model.

In order to apply product-form queueing network for analysis of store-and-forward network, several simplified assumptions are necessary. The key one being the independence assumption originally introduced by Kleinrock [Klei64] : Each time a packet is received at a node in the network, its length is determined independently from the probability density function

$$b(x) = \mu e^{-\mu x} \qquad x \geq 0 \tag{3.1}$$

where $1/\mu$ is the mean packet length.

This states that the exponential distribution is used in generating a new length each time a packet is received by a node. This removes the statistical dependence of the transmission time of a packet at various channel because packets maintain their length as they pass through the network. Without this assumption, the analysis is not mathematically tractable.

Our model for the network consists of M nodes and R chains. The communication channels are modeled by first-come-first-served (FCFS) servers and the routing behavior of the chain i is modeled by a first-order Markov chain with transition probabilities ${p_{ab}}^{i}$ $a,b = 1,...,M$.

We assume that the arrival of chain i packets is a Poisson process with rate d_i packets per second. Therefore $d = d_1 + d_2 + \cdots + d_R$ is the total external arrival rate to the network. If the corresponding throughput rates are denoted by λ_i for $i = 1,...,R$, we must have

$$\lambda_i \leq d_i \qquad i = 1,...,R$$

Since external arrivals to a chain may not always be accepted due to the input flow control, the throughput may be smaller than the corresponding arrival rate. The difference between λ_i and d_i is the rate at which chain i arrivals are rejected.

The mean rate γ_{ik} of chain i packet arrivals to node k is found by solving

$$\gamma_{ik} = d_i u_{ik} + \sum_{j=1}^{M} \gamma_{ij} p_{jk}{}^{i} \tag{3.2}$$

where

$$u_{ik} = \begin{cases} 1 & \text{if } k \text{ is the first service center of chain } i \\ 0 & \text{otherwise} \end{cases}$$

The arrival rate of packets from all chains to service center k is

$$\gamma_k = \sum_{i=1}^{R} \gamma_{ik} \tag{3.3}$$

If the chain is open (i.e., allowing external arrivals and departures), the number of packet in a chain can range from 0 to ∞. The equilibrium probability of the network state has a product form solution [Bask75] and the measurements of network performance (typically throughput and network delay) can be easily obtained from this equilibrium probability.

In chapter 4 this input rate flow control model will be used to obtain the mean end-to-end delay over all packets in the network. This delay is the basic formula used in our formulation in the fairness optimization in open networks.

3.2 Closed Network Models

In this section we will describe the closed network model (in the sense a fixed and finite number of customer in the system without external arrivals and

departures). As in the case of open network model we limit our scope to the class of queueing networks suitable for modeling store-and-forward networks. Multiple closed chain model and window flow control model will be presented in 3.2.1.and 3.2.2 respectively.

3.2.1 Multiple Chain Model

This model corresponds to Jackson's model in which $\sum_{b=1}^{M} p_{ab}{}^{i} = 1$ for all a and $d_i = 0$, i.e.,

$$\gamma_{ik} = \sum_{j=1}^{M} \gamma_{ij} p_{jk}{}^{i} \qquad k = 1,...,M \tag{3.4}$$

where γ_{ik} is the packet arrival rate to service center k for closed chain i

The model consists of a finite number of service centers and a fixed number of customers which are competing for service at the various centers. Therefore the number of state of the queueing network is finite.

Let the state of the queueing network be denoted by

$$\chi = (\mathbf{n}_1,...,\mathbf{n}_M)$$

where

$$\mathbf{n}_k = (n_{1k}, \ldots, n_{Rk})$$

where n_{ik} is the number of chain i packets at service center k and $\mathbf{n} = (n_1, \ldots, n_M)$

From Eq. (3.4) the packet arrival rate to service center k for a closed chain i can only be determined within a multiplicative constant. In our model we consider FCFS service center k working at a constant rate of C_k bits per second for chain i packet. The traffic intensity of chain i packet at service center k is given by

$$\rho_{ik} = \frac{\gamma_{ik}}{\mu C_k} \tag{3.5}$$

where $\frac{1}{\mu}$ is the mean packet length

and the overall traffic intensity of service center k is given by

$$\rho_k = \sum_{i=1}^{R} \rho_{ik} \tag{3.6}$$

Let N_i be the population size of chain i and

$$\mathbf{N} = (N_1, \ldots, N_R)$$

The equilibrium network probability has been derived and has the product form solution [Bask75]

$$P(\chi) = \frac{1}{G(\mathbf{N})} \prod_{k=1}^{M} p_k(\mathbf{n}_k) \tag{3.7}$$

where

$$p_k(\mathbf{n}_k) = n_k! \prod_{i=1}^{R} \frac{\rho_{ik}^{n_{ik}}}{n_{ik}!}$$

and $G(\mathbf{N})$ is the normalization constant and is equal to

$$G(\mathbf{N}) = \sum_{\chi \in \delta(\mathbf{N})} \prod_{k=1}^{M} p_k(\mathbf{n}_k) \tag{3.8}$$

The summation is done over the set of feasible network states:

$$\delta(\mathbf{N}) = \left\{ \chi : \sum_{k=1}^{M} \mathbf{n}_k = \mathbf{N} \right\} \tag{3.9}$$

From Eq. (3.7) the following can be derived for FCFS server:

$$P(\mathbf{n}) = \frac{1}{G(\mathbf{N})} \prod_{k=1}^{M} \rho_k^{n_k} \tag{3.10}$$

From this equilibrium probability of the network state, performance measures of throughputs and delays of individual virtual circuits can be computed [Buze73]. Both the throughput $\lambda_{mk}^{*}(\mathbf{N})$ and mean number of customers $q_{mk}^{*}(\mathbf{N})$ of chain m at a FCFS service center k can be obtained in terms of variants of the normalization constant $G(\mathbf{N})$ [Reis75]. Thus the throughput of chain m is equal to the throughput of the chain at the source service center and the mean delay incurred by chain m customer at service center k is given by Little's formula to be

$$\frac{q_{mk}^{*}(\mathbf{N})}{\lambda_{mk}^{*}(\mathbf{N})}$$

From this the mean end-to-end delay can be obtained.

We notice that the normalization constant $G(\mathbf{N})$ is the sum of an extremely large number of product terms when R is large or when the set of $\{N_i\}$

are large. We will present an algorithm in 3.3.1.2 to find the performance measures such as the mean throughputs and delays without calculating this normalization constant $G(\mathbf{N})$.

3.2.2 Window Flow Control Model

The majority of today's operational packet switching networks provide virtual circuits that use flow control and non-bifurcated routing. In the corresponding queueing network model, only end-to-end (window) flow control between source and destination nodes is considered. This form of flow control is important in synchronizing different source input rate and destination acceptance rate. Figure 3.2 shows a queueing network model of a store-and-forward packet switching network with R virtual circuits.

Each virtual circuit corresponds to a source destination pair. Packets in the same virtual circuit follow a fixed path which may be chosen probabilistically from the possible set of paths between source and destination. They pass through intermediate nodes which are modeled as FCFS servers with exponentially distributed service times. Acknowledgement from the destination to the source indicating receipt of a packet can be modeled by an independent random variable or assumed to be instant i.e., no delay for the acknowledgement packet.

The window size of a virtual circuit is the maximum number of packets in transit at the same time within the circuit in the network. If the number of packets in transit within the circuit is equal to its window size then the source is

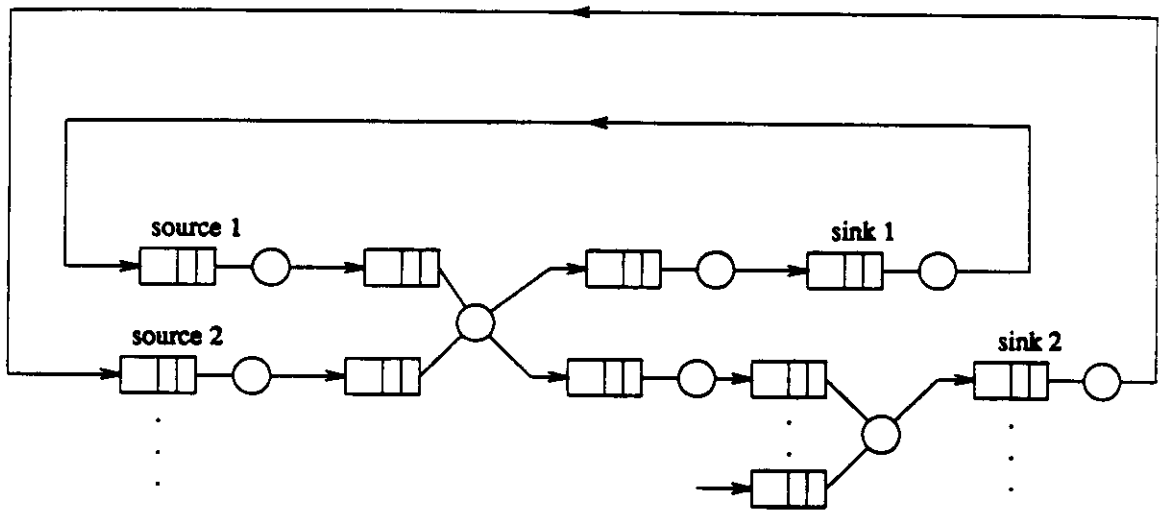

Fig. 3.2 Example of a closed network.

blocked. Thus window flow controlled virtual circuits are modeled as multiple closed chains with a fixed number of circulating customers. Each customer corresponds to a permit. When a packet is admitted at the source, it carries a permit with it. The packet is blocked when permits run out at the source. The permit will be returned as an ACK to the source when the packet arrives at the destination. Thus the circulating permits (window size) of a virtual circuit correspond to the circulating customers of a closed chain.

The source, modeled by a single-server queue with FCFS discipline, is described by a packet length distribution and a packet interarrival time distribution. When the source is not blocked, it generates a new packet for input into the

network at rate d_i. When the source is blocked the packet is lost i.e., the input rate is zero and we avoid having a queueing system with blocking by modeling the source as a loss system. Therefore the delay that we calculate is the transit delay from the time of admission to the time of arrival at the destination. The destination, also modeled by a single-server queue with FCFS discipline, is described by absorbing time distribution. Modeling this absorbing time delay is not important since the ACK packet consumes relatively small amount of resources in the network. However the ACK traffic may be accounted for by reducing the channel capacities by amounts equal to the throughputs of ACK packets as suggested by Reiser [Reis79]

In summary, a packet switching store-and-forward network with flow controlled virtual circuits is modeled by a multiple closed chain queueing network. Each closed routing chain corresponds to a virtual circuit with the chain population size equal to the circuit window size. This is a general model which we adopt for window flow controlled packet switching network. In Chapter 5 a formulation for fairness optimization in closed networks with this model will be presented.

3.3 Analysis and Methods of Solution

We mentioned in 3.2 that the normalization constant $G(\mathbf{N})$ is the sum of an extremely large number of terms when R is large and also when the population sizes are large. The computational time and space requirements are

extremely large. In recent years, approximate solutions have been introduced to lessen this space and time requirements. We will review these in 3.3.2. The exact solution where the approximations will be based on is presented next.

3.3.1 Exact Analysis

Three exact solution, namely convolution, tree convolution algorithms and Mean Value Analysis (MVA) algorithm will be given. We assume FCFS servers for our purpose of discussion.

3.3.1.1 Convolution Algorithm

We mentioned in 3.2.1 that $G(\mathbf{N})$ in (3.8) is computed where the summation is done over set of feasible network states $\delta(\mathbf{N})$. This is done with the convolution algorithm. The algorithm derives its name in the way $G(\mathbf{N})$ is calculated from $p_k(\mathbf{n}_k)$ similarly to the convolution of discrete probability distributions. If we represent each $p_k(\mathbf{n}_k)$ in (3.8) by a R-dimensional array between $\mathbf{0}$ and $\mathbf{N}$, and $G_k(\mathbf{n})$ is a real-valued function indexed by an array with R dimensions which contains the summation over the first k service centers for each possible vector of numbers of customers $\mathbf{n}$ in those centers for $\mathbf{0} \le \mathbf{n} \le \mathbf{N}$.

$$G_1(\mathbf{n}) = p_1(\mathbf{n})$$

or written as

$$G_1 = p_1$$

and $G_2(\mathbf{n})$ is defined as follows [Chan75] :

$$G_2(\mathbf{n}) = \sum_{j_1=0}^{n_1} \cdots \sum_{j_R=0}^{n_R} p_1(\mathbf{j}) p_2(\mathbf{n} - \mathbf{j}) \qquad \text{for } 0 \le \mathbf{n} \le \mathbf{N}. \tag{3.11}$$

or written as

$$G_2 = p_1 \textcircled{*} p_2 = p_2 \textcircled{*} p_1 \tag{3.12}$$

In a similar fashion the algorithm finds $G(\mathbf{N})$ by performing the convolution sequentially:

$$G_i(\mathbf{n}) = \sum_{j_1=0}^{n_1} \cdots \sum_{j_R=0}^{n_R} G_{i-1}(\mathbf{j}) p_i(\mathbf{n} - \mathbf{j}) \qquad \text{for } 0 \le \mathbf{n} \le \mathbf{N}. \tag{3.13}$$

or written as

$$G_i = G_{i-1} \textcircled{*} p_i \qquad \text{for } i = 2,\ldots,M$$

$G(\mathbf{N})$ is then the element indexed by N in the following array

$$G(\mathbf{N}) = G_M(\mathbf{N}) = p_1 \textcircled{*} \cdots \textcircled{*} p_M \tag{3.14}$$

The time requirement to compute the array $G_M(\mathbf{N})$ and hence $G(\mathbf{N})$ is of the order of

$$(M - 1) \prod_{i=1}^{R} \frac{(N_i + 2)(N_i + 1)}{2} \tag{3.15}$$

while the space requirement is of the order of

$$2\prod_{i=1}^{R}(N_i + 1) \tag{3.16}$$

Both requirements grow exponentially with R.

3.3.1.2 MVA Algorithm

The MVA algorithm [Reis80] bypasses the evaluation of the normalization constant $G(\mathbf{N})$ and computes the performance measures of mean queue lengths and chain throughputs directly. We begin by defining the following notations that will be used in this chapter and chapter 5.

R: number of closed routing chains

N_i: population of chain i

N: population array $(N_1, \ldots, N_R)$

$L_k(\mathbf{N})$: mean number of customers at center k with population N

$L_{ik}(\mathbf{N})$: mean number of customers of chain i at center k with population N

$\lambda_i(\mathbf{N})$: mean throughput of chain i with population N

$W_{ik}(\mathbf{N})$: mean waiting time (queueing time + service time) of chain i customer at center k with population N

x_{ik}: mean service time of chain i customer at center k

θ_{ik}: visit ratio of chain i at center k

a_{ik}: relative utilization of chain i at center k

$\mathbf{e}_i$: R dimensional vector where ith element is one and other elements are zero

For FCFS servers multiple chain network to have a product form solution, all chains visiting center k must have the same exponential service time distribution at the center i.e., $x_{ik} = x_k$ which is the mean service time at center k [Bask75].

The key element to MVA is a recursive expression for the mean response time of a service center in terms of measures for the network with one less customer [Reis80]. Using Little's formula first for chain i, and then for chain i at center k, the following relations hold:

$$W_{ik}(\mathbf{N}) = x_{ik}[1 + L_k(\mathbf{N} - \mathbf{e}_i)] \qquad 1 \le i \le R, 1 \le k \le M, \mathbf{N} \ge \mathbf{e}_i \tag{3.17}$$

$$\lambda_i(\mathbf{N}) = N_i \sum_k W_{ik}(\mathbf{N}) \qquad 1 \le i \le R, 1 \le k \le M, \tag{3.18}$$

$$L_{ik}(\mathbf{N}) = \lambda_i(\mathbf{N}) W_{ik}(\mathbf{N}) \qquad 1 \le i \le R, 1 \le k \le M, \tag{3.19}$$

Eqs. (3.17) to (3.19) can be solved by recursion starting with $L_{ik}(0) = 0$. Only mean values are obtained using this method and the computational complexity is of the order $\prod_{i=1}^{R}(N_i + 1)$ which grows exponentially with R.

3.3.2 Approximate Analysis

We have already remarked that the computational time and space requirements of both the (sequential) convolution algorithm and the MVA algorithm grow exponentially with R. Approximate methods of solution are seeked for network models with dozens of virtual circuits. Three approximate techniques, namely the Schweitzer's, Linearizer approximation and a heuristic based upon the MVA algorithm will be presented.

3.3.2.1 Schweitzer's Approximation

Schweitzer proposed the following approximation to avoid the recursive computation in the MVA equations: [Schw79]

$$L_{ik}(\mathbf{N} - \mathbf{e}_c) = L_{ik}(\mathbf{N}) \qquad i \neq c \tag{3.20}$$

and

$$L_{ck}(\mathbf{N} - \mathbf{e}_c) = \frac{N_c - 1}{N_c} L_{ck}(\mathbf{N}) \qquad i = c \tag{3.21}$$

The approximation states that the mean fraction of a chain's customers in service center k does not change if there is one less chain c customer in the network i.e., with large population, there will be no significant changes in mean queue lengths by one less customer.

Summing over i with Eqs. (3.20) and (3.21) yields

$$L_k(\mathbf{N}-\mathbf{e}) = \sum_{i=1}^{R} L_{ik}(\mathbf{N}-\mathbf{e})$$

$$= 1 + L_k(\mathbf{N}) - \frac{L_{ck}(\mathbf{N})}{N_c} \tag{3.22}$$

Substitute this into (3.17) will give

$$W_{ik}(\mathbf{N}) = x_{ik}\left[1 + L_k(\mathbf{N}) - \frac{L_{ik}(\mathbf{N})}{N_i}\right] \tag{3.23}$$

With Eqs. (3.18) and (3.19) this forms a set of simultaneous nonlinear equations in the unknowns $W_{ik}(\mathbf{N})$. These can be solved in a straightforward way.

3.3.2.2 Linearizer Approximation

Chandy and Neuse [Chan82] proposed a more accurate but more costly algorithm based on Schweitzer's approximation:

Let $D_{ikc}(\mathbf{N})$ denotes the change in the fraction of chain i at queue k resulting from on less chain c customer

$$D_{ikc}(\mathbf{N}) = \begin{cases} \dfrac{L_{ik}(\mathbf{N}-\mathbf{e}_c)}{N_i} - \dfrac{L_{ik}(\mathbf{N})}{N_i} & i \neq c \\ \dfrac{L_{ck}(\mathbf{N}-\mathbf{e}_c)}{N_c - 1} - \dfrac{L_{ck}(\mathbf{N})}{N_c} & i = c \end{cases} \tag{3.24}$$

If we assumed $D_{ikc}(\mathbf{N})$ are known then

$$L_{ik}(\mathbf{N}-\mathbf{e}_c)=L_{ik}(\mathbf{N})+N_i D_{ikc}(\mathbf{N}) \qquad i\neq c \tag{3.25}$$

$$L_{ck}(\mathbf{N}-\mathbf{e}_c)=\frac{N_c-1}{N_c}L_{ck}(\mathbf{N})+(N_c-1)D_{ckc}(\mathbf{N}) \qquad i=c \tag{3.26}$$

These reduce to Eqs. (3.20) and (3.21) if $D_{ikc}(\mathbf{N})$ are assumed to be 0.

The algorithm estimates the error $D_{ikc}(\mathbf{N})$ from Schweitzer's approximation by solving R networks identical to the original one, but each network with one less customer at chain c. It refines the value of this error in successive iterations with zero error as a starting solution.

3.3.2.3 MVA Approximation

As in the case of Schweitzer's and Linearizer approximation, MVA approximation replaces the recursion by an iteration method [Reis79].

Let D_{ikc} denotes the change in mean queue length of chain i at queue k resulting from on less chain c customer

$$D_{ikc}=L_{ik}(\mathbf{N})-L_{ik}(\mathbf{N}-\mathbf{e}_c) \qquad 1\le i\le R,\ 1\le k\le M \tag{3.27}$$

If we assume D_{ikc} are known, then the MVA equations can be written as

$$W_{ik}(\mathbf{N})=x_{ik}[1+\sum_i(L_{ik}(\mathbf{N})-D_{ikc}] \tag{3.28}$$

$$\lambda_{ik}(\mathbf{N})=\frac{N_i}{\sum_k W_{ik}(\mathbf{N})}$$

(3.29)

$$L_{ik}(\mathbf{N}) = \lambda_i(\mathbf{N}) W_{ik}(\mathbf{N}) \tag{3.30}$$

The set of equations (3.28) - (3.30) can be solved with a initial set of $\{L_{ik}\}$ and iterating sequentially until convergence is observed. Reiser [Reis79] gives a heuristic for evaluating D_{ikc} as follows:

Assuming that the chain with on less customer is affected the most, for $i \neq c$

$$D_{ikc} = 0 \tag{3.31}$$

i.e.,

$$L_{ik}(\mathbf{N}) = L_{ik}(\mathbf{N} - \mathbf{e}_c)$$

the same as Schweitzer's approximation but D_{ckc} is estimated by a single chain network where a chain c customer "sees" only a fraction of the channel capacity [Penn75]. The redefined mean service time for all FCFS servers in the single-chain network is

$$x_{ik} = \frac{x_{ik}}{1 - \sum_{j \neq i} \lambda_i(\mathbf{N}) x_{jk}} \tag{3.32}$$

Let $L_k(N_i)$ denotes the mean queue size of server k in the single-chain network with service time given by (3.32) Then we set

$$D_{ckc} = L_k(N_i) - L_k(N_i - 1) \tag{3.33}$$

(3.31) together with (3.33) constitute the heuristic in estimating $W_{ik}(\mathbf{N})$ in (3.28) thus solving the set of equations (3.28) - (3.30).

3.4 Concluding Remarks

In this chapter we reviewed the network models that will be used in our fairness optimization problems. The input rate flow control model is used when the parameter of optimization is the input rate. The model gives us the throughputs and delays that will be used in our downhill technique in achieving the fairness objective. Optimality conditions are obtained for our formulation of the fairness problem and a simple approach is derived to achieve the optimal solution. On the other hand the window flow control model is used when the parameter of optimization is the set of window sizes. The model gives us the throughputs and delays that will be used in achieving the fairness objective. Though the throughputs and delays cannot be calculated from a closed-form formula, they are computed using an algorithmic approach. An approximation technique is derived to obtain the optimal set of window sizes. The details will be given in Chapter 4 and Chapter 5.

The complexity of an exact analysis of the closed network model makes it practically impossible to obtain throughputs and delays for networks with reasonably number of chains. Researchers have long been looking for either approximation solution techniques for large networks or exact solution for networks with

special structure such that the complexity of the problem can be reduced. Recently Lam [Lam83] developed a tree convolution algorithm which requires substantially less space and time in calculating the normalization constant for product-form queueing networks. The algorithm exploits the two common properties in communication networks, namely the sparseness property i.e., chains visit only a small fraction of all queues in the network and the locality property i.e., chains are clustered in certain parts of the network.

The algorithm is based on two ideas. First the convolution in (3.13) can be performed in any order to obtain $G_M(\mathbf{N})$. Specifically it places the array $\{p_m\}$ at the leaf nodes of a tree and the nodes are visited according to some order of tree traversal. Second the concept of fully covered, noncovered and partially covered of a chain which visits a set of service centers with respect to a subset of the M service centers is introduced. It is noted that not all the elements of G_m in the process of convolution are needed to calculate $G(\mathbf{N})$. Only those belonging to partially covered chains in subsets are involved. Thus if convolution is done in a "clever" sequence with minimum numbers of partially covered chains, the algorithm will give a substantial saving in time and space. Decision on the sequence is termed "tree planting" which involves the placement of service centers at leaf nodes. A detailed discussion of this algorithm and its complexity is given in [Lam83]

For arbitrary network with no special structure, approximate technique has to be used. An approach based on clustering of chains and service centers is proposed by Edmundo Silva [Silv84]. The method is applicable to queueing networks with single server, infinite server and multiple server service center. The approach is found to be fairly accurate compared to the current existing approximation techniques such as the Schweitzer's and Linearizer approximations. However our solution technique used in the fairness optimization in closed networks is independent of the method used in computing the throughputs and delays. Therefore any new method e.g. the tree MVA algorithm [Tucc85] and methodology in computing the throughputs and delays can be directly used in our solution without any modification.

CHAPTER 4

FAIRNESS OPTIMIZATION IN OPEN NETWORKS

4.1 Introduction

In this chapter, we present the solution to our fairness optimization problem in open networks. Multipath routing is considered together with the flow control procedure. We chose to use the open network model which is appropriate for input rate flow control [Klei76]. We assume we can directly control the input rates which are continuous variables to satisfy our fairness condition. The main difficulty is how to selectively adjust these rates such that the overall fairness condition is met. We will describe our problem in section 4.2. We will present our model in section 4.3 followed by the solution approach in section 4.4. It will be shown that the problem has only one local optimum which consequently is the global optimum. Finally an algorithm for both finding the route and selectively adjusting the user input rates to satisfy the optimality condition is given in section 4.5. A large part of the algorithm contributes to satisfying the optimality condition in an efficient way. The last section of this chapter will be devoted to the experimental results by running the algorithm on a couple of network examples.

4.2 The Problem Description

Packet switching networks, like ARPANET shown in Figure 4.1, is an attractive way for transmitting bursty data traffic. In order to produce an orderly sharing of resources to achieve efficiency for the network, it is necessary to exert some control. Routing and input rate flow control are two forms of control to help to improve the performance of the network.

Traditionally these two forms of control were studied separately; consequently, routing procedures have been designed independently of flow control schemes. They are closely related in such a way that choice of route will change the traffic pattern which in turn will affect level of congestion of the network. A better approach would be to combine the routing and flow control decisions. If routing and flow control procedures are properly designed, efficient use of network resources will be achieved.

One area of concern which were mentioned in Chapter 3 is the fairness issue. The problem is how to design the routing and flow control algorithm which may involve favoring some users over others in throughput in order to achieve a global fairness condition. For example in Figure 4.1, if there is a user session from AMES to RCC5, another user session from TYM93 to MITRE and other session from NBS to NPS, our goal is to find proper routes and the input rates for these sessions such that fairness condition can be met. We will present our model in the next section followed by the solution for our problem.

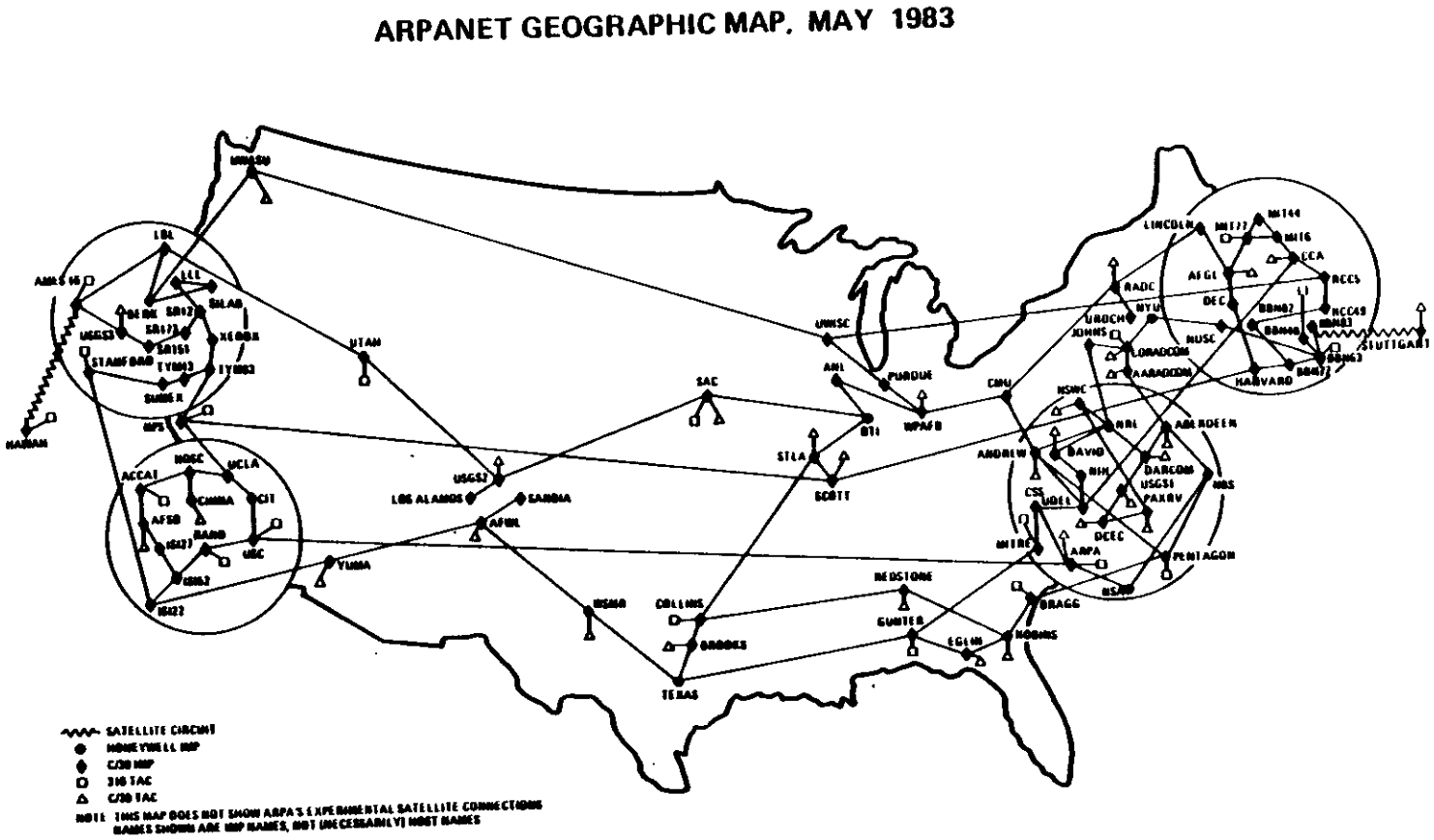

Fig. 4.1 Arpa Network.

4.3 The Model

We use the open network of queues for our integrated routing and flow control problem where the control is imposed on the input rates, i.e., the entry node can directly control the rate of each user session. One important aspect of the solution proposed here is that the fairness issue is embedded in the objective function presented later in this section. We will first look at the queueing aspect of the network.

4.3.1 Delay Analysis

For the queueing aspect, the usual assumptions for open network of queues will be made. These assumptions permit us to model the stochastic flow of packets as a network of independently M/M/1 queues. We assume that R user sessions are present in the network, where each session corresponds to a virtual circuit between a given source and destination. The traffic demand of user i is d_i. Due to flow control, the actual throughput achieved by this user is λ_i where $0 \leq \lambda_i \leq d_i$. Both λ_i and d_i are measured in units of information per unit of time. The communications subnetwork consists of M store-and-forward nodes which are linked by NL links. The capacity of link k is C_k and the flow f_k traversing the link is the sum of all contributions λ_i which traverse link k.

Using the usual assumptions, namely poisson arrivals, exponential packet length distribution, independence between service and arrival times [Klei76], the average network delay T can be expressed as:

$$T = \frac{1}{\sum_{i=1}^{R} \lambda_i} \sum_{j=1}^{NL} \frac{f_j}{C_j - f_j} \tag{4.1}$$

This is very similar to the conventional routing problem formulation [Frat73] except the total traffic $\sum_{i=1}^{R} \lambda_i$ allowed into the network is not a constant.

4.3.2 Objective Function

One possible fairness measure is as follows:

$$P = \sum_{i=1}^{R} \frac{d_i - \lambda_i}{\lambda_i}$$

The formulation of our problem includes minimizing the function P which is the sum of normalized relative difference in the demand and throughput of the user. There are two pleasing properties of this fairness measure, namely the throughput of the individual user cannot be driven to zero otherwise the value of the objective function becomes infinite and the actual throughput achieved by a user tends to be proportional to his traffic demand.

Minimizing this function with all user sessions having the same demands and no other constraints attached will give equal throughput for all user session. This conforms to our discussion of fairness in the previous chapter. Notice that since we minimize P, we can consider P as unfairness or penalty to be minimized rather than "fairness" to be maximized.

4.3.3 The Formulation

We formulate the problem of "fair" routing and flow control as an optimization problem where the objective function is P subject to an average delay constraint. The formulation is now stated:

- Given: Network topology and link capacities
  Demands of user sessions

- *Minimize*: $P = \sum_{i=1}^{R} \frac{d_i - \lambda_i}{\lambda_i}$ (4.2)

- *Over the variables*: $\{\lambda_1 \lambda_2, \cdots, \lambda_R\}$ and *routing*

- *Subject to the constraints*:

$$T \le T_{max} \tag{4.3}$$

$$0 \le \lambda_i \le d_i, \qquad i = 1,\ldots,R \tag{4.4}$$

$$f_j \le C_j, \qquad j = 1,\ldots,NL \tag{4.5}$$

where T_{max} is the maximum average delay tolerated by the network.

Constraint 4.3 expresses the condition that the maximum average network delay must not be exceeded by the average network delay.

Constraint 4.4 expresses the condition that the demand of user must not be exceeded by its throughput.

Constraint 4.5 expresses the condition that the capacity of the link must not be exceeded by its flow.

As stated before the choice of the objective function P was determined by fairness consideration mentioned in the last section. With a little thought, it can be seen that the function P cannot be minimized while one of the user has a throughput much below its desired rate.

In the formulation, it is obvious that, without the delay and capacity constraints, the optimal solution is that each user gets what he requests, i.e., $\lambda_i = d_i$ for all i. This will give $P = 0$ and optimally fair. The delay constraint upper bounded by the maximum average network delay T_{max} is then introduced. The formulation has two underlying considerations in mind, namely what the network considers fair as opposed to what the individual user considers fair. As mentioned before the individual user's point of view of fairness is incorporated in the objective function. The network point of view of fairness is indirectly given in the average network delay.

4.4 The Optimal Solution

We will give our approach in solving the problem formulated in the last section. We then derive the optimality conditions which are necessary in designing an efficient algorithm in finding the global optimal point.

4.4.1 The Approach

The formulation presented in the last section was derived by the combined consideration of two well-known performance measures, namely throughput and delay with the objective of fairness. Routing was incorporated in the formulation of the problem. Optimal routing will reduce the average network delay. Therefore without violating the delay constraint, scaling up the throughputs uniformly will increase the total throughput. This scaling will help minimizing the objective function. This property allows us to iterate between routing and flow control to minimize the average network delay and the objective function by uniformly scaling up the throughputs.

It is easy to show that objective function defined in Eq. (4.2) is a convex function with respect to $\{\lambda\}$. The constraints in Eq. (4.4) and Eq. (4.5) where f_j is the sum of all contributions λ_i traversing link k are linear and therefore convex. Also from Eq. (4.1), constraint (4.3) can be written as

$$\sum_{j=1}^{NL} \frac{f_j}{C_j - f_j} - T_{\max} \sum_{i=1}^{R} \lambda_i \leq 0$$

The left hand side of the inequality can be interpreted as the sum of convex functions and is therefore convex in the routing variables and the input rates. Hence, the constraint set defined by Eq. (4.3) Eq. (4.4) and Eq. (4.5) is convex.

It is well known that the stationary point of a convex function, defined over a convex set, is the global minimum of the function over the set. In the next section we will present an efficient algorithm for finding the stationary point which is, therefore also the global minimum.

The solution approach consists of finding the stationary point using a downhill technique. Namely, at each iteration the routing pattern is optimized (to yield minimum delay) and user throughputs are adjusted (to improve fairness, yet maintaining the delay fixed), until the optimality conditions are met. When this happens, we know that we have a global minimum.

4.4.2 Optimality Conditions

Two separate conditions must be satisfied for optimality; namely, a condition on the routing pattern and a condition on user throughputs.

A necessary condition on the routing pattern is that such pattern yield the minimum delay when user throughputs are kept fixed. This condition must hold unless $\lambda_i = d_i$ for $i = 1,...,R$.

Proof:

Let us assume we have a feasible solution, with $T = T_{max}$. The routing pattern, however, is not optimal; that is, there exists another pattern which provides a delay $T' < T_{max}$. Furthermore, let us assume that there exists at least one user, say i, for which $\lambda_i < d_i$.

Under these assumptions, if we choose the improving pattern, we can scale up some of the user throughputs until the delay constraint T_{max} is satisfied with equality. The throughput increase clearly improves the objective P in Eq. (4.2). Thus, we have proved that a non optimal routing pattern cannot yield an optimally fair solution, unless all user throughput demands are satisfied as equalities; i.e., $\lambda_i = d_i$ for all $i = 1,...,R$.

Next, we consider the condition on user throughputs. Let w_i be the partial derivative of the average network delay with respect to the throughput of user i,

$$w_i = \frac{\partial T}{\partial \lambda_i} \tag{4.6}$$

A necessary condition which the set $\{\lambda\}$ must satisfy in order to minimize (4.2) subject to (4.3), (4.4) and (4.5) is that, for all $i=1,...,R$:

$$\nabla_i = \frac{\lambda_i^2 w_i}{d_i} \begin{cases} = K & \text{if } i \notin D \\ \leq K & \text{if } i \in D \end{cases} \tag{4.7}$$

where K is a constant to be determined and D is the set:

$$D = \{ i \mid \lambda_i = d_i \} \tag{4.8}$$

Proof:

Let users j and k be such that $\nabla_j > \nabla_k$. It will be shown that the objective function P can be reduced by reducing user j's throughput and increasing k's.

Assume first that λ_j is reduced by an (infinitesimal) amount $\Delta\lambda$. From the definition of w_j, the average network delay decreases by approximately:

$$\Delta T \cong w_j \Delta \lambda_j$$

After this reduction, λ_k can be increased by:

$$\Delta \lambda_k \cong \frac{\Delta T}{w_k} \cong \frac{w_j \Delta \lambda_j}{w_k} \tag{4.9}$$

without exceeding the original value of T.

Let us now evaluate the variation of the objective function ΔP due to the changes in both λ_j and λ_k. From (4.2), it can be seen that,

$$\frac{\partial P}{\partial \lambda_i} = -\frac{d_i}{{\lambda_i}^2}$$

Thus:

$$\Delta P = -\frac{d_k}{{\lambda_k}^2} \Delta \lambda_k - \frac{d_j}{{\lambda_j}^2} (-\Delta \lambda_j) \tag{4.10}$$

Using (4.9) in the last expression, we get:

$$\Delta P = \Delta \lambda_j \left(\frac{d_j}{{\lambda_j}^2} - \frac{d_k w_j}{{\lambda_k}^2 w_k}\right) \tag{4.11}$$

Since ∇_j was assumed larger than ∇_k, we have:

$$\frac{d_j}{{\lambda_j}^2} < \frac{w_j d_k}{w_k {\lambda_k}^2}$$

which from (4.10), implies that $\Delta P < 0$. This means that by transferring traffic

from j to k, the objective function can always be improved when $\nabla_j > \nabla_k$.

This proof can be generalized to show that as long as a user m has a value of ∇_m larger than other (flow controlled) users, the objective can be decreased. This improvement is accomplished by decreasing traffic of user m and increasing that of other users (except those which have attained the constraint $\lambda_i = d_i$) until ∇ is the same for all users, again except for those which belong to the set D defined in (4.8).

Note that Eq. (4.7) could have been written directly as the Kuhn-Tucker condition for the problem. We chose a different derivation to provide a better physical interpretation of the solution.

The optimality condition just defined reflects also the fairness properties of the method. Basically it says that a user session which has a high value of λ_i / d_i, must have a small value of w_i. The quantity w_i, in some sense indicates how much congestion will be caused in the network by an increase in user i's throughput. Therefore, the solution to the integrated routing and flow control problem posed in this section satisfies the fairness concept defined in [Wong] :" a user should not be able to congest the network more than it is being penalized (in terms of throughput limitation) by the network controlling mechanism.

We now show that w_i can be readily computed if the routing pattern was optimized using the Flow Deviation method [Frat73]. From Eq. (4.1) and Eq. (4.6) we find:

$$w_i = \frac{1}{\sum_{i=1}^{R} \lambda_i} \left[\frac{\partial}{\partial \lambda_i} \sum_{j=1}^{NL} \frac{f_j}{C_j - f_j} - T \right]$$

The first term within the brackets is the incremental delay caused by a unit increase in user i throughput. If the routing pattern has been optimized, then we know that this incremental delay must be the same on all paths on which user i is currently sending flow. In particular, it is equal to the incremental cost of the shortest path from source to destination for user i, where the cost of each link along the path is defined as the derivative of link delay versus link flow. The cost of the shortest path for each source-destination pair is computed as an intermediate step in the Flow Deviation method. Likewise, the term T is directly available from the routing solution.

Above we showed how to calculate w_i (Step 3 of the algorithm given in the following section) which may turn out to be negative, i.e., increasing λ_i will not increase the average network delay. For these users, we calculate the maximum amount of flow that can be increased without exceeding the capacity of the link which the user traverses. The flow after this increment is used as the upper bound in the bisection method to obtain the new flow without violating the delay constraint.

4.5 The Algorithm

The algorithm to be described has two basic steps. In the first one, the traffic is optimally routed using the Flow Deviation (FD) algorithm [Frat73]. Recall that an optimal routing procedure can be used in two ways. One is to reduce the average delay in the network; the other is to increase the total throughput of the net, keeping the average delay constant. This last property of the FD algorithm is the one which interests us here, since the objective is to minimize the function P in Eq. (4.2).

The other step attempts to implement the optimality condition, given in Eq. (4.7) without changing the average network delay, i.e., it equates the values of ∇ for those users which do not belong to D, keeping $T = T_{max}$. The explanation of this step will be given in section 4.5.2. The algorithm used to generate the optimal solution is outlined below.

Step 1 - Set $n = 0$; let $\lambda_i^{(0)}, i = 1,...,R$ be an initial feasible allocation of throughputs (i.e., $\lambda_i^{(0)} \leq d_i \ \forall \ i$)

Step 2 - Use Flow Deviation to optimize routing and scale up the throughputs until $T \leq T_{max}$. Evaluate $P^{(0)}$.

Step 3 - Compute the weights $w_i^{(n)}, i = 1,...,R$.

Step 4 - Compute $\nabla_i^{(n)}, i = 1,...,R$.

Step 5 - Provide incremental adjustments to the throughputs $\{\lambda_i^{(n)}\}$ so that $\{\nabla_i^{(n)}\}$ moves toward a central value.

Step 6 - Use Flow Deviation to minimize delay and scale throughputs uniformly so that $T = T_{max}$ and routing is optimal. Evaluate $P^{(n)}$.

Step 7 - If $P^{(n-1)} - P^{(n)} < \varepsilon$ (where ε is a properly chosen tolerance), stop and print "Global minimum is found"; else, let $n \leftarrow n + 1$ and go to Step 3.

The reason we go back to Step 3 from Step 7 is that both adjustment of flow in Step 5 and scaling in Step 6 changes λ_i, $w_i = \frac{\partial T}{\partial \lambda_i}$ and subsequently ∇_i. Therefore λ_i has to be readjusted again with a new constant K.

It can be seen that at each iteration, the value of the objective function is reduced. The flow deviation step allows the input rates to be increased (and hence a decrease in P) through an improvement in the routing procedure. Step 5 then rearranges the input rates so that they satisfy the optimality condition.

4.5.1 Initial Feasible Solution

The initial feasible solution for the algorithm consists of both the routing and the feasible flows for this route. The initial routing can be user-defined, min-hop or random. Since our problem is minimizing the objective function over the $\{\lambda_i\}$ and the routing and we have proved that there is one global minimum, it does not matter what initial route we choose to start with.

The initial feasible flows for the user session are found by first allowing their flow demands traversing along the initial routes in the network. The bottleneck link is identified. If this link flow exceeds the capacity of the link, all the flows of the user sessions are scaled down such that the flow in the bottleneck satisfies the capacity constraint (4.5).

Thus, the initial feasible solution consists of the initial feasible allocation of throughputs which may be a fraction of the demands in order to satisfy the capacity constraint and any route that we may choose to start with.

4.5.2 Equating ∇_i for Optimality Condition

Step 5 is probably the most delicate step in the algorithm. Step size should be small enough so that relative delay variations are much smaller (i.e., infinitesimal) with respect to relative throughput changes. This condition is necessary in order to guarantee monotonic convergence to the stationary point. On the other hand, reasonably large step sizes are desirable to obtain fast convergence. Below we give the main steps in calculating the new throughputs.

Note that this step basically solves for $\{ \lambda_i \}$ the following system of equation.

$$\frac{\lambda_i^2 w_i}{d_i} = K \qquad i = 1,\ldots,R \tag{4.12}$$

$$\sum_{i=1}^{R} w_i \Delta\lambda_i = 0 \tag{4.13}$$

$$\lambda_i \le d_i \tag{4.14}$$

From (4.12)

$$\lambda_i' = \left[\frac{Kd_i}{w_i}\right]^{1/2} \tag{4.15}$$

where λ_i' are the new throughputs after adjustment.

Substituting in (4.13) gives

$$\sum_{i=1}^{R} w_i \left[\left[\frac{Kd_i}{w_i}\right]^{1/2} - \lambda_i\right] = 0$$

Rearranging terms

$$\sum_{i=1}^{R} (w_i K d_i)^{1/2} = \sum_{i=1}^{R} w_i \lambda_i$$

$$K^{1/2} = \frac{\sum_{i=1}^{R} w_i \lambda_i}{\sum_{i=1}^{R} (w_i d_i)^{1/2}}$$

From (4.15)

$$\lambda_i' = \frac{\sum_{i=1}^{R} w_i \lambda_i}{\sum_{i=1}^{R} (w_i d_i)^{1/2}} \left[\frac{d_i}{w_i}\right]^{1/2}$$

So after Step 2 of the algorithm, K and the new values of λ_i can be computed to satisfy the optimality condition $\nabla_i = K$ without changing T. This is applied to the set of $\{ \lambda_i \}$, $i \notin D$. We may find in some cases $\lambda_i > d_i$ so in general we have to eliminate those set $\{ j \mid \lambda_j \geq d_j \}$ in calculating K. Also after adjusting λ_i, w_i changes which in turn changes ∇_i and capacity constraint in Eq. (4.5) may be violated after the adjustment. Essentially the three constraints (4.3), (4.4), (4.5) have to be taken care of in the algorithm. Details are presented in Appendix A.

4.6 Numerical Results

The integrated routing and flow control algorithm was tested on several network examples. First, the four node topology shown in Figure 4.2 is considered. Two user sessions, from 1 to 4 and from 2 to 3, are present. The demand is $d = 4$ for both sessions. Link capacity is $C = 1$ for all links. Packet length is $b = 1$. The maximum delay is $T_{max} = 4$. Optimal throughputs are 0.91 and 1.05 for sessions 1 and 2 respectively. Optimal delays are 4.6 and 3.4 for sessions 1 and 2 respectively. We note that the "fair" solution provides a slightly higher throughput from 2 to 3 than from 1 to 4. This is consistent with the fact that there are three paths from 2 to 3, while there are only two (disjoint) paths from 1 to 4. Likewise, delay for session 1 is higher than for session 2 since paths for session 1 are longer. The results are shown in Table 4.1.

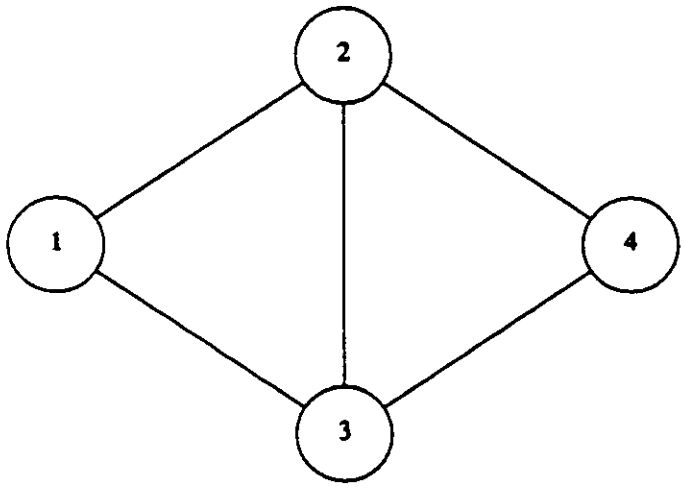

Fig. 4.2 Four node network example.

Table 4.1
Results of the four node example with selective adjustment of user throughtput and $T_{max} = 4$.

Session	Source	Dest.	Demand	Thrup.	Delay
1	1	4	4	0.91	4.6
2	2	3	4	1.05	3.4

Next, the six node topology shown in Figure 4.3 is considered. Link capacity is $C = 4$ for all links. Delay is $T_{max} = 2$ and packet length is $b = 1$.

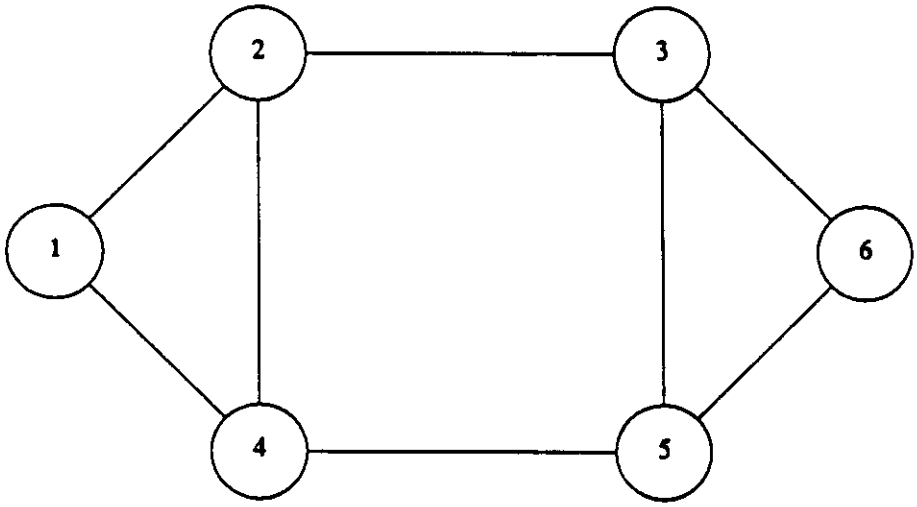

Fig. 4.3 Six node network example.

Eight user sessions are present with uniform demand $d = 6$. The results are shown in Table 4.2. The fairness of the solution can be verified by examining the throughput of sessions which share the same bottlenecks. For example, sessions 1, 6 and 8 share the bottleneck represented by links (1,2) and (3,4). As expected, they achieve comparable throughputs. Session 8 has a slightly higher throughput than the other two sessions; this is justified by the shorter path length (1 hop versus 2 and 3 hops). A similar observation can be made about sessions 2, 5 and 7. It is also interesting to note that sessions 3 and 4 suffer little interfer-

Table 4.2
Six node network results

Session	1	2	3	4	5	6	7	8
Source	1	6	2	5	5	4	3	4
Dest.	6	1	4	3	2	3	2	5
Demand	6	6	6	6	6	6	6	6
Thrup.	2.0	2.1	5.3	4.5	2.1	2.1	2.6	2.6
Delay	3.0	3.0	1.2	1.3	2.8	2.8	1.7	1.8

ence on their network paths and therefore achieve much higher throughput than the other sessions. In all, the solution is both fair and efficient.

Different values of T_{max}, C and d are used in this example. While keeping other parameters the same, larger T_{max} only gives slightly higher individual throughputs due to the capacities of the links. Also if different user demands are used instead of uniform demands, users with higher demands will get higher throughputs. Users with little demands may get all their requirements. This conforms to our notion of fairness in our previous discussion. Our objective function P does not favor users with high demands over users with little demands.

Next we use the example given in [Gerl82] by considering the topology in Figure 4.4. Initially only links 1 through link 7 are present (i.e., link 8 and link 9 are removed). This renders the topology a tree and forces a single-path routing solution. Five user sessions with uniform demand $d = 1$ are considered, as shown by the dashed lines in Figure 4.4. Link capacity is $C = 1$ for all links.

Delay is T_{max} = 20 and packet length is b = 1. The solution is shown in Table 4.3.

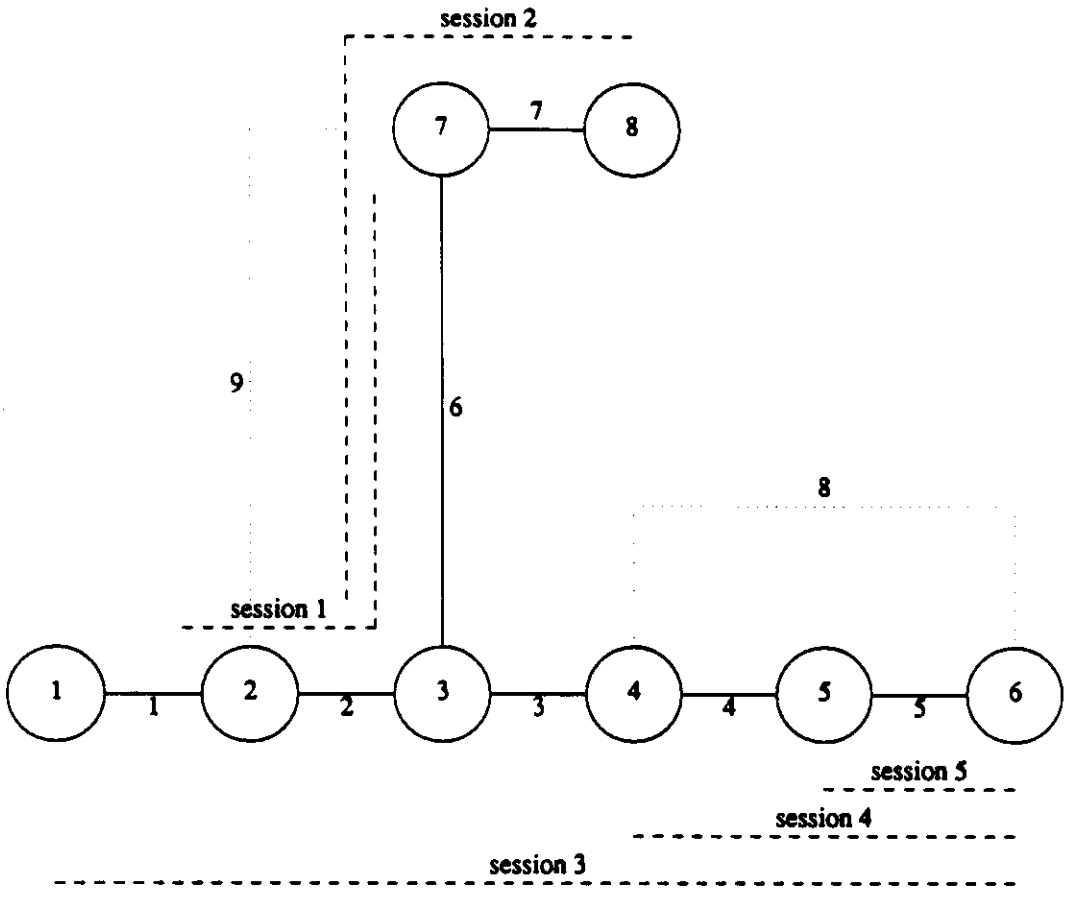

Fig. 4.4 Eight node network example.

The first column showed the result for tree topologies where the fixed path solution corresponds to the optimal routing solution. Link 6 is the bottleneck for session 1 and session 2, and therefore its capacity is equally subdivided between

them. Likewise, link 5 is the bottleneck for session 3, 4 and 5.

Next, we add links to the network to introduce multi-path routing. Adding link 8 relieves the bottleneck at link 5, session 3 is now bottlenecked at link 2, and the residual capacity of the distributed bottleneck leading into node 6 is divided evenly among sessions 4 and 5. Similarly, adding link 9 relieves the bottleneck of sessions 1 and 2, doubling their throughputs; finally, adding both links 8 and 9 doubles the session throughputs from what they were with links 1 through link 7 alone.

Table 4.3
Eight node network results

Session	Source	Dest.	Links 1-7 only	With link 8	With link 9	With links 8 & 9
1	2	7	0.45	0.49	1.00	1.00
2	3	8	0.47	0.50	0.98	0.98
3	1	6	0.32	0.50	0.33	0.66
4	4	6	0.32	0.74	0.33	0.66
5	5	6	0.32	0.75	0.33	0.66

The results showed in Table 4.3 match with what was presented in [Gerl82]. The notion of fairness in [Gerl82] and the notion of fairness presented in this chapter gives the same result except we have the delay constraint here which guarantees the average network delay is within certain limit while in [Gerl82] there is no such delay constraint.

Finally, the ARPANET topology shown in Figure 4.1 is considered. For this topology, link capacity $C = 1$ for all links, and packet length $b = 1$. Thirty user sessions are present, with uniform demand $d = 4$. The delay constraint is $T_{max} = 50$. The optimal solution is shown in Table 4.4. Total network throughput, sum of all individual throughputs, is 13.44. We note that there is a considerable variance in throughput and delay from session to session, due to the different relative positions of the users.

In order to evaluate the effectiveness of our "fair" solution, we compare it with an optimal routing solution in which individual throughputs are uniformly scaled up (instead of being selectively adjusted) so that delay $T_{max} = 50$. The total achievable throughput in the latter case is 10.35, corresponding to individual session throughputs = 0.34. We found that the total throughput with selectively adjustment is always better than that when the throughputs are uniformly adjusted. The "fair" solution is much more efficient. In fact, total throughput is 30% higher. Furthermore only nine (out of thirty) sessions in our solution yield a throughput lower than 0.34!

4.7 Concluding Remarks

In this chapter we define a proper fairness measure which is to be optimized as an objective function. We propose an integrated routing and input rate control algorithm to optimize the objective function such that a given delay constraint is met. The algorithm yields the stationary point which is the global

Table 4.4
Results of ARPANET with selective adjustment of user throughtput and $T_{max} = 50$.

Session	Source	Dest.	Thrup.	Delay
1	AMES	RCC5	.395	42.9
2	TYM93	MITRE	.331	61.1
3	NOSC	IS122	.701	29.8
4	ROBINS	SR151	.517	48.6
5	UCLA	LINCOL	.292	77.2
6	USGS1	MIT44	.780	36.5
7	BBN83	BERK	.447	48.9
8	CIT	UTAH	.344	85.8
9	DEC	TEXAS	.514	39.8
10	AFWL	NSWC	.331	61.3
11	NBS	LINCOL	.675	32.1
12	DEC	GUNTER	.506	48.0
13	RCC49	TYM93	.480	45.8
14	ACCAT	LBL	.352	80.7
15	SR173	WPAFB	.381	66.6
16	USGS3	DCEC	.330	84.6
17	STANFO	BBN83	.323	74.2
18	CHINA	IS122	.330	48.4
19	USC	MIT77	.310	58.3
20	XEROX	HARVAR	.344	47.1
21	BBN82	RAND	.536	45.3
22	CMU	SR12	.507	40.2
23	PURDUE	LOS ALA	.473	44.7
24	UTAH	PENTAG	.395	59.3
25	TYM43	BBN63	.323	65.6
26	NYU	MITRE	.539	39.5
27	MIT6	ANDREW	.645	29.8
28	UWASH	ROBINS	.358	61.5
29	IS122	UROCH	.297	86.9
30	NBS	NPS	.686	36.2

minimum due to the convexity of the objective function and the feasible solutions.

The algorithm is computationally efficient that it optimizes the routing pattern and adjusts the user throughputs alternately until the optimality conditions are met. It is shown to provide higher overall throughghputs while at the same time improving fairness, and maintaining the delay requirement.

One drawback is the assumption that the user input rate can be controlled. In practice the rates are controlled by window or by a passive mechanism known as backpressure. We will use window as the flow control mechanism and give out solution in the optimization in the following chapter. Behaviour of backpressure mechanism is covered in chapter 6.

CHAPTER 5

FAIRNESS OPTIMIZATION IN CLOSED NETWORKS

5.1 Introduction

In this chapter we present the solution to our fairness optimization problem in closed networks. Fixed path routing is considered together with the window flow control procedure. We use the closed multiple chain network model given in Chapter 3 [Reis79]. We assume we can adjust the set of window sizes which are discrete multivariables to satisfy our fairness condition. The main difficulty is how to selectively adjust these window sizes such that the overall fairness condition is met. Complete enumeration of the fairness objective for all the possible combinations of window sizes is infeasible when the number of chain is large (which is typical large in computer networks). Therefore a heuristic approximation method is used in our solution. There are three criteria for the method to be reasonable good. It has to be fast, stable in a sense that it converges and fairly accurate. We show that the heuristic technique we use satisfies all these requirements. We will describe our problem in section 5.2. and our model in section 5.3, which is the same as the one in section 4.3. Finally a heuristic algorithm for finding an approximate set of window size for our problem is given in section 5.4. Part of the algorithm contributes to searching the

solution space in an efficient way and the equations for it are derived in the Appendix B. The last section of this chapter will be devoted to the experimental results by running the algorithm on some network examples.

5.2 The Problem Description

In section 1.2 and section 4.2 we mentioned that flow control is necessary to regulate the flow of packets to avoid congestion in networks. One form of flow control is end-to-end window flow control. In most network implementations, the window can be negotiated at connection set up time (e.g. X.25). In some cases, it can be changed dynamically during a session (e.g. TCP). The issue then arises of how to optimally choose window sizes for the various sessions present in the network. The goal is to optimize the throughput within given delay and fairness constraints. It should be clear, that as window sizes are progressively increased, throughput increases, but queueing delay also increases, since more packets are allowed in the network. The delay increase has a feedback effect on throughput via window flow control. This way, congestion is prevented. In this process, however the throughputs on different user sessions are not always evenly regulated, even if window sizes are the same for all sessions. To achieve a fair distribution of throughputs, and at the same time meet some specified delay constraints, the window sizes must be properly adjusted.

Flow control, fairness and, therefore, window optimization problems are particularly critical in a network with heavy file transfer requirements. Here, we

want to maximize network throughput, since this will minimize file transfer times. Yet, we must maintain the delay within given bounds, otherwise interactive and control traffic (here assumed to be a small fraction of the total traffic) will be adversely affected. Also, we must provide a fair share of the bandwidth to the various file transfers, since the users expect equal quality of service.

Here we consider a file transfer environment, where several transfers are in progress at the same time. A virtual circuit transport scheme is assumed, where a virtual circuit is established between each source/destination pair, and data flows on this virtual circuit. For each file transfer we define a demand d, which is the rate at which packets could be transferred on the virtual circuit if no other (interfering) traffic was present in the network. Clearly, if other traffic is present, the throughput is less than d.

5.3 The Model

Here we define the fairness measure which implicitly includes the notion of throughput efficiency. The model follows from discussions in [Reis79, Lam82] and section 3.2.2. We shall consider networks that provide virtual circuits between packet sources and sinks. The virtual circuits are end-to-end flow controlled. This means that the flow control is imposed indirectly through the window sizes i.e., limiting number of packet in each chain at any time. An important function of such end-to-end control is synchronization of the source input rate to the sink acceptance rate. All of them work by limiting the number

of packets that a virtual circuit can have in transit within the network. We shall not cover the model presented in section 3.2.2. again here. However we summary the assumptions below:

- Negligible processing time at the switching nodes
- Negligible propagation delay
- Error free channels
- Infinite buffers at all nodes
- Zero delay individual acknowledgement for data packet
- Same virtual circuit packets are routed along the same path
- Poisson arrival process at the source i.e., exponential intergeneration time
- Kleinrock's independence assumption
- Zero absorption time at the destination
- Exponential packet length
- FIFO queues for the data link

5.3.1 Delay Analysis

The packet network consists of M nodes linked by NL communication channels. R user sessions are present in the network. Each session is supported by a virtual circuit between a given source and destination. Traffic on each virtual circuit is controlled by a window mechanism which limits the number of packets (the window size) traveling on the circuit. The traffic demand of user i is d_i. Owing to window flow control, the throughput of user i is $\lambda_i \leq d_i$. Both λ_i and d_i are measured in units of information(e.g.,bits) per unit of time.

In our model, we assume single path routing i.e., all packets on a given virtual circuit flow along a fixed path instead of being distributed over multiple paths. The usual assumptions regarding input traffic and service statistics are made; namely, Poisson arrivals, exponential packet length distribution and Kleinrock's independence assumption [Klei76].

In contrast to the open network model in Chapter 4 the average network delay cannot be obtained in a closed form formula. Instead individual user network delays are obtained. Hence we define the average network delay as follows:

$$T = \frac{\sum_{i=1}^{R} \lambda_i T_i}{\sum_{i=1}^{R} \lambda_i} \leq T_{\max} \tag{5.1}$$

Therefore once the individual user network delays are obtained, the average

network delay T can be found.

5.3.2 Objective Function

The following fairness measure was chosen for the optimization of window sizes: [Gerl84]

- *Minimize*: $P = \sum_{i=1}^{R} \frac{d_i - \lambda_i}{\lambda_i}$

The system is optimally fair if $P = 0$. Thus, the goal of the problem is to minimize P.

This fairness measure has two pleasing properties:

(a) the throughput of the individual user cannot be driven to zero otherwise the value of the objective function becomes infinite; and

(b) the throughput achieved by a user tends to be proportional to his traffic demand.

This is the same fairness measure we use in our open network model except $\{\lambda_i\}$ can no longer be directly controlled but indirectly controlled through the window sizes.

5.3.3 The Formulation

With the above objective in mind, we formulate our fair window flow control problem in large network where the objective function is P subject to an average delay constraint as follows:

- *Given:* Network topology and link capacities
  Demands of user sessions

- *Minimize:* $$P = \sum_{i=1}^{R} \frac{d_i - \lambda_i}{\lambda_i} \tag{5.2}$$

- *Over sets of window sizes:* $\mathbf{N} = \{N_1, N_2, \cdots, N_R\}$

- *Subject to the constraints:*

$$T = \frac{\sum_{i=1}^{R} \lambda_i T_i}{\sum_{i=1}^{R} \lambda_i} \leq T_{\max} \tag{5.3}$$

$$0 \leq \lambda_i \leq d_i, \qquad i = 1,...R \tag{5.4}$$

where

d_i - traffic demands of user i

λ_i - actual throughput of user i

T - average network delay

T_i - average delay of user i

T_{max} - maximum delay

N_i - window size of virtual circuit i

Constraint 5.3 expresses the condition that the maximum network average delay must not be exceeded by the average network delay.

Constraint 5.4 expresses the condition that demand of user must not be exceeded by its throughput.

The choice of the objective function P was determined by fairness consideration mentioned in the last section. The function P cannot be minimized while one of the user has throughput much below its desired rate.

The quantities λ_i and T are functions of the set of window sizes and should be represented as $\lambda_i(\mathbf{N})$ and $T(\mathbf{N})$. However for simplicity of notation we will at times express T and λ_i without explicitly showing the dependence on $\mathbf{N}$, when this does not cause ambiguity.

5.4 The Solution

We will give our approach in solving the problem formulated in the previous section. First we give the principle of Lagrangian Decomposition which is used in our approach in finding the set of window sizes which minimize the objective function for a given delay constraint.

5.4.1 Lagrangian Decomposition

Lagrangian Decomposition is a method for optimization of non-linear problems subject to nonlinear constraints. The heuristic approach presented in the next section is based on this method. Let us consider the following problem:

$$\min P(\mathbf{N})$$

subject to:

$$T(\mathbf{N}) \leq T_{max} \tag{5.5}$$

where $P(\mathbf{N})$ and $T(\mathbf{N})$ are arbitrary functions of the R-dimensional variable N. The Lagrangian $H(\mathbf{N},\beta)$ for the problem is defined as follows [Ever63] :

$$H(\mathbf{N},\beta) = P(\mathbf{N}) + \beta(T(\mathbf{N}) - T_{max}) \tag{5.6}$$

where $\beta \geq 0$

Let $\beta = \beta^*$, where β^* is a positive multiplier and let $\mathbf{N}^*$ be the minimizer of $H(\mathbf{N},\beta)$. Then $\mathbf{N}^*$ is the solution of Eq. (5.6) if $T(\mathbf{N}^*) = T_{max}$ [Ever63]. Therefore if $\mathbf{N}^*$ minimizes Eq. (5.6) then $\mathbf{N}^*$ minimizes $P(\mathbf{N})$ such that $T(\mathbf{N}^*) \leq T(\mathbf{N})$ provided that $T(\mathbf{N}^*) = T_{max}$. In other words if an *unconstrainted* minimum of the Lagrangian function Eq. (5.6) can be found then this is a solution to the *constrained* minimization problem (5.5). This implies that (5.5) can be solved by minimizing $H(\mathbf{N},\beta)$ over N for different value of β until $T(\mathbf{N}^*) = T_{max}$. The problem of course is to find the right β such that the particular constrained problem is solved.

This method is valid for arbitrary $P(\mathbf{N})$ and $T(\mathbf{N})$ and is particularly efficient for the optimization of separable problems where the R-dimensional optimization of Problem (5.5) is decomposed into R separate one dimensional optimization. In general different choices of the β lead to different solutions. However by sweeping through the values of β, a spectrum of problems can be solved for the entire range of T_{max}. Thus, we apply this method to our window assignment problem.

Now we describe an efficient algorithm for the determination of this range of "Lagrangian" solutions [Whit72]. The algorithm is based on the principle of non-dominated solutions. We recall that solution A dominates solution B iff:

$$P(\mathbf{N})_A < P(\mathbf{N})_B \text{ and } T(\mathbf{N})_A \leq T(\mathbf{N})_B$$

Geometrically, finding all the nondominated solutions is equivalent to find the lower envelope of the set of points in the $(P(\mathbf{N}),T(\mathbf{N}))$ plane. In the terminology of our formulation in the previous section, suppose at step k the solution $(N_1', \ldots, N_R')$ has been found with objective value $P(\mathbf{N}')$ and average network delay $T(\mathbf{N}')$. In order to obtain the next solution, consider the ratio η_i defined as follows for each user i:

$$\eta_i \triangleq \frac{P(\mathbf{N}_i) - P(\mathbf{N}')}{T(\mathbf{N}_i) - T(\mathbf{N}')} \tag{5.7}$$

when $\mathbf{N}_i$ is a new set of window sizes with change in window size of user i.

Let m^* be the user such that

$$\eta_{m^*} = \min_i \eta_i$$

The next solution at step k+1 is obtained by increasing the window size of user m^* and by leaving all other users' window size unchanged.

The method is valid only if the functions $P(\mathbf{N}_i)$ vs $T(\mathbf{N}_i)$ are decreasing and convex for all $i = 1,...,R$ and the following conditions on $P(\mathbf{N})$ and $T(\mathbf{N})$ are met:

- $P(\mathbf{N})$ and $T(\mathbf{N})$ are separable, i.e. are expressed by sums of terms, where each term depends only on one variable. Namely:

$$P(\mathbf{N}) = \sum a_i P_i(N_i)$$

$$T(\mathbf{N}) = \sum b_i T_i(N_i)$$

- $P_i = P_i(T_i)$ is convex.

Unfortunately, these conditions on $P(\mathbf{N})$ and $T(\mathbf{N})$ are not verified in our problem except in the (not very interesting) case in which there is no interference among chains. In practical situations, each term P_i and T_i depends on *all* window sizes, and not only on N_i. (However, the strongest dependence is on N_i, i.e. a chain's delay and throughput is affected the most by a change in its own window size). Therefore heuristic is used and is presented in the next section.

5.4.2 The Solution Approach

As it may be noticed from Eq. (5.2) and (5.3) the solution to our problem requires the evaluation of λ_i and T for a given window allocation **N**. This is done by using a closed queueing network model with multiple classes of customers, where each class represents a different virtual circuit. The service centers are the communications channels [Reis79]. The exact solution is computationally too cumbersome, except for very small cases. Thus, approximations must be used. Two different types of approximations have been used in this work, namely, Linearizer [Chan82] and Schweitzer [Schw79] approximation.

After defining a procedure for the computation of λ_i and T for a given **N**, we must solve the problem of optimizing **N**. A simple inspection shows that this problem is of formidable complexity, since the variable **N** is a vector of integers, and objective and constraints are nonlinear. One solution is to try all possible window combinations **N**. This, however, is feasible only for small networks, with a small number of virtual circuits and small window range. For situations of practical interest, a different approach must be used.

The proposed solution is a based on the Lagrangian Decomposition method mentioned in the previous section. The chain which minimizes the ratio of fairness increment over delay increment is selected. More specifically, the procedure starts with unit window assignment to all chains, i.e. **N** = (1,1,1, ... ,1) and progressively increases windows step-by-step. At each step, the window is

increased for the chain minimizing the ratio:

$$\frac{\Delta P}{\Delta T} = \frac{\text{fairness increment}}{\text{delay increment}}$$

The procedure stops when:

$$\lambda_i = d_i \qquad \forall\, i\ ; \qquad or$$

$$N_i = M_i \qquad \forall\, i$$

where M_i is the maximum admissible window size for chain i.

This procedure will generate a curve $P = P(T)$ of the type shown in Figure 5.8. The desired solution is represented by the point (P^*, T^*) on this curve such that T^* is the largest value $\leq T_{max}$.

The heuristic is intuitively appealing since at each step we choose the direction which gives the best reduction in P for the same increase in delay T. We expect the method to generate a suboptimal set of solutions, for a range of values of T. From this set, we obtain the solution for the desired value T_{max}.

According to the Lagrangian Decomposition principle, the incremental method yields all the nondominated solutions in the (P, T) plane. The application of the method thus provides a set of suboptimal solutions i.e., not all the nondominated solutions are generated. The numerical results, however, show that the set of points so generated approximates closely the lower envelope (see

Figure 5.6).

In order to improve the computational efficiency of the algorithm in handling large networks, another approximation is introduced. We recall that, at each step of the algorithm the ratio $\Delta P/\Delta T$ must be evaluated for each chain. This implies recomputing λ_i and T for each window increment. The exact approach requires solving for the entire network again after each increment - a *very* costly proposition even if Schweitzer or Linearizer approximations are used. To simplify the computation, an approximation technique is used, as reported in Appendix B. This approximation allows us to compute the changes in λ_i and T due to a window increment directly from the current solution. More specifically, it allows us to increase window size by 1 in one of the chains, to estimate the throughput increase in that chain and the throughput decrease in the remaining chains.

5.5 The Algorithm

This algorithm generates the set of nondominated (P,T) solutions.

1. Set the maximum number of iterations

2. Set the window size for each chain to be 1, i.e. $N_i = 1$ for $i = 1,2, \ldots, R$.

3. Calculate the throughput λ_i of each chain and the average delay of the network using MVA with this set of window sizes.

ROUTING, FLOW CONTROL AND FAIRNESS IN COMPUTER NETWORKS

Hak-Wai Chan

October 1986
CSD-860067

Repeat steps 4 through 7 for all the chains $i = 1,2,...,R$

4. Let $N_i = N_i + 1$, other window sizes unchanged.

5. Calculate the throughput of chain i using the formula [Reis80] :

$$\lambda_i(\mathbf{N} + \mathbf{e}_i) = \frac{N_i + 1}{\sum_{k=1}^{NL} \theta_{ik} x_{ik} \left[1 + L_k(\mathbf{N})\right]}$$

6. Calculate other chains $j (\neq i)$throughputs using the formula

$$\lambda_j(\mathbf{N} + \mathbf{e}_i) = \frac{N_j}{\frac{N_j}{\lambda_j(\mathbf{N})} + \sum_{k=1}^{NL} a_{jk} \frac{L_{ik}(\mathbf{N})}{N_i}}$$

derived from Schweitzer's approximation as shown in Appendix B, Eq. (B2).

7. Compute the unit decrease in objective value per unit increase of average delay, that is

$$\eta = \frac{P - P'}{T - T'}$$

where P is the new objective value and P' is the previous objective value; T is the new average delay and T' is the previous average delay.

Note: T is always increasing for increasing window size. P however, may increase or decrease.

8. Let $N_i = N_i - 1$;

9. Find the chain m^* which minimizes η

10. Let $N_{m^*} = N_{m^*} + 1$

11. Go back to 3 until the maximum window sizes have been reached or $T > T_{max}$.

The above algorithm identifies at each step the best chain on which the window should be increased for the given set of window sizes. Starting from N = (1,1, ... ,1) the algorithm finds the next window increment. This gives us a new set of objective values and average delays. The algorithm stops when all the windows have reached their maximum values or the maximum number of iterations has been reached. The result is a curve of fairness values versus network average delays.

The algorithm has several nice characteristics :

1. It is simple and requires no complete enumerations of all possible window combinations.

2. The step to find the chain candidate for window increase is fast because of the ease of estimating throughputs from the approximate formula.

3. There is an upper bound of $\sum_{i=1}^{R} M_i - R + 1$ iterations as compared to $\prod_{i=1}^{R} (M_i - 1)$ number of possible window selection in the exhaustive

method, where M_i is the maximum window size of chain i.

4. The algorithm gives a range of objective values versus average delays, rather than one single solution. Thus, sensitivity studies can be performed at no extra cost.

The algorithm just presented is an approximate algorithm. The approximation in the algorithm comes from three steps. First, for large networks, exact MVA cannot be used. Therefore, an error is introduced by the approximate technique (e.g. Linearizer or Schweitzer's) used in step 3. Secondly, the incremental throughputs in step 6 are based on Schweitzer's approximation. Thirdly, the objective and the constraint are not separable functions, thus violating the Lagrangian Decomposition requirements for the generation of all nondominated solutions.

Due to the approximate nature of the algorithm, it is important to validate the results with several examples. This is done in the next section.

5.6 Numerical Results

In this section we present the results of the application of the algorithm to several network examples. First we use the four chain and eight queue example shown in Figure 5.1. The traffic requirements and the paths used by the chains are shown in Table 5.1. Queues are assumed to be FIFO with exponential service time distributions. We limit the individual window size to be 8. This exam-

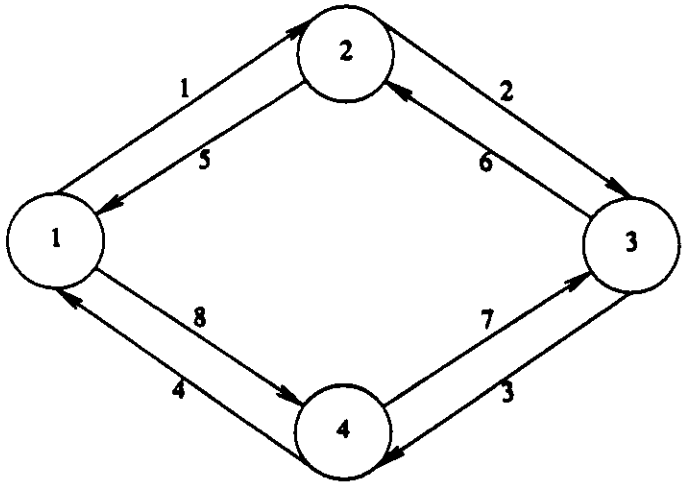

Fig. 5.1 Four node example.

Table 5.1
Traffic requirements and routes for
the four node example - case 1.

chain	source	dest.	demand	route
1	1	4	2	1-2-3
2	1	2	2	8-7-6
3	3	1	2	6-5
4	4	2	2	4-1

ple is small enough to permit us to explore the set of all possible (4096) window size combinations. In Figure 5.2, we plot the objective values against network average delays for the lower 1000 objective value points. Our algorithm attempts to find the lower envelope points using the heuristic presented in the previous section. The connected points in the figure are the points generated by the algorithm using Mean Value Analysis (MVA) in step 3. They do represent most of the nondominated solutions and thus provide us with a suboptimal set of window sizes.

The near optimality of the above results is mainly due to the fact that there is little interference among the chains. From Table 5.1 see that chain 1 interferes with chain 4 only at link 1 and chain 2 interferes with chain 3 only at link 6. Our algorithm cannot sense this second order interference among the chains since it is based on Schweitzer's approach derived in Appendix B.

In Figure 5.3 we plot the three sets of suboptimal solutions when different network of queue algorithms are used in step 3; namely MVA, Schweitzer [Schw79] and Linearizer [Chan82]. We see that Linearizer and MVA give comparable curves. Table 5.2 gives us the run time (in CPU seconds) using the three aforementioned approaches for our example. We see that MVA tops the list while Linearizer and Schweitzer have comparable run times. From this we see that Linearizer is probably the best choice for large network.

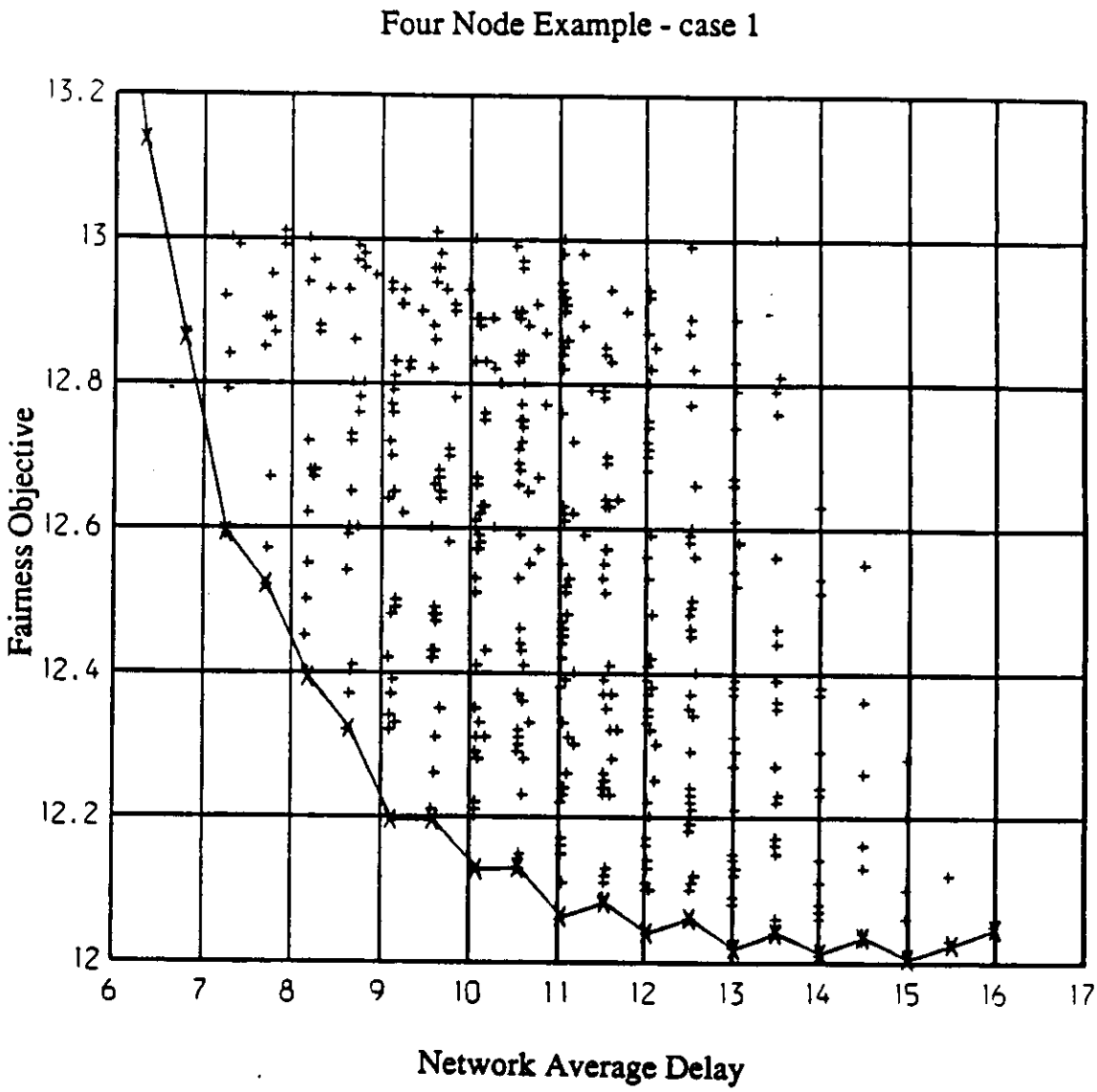

Fig. 5.2 Family of suboptimal solutions for the four node example in Table 5.1.

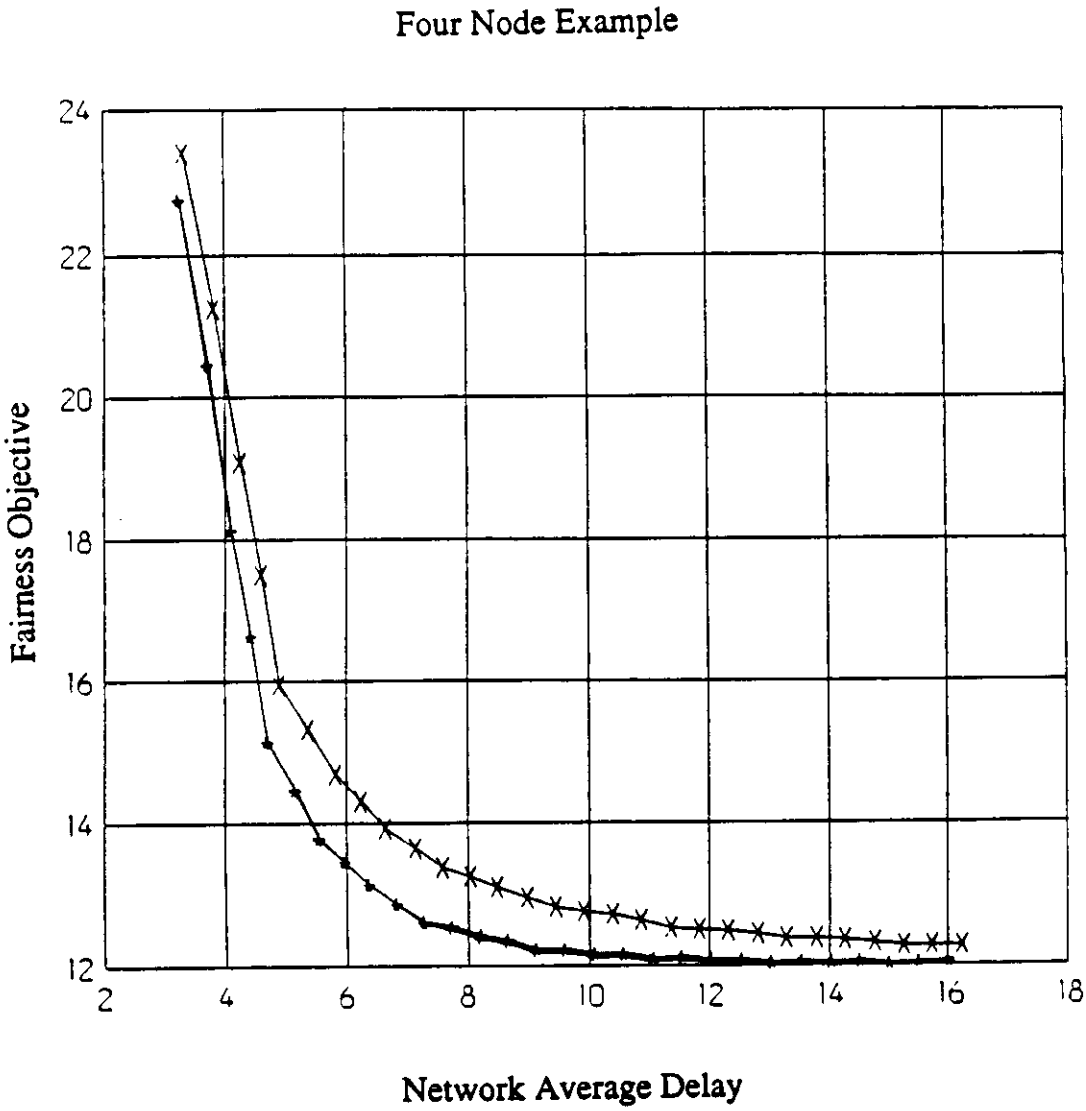

Fig. 5.3 Plot of lower envelopes from MVA(+), Linearizer(*) and Schweitzer(X) for the example in Table 5.1.

Next we use the ten node example shown in Figure 5.4. The links are assumed to be full duplex. The demands and the routes used by the chains are shown in Table 5.3.

Table 5.2
Run time for the four nodes example in Table 5.1.

	MVA	Linearizer	Schweitzer
Run time	34.4s	6.6s	4.1s

In the example we have three chains. Chain 2 has interaction at link 6 with chain 1 which in turn has interaction with chain 3 at link 10. Limiting maximum window size to 8, all 512 possible window size combinations can be plotted and compared with the set of window sizes obtained from our algorithm. Again Figure 5.5 shows that the results are near optimal in spite of the fact that more interference is experienced in this example. Table 5.4 summarizes the set of window sizes generated by the algorithm. For a given T_{max} one can select the window set N that gives the best objective value.

Next we consider again the same four node example in Figure 5.1, except that this time we introduce substantial interaction among the chains. The demands and the paths in use are shown in Table 5.5. Chain 3 interacts with chain 4. Chain 4 interacts with chain 2 and chain 2 interacts with chain 1. Figure 5.6 gives the plot of fairness values versus network average delays. We observe

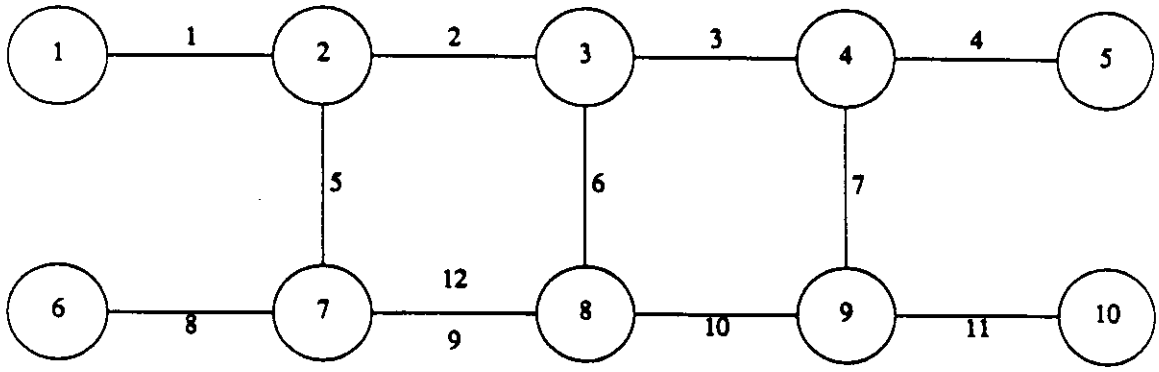

Fig. 5.4 Ten Node example.

Table 5.3
Traffic requirements and routes for
the ten node example.

chain	source	dest.	demand	route
1	1	10	4	1-2-6-10-11
2	5	6	4	4-3-6-12-8
3	2	4	4	5-9-10-7

that the set obtained from the algorithm is again very close to the optimal. Table 5.6 tabulates the window sizes when different techniques are used in step 3 of the algorithm. There is only a slight difference between the set of windows found by Schweitzer's approximation and the set obtained using MVA. The two sets of

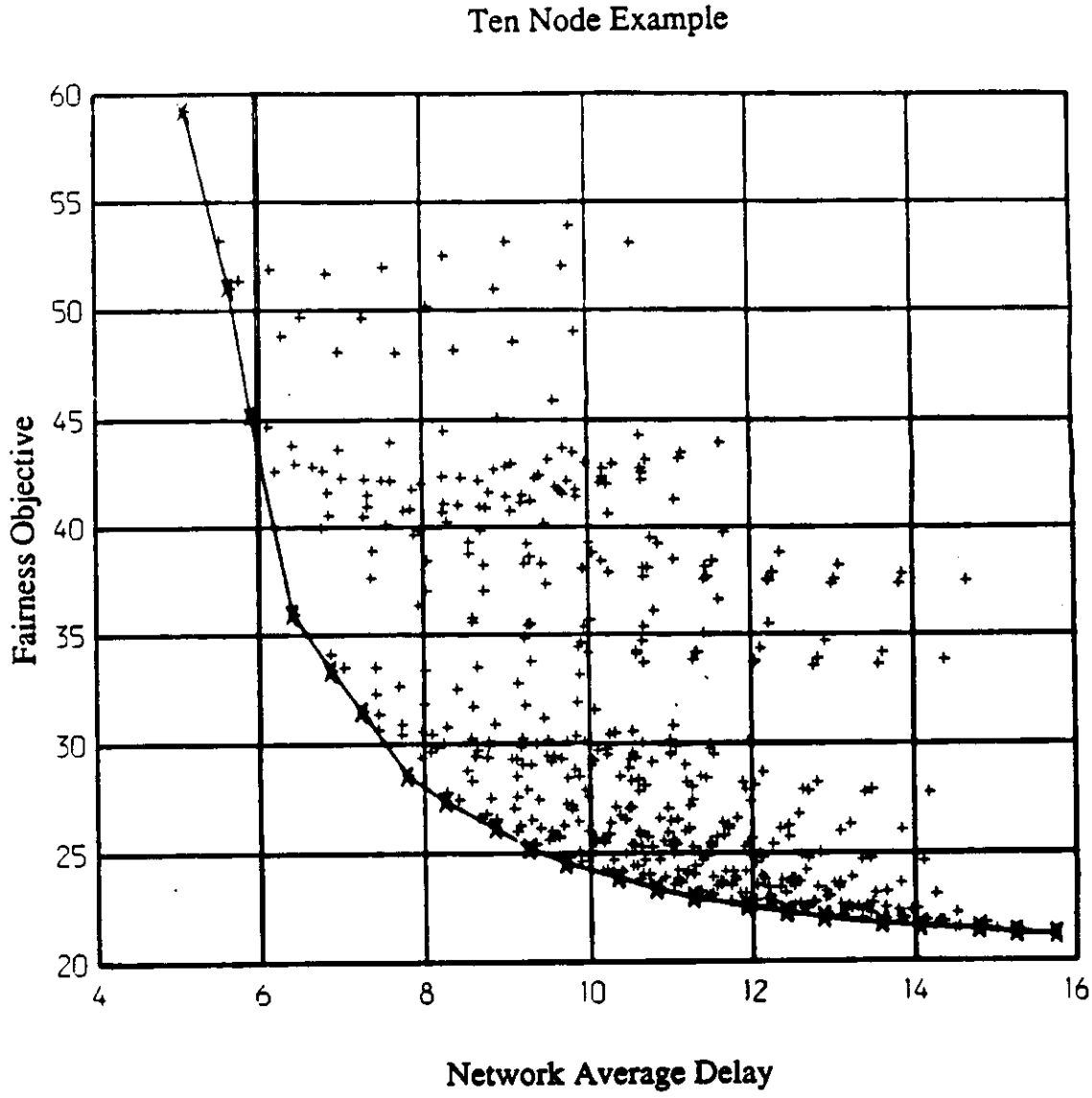

Fig. 5.5 Family of suboptimal solutions for the ten node example.

Table 5.4
Delays and window sizes for the ten node example.

T	λ_1	λ_2	λ_3	N	P
5.1211	0.1762	0.1841	0.2255	(1,1,1)	59.1675
5.6534	0.1704	0.3113	0.2259	(1,2,1)	51.0343
5.9204	0.1637	0.3117	0.3691	(1,2,2)	45.1006
6.4210	0.2775	0.3020	0.3549	(2,2,2)	35.9286
6.8846	0.2689	0.3920	0.3559	(2,3,2)	33.3198
7.2557	0.2589	0.3930	0.4506	(2,3,3)	31.5022
7.8081	0.3371	0.3815	0.4341	(3,3,3)	28.5664
8.2591	0.3270	0.4478	0.4360	(3,4,3)	27.3393
8.8651	0.3854	0.4343	0.4211	(4,4,3)	26.0872
9.2774	0.3718	0.4370	0.4846	(4,4,4)	25.1643
9.7300	0.3611	0.4874	0.4876	(4,5,4)	24.4877
10.3702	0.4050	0.4733	0.4717	(5,5,4)	23.8071
10.8196	0.3914	0.4771	0.5178	(5,5,5)	23.3277
11.2776	0.3806	0.5163	0.5219	(5,6,5)	22.9229
11.9491	0.4149	0.5021	0.5057	(6,6,5)	22.5169
12.4224	0.4017	0.5068	0.5405	(6,6,6)	22.2512
12.8864	0.3911	0.5379	0.5455	(6,7,6)	21.9979
13.5846	0.4190	0.5238	0.5295	(7,7,6)	21.7380
14.0780	0.4077	0.5487	0.5352	(7,8,6)	21.5736
14.7993	0.4315	0.5341	0.5210	(8,8,6)	21.4372
15.2628	0.4196	0.5406	0.5467	(8,8,7)	21.2481
15.7605	0.4078	0.5469	0.5681	(8,8,8)	21.1641

windows obtained using MVA and Linearizer coincide.

Lastly, the algorithm was tested on the network shown in Figure 5.7 corresponding to an early 8 node ARPANET configuration [Klei76]. Ten virtual circuits are present. Source, destination, demand and the path of each circuit are shown in Table 5.7. Figure 5.8 shows the curve of objective values versus network average delay. The slight increase of fairness values towards the end of the tail is due to the fact some of the window sizes have reached their maximum values 8. Apparently, fairness deteriorates by further increasing the windows on the remaining chains. By examining Figure 5.8, we observe that the set of suboptimal solutions is very dense, that is, the gaps between suboptimal solutions are very small. Also, the curve tends to be nearly convex. These two facts guarantee that the set of solutions is close to optimal. We expect this trend to be maintained as network size increases.

Table 5.5
Traffic requirements and routes for
the four node example - case 2.

chain	source	dest.	demand	route
1	1	3	75	1-2
2	4	3	75	4-1-2
3	3	4	75	3
4	3	1	75	3-4

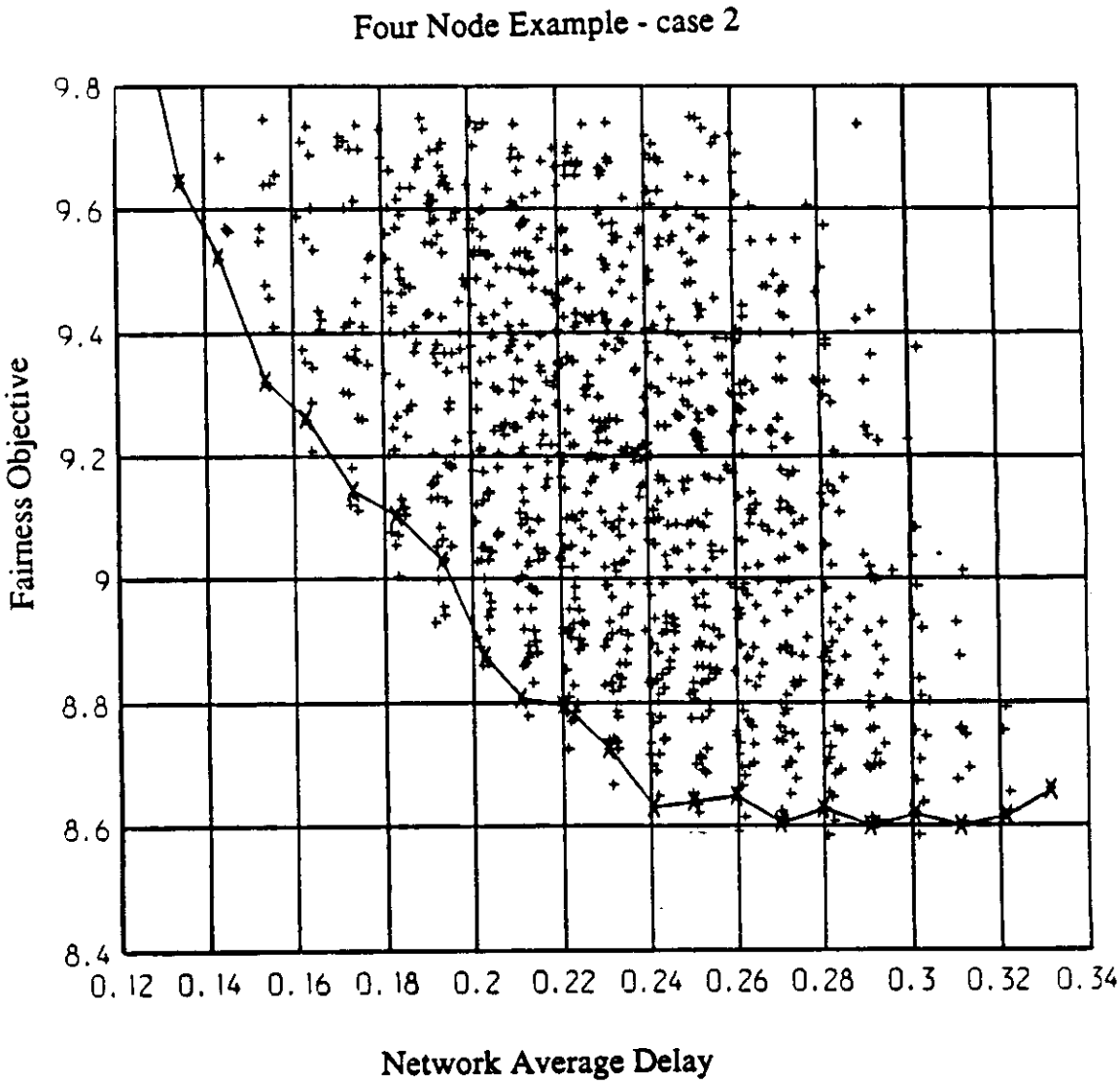

Fig. 5.6 Family of suboptimal solutions for the four node example in Table 5.5.

Table 5.6
Window sizes for the four node example in Table 5.5.

Iter.	Window Sizes		
	MVA	Linearizer	Schweitzer
1	(1,1,1,1)	(1,1,1,1)	(1,1,1,1)
2	(1,2,1,1)	(1,2,1,1)	(1,2,1,1)
3	(2,2,1,1)	(2,2,1,1)	(2,2,1,1)
4	(2,2,1,2)	(2,2,1,2)	(2,2,1,2)
5	(2,2,2,2)	(2,2,2,2)	(2,2,2,2)
6	(2,3,2,2)	(2,3,2,2)	(2,3,2,2)
7	(3,3,2,2)	(3,3,2,2)	(3,3,2,2)
8	(3,3,2,3)	(3,3,2,3)	(3,3,2,3)
9	(3,4,2,3)	(3,4,2,3)	(3,4,2,3)
10	(4,4,2,3)	(4,4,2,3)	(3,4,3,3)
11	(4,5,2,3)	(4,5,2,3)	(4,4,3,3)
12	(5,5,2,3)	(5,5,2,3)	(4,4,3,4)
13	(5,6,2,3)	(5,6,2,3)	(4,5,3,4)
14	(6,6,2,3)	(6,6,2,3)	(5,5,3,4)
15	(6,7,2,3)	(6,7,2,3)	(5,6,3,4)
16	(6,7,2,4)	(6,7,2,4)	(6,6,3,4)
17	(6,7,3,4)	(6,7,3,4)	(6,7,3,4)
18	(7,7,3,4)	(7,7,3,4)	(6,7,3,5)
19	(7,8,3,4)	(7,8,3,4)	(6,7,4,5)
20	(7,8,3,5)	(7,8,3,5)	(6,7,4,6)
21	(8,8,3,5)	(8,8,3,5)	(6,7,5,6)
22	(8,8,4,5)	(8,8,4,5)	(7,7,5,6)
23	(8,8,4,6)	(8,8,4,6)	(7,8,5,6)
24	(8,8,5,6)	(8,8,5,6)	(7,8,5,7)
25	(8,8,5,7)	(8,8,5,7)	(7,8,6,7)
26	(8,8,6,7)	(8,8,6,7)	(7,8,6,8)
27	(8,8,6,8)	(8,8,6,8)	(7,8,7,8)
28	(8,8,7,8)	(8,8,7,8)	(8,8,7,8)
29	(8,8,8,8)	(8,8,8,8)	(8,8,8,8)

We next examine the result of an experiment to investigate the effect on the total throughputs by the virtual circuit loads with different delay constraints. In Figure 5.9 the total throughput rate is plotted as a function of the uniform input rates $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4$ for T_{max} = 4,5,6,7,8 of the network configuration in Figure 5.1 with 4 chains and their route shown in Table 5.1. Note that there is a little increase in total throughput rate when T_{max} = 7 and T_{max} = 8 while wider difference in total throughputs is obtained for T_{max} = 4,5 and 6. In other words up to a certain value of T_{max} the total throughput cannot increase due to the capacity constraint. On the other hand as λ's increase the number of packets in transit will be equal to the window size much of the time. As a result the rate at which packet are admitted by a virtual circuit levels off as λ's increase. Further increase in λ's will only have marginal effect on the network loading. As expected the total throughput rate of each curve in Figure 5.9 increase as the virtual circuit loads increase.

5.7 Concluding Remarks

In this chapter we define a proper fairness measure which includes the notion of throughput efficiency. We then proposed an approximate algorithm to optimize fairness such that a given delay constraint is met by adjusting the window sizes. This is a suboptimal algorithm which yields a subset of global set of optimal solutions corresponding to different average network delay constraints.

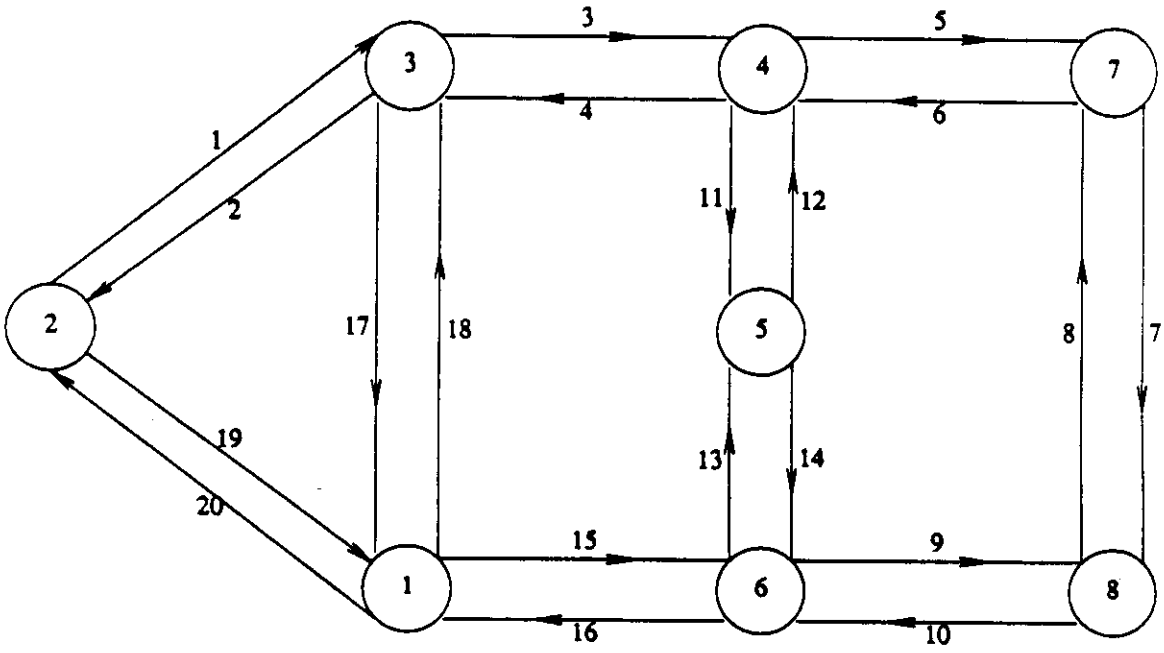

Fig. 5.7 Early ARPRNET example.

File transfer environment i.e., there is always packets waiting at the source is the situation under consideration. This makes the model valid with window as the only flow control mechanism. However the method may be extended to the optimization of window in a window flow controlled network with mixed traffic environments, where both the file transfer and interactive traffic components are present.

Table 5.7
Traffic requirements and routes for
the early ARPANET example

chain	source	dest.	demand	route
1	1	7	5	18-3-5
2	3	8	5	17-15-9
3	2	5	5	19-15-13
4	4	6	5	11-14
5	1	5	5	15-14
6	4	2	5	4-2
7	3	7	5	3-5
8	6	8	5	9
9	1	3	5	18
10	7	4	5	6

The algorithm is computationally efficient that it progressively finds the proper set of window to achieve throughput efficiency within given delay and fairness constraints. In the process it avoids the costly step of recomputing throughputs and delays of individual chains with different sets of window under consideration. It is also shown to provide nearly optimal solution.

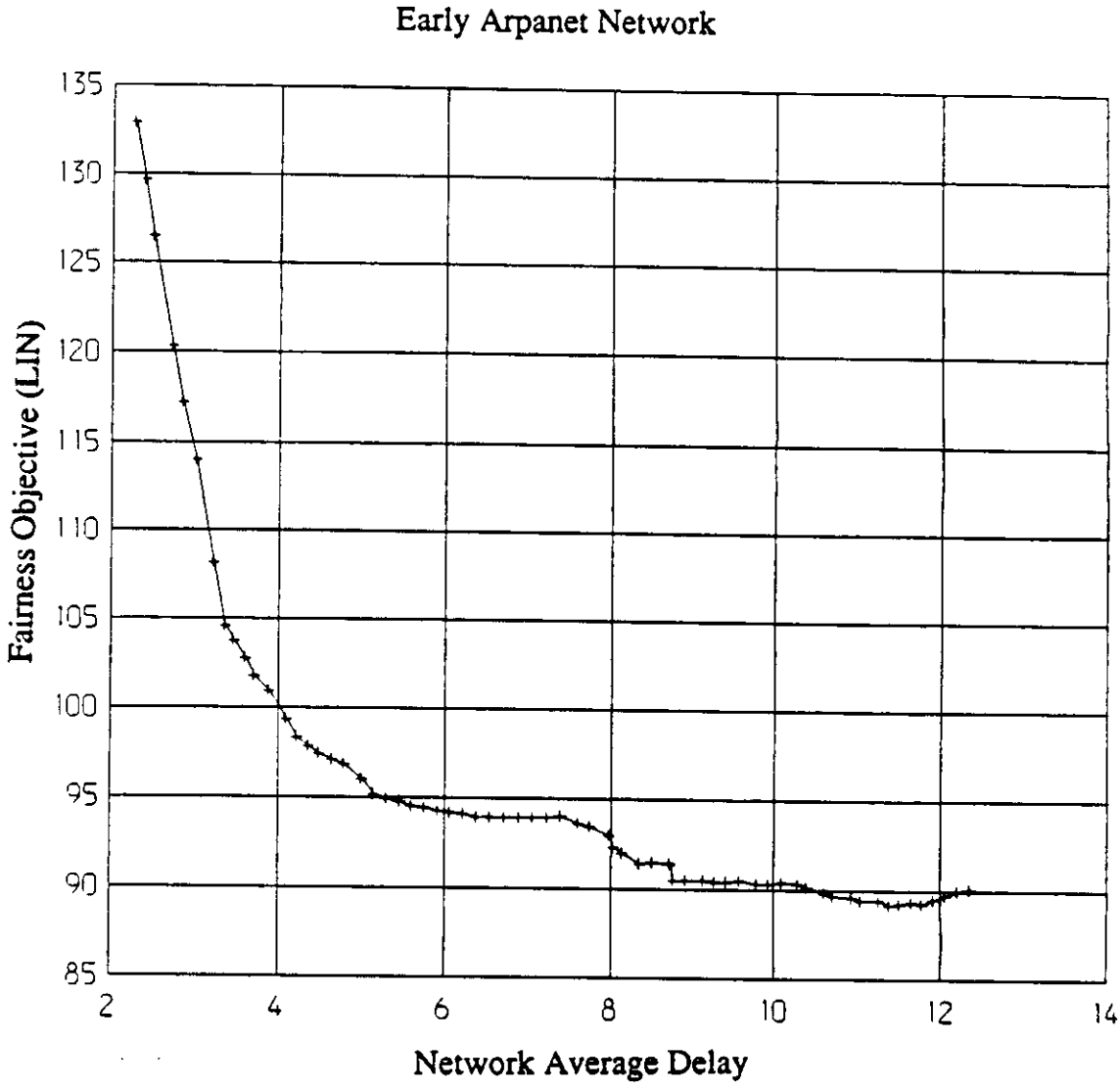

Fig. 5.8 Family of suboptimal solutions for early ARPANET example.

Four Node Example

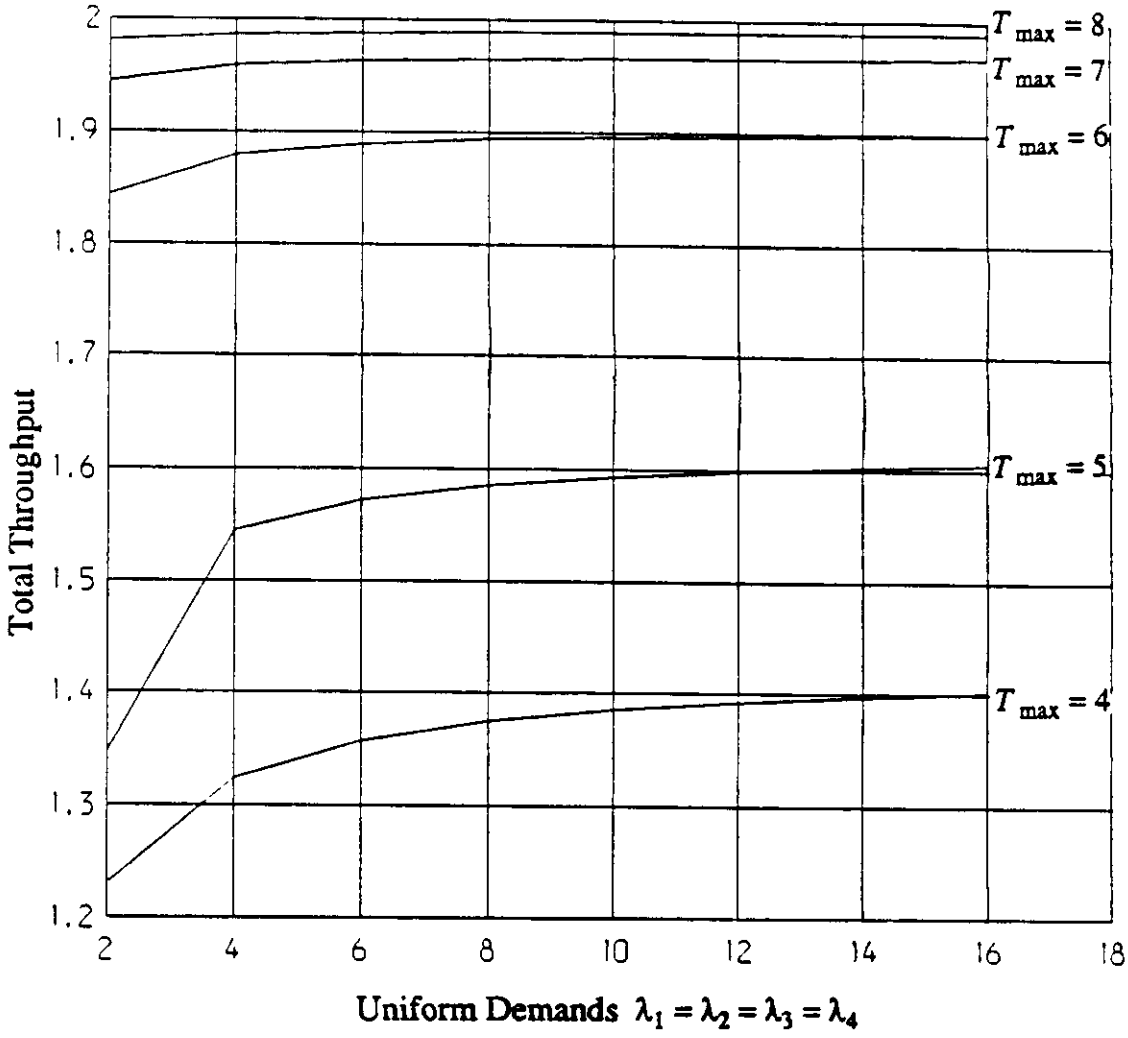

Fig. 5.9 Network throughput rate as a function of λ and T_{max}.

CHAPTER 6

SIMULATION OF FLOW CONTROL SCHEMES - BACKPRESSURE AND WINDOW CONTROL

6.1 Introduction

We have so far presented the results of input rate flow control and window flow control mechanisms in the context of a fairness optimization problem. In our analysis, we have assumed unlimited nodal buffers. In fact, nodal buffers are not infinite, and limited buffer size as well as buffer allocation discipline among competing users will affect performance characteristics such as delay, throughput and fairness.

In this chapter, we relax the infinite buffer assumption and extend our investigation to a flow control technique known as backpressure which is based on nodal buffer management. We also consider the combination of this control mechanism with the window flow control mechanism described in the previous chapter. One factor that we will consider is the service discipline in a node to achieve the fairness objective. Full duplex virtual circuits are considered here. Control messages are assumed instantaneous and not consuming network bandwidth.

We will consider three schemes. First, the window flow control scheme, studied using closed network models and Kleinrock's independence assumption. The second and third scheme are backpressure and a combination of backpressure and window control respectively. With multiple virtual circuits in a typical computer communication network, the complexity of the backpressure model with limited nodal buffers is beyond the reach of mathematical analysis. We therefore resort to simulation for our study. We will relax the independence assumption and will assume fixed packet length throughout the virtual circuit session. The schemes will be compared under different traffic load patterns. We will describe our problem in section 6.2 and the dynamic window control scheme in section 6.3. Then, the backpressure control algorithm is given in section 6.4 followed by the simulation result in the section 6.5.

6.2 The Problem Description

The problem description follows the definitions in Chapters 4 and 5 along with the notion of virtual circuit which is inherent in the backpressure scheme. The key issue here is how to allocate buffer and channel resources fairly among the users. Clearly, if buffer sizes are increased, throughput will increase but queue length at each node will also increase. Congestion could be prevented if queue length has the feedback effect of restricting incoming traffic. This is the essence of the backpressure technique presented in the next session. At the same time, increasing queue length triggers the window control mechanism. So, there is an interplay between these two control mechanisms. Both have effect on the

network throughput and delay. We would also like to study this interplay.

Here we consider a network environment where virtual circuits are set up and remain static for the entire duration of the session. Routing is done only once, when the user first requests the connection.

6.3 The Schemes

The underlying model is still the network of queues model. An example of simple virtual circuit is given in Figure 6.1. Source A using channel 4 is connected through node 1,2 and 3 to destination B using channel 7. Buffers are associated with channel 4 in node 1 and with channel 7 in node 3. Buffers are also used in node 2 for traffic passing through it. If we assume no channel errors the packets will be delivered in order to the destination. We now briefly describe the control schemes under study.

6.3.1 Dynamic Window Control

We summarize the approach based on the window pacing concept [Schw82, Atki80] in Figure 6.2.

Initially, the window size is set to the number of communication channels the virtual circuit traverses. Window size is then dynamically adjusted based on the load. A minimum window size may also be defined for each virtual circuit, and may be enforced in case of severe congestion. Of course, parameters such as severeness of congestion, duration of the congestion and the minimum window

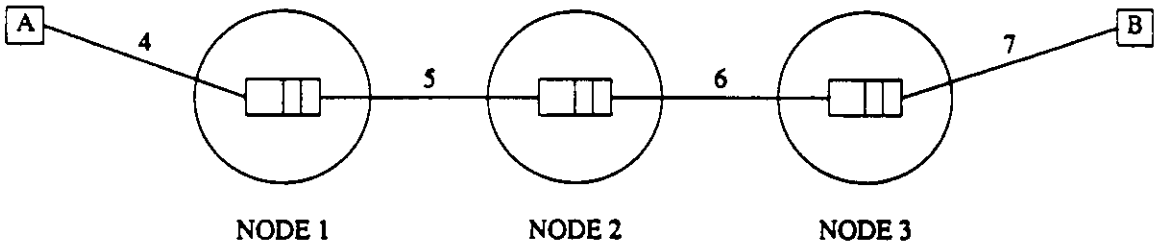

Fig. 6.1 Typical virtual channel in a network.

size must be chosen judiciously by the network designer. Notice that in order to select the virtual circuit on which to increase or decrease the window, messages must be exchanged among the sessions. This may involve a lot of messages for a large network, thus contributing to congestion. This motivates us to propose the "passive" fair flow control scheme described in section 6.4.

6.3.2 The Backpressure

The following describes the dynamics of the backpressure scheme. For each virtual circuit (VC), there is a limit U of the number of messages which may be transmitted from one node to the next. This limit is assigned at the beginning of the session and will depend on the expected peak data rate of the VC. Once the sending node has reached this limit, it may not send any more packets on that VC until it gets the acknowledgement from the receiving node.

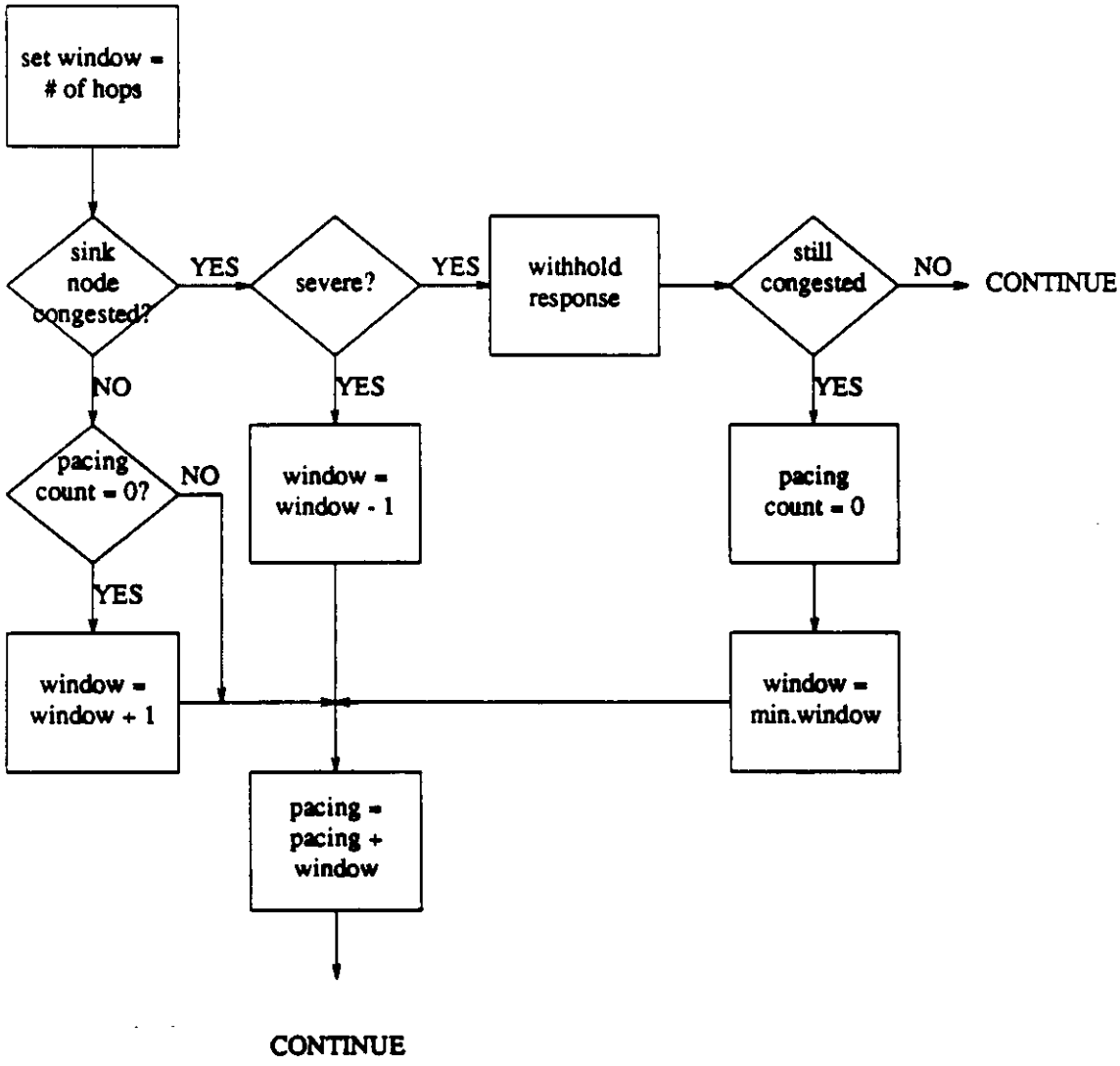

Fig. 6.2 Dynamic window control.

The receiving node knows how much data is in the VC buffer. When the buffer is low or exhausted, the receiving node does not send back permission, thus stopping the flow of data on the VC. This is the passive nature of the back-pressure scheme. Doing nothing is the way to stop the influx of data. If a node is overloaded, one effect of backpressure is to reduce the load. Backpressure propagates from node to node back to the source and effectively shuts it off. It does not matter whether the cause of the backpressure is inability of the destination to consume the data as fast as the source can supply it, or congestion within the net. Either way, the source is slowed down or shut off. Only VC's causing buffer overload are affected.

The operation of the backpressure scheme can be seen from Figure 6.1 with details of a node shown in Figure 6.3. The source A (in Fig. 6.1) is sending messages at a rate faster than B can consume. As the data arrive they start to flow through the VC. The data will start to grow at node 3. After the number of messages buffered in node 3 exceeds U for this VC, an RNR (Receiver Not Ready) control packet is sent back to node 2. Thus, node 3 will have at most U messages buffered for this channel. Node 2 will go through the same process resulting in at most U messages buffered. The buffer in node 1 will start to grow. When the buffer exceeds U messages, the source will be requested to stop transmitting. No data will flow in the VC until the number of messages at node 2 drops below L. L is chosen to be less than U, to avoid oscillation in the control. At this point the RR (Receiver Ready) control packet will be sent and data will

start to flow from node 1 to node 2. Similarly when the number of messages buffered for node 3 drops below L, data will start to flow from node 2 to node 3. The source will be permitted to transmit again when the number of messages in node 1 becomes less than L. This form of flow control requires no explicit end-to-end control. If a node is heavily loaded, the result is the stopping of data flow to the heavily loaded node.

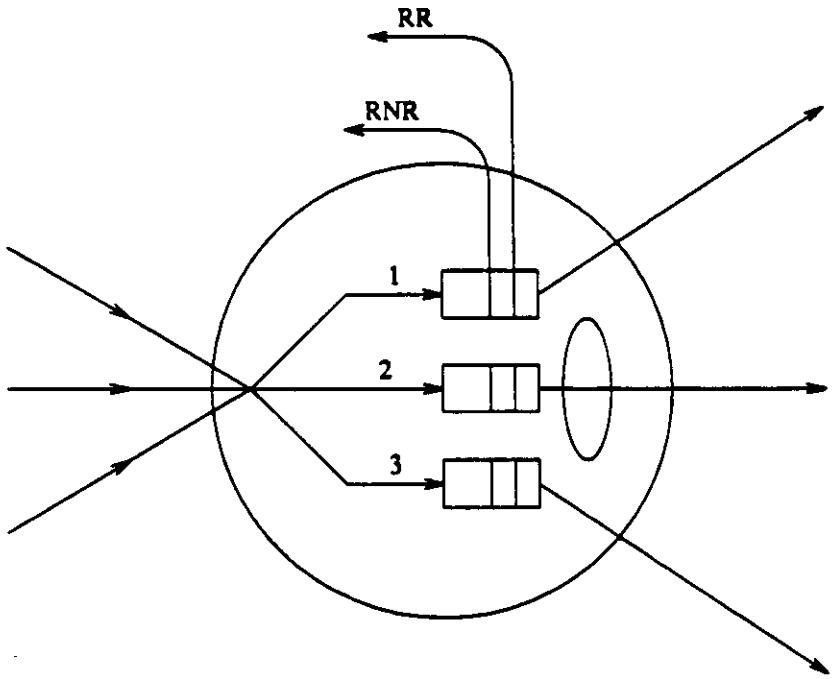

Fig. 6.3 Model of a node in a network.

An implicit service discipline in backpressure is depicted in Figure 6.3 with separate buffers for each VC in a node. Arrivals of VC 1 packets to a node will be put in buffer 1, VC 2 packets in buffer 2, and so on. Each time a packet is sent from the node to another node, it is selected from first from buffer 1, then from buffer 2, and so on in a round-robin fashion.

6.4 The Control Algorithm

We will first describe the principles of operation for the control algorithm followed by the description of the PAWS (The Performance Analyst's Workbench System) [Berr82] simulation model. The overall underlying operation of our simulation will be outlined and the simulation model for our study will be given.

6.4.1 Principles of Operation

Our network is characterized as follows:

1. Each VC employs the backpressure mechanism described earlier. End-to-end flow control will also be incorporated in our study.

2. Each virtual circuit has a traffic source and sink. Messages are generated by the source according to a traffic rate probability distribution. In our study, each message generated consisted of a single fixed length packet. The interarrival time of the messages was assumed to be exponentially distributed.

3. Both nodal processing times and sink absorption times of packets are assumed to be negligible compared to channel delays.

4. Each separate queue in a node employs a FCFS discipline. Fixed routing is used

5. A packet must acquire a channel to be transmitted and must be individually acknowledged. It is assumed that errors due to channel noise are negligible and that positive acknowledgements are always accepted.

6. Each node has a finite number of buffers. Fixed buffer are assigned to the VC along with U and L thresholds as discussed in 6.3.2. Typical values are $U = 8$ and $L = 4$, but other values of U and L are also used to study the effect of these limits on throughput and delay.

7. Node-to-node acknowledgement is not explicitly modeled in the backpressure environment. This means that when a packet is delivered to the next node, this fact is known to the sending node right away. Similar assumption is made for the end-to-end acknowledgements so that when a packet is delivered to the sink, this is known to the source right away. However these acknowledgements may be incorporated in the simulator fairly easily.

6.4.2 PAWS Simulation Model

PAWS (The Performance Analyst's Workbench System) is a simulation language where high-level description of our model is possible. Information Processing Graphs, which is the pictorial representations of the models, will be used to describe our model. In the PAWS simulation, communication channels are modeled as servers. Generated packets are represented as customers (transactions) requesting service. The service times correspond to the transmission delays while the waiting times for the packets at a node correspond to the queueing delays. After receiving service, the packet may have to wait for acknowledgement before proceeding for additional service.

Several network and traffic configurations have been simulated. They will be described and their experimental results will be reported later. Most experiments were run for duration of 1000s of simulation time so that each VC delivered about 2000 packets during a simulation run. Each simulation experiment is repeated five times with consecutive time intervals to obtain the statistical analysis of the results.

We will not describe the model in detail. Rather, the flow of the packets within the network will be given and summarized in a flow chart at the end of this section. Flow of packets for a particular VC is considered here. Packets are generated at the source with exponentially distributed interarrival times. Packets are queued in a FCFS manner for permission to be sent at the buffer in the source

node. If the buffer is full at the source node, the packet is dropped. After obtaining permission, the packets queue for the communication channel which serves packets from different VC in a round-robin fashion. The service times are constant since we consider fixed length packets here. After completing service from the channel, the buffer availability at the next node is checked. If buffer occupancy exceeds the limit U at the next node, RNR will be sent i.e., the source will be informed that no more packets can be delivered. If buffer occupancy falls below the limit L at the next node RR will be sent i.e., the source will be informed that packets can be sent again. This sequence is repeated for each communication channel which the packets traverse. Flow of packets for other VC's follows the same pattern. Therefore multiple VC configurations can be easily described in the model. Figure 6.4 summarizes the logical flow of the packets.

6.5 The Simulation Results

The objective of our experiment was to investigate the behavior of different schemes - backpressure mechanism alone, backpressure with window control mechanism and window flow control alone - under different traffic environments. Values of the fairness measure are also compared under the three schemes.

6.5.1 Backpressure

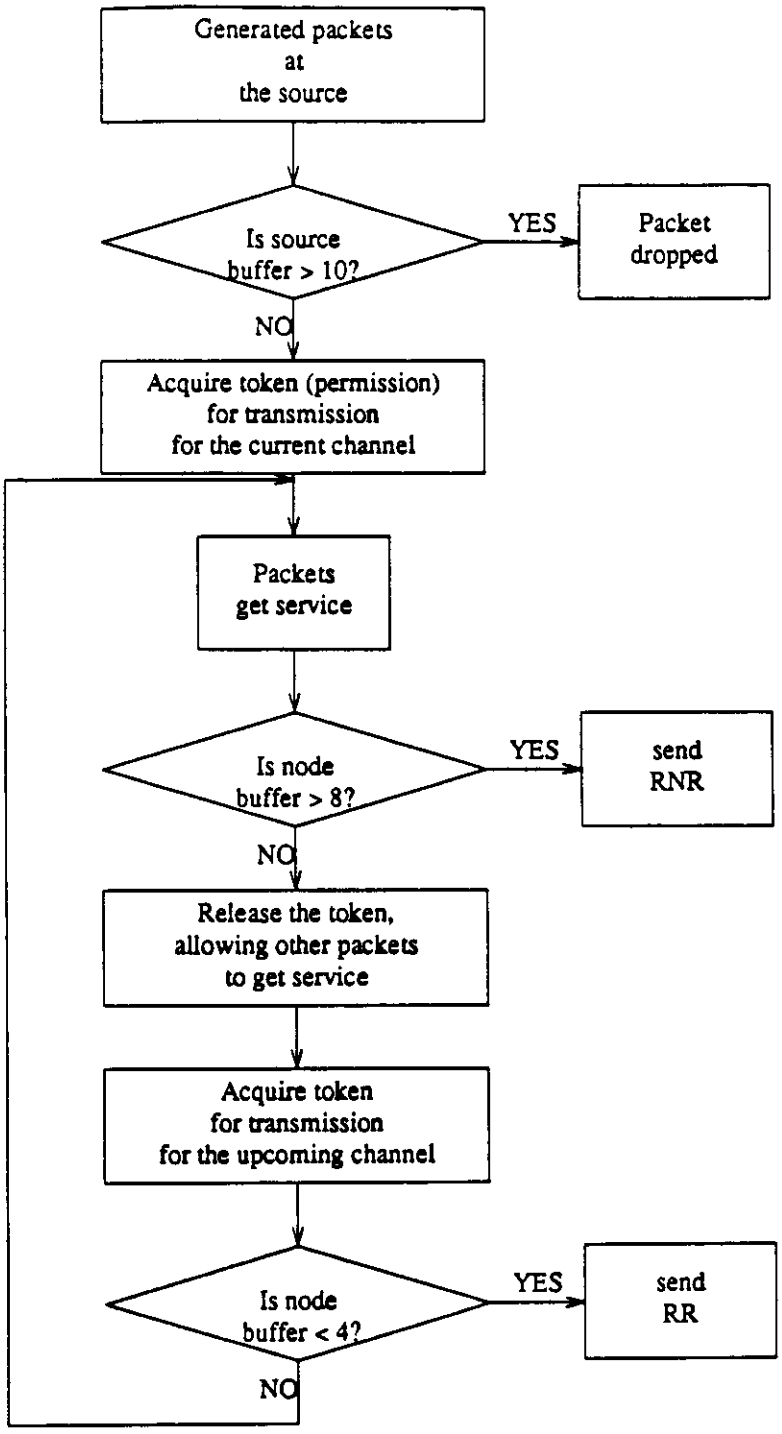

Fig. 6.4 Flow of packets using backpressure scheme.

The first network example (used in our study of the backpressure mechanism) is the eight node example given in Figure 4.4 and reproduced here in Figure 6.5.

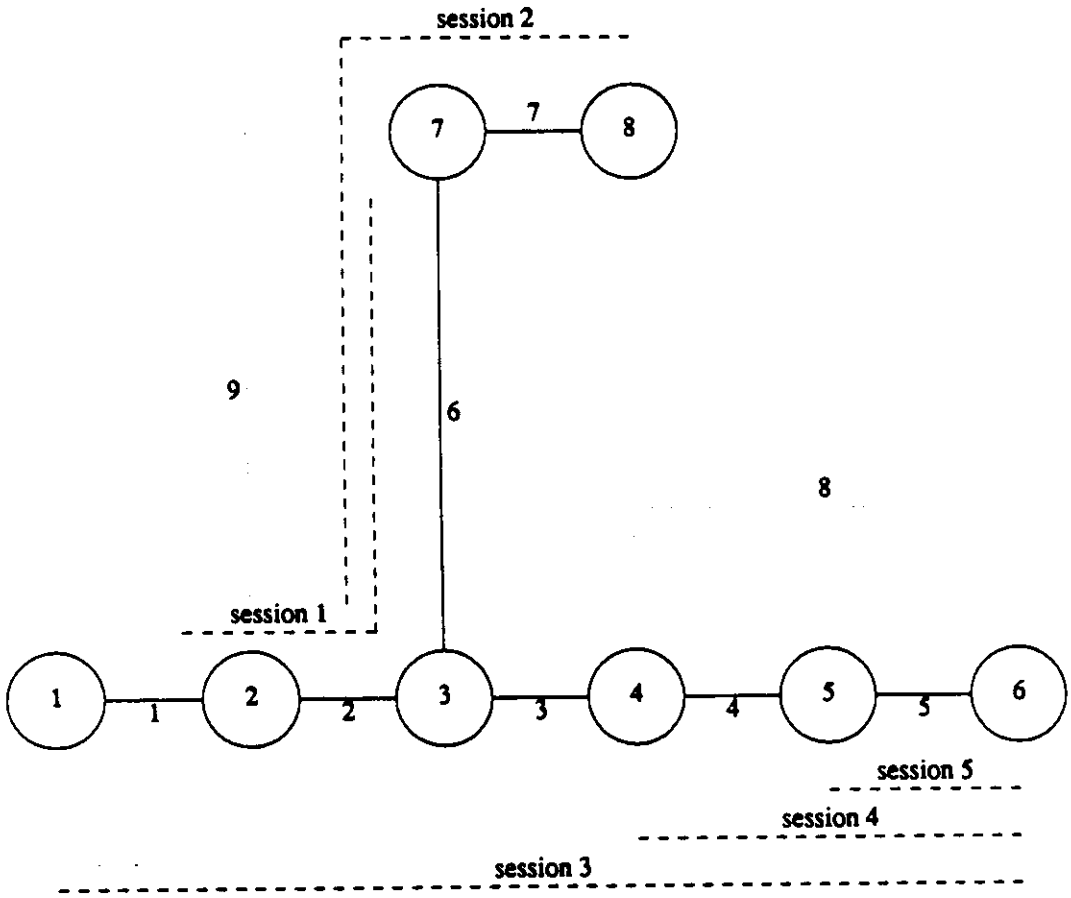

Fig. 6.5 Eight node network example.

Notice that three sessions, namely session 3,4,5 are sharing channel 5 and two sessions, namely session 1,2 are sharing channel 6. The VC input rates are

$d_i = 1\ \forall i$, the channel capacities are $C_i = 1\ \forall i$,and the packet length for the sessions are $b_i = 1\ \forall i$. These values are the same as those in section 4.6 i.e., the service time for the packet at the channels is $S = 1$. The values of U and L are respectively 8 and 4.

VC throughput results are shown in Table 6.1. Note that channel 6 capacity is equally subdivided between sessions 1 and 2 while channel 5 capacity is equally subdivided among sessions 3,4 and 5. This subdivision is expected since channel 6 is the bottleneck for sessions 1 and 2 while channel 5 is the bottleneck for sessions 3,4 and 5 under heavy traffic load. Thus, the backpressure mechanism achieves the fairness conditions proposed by Jaffe [Jaff81b] i.e., the bottleneck sharing given in Chapter 2.

Table 6.1
Network throughputs with uniform heavy traffic loads

Session	1	2	3	4	5
Source	2	3	1	4	5
Dest.	7	8	6	6	6
Demand	1.000	1.000	1.000	1.000	1.000
Thrup.	0.500	0.500	0.333	0.333	0.333

We repeated the experiment with the same set of parameters except that nonuniform traffic pattern was used. The demands and throughputs are given in Table 6.2. In this case the demands for sessions 2 and 3 are only 0.2. We

wanted to see how the backpressure mechanism would affect users with light traffic loads among other users with heavy traffic loads. We expected sessions 2 and 3 to satisfy their traffic requirements while other users with heavy traffic demands would be getting their share accordingly.

Table 6.2
Network throughputs with nonuniform
light to heavy traffic loads - case 1

Session	1	2	3	4	5
Source	2	3	1	4	5
Dest.	7	8	6	6	6
Demand	1.000	0.200	0.200	1.000	1.000
Thrup.	0.785	0.200	0.192	0.404	0.404

From the experiments we observed that channel 5 capacity is subdivided for session 3,4 and 5 where session 3 gets its full demand while sessions 4 and 5 equally share the remaining capacity. Session 2 has a throughput of 0.2 while session 1 captures the remaining capacity of channel 6.

If the demand for session 3 were 1.0 instead of 0.2, we would expect channel 5 capacity would be equally subdivided among sessions 3,4 and 5 since they have equal demands and share the same channel 5. This is indeed the case and the result is given in Table 6.3. We note that session 1 gets less throughput than the previous case. This is because sessions 1 and 3 are utilizing the same channel 2. Session 3 has higher throughput which results in session 1 having less

Table 6.3
Network throughputs with nonuniform
light to heavy traffic loads - case 2

Session	1	2	3	4	5
Source	2	3	1	4	5
Dest.	7	8	6	6	6
Demand	1.000	0.200	1.000	1.000	1.000
Thrup.	0.667	0.200	0.333	0.333	0.333

throughput in this case.

In the three cases, the results satisfy the notion of fairness first defined in [Gerl82] and later applied in Chapter 3 where input rates were adjusted as a means of flow control. Thus, the backpressure mechanism which we presented is a fair scheme under different traffic load situations.

The effect of traffic load and buffer size on throughput and delay of a network using different control schemes have been reported in [Gies78]. We found that the backpressure mechanism exhibits the same effect on average network delay with regard to increasing traffic load i.e., increasing the traffic load will increase the average network delay. However it is not clear what is the effect of different values of U and L on VC throughput and delay under heavy traffic load. Here we consider a range of values, not just a single value of buffer size. We use the network shown in Figure 6.6 which consists of 10 nodes, 9 VC's and 13 full-duplex channels for our study.

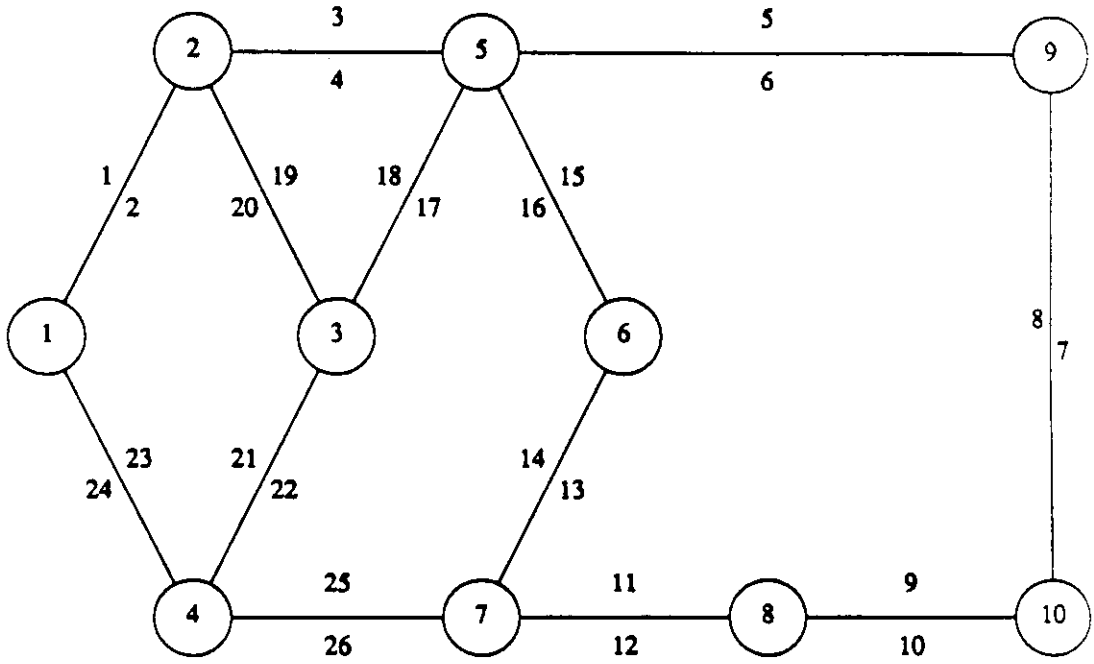

Fig. 6.6 A ten node network.

The traffic requirements and the VC paths are given in Table 6.4. The direction of the channel specified in the "route" column should be obvious from Figure 6.6 given the source and destination of the session. The capacities are $C = 2.50$ for all channels.

The average network delay (averaged over all VC's) for different values of U and L is tabulated in Table 6.5. We limit the lowest value of L to be 1 and the highest value of U to be 10.

Table 6.4
Traffic requirements and routes for
the network in Figure 6.6.

session	source	dest.	demand	route	shared channels
1	2	6	2	4-15	15
2	4	9	2	21-18-5	21,5
3	5	1	2	15-14-26-24	15,14,26
4	10	3	2	10-12-26-21	26,21
5	3	8	2	22-25-11	22,11
6	6	10	2	14-11-9	14,11,9
7	7	2	2	13-16-17-19	17
8	8	5	2	9-8-6	9,6
9	9	4	2	6-17-22	6,17,22

We find that the individual session throughputs all equal 1.25 (which is half the channel capacity) regardless of the value of U and L. If we take a closer look at the number of competing VC's at any communication channel, we note that it is at most 2. This is the reason for the equal sharing of the channel capacity since all sessions mutually interfere with one another.

If we look at the average network delay for various values of U and L, we note that for the same range U - L (say 4), the higher the value of U, the higher is the average network delay. Likewise, if we fix the value of L, we notice that the smaller the range, the lower is the average network delay. And, if we fix the value of U, the smaller the range, the higher is the average network delay. This can be explained as follows. Under heavy traffic load and with respect to a fixed

Table 6.5
Average network delays with different values of U and L

values of L	values of U	range	average network delay	values of L	values of U	range	average network delay
1	5	4	10.410	1	5	4	10.410
1	9	8	11.165	4	5	1	11.186
2	6	4	10.873	2	6	4	10.873
2	8	6	11.231	3	6	3	11.035
3	6	3	11.035	5	7	2	11.852
3	10	7	11.630	6	7	1	12.110
4	5	1	11.186	2	8	6	11.231
4	8	4	11.783	4	8	4	11.783
5	7	2	11.852	1	9	8	11.165
5	10	5	12.383	6	9	3	12.461
6	7	1	12.110	3	10	7	11.630
6	9	3	12.461	5	10	5	12.383

value of L, the value of U is the upper limit of the number of packets at the node. More packets at the node will lead to higher node delay. This corresponds to the fact that (for a fixed value of L) the larger the range, the higher is the average delay. In contrast, with a fixed value of U under heavy traffic load, the value of L is the number of packet at the node when RR is sent back. Thus the lower the value of L, the longer the node is in the RNR mode and therefore the fewer the packets in the network. This implies that (with a fixed value of U) the smaller the range, the higher is the average network delay. Of course this will affect the total throughput of the network. In our case all sessions have the same throughput due to the fact that with U = 5, there is enough packets at a node to

fully load the channels. Also we notice insensitivity of the delay with different values of U and L. This is due to the fact that with 5 as the lowest value for U, we have about 10 packets for each chain in the network to overload the channel (2 is the average number of hops traversed).

6.5.2 Comparison of Input, Window and Backpressure control with respect to fairness

We have presented results for network throughputs with backpressure control mechanism under different traffic loads in the previous section. It is shown that backpressure alone is a fair scheme. We next use the network in Figure 5.4 which is again reproduced here (Fig. 6.7) for the comparison of input, window and backpressure control. We use the values of fairness measure as a function of network average delay under the three control schemes for our comparison. Different values of T_{max} were used in the input rate control experiment. Different values of U were used in the backpressure control experiment. The comparison of values of fairness measure with the window control is shown in Figure 6.8. It is seen that the window control is better than the input rate control in terms of smaller fairness value for a fixed T_{max}. (Recall that optimal fairness corresponds to fairness measure = 0). However for delay greater than a certain T_{max}, the input rate control shows a better performance. This is obvious since an increase in window sizes will eventually overload the network with packets to render the delay higher without increasing throughput. On the other hand input rates are adjusted to satisfy the delay constraint.

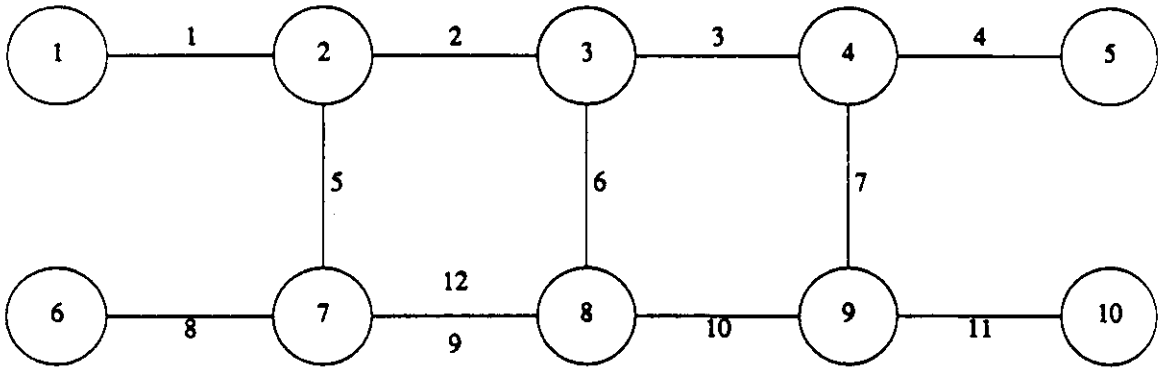

Fig. 6.7 Ten Node example.

We include the backpressure control in our comparison in Figure 6.9. We notice that window control is the best among the three control schemes in terms of minimizing fairness value. It has a lower fairness value than the backpressure control for the range of T_{max} considered. As a fact, we expected input rate control to be the best among the three schemes. Instead, we found that window control is better than both the input rate control and backpressure. One possible explanation is that the closed network model for window control assumes unlimited number of packets at the source. When acknowledgement is received at the source, a packet is always available to be sent. This is not necessarily true in the open network model because of the statistical fluctuation of traffic at the source. This accounts for the higher average throughput in the window control model than the input rate control model for the same delay constraint. This is also valid

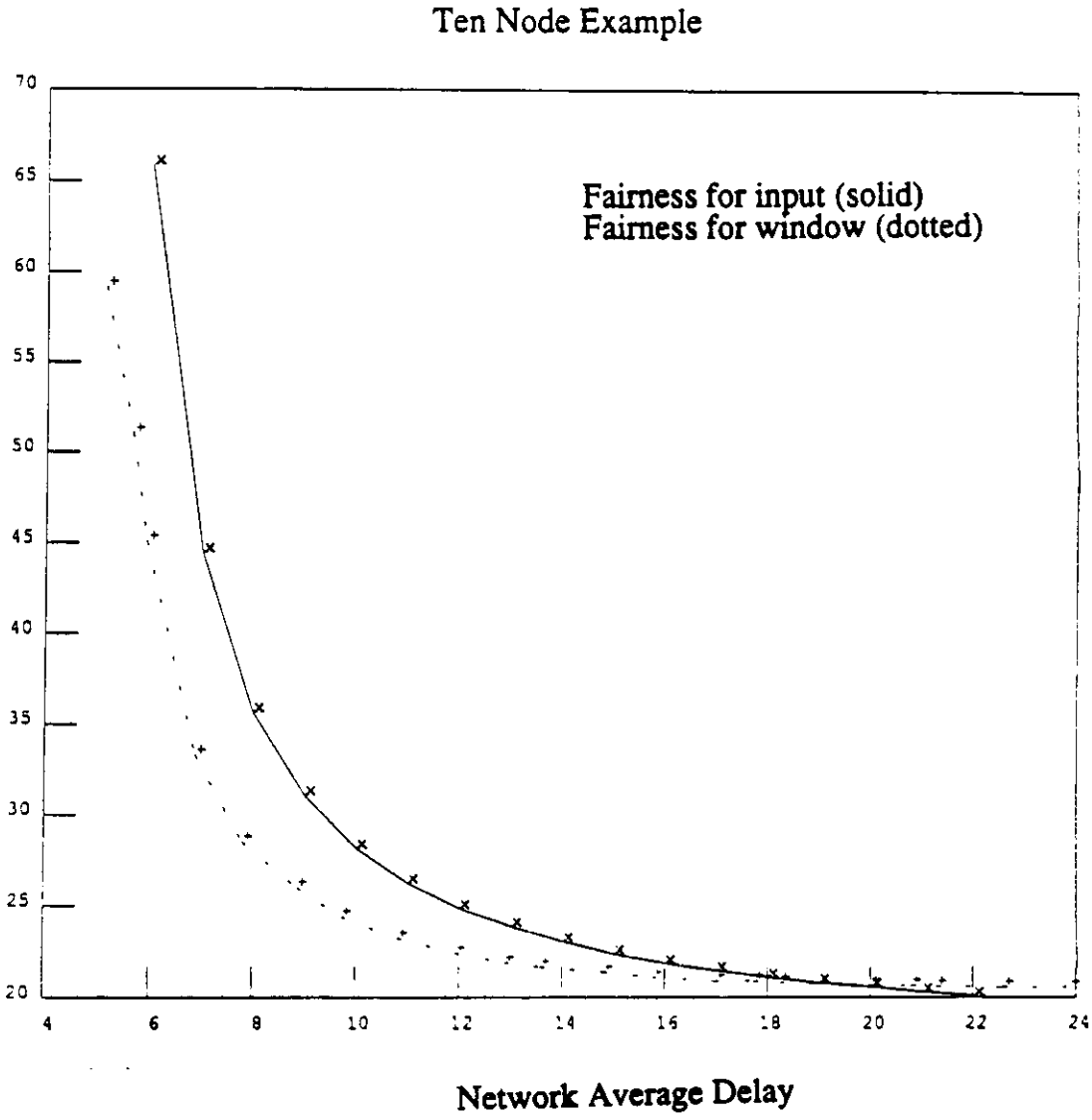

Fig. 6.8 Comparison of input and window control with respect to the fairness measure.

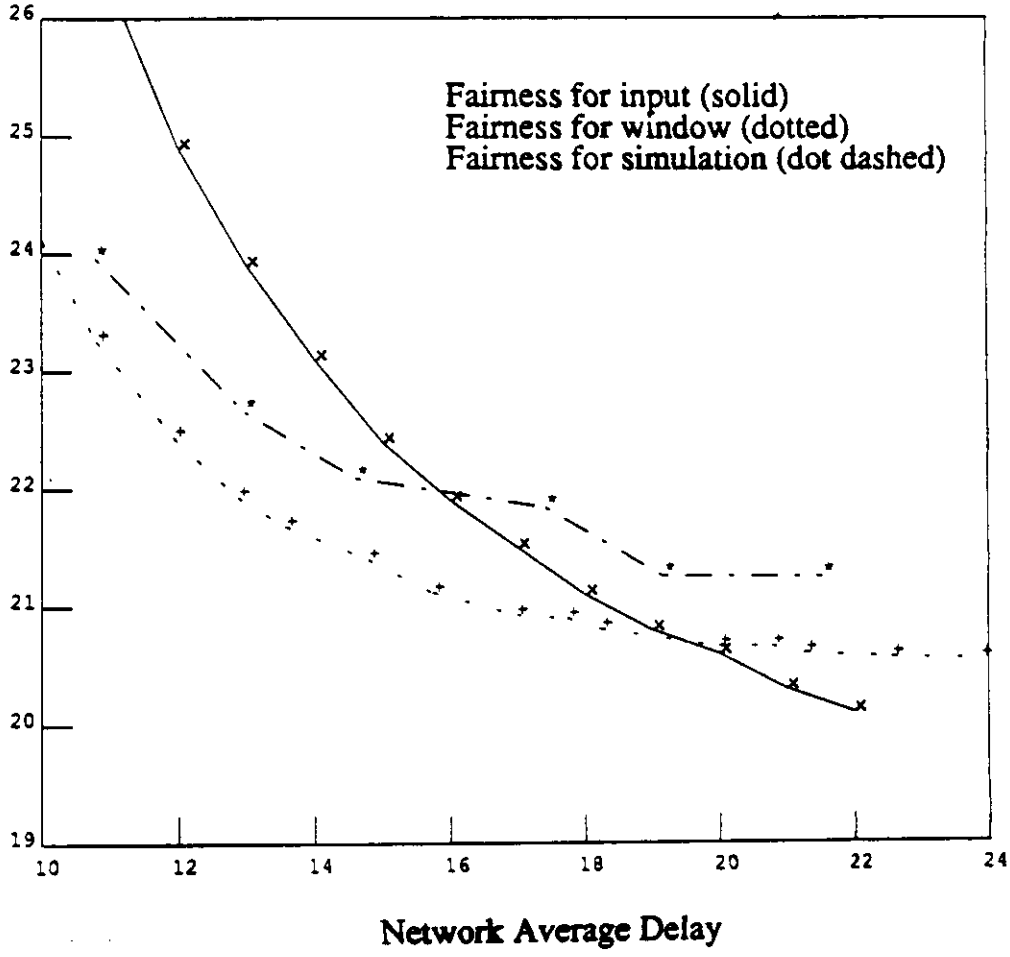

Fig. 6.9 Comparison of input, window and backpressure with respect to the fairness measure.

for comparison between window control and backpressure control. Similar behaviour is observed between the backpressure control and the input rate flow control and can be accounted for in the same way.

6.5.3 Comparison of Window Control and Window with Backpressure under different Traffic Load

We next examine the impact on network performance by VC window control and combination of backpressure with window flow control using the example in Figure 6.7. In this network we have three sessions (shown in Table 6.6). Session 2 interacts at channel 6 with session 1 which has interaction with session 3 at channel 10. Therefore, changing the window size of one session will alter the throughputs of all sessions. It is not clear a priori which set of window sizes should give the best performance in terms of throughput and delay.

Two sets of experiments were run for this example. The respective traffic requirements are given in Table 6.6. The first set uses nonuniform light to heavy traffic inputs and the second set uses uniform, heavy traffic inputs. Also for each set of the experiment, eight different sets of window size (from (1,1,1) to (8,8,8)) are used to study the interplay between backpressure and window control. Table 6.7 shows the different situations under which we ran our experiment.

Table 6.6
Traffic requirements and routes for
the ten node example.

session	source	dest.	heavy traffic demand	light to heavy traffic demand	route
1	1	10	4	0.1	1-2-6-10-11
2	5	6	4	5.0	4-3-6-12-8
3	2	4	4	4.0	5-9-10-7

Table 6.7
Situations under which the experiment is run

Uniform heavy traffic load		Nonuniform light to heavy traffic load	
8 sets of windows		8 sets of windows	
window control alone	window + backpressure control	window control alone	window + backpressure control

The Effect on the Fairness Measure

In Figure 6.10 and Figure 6.11 we plot the fairness measure $P = \sum_{i=1}^{3} \frac{d_i - \lambda_i}{\lambda_i}$ against different sets of window size (the number w in the horizontal axis stands for window set [w,w,w]) under the uniform heavy traffic load and nonuniform light to heavy traffic load using window scheme (Fig. 6.10) and

combination of window with backpressure scheme (Fig.6.11) as flow control mechanisms. We notice that the value of the fairness measure remains approximately constant after a certain set of window size, namely (3,3,3) under the heavy traffic load and (5,5,5) for the light to heavy traffic load for both control mechanisms. This is because the network is overloaded after these sets of window. The throughputs are constrained by the capacities of the communication channels. Increasing the window sizes will not increase the network throughputs. In contrast, increasing window sizes before this threshold will increase the throughputs of individual sessions and thus decrease (i.e., improve) the value of the fairness measure. There is almost no difference between the fairness curve achieved under the two different control schemes. This motivates us to further investigate the effect on individual virtual circuit throughputs under different schemes.

The Effect on the Average Network Delay

In Figure 6.12 the average network delay is plotted as a function of the window set for uniform heavy traffic load and nonuniform light to heavy traffic load using only window as the control scheme. Figure 6.13 is the plot using a combination of window and backpressure as the control scheme.

Note the decrease in the average network delay before certain sets of window and the gradual increase in the average network delay after these sets. For the uniform heavy traffic load, this set is (3,3,3) and for the light to heavy traffic

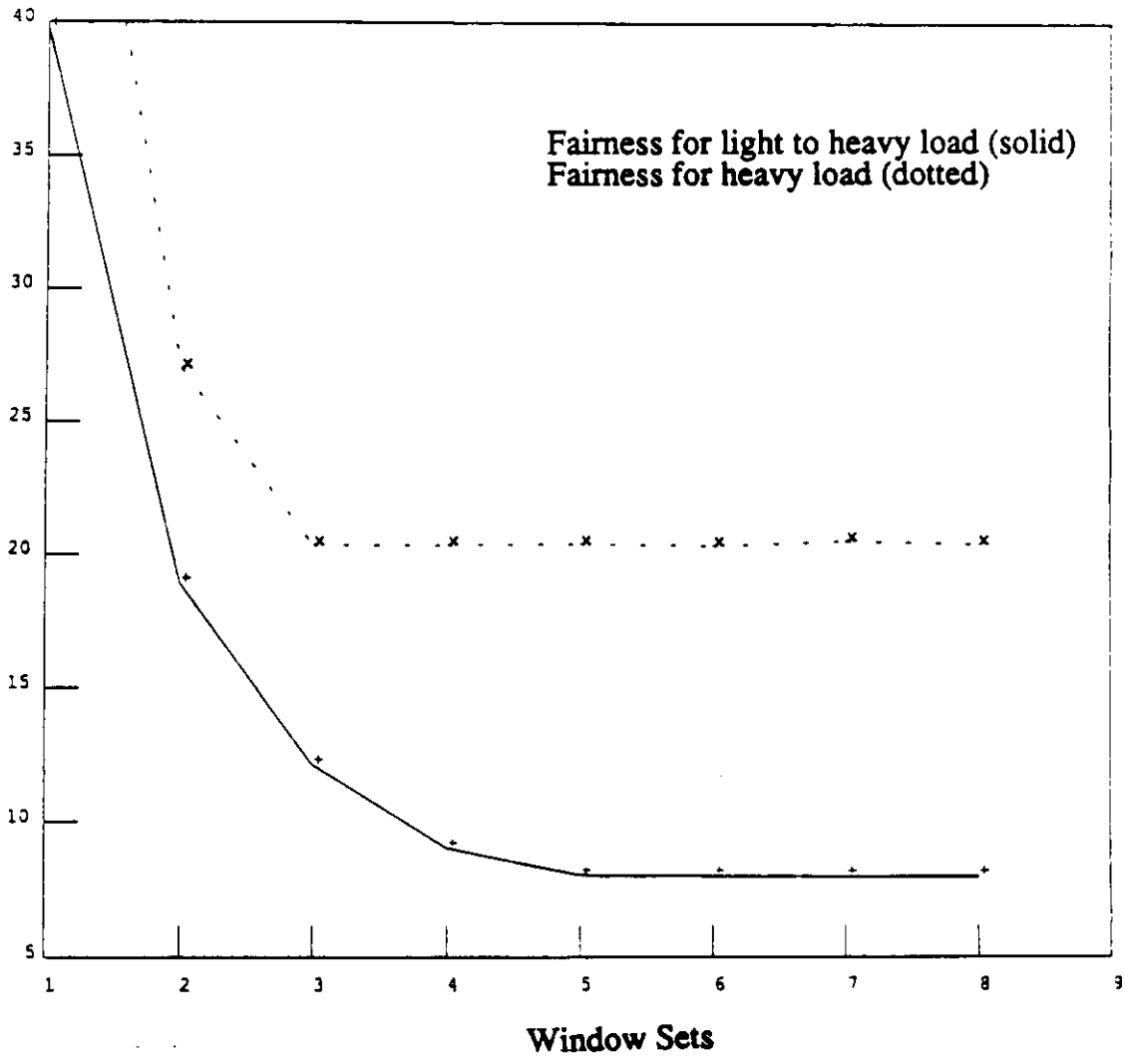

Fig. 6.10 Fairness as a function of window under window flow control alone.

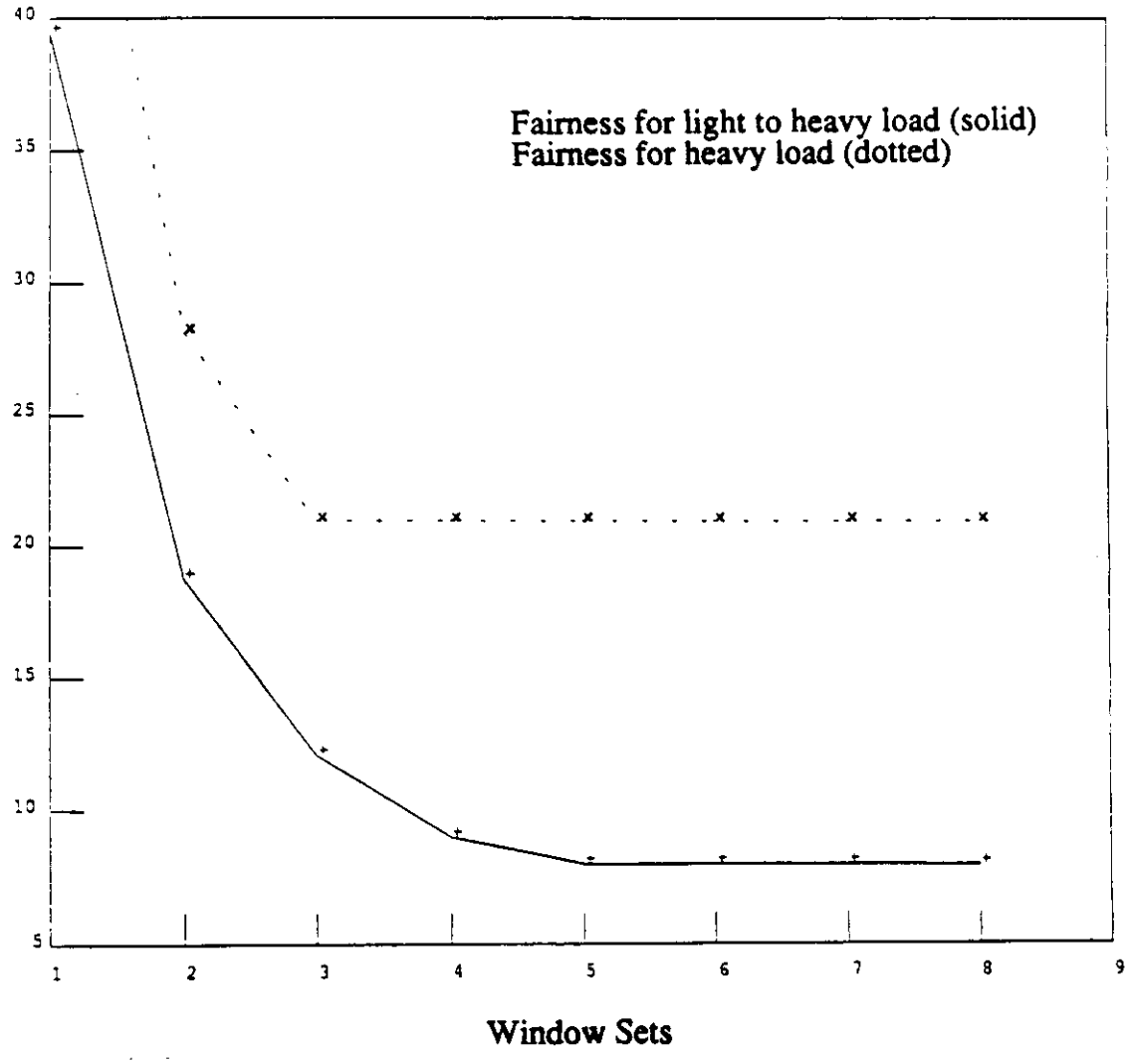

Fig. 6.11 Fairness as a function of window under window with backpressure control.

load, this set is (5,5,5). The smaller set for the heavy traffic load is due to the fact that for this set the traffic is heavy enough to saturate the network. A larger set of windows is required for the light to heavy traffic load to achieve the same effect.

The increase in average network delay for large windows is due to the fact that further increase in window will allow more packets waiting at the intermediate nodes. In addition, rate of delay increase under the heavy traffic load grows sharply when compared to the rate of increase under the light to heavy traffic load. This is also expected.

If we compare both the values of the fairness measure and the average network delay for the two control schemes, they display similar trends with different window sets. The addition of backpressure to the window control does not change the performance in regard to the "average" behavior of the network.

The Effect on Individual Virtual Circuit Throughput

(a) Light to heavy traffic load

In Figure 6.14 and Figure 6.15 the individual VC throughput is plotted as a function of window set for light to heavy traffic load under the two control schemes.

Increasing the window size increases the throughputs for VC 2 and VC 3 while throughput for VC 1 remains the same due to the small input traffic

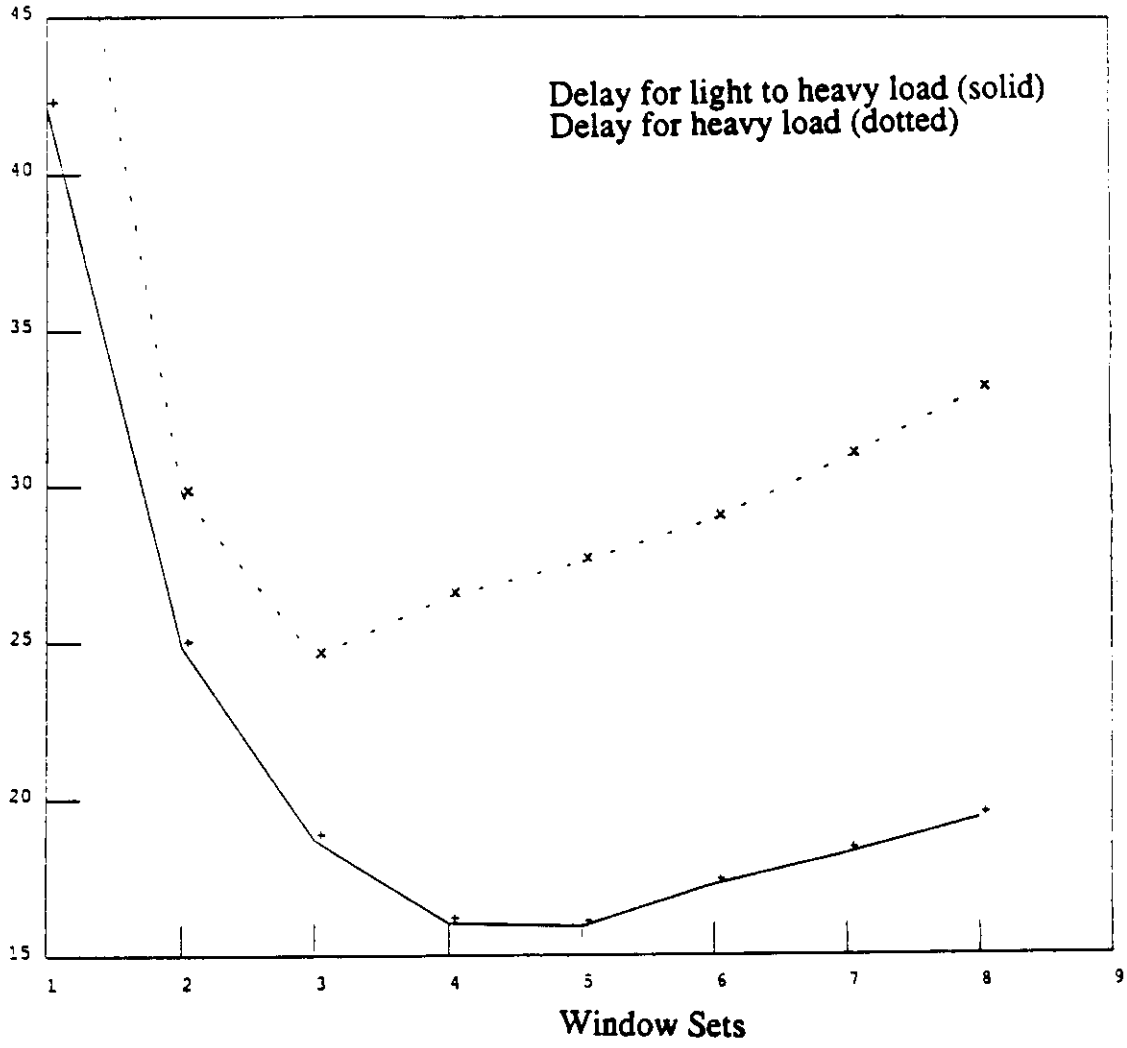

Fig. 6.12 Average delay as a function of window under window flow control alone.

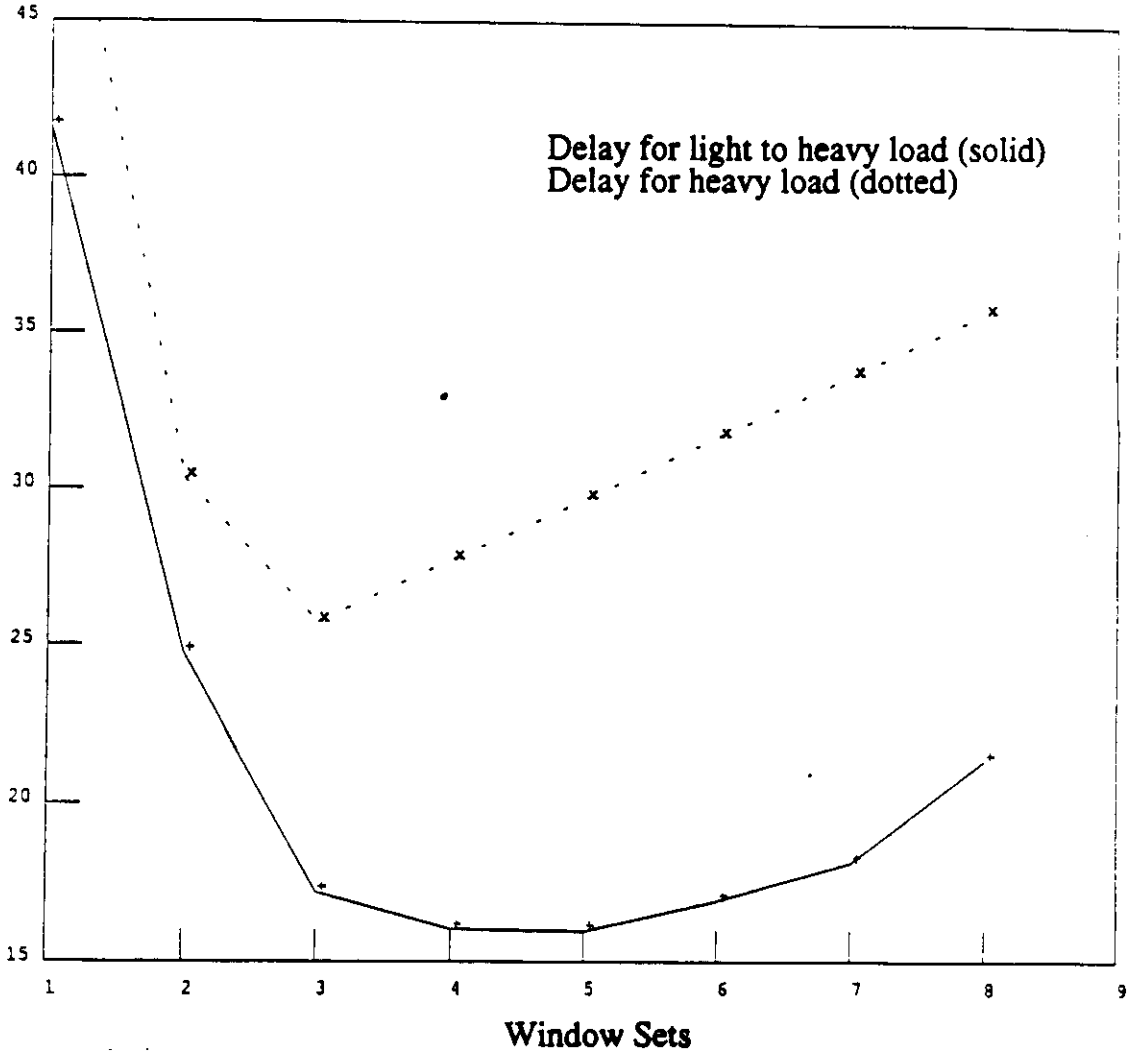

Fig. 6.15 Individual throughput as a function of window for light to heavy traffic load under window with backpressure control.

requirement. Therefore traffic demand for VC 1 is satisfied independent of the window set. In this case, the threshold value of the window set is again (5,5,5). Any window set larger or equal to (5,5,5) has enough packets in transit to saturate the network and the throughput rate of a VC levels off as window set increases.

VC's 2 and 3 have the same individual throughput since VC 1 interacts with VC 2 at channel 6 and VC 3 at channel 10. The input rates of VC 2 and 3 are heavy enough to saturate the remaining capacities of channel 6 and 10. However, before the threshold window set of (5,5,5) it is observed that VC 3 has higher throughput than that of VC 2. The explanation is that the number of channels traversed by VC 3 (4 hops) is less than that of VC 2 (5 hops). This leads to a faster return of the acknowledgement to the source resulting in higher throughput for VC 3.

If we compare the individual throughput for the two control schemes, they also display similar trends under light to heavy traffic load. The addition of backpressure to window control does not change the behavior in regard to the individual chain throughputs.

(b) **Heavy traffic load**

In Figure 6.16 and Figure 6.17 the throughput of individual VC's is plotted as a function of window set for heavy traffic load under the window control scheme and combination of window with backpressure control scheme

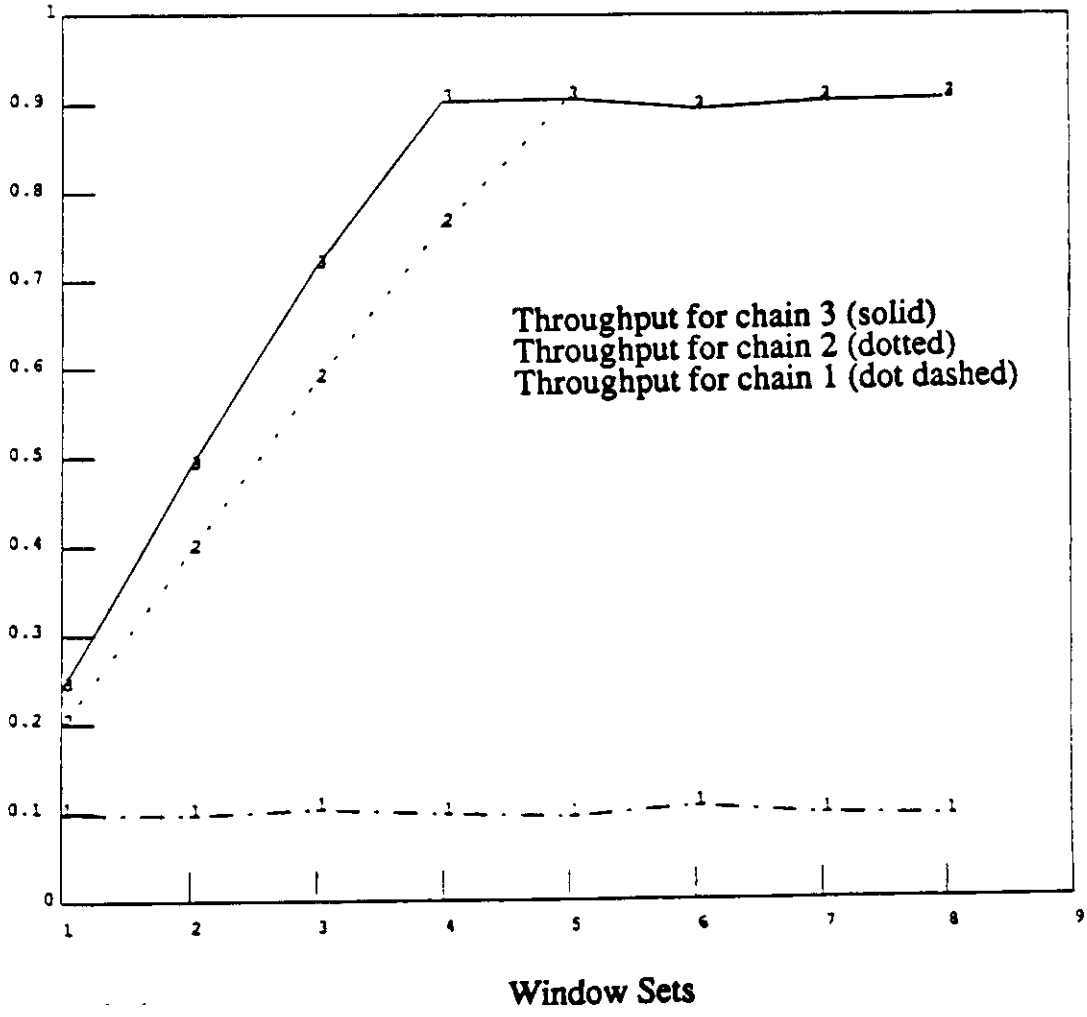

Fig. 6.14 Individual throughput as a function of window for light to heavy traffic load under window flow control alone.

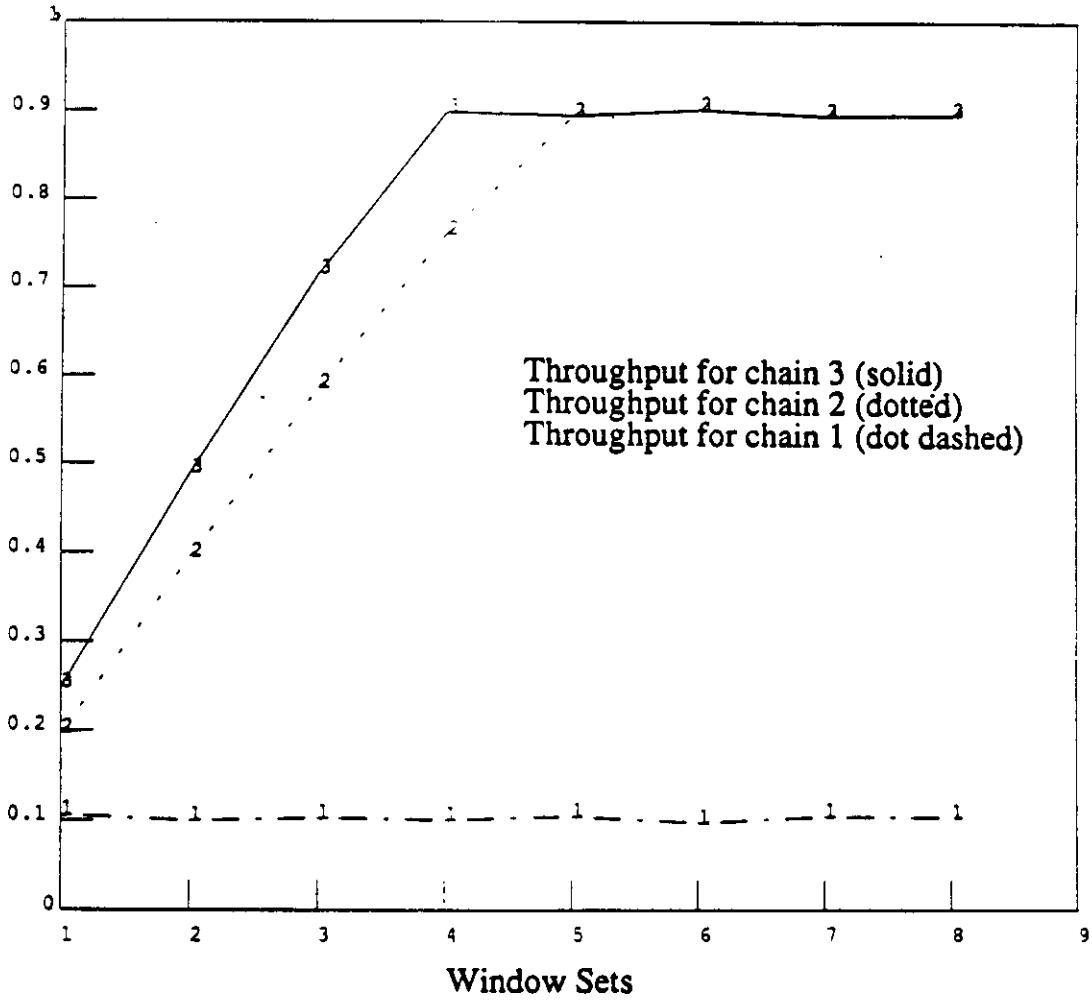

Fig. 6.15 Individual throughput as a function of window for light to heavy traffic load under window with backpressure control.

respectively.

Note that increasing VC window sizes has different effects under the two control schemes. The effects observed under the light to heavy traffic loads given in (a) are no longer present. Here, the threshold value of the window set is (3,3,3) but the behavior of the curve before and after this set is much different for the two control schemes.

With window flow control and window set equal or larger than (3,3,3), VC 1 has less throughput than that of VC 2 and 3 while VC 2 and 3 have equal throughputs. The explanation is as follows. While virtual channel windows provide some form of input control for the network, uniform window set does not necessary impose 'equal' control. The interaction among the VC's determines the individual throughput. In this case VC 1 was interfered twice, once at channel 6 with VC 2 and once at channel 10 with VC 3. This leads to the observed phenomenon. Window control with window set smaller (3,3,3) has the interesting effect that all three VC's have equal throughputs. The result is explained by the fact that the number of packets admitted into the network is small (imposed by the small window set) such that no interference occurs at the channels resulting equal throughputs for the VC under the heavy traffic load.

With combination of window and backpressure scheme and window set equal or larger than (3,3,3), VC's 1,2 and 3 all have equal throughputs. A possible explanation is that for heavy traffic and large set of window, the flow control

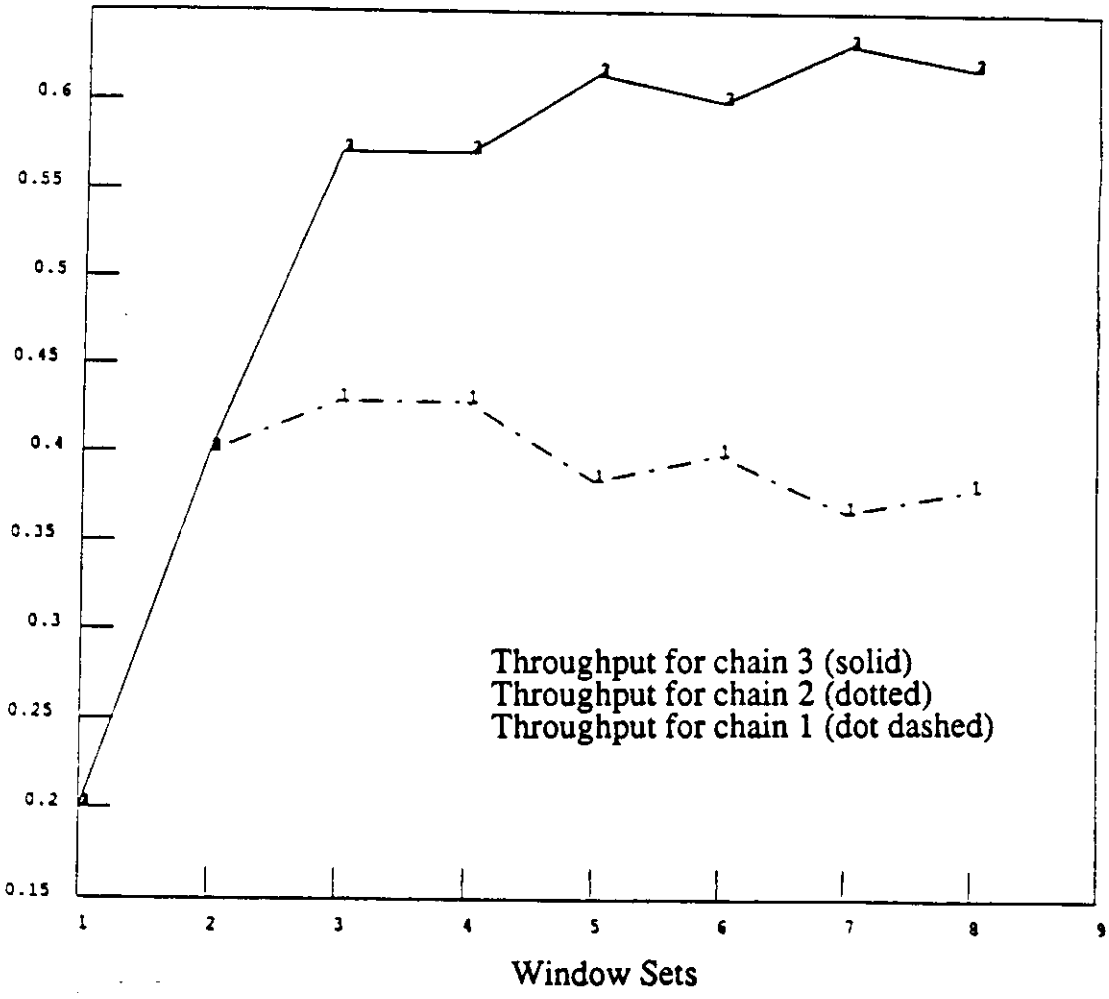

Fig. 6.16 Individual throughput as a function of window for heavy traffic load under window flow control alone.

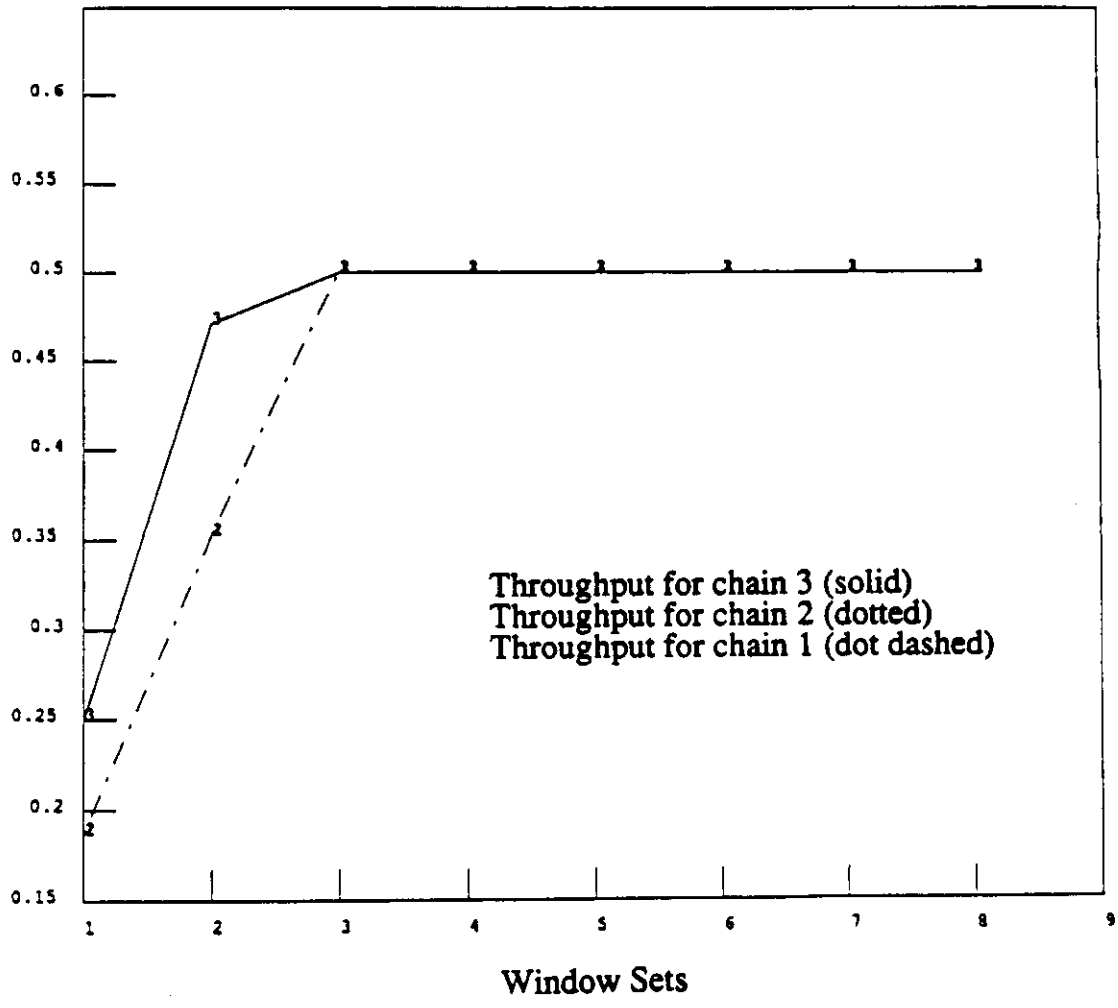

Fig. 6.17 Individual throughput as a function of window for heavy traffic load under window with backpressure control.

is done by backpressure mechanism. Due to the round robin service scheme discipline used at the intermediate nodes, the consequence is that the VC's have equal throughputs.

Combination of window and backpressure control and window set smaller than (3,3,3) has the effect that VC's 1 and 2 have the same throughput while VC 3 has a higher throughput than both VC's 1 and 2. The result can be explained because VC 3 has a shorted path length (4 hops) than VC's 1 and 2 (5 hops) leading to faster return of acknowledgement thus higher throughput. Notice how the path length makes a difference in throughput here while it is not true in window flow control alone.

An interesting phenomenon is observed when the individual throughputs under the two control mechanisms are compared. The VC's have equal throughputs before window set (3,3,3) under window flow control alone. The same effect is observed after window set (3,3,3) under combination of window and backpressure control. Difference in throughputs is observed after the window set (3,3,3) under window flow control alone and before window set (3,3,3) under combination of window and backpressure control. These observations can be accounted for when we consider the effect of window sets on the individual VC delays.

If we compare the individual throughputs for the two control schemes, backpressure with window control displays far better behavior with large win-

dow set under heavy traffic load. The addition of backpressure to window control "regulates" the individual chain throughput to be the same.

The Effect on Individual Virtual Circuit Delay

(a) Light to heavy traffic loads

In Figure 6.18 and Figure 6.19 the individual VC delay is plotted as a function of window set for light to heavy traffic load under the two control schemes, namely window control scheme and combination of window with backpressure control scheme.

Increasing the VC window sizes decreases the VC delays with window set less than (5,5,5) because more packets are allowed into the network reducing the waiting time at the source nodes. Again the threshold value of the window set is (5,5,5). Equal delays for VC's 2 and 3 are observed while VC 1 manages to have the same delay for the backpressure with window control scheme and increases slightly for the window control scheme. The reason for this is that VC 1 interacts with VC's 2 and 3 at channel 6 and channel 10. Increasing VC's 2 and 3 window size will in turn increase delay at channel 6 and 10 thus delay for VC 1. However in the backpressure with window scheme, delay at channel 6 and 10 does not increase because the backpressure scheme has the effect of limiting the number of packets at the intermediate nodes.

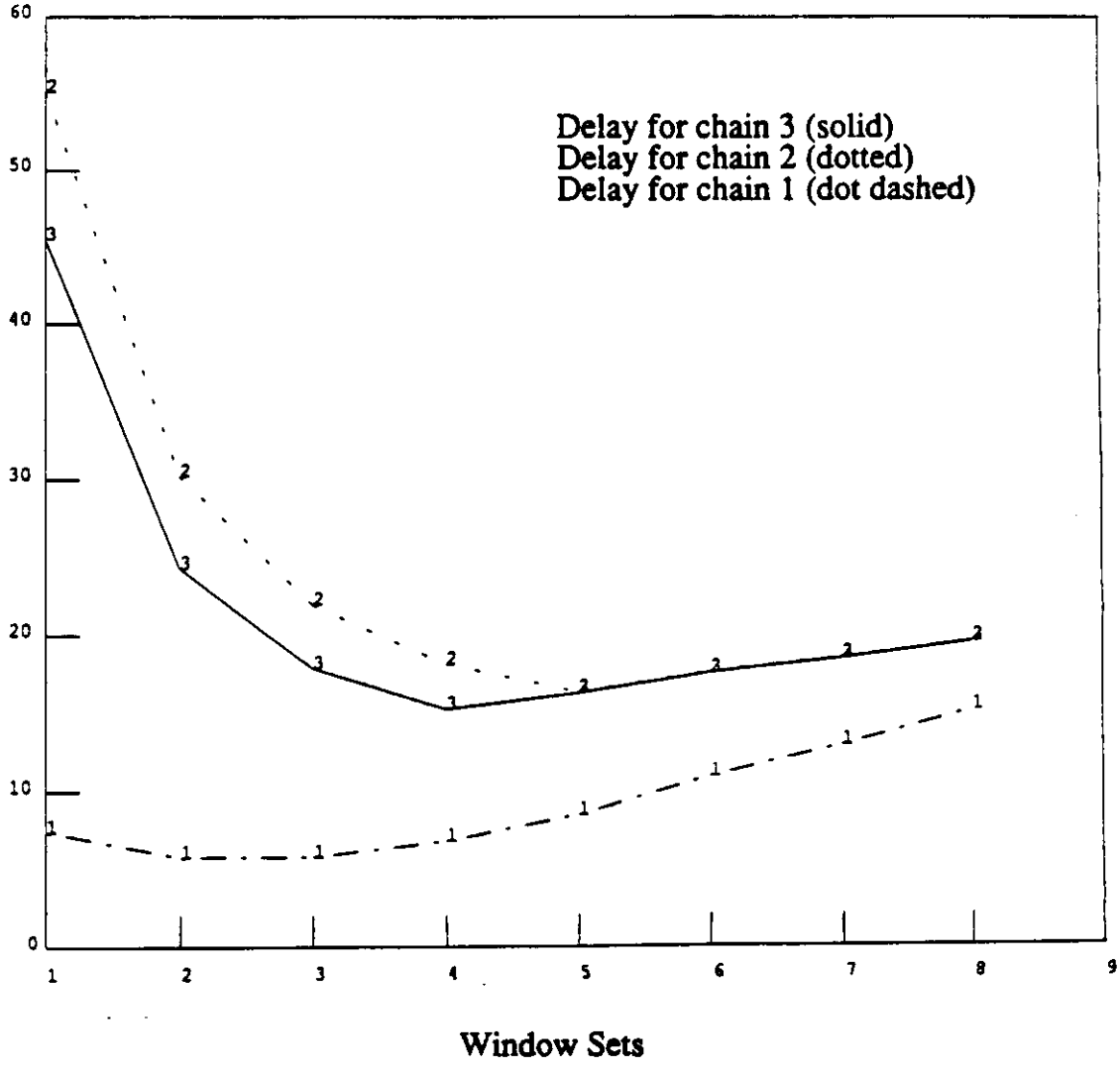

Fig. 6.18 Individual delay as a function of window for light to heavy traffic load under window flow control alone.

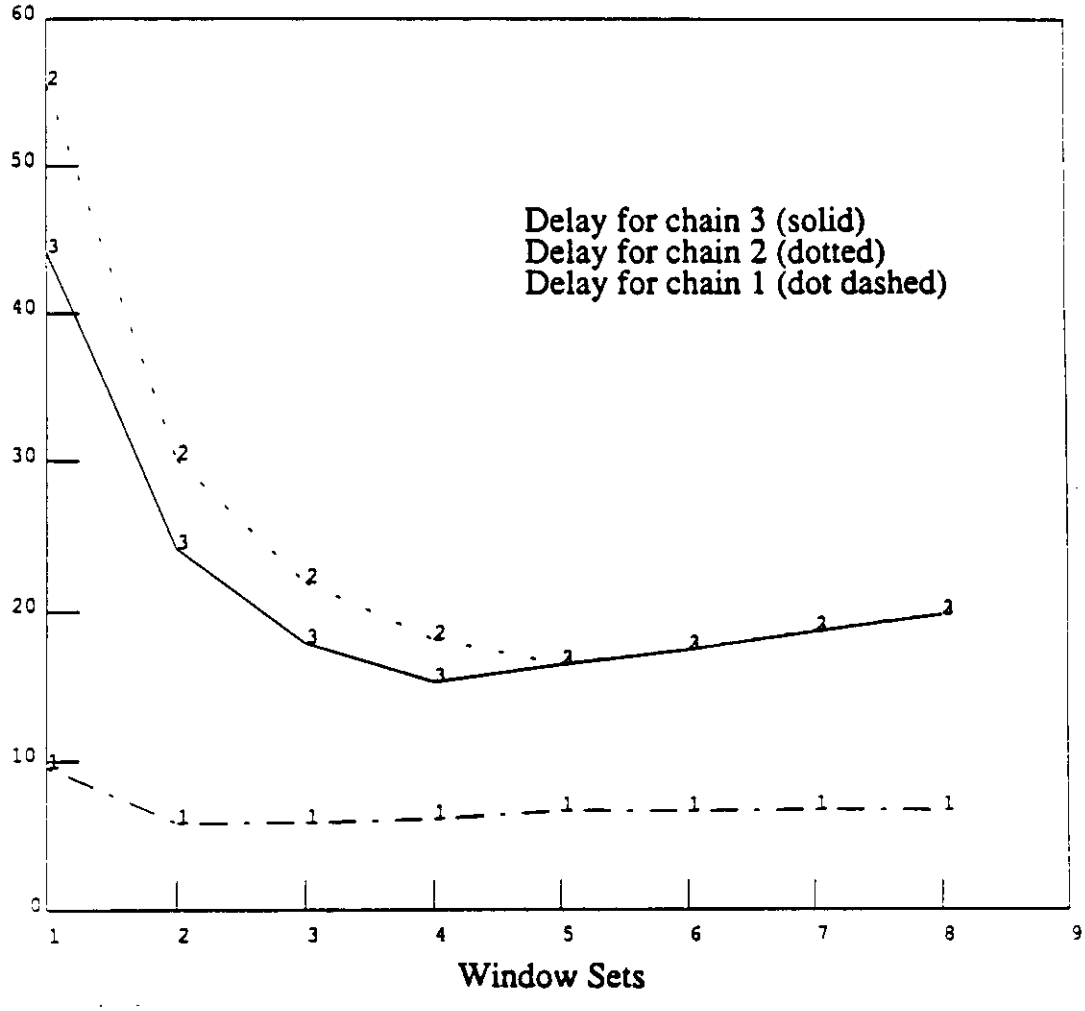

Fig. 6.19 Individual delay as a function of window for light to heavy traffic load under window with backpressure control.

The phenomenon that VC's 2 and 3 has the same individual delay after window set (5,5,5) can be explained. Although VC 2 has longer path length (5 hops) than that of VC 3 (4 hops), the node delay at node 8 is greater than the node delay at node 3. This is because shorter path allows faster return of acknowledgement which in turn allows more VC 3 packets to get into the network, in particular more packet at node 8. However before threshold window set (5,5,5) it is observed that VC 3 has a smaller delay than that of VC 2. The reason is that the number of channel traversed by VC 3 (4 hops) is less than that of VC 2 (5 hops). This leads to a smaller delay at the source for VC 3. Unlike the previous observation, here there is not enough traffic (due to smaller window set) to make the node delay at node 8 greater than the node delay at node 3.

If we compare the individual chain delays for the two control schemes, they display similar trends under the light to heavy traffic load. The addition of backpressure to window control does not change the behavior with respect to the individual chain delays.

(b) Heavy traffic load

In Figure 6.20 and Figure 6.21 the individual VC delay is plotted as a function of window set for heavy traffic load under window control scheme and combination of window with backpressure control scheme respectively.

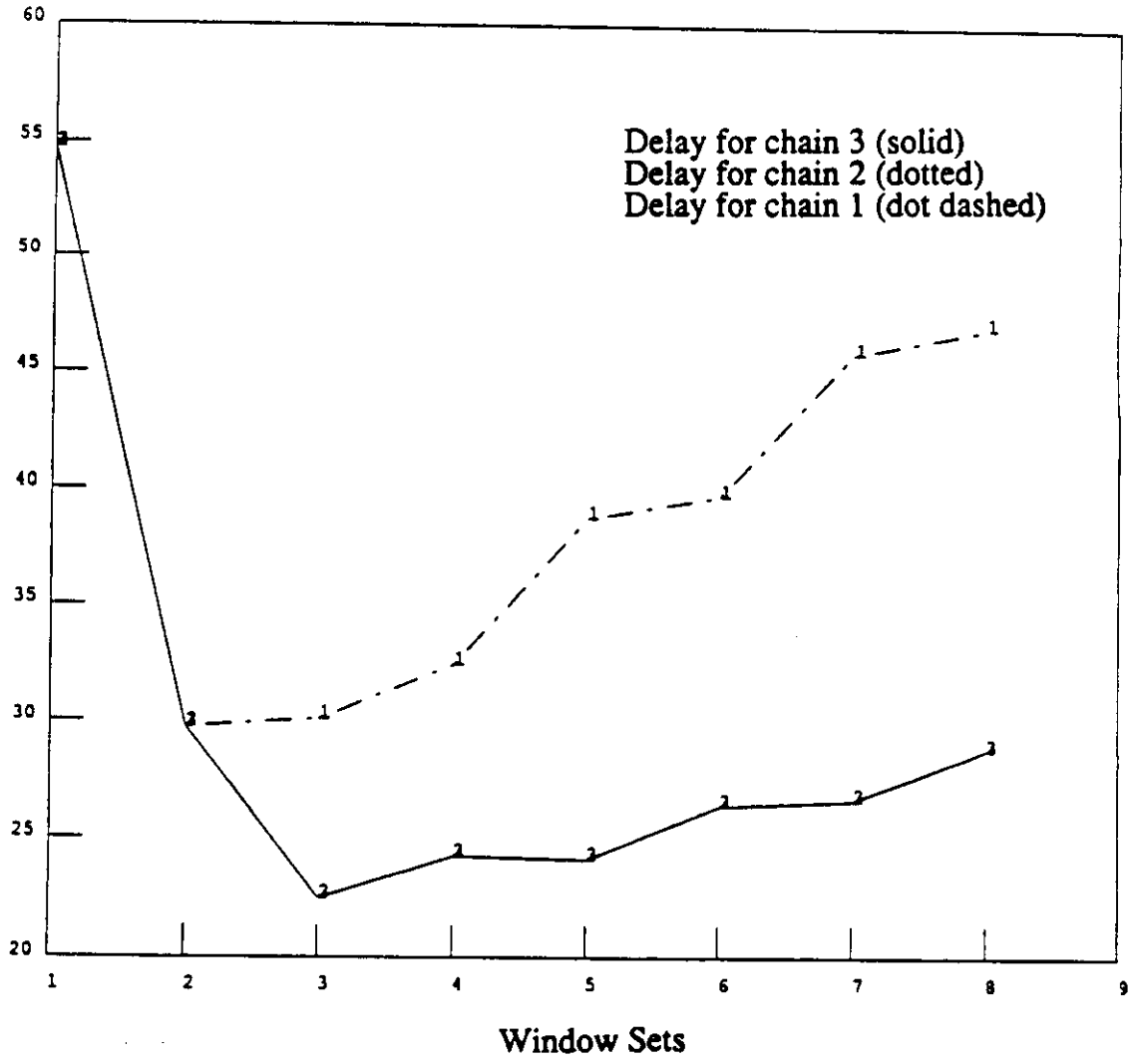

Fig. 6.20 Individual delay as a function of window for heavy traffic load under window flow control alone.

Note that when VC window sizes increase the individual VC delay decreases for window set less than (3,3,3) but increases for window set larger than (3,3,3). It is because before (3,3,3) increasing window will allow more packets into the network thus reducing the waiting time at the source node without overloading the intermediate nodes. This is not true when window set increases beyond (3,3,3).

With window flow control and window set equal or larger than (3,3,3), VC 1 has higher delays than that of VC's 2 and 3 while VC's 2 and 3 have equal delays. The explanation is as follows. Virtual channel 1 interacts with both VC's 2 and 3, it is interfered twice accounting for higher delay. One may argue that this may delay the return of acknowledgement to the source thus making the node delay smaller. However, after window set (3,3,3) the traffic is heavy enough to have high delay at the source. Virtual channels 2 and 3 have the same delays after (3,3,3) is due to the fact although VC 2 has a longer path length than that of VC 3, the node delay at node 8 is greater than the node delay at node 3. It can be seen how unfair treatment VC 1 receives under uniform window set. It has less throughput compared with VC's 2 and 3 (Fig. 6.16) but it also suffers higher delay.

Window flow control and window set smaller than (3,3,3) also has an interesting effect that all three VC's have the same delays. The explanation is that VC 3 has higher node delay at node 8 due to its shorter path length. This allows faster return of acknowledgement to the source which in turn allows more

VC 3 packets into the network, in particular more packets at node 8. Notice how this differs from that of light to heavy traffic load (Fig. 6.18) with window less than (3,3,3). There the path length makes a difference in the individual VC delays. With the present heavy traffic load, the path length does not make any difference.

With combination of window and backpressure scheme and window set equal or larger than (3,3,3), VC's 1,2 and 3 have equal delays. A possible explanation is that for heavy traffic load and large window set, the control is essentially done by backpressure mechanism. Again because of the round robin service discipline used at the intermediate nodes, the consequence is that VC's have the same delays when the traffic is heavy enough to overload the network. This is the case for window set larger than (3,3,3).

Combination of window and backpressure control and window set smaller than (3,3,3) has the effect that VC's 1 and 2 have the same delays while VC 3 has a lower delay than both VC's 1 and 2. This is due to the fact that VC 3 has a shorter path length than that of VC's 1 and 2. This leads to a faster return of acknowledgement to the source thus giving smaller delay at the source. One may argue that this will result in higher node delay at node 8 but again the round robin service discipline of the backpressure mechanism at the intermediate node has the effect of preventing accumulation of packets at node 8. It is seen what a favorable treatment VC 3 receives. It has more throughput than that of VC 1 and VC 2 (Fig. 6.17) but it also suffers less delay under heavy traffic load.

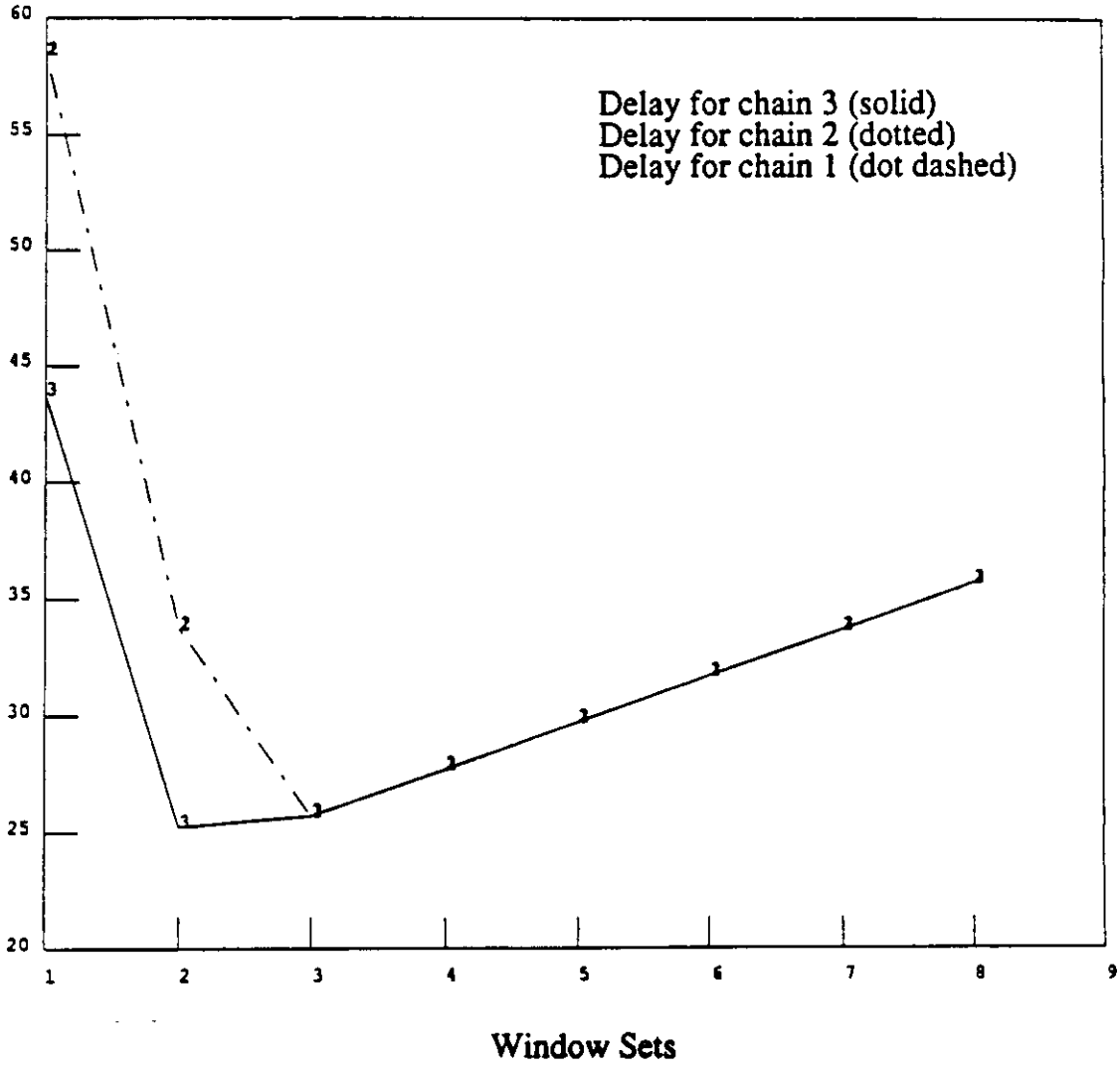

Fig. 6.21 Individual delay as a function of window for heavy traffic load under window with backpressure control.

Now the phenomenon observed in the section "effect of window set on throughputs for heavy traffic load" can be explained. The difference in throughput after window set (3,3,3) under window control alone is due to the corresponding difference in delay. It results in higher delay for VC 3 thus smaller throughput. The difference in throughput before window set (3,3,3) under window and backpressure control is also due to the corresponding difference in delay. It results in lower delay for VC 3 thus higher throughput.

If we compare the individual chain delays under the two control schemes, backpressure with window control displays better behavior with large window set under heavy traffic load. The addition of backpressure to window control "regulates" the individual chain delays to be the same.

6.6 Conclusion

In this chapter we have studied the network performance in terms of fairness, average network delay, individual chain throughputs and delays for the two control schemes with different sets of windows under different traffic loads. We find that there is a critical window set where performance behavior differs before and after this set. In general this window set is smaller for heavy traffic load than the light traffic load.

The sets of end-to-end window play an important role in the network performance. They serve as a "global" control over the number of packets in the network, whereas the backpressure serves as a "local" node to node control. The

backpressure has a significant impact to regulate the individual chain throughputs and delays when there is enough traffic in the network to cause congestion i.e., when the window sizes are large. Thus, under users' heavy demands, backpressure with window control seems to be the best scheme one can use in regard to fairness.

In summary, the choice of the "right" set of window has a significant effect on individual chain throughputs and delays. This window set depends on parameters such as the interactions among the chains and individual traffic demands. The backpressure has the effect of distributing the resources equally among the competing users provided that enough traffic is in the network.

CHAPTER 7

CONCLUSIONS AND SUGGESTIONS FOR FUTURE RESEARCH

7.1 Conclusions

The present research was motivated by the potential unfairness in a network environment where users compete for common resources. Basically, flow control is the key to maintain fairness. Our main contribution was the development of efficient algorithms to achieve fairness under different flow control mechanisms, namely input rate flow control, window flow control and backpressure. The algorithms are very efficient, allowing the network designer to select network parameters in a straight forward manner so as to improve fairness. The simplicity of the quantitative measure of fairness used in our study allows a fairly easy optimization of such measure.

This research probably represents the first attempt (in the published literature) at defining a simple and robust measure of fairness and playing with both routing and flow control to optimize such measure. An important contribution of this research is the development of an algorithm for window selection. In fact the intractability of the problem of finding the exact window set solution makes our heuristic algorithm very attractive. Although approximate, our solutions are shown to be very close to optimum.

Another important contribution is the algorithm which manipulates both routing and input rate control to achieve fairness. In this case, we were able to fine the optimal solution. Finally, the interrelation between backpressure and window flow control and their effect on fairness was studied under different traffic loads. This study has revealed that there exists a critical set of windows beyond which the backpressure provides a fair allocation of resources under heavy traffic load.

7.2 Directions for further Research

In our window optimization problem we have assumed fixed routing. The fixed routing assumption could be relaxed by combining both routing and window flow control in the optimization. Combining both routing and window flow control to optimize our measure of fairness is a very challenging task since both routing and window combination are difficult combinatorial problems. Approximate algorithm will certainly be the approach to solve it. Some preliminary work on a related problem has recently been reported. More precisely, the routing problem on a network of M closed chains was transformed into a simpler routing problem on a mixed network with $M-1$ open chains and a single closed chain. The objective was to minimize network delay [Silv84].

An obvious question which arises during fairness optimization is whether fairness across all users is indeed the best solution under all circumstances. We know, for example, that control messages are much more sensitive to delay than

file transfers. Thus, priorities must be assigned to utilize network resources efficiently. Furthermore, users who are willing to pay more should be entitled to receive better service. Thus, one should find the way of incorporating priorities in a fairness optimization problem.

A reasonable way to combine priorities and fairness is to assign priority based on user group characteristics, e.g., the interactive traffic group and the file transfer group. Fairness may then be enforced within each group, but not necessarily across groups. This gives more flexibility in designing fair networks, yet accounting for distinct user requirement.

The issue of fairness arises because of the interaction among competing users for network resources. A formal mathematical definition of degree of interaction among the users may be helpful for further study on this topic. In fact, the degree of the congestion and the fair allocation of resources are closely related to the way users interact (and interfere) with one another using such resources. Investigation can then be based on this interaction as the key parameter. This will help in giving us insight in the interaction between routing and flow control because a change of route will change the whole pattern of interaction among the users.

Finally, we have only considered centralized solutions to the fairness problem. Namely, flow control and routing parameters have been optimized using centralized algorithms Distributed solutions (i.e., solutions generated by

the network itself via distributed computation) should also be pursued because of their potential benefits in overhead reduction (by eliminating the need for control information transfer with a central controller), in response time improvement and in fault tolerance.

APPENDIX A

MEETING THE CONSTRAINT

As we mentioned in section 4.5.2, three constraints have to be satisfied after the adjustment and scaling of user throughputs. The three constraints (4.3), (4.4), (4.5) are repeated below, followed by the approach we use to insure constraint requirements are met.

$$T \leq T_{max} \tag{A.1}$$

$$0 \leq \lambda_i \leq d_i \qquad i = 1,...R \tag{A.2}$$

$$f_j \leq C_j \qquad j = 1,...NL \tag{A.3}$$

In Step 6 of the algorithm presented in section 4.5, the factor used in scaling was found as follows

a. Obtain the minimum ratio $C_j/f_j \qquad \forall$ j

b. The ratio obtained in (a) is used as the maximum scaling factor in the bisection method to meet $T \leq T_{max}$ requirement.

c. The optimal scaling factor found in the bisection method such that $T \leq T_{max}$ is used to obtain the new set $\{\lambda_i\}$. Any λ_i exceeding d_i will be truncated to d_i.

Now we go into the most delicate step of the algorithm, namely Step 5. We have showed how to calculate the new $\{\lambda_i\}$ in section 4.5.2. In doing that we excluded those users with the negative ω_i. Therefore

$$K^{1/2} = \frac{\sum_{\substack{i=1 \\ i \in P}}^{R} \omega_i \lambda_i}{\sum_{\substack{i=1 \\ i \in P}}^{R} (\omega_i d_i)^{1/2}}$$

$P = \{i \mid \omega_i \text{ is non negative }\}$ and

$$\lambda_i' = \left[\frac{K d_i}{\omega_i}\right]^{1/2}$$

After calculating the new $\{\lambda_i\}$, any λ_i exceeding d_i will be truncated to d_i.

The factor we use in the convex combination of the new λ_i' and old λ_i is found as follows.

a. For constraint (A.2) we have

$$\lambda_i^{next} - \lambda_i + \alpha_1(\lambda_i' - \lambda_i) \geq 0 \qquad \text{(A.4)}$$

Solving Eq.(A.4) gives

$$\alpha_1 < \min\left[\frac{\lambda_i}{\lambda_i - \lambda_i'}\right] \quad \text{for } \lambda_i' < \lambda_i$$

and for $\lambda_i' \geq \lambda_i$ the condition is always satisfied.

b. For constraint (A.3) we have

$$f_j + \alpha_2(f_j' - f_j) \le C_j \tag{A.5}$$

Solving Eq.(A.5) gives

$$\alpha_2 \le \frac{C_j - f_j}{f_j' - f_j} \quad \text{for } f_j' > f_j$$

and for $f_j' < f_j$ the condition is always satisfied.

c. The minimum of α_1, α_2 is then used in the convex combination in finding the new λ_i 's.

d. Any new λ_i exceeding d_i will be truncated to d_i.

APPENDIX B

APPROXIMATION FOR $\lambda_i(\mathbf{N} + \mathbf{e}_c)$

The goal is to express $\lambda_i(\mathbf{N} + \mathbf{e}_c)$ as a function of $\lambda_i(\mathbf{N})$, $\forall\, i$. We start by using Schweitzer's approach to approximate average queue length L_{ik} at center k for chain i, after the addition of one more customer in chain c, namely:

$$L_{ik}(\mathbf{N} + \mathbf{e}_c) = L_{ik}(\mathbf{N}) \qquad i \neq c$$

$$L_{ck}(\mathbf{N} + \mathbf{e}_c) = \frac{N_c + 1}{N_c} L_{ck}(\mathbf{N}) \qquad i = c$$

We can rewrite the above expressions for population $\mathbf{N} - \mathbf{e}_i$, i.e. one less customer in chain i as follows:

$$L_{ik}(\mathbf{N} + \mathbf{e}_c - \mathbf{e}_i) = L_{ik}(\mathbf{N} - \mathbf{e}_i) \qquad i \neq c$$

$$L_{ck}(\mathbf{N} + \mathbf{e}_c - \mathbf{e}_i) = \frac{N_c + 1}{N_c} L_{ck}(\mathbf{N} - \mathbf{e}_i) \qquad i = c$$

Therefore:

$$L_k(\mathbf{N} - \mathbf{e}_i + \mathbf{e}_c) = L_k(\mathbf{N} - \mathbf{e}_i) + \frac{L_{ck}(\mathbf{N} - \mathbf{e}_i)}{N_c}$$

We recall that from MVA principles [Reis80] and from Little's Result the waiting time W_{ik} can be expressed as:

$$W_{ik}(\mathbf{N}) = x_{ik}[1 + L_k(\mathbf{N} - \mathbf{e}_i)]$$

where x_{ik} is the mean service time of chain i at center k.

If one customer is added to chain c the expression becomes:

$$W_{ik}(\mathbf{N} + \mathbf{e}_c) = x_{ik}[1 + L_k(\mathbf{N} - \mathbf{e}_i + \mathbf{e}_c)]$$

$$= x_{ik}\left[1 + L_k(\mathbf{N} - \mathbf{e}_i) + \frac{L_{ck}(\mathbf{N} - \mathbf{e}_i)}{N_c}\right] \qquad \text{(B1)}$$

Using Schweitzer's approximation we get:

$$L_k(\mathbf{N} - \mathbf{e}_i) = L_k(\mathbf{N}) - \frac{L_{ik}(\mathbf{N})}{N_i}$$

and

$$L_{ck}(\mathbf{N} - \mathbf{e}_i) = L_{ck}(\mathbf{N}) \qquad i \neq k$$

Substituting in (B1) we get:

$$W_{ik}(\mathbf{N} + \mathbf{e}_c) = x_{ik}\left[1 + L_k(\mathbf{N}) - \frac{L_{ik}(\mathbf{N})}{N_i} + \frac{L_{ck}(\mathbf{N})}{N_c}\right]$$

$$= W_{ik}(\mathbf{N}) + x_{ik}\frac{L_{ck}(\mathbf{N})}{N_c}$$

This follows from the fact that

$$W_{ik}(\mathbf{N}) = x_{ik}\left[1 + L_k(\mathbf{N}) - \frac{L_{ik}(\mathbf{N})}{N_i}\right]$$

Hence, chain i mean throughput with one more customer in chain c can be

expressed as:

$$\lambda_i(\mathbf{N}+\mathbf{e}_c)=\frac{N_i}{\sum_k \theta_{ik} W_{ik}(\mathbf{N})+\sum_k \theta_{ik} x_{ik}\dfrac{L_{ck}(\mathbf{N})}{N_c}}$$

$$=\frac{N_i}{\dfrac{N_i}{\lambda_i(\mathbf{N})}+\sum_k a_{ik}\dfrac{L_{ck}(\mathbf{N})}{N_c}} \qquad \text{(B2)}$$

where θ_{ik} is chain i visit ratio at center k

Note that all the quantities on the RHS of Eq. (B2) are known since they are part of the current solution.

REFERENCES

[Atki80] Atkins, J., "Path Control: The Transport Network of SNA," *IEEE Transactions on Communications*, Vol. 28, No. 4, April 1980, pp. 527-538.

[Bask75] Baskett, F., K.M. Chandy, R.R. Muntz, and F.G. Palacios, "Open, Closed, and Mixed Networks of Queues with Different Classes of Customers," *Journal of the ACM*, Vol. 22, No. 2, April, 1975, pp. 248-260.

[Berr82] Berry, R., K.M. Chandy, J. Misra, and D. Neuse, "PAWS 2.0 Performance Analyst's Workbench System," Information Research Associations, Austin, TX, Dec. 1982.

[Bhar81] Bharath-Kumar, K. and J. Jaffe, "A New Approach to Performance-Oriented Flow Control," *IEEE Transactions on Communications*, Vol. 29, No. 4, April 1981, pp. 427-435.

[Buze73] Buzen, J.P., "Computational Algorithms for Closed Queueing Networks with Exponential Servers," *Communications of the ACM*, Vol. 16, No. 9, Sept. 1973, pp. 527-531.

[Chan75] Chandy, K.M., U. Herzog, and L.S. Woo, "Parametric Analysis of Queueing Networks," *IBM J. Res. Develop.*, Vol. 19, 1975, pp. 36-42.

[Chan82] Chandy, K.M. and D. Neuse, "Linearizer: A Heuristic Algorithm for Queueing Network Models of Computing Systems," *Communications of the ACM*, Vol. 25, No. 2, Feb. 1982, pp. 126-134.

[Ever63] Everett, H., "Generalized Langrange Multiplexers Method for Solving Problems of Optimal Allocation of Resources," *Operations Research*, Vol. 11, 1963, pp. 399-418.

[Frat73] Fratta, L., M. Gerla, and L. Kleinrock, "The Flow Deviation Method: An Approach to Store-and-Forward Communication Network Design," *Networks*, Vol. 3, 1973, pp. 97-133.

[Gail81] Gail, R. and L. Kleinrock, "An Invariant Property of Computer Network Power," in *Proceedings ICC*, Denver, Colorado: June 1981, pp. 63.1.1-63.1.5.

[Gall80] Gallager, R. and S. Golestaani, "Flow Control and Routing Algorithms for Data Networks," in *Proceedings ICCC*, Atlanta, Georgia: Oct. 1980, pp. 779-784.

[Gerl80a] Gerla, M. and P. Nilsson, "Routing and Flow Control Interplay in Computer Networks," in *Proceedings ICCC*, Atlanta, Georgia: Oct. 1980, pp. 84-89.

[Gerl80b] Gerla, M. and L. Kleinrock, "Flow Control: A Comparative Survey," *IEEE Transactions on Communications*, Vol. 28, No. 4, April 1980, pp. 553-574.

[Gerl82] Gerla, M. and M. Staskauskas, "Fairness in Flow Controlled Networks," *Journal of Telecommunication Networks*, Vol. 1, No. 1, Spring 1982, pp. 29-38.

[Gerl84] Gerla, M., H.W. Chan, and J. Marca, "Routing, Flow Control and Fairness in Computer Networks," in *Proceedings ICC*, Amsterdam, Holland: May 1984, pp. 1272-1276.

[Gies78] Giessler, A., J. Hanle, A. Konig, and E. Pade, "Free Buffer Allocation - An Investigation by Simulation," *Computer Networks*, 1978, pp. 191-208.

[Jack57] Jackson, J.R., "Networks of Waiting Lines," *Operations Research*, Vol. 5, 1957, pp. 518-521.

[Jack63] Jackson, J.R., "Jobshop-Like Queueing Systems," *Mangement Science*, Vol. 10, No. 1, 1963, pp. 131-142.

[Jaff81a] Jaffe, J., "Flow Control Power is Nondecentralizable," *IEEE Transactions on Communications*, Vol. 29, No. 9, Sept. 1981, pp. 1301-1306.

[Jaff81b] Jaffe, J., "Bottleneck Flow Control," *IEEE Transactions on Communications*, Vol. 29, No. 7, July 1981, pp. 954-962.

[Klei64] Kleinrock, L., *Communication Nets: Stochastic Message Flow and Delay*, New York: McGraw-Hill, 1964.

[Klei76] Kleinrock, L., *Queueing Systems, Vol. 2: Computer Applications*, New York, New York: John Wiley & Sons, 1976.

[Klei79] Kleinrock, L., "Power and Deterministic Rules of Thumb for Probabilistic Problems in Computer Communications," in *Proceedings ICC*, Boston, MA: June 1979, pp. 43.1.1-43.1.10.

[Lam82] Lam, S.S. and J.W. Wong, "Queueing Network Models of Packet Switching Networks Part 2: Networks with Population Size Constraints," *Performance Evaluation*, Vol. 2, 1982, pp. 161-180.

[Lam83] Lam, S.S. and Y.L. Lien, "A Tree Convolution Algorithm for the Solution of Queueing Networks," *Communications of the ACM*, Vol. 26, No. 3, March 1983, pp. 203-215.

[Marc84] Marca, J. Ricardo B. de, *A Study on Fairness in Packet Switching Networks*, RJ Brazil: PUC/RJ, March 1984.

[Penn75] Pennotti, M. and M. Schwartz, "Congestion Control in Store and Forward Tandem Links," *IEEE Transactions on Communications*, Vol. 23, No. 12, Dec. 1975, pp. 1434-1443.

[Reis75] Reiser, M. and H. Kobayashi, "Queueing Networks with Multiple Closed Chains: Theory and Computational Algorithms," *IBM J. Res. Develop*, Vol. 19, No. 5, May 1975, pp. 283-294.

[Reis79] Reiser, M., "A Queueing Network Analysis of Computer Communication Networks with Window Flow Control," *IEEE Transactions on Communications*, Vol. 27, No. 8, Aug. 1979, pp. 1199-1209.

[Reis80] Reiser, M. and S. Lavenberg, "Mean-Value Analysis of Closed Multichain Queueing Networks," *Journal of the Association for Computing Machinery*, Vol. 27, No. 2, April 1980, pp. 313-322.

[Sauv82] Sauve, J.P., J.W. Wong, and J.A. Field, "Improving Total Throughput in Packet Switching Networks with Window Flow Control," in *Proceedings Glocom*, Miami, Florida: Dec. 1982, pp. F1.5.1-F1.5.6.

[Schw80] Schwartz, M. and T. Stern, "Routing Techniques Used in Computer Communication Networks," *IEEE Transactions on Communications*, Vol. 28, No. 4, April 1980, pp. 539-552.

[Schw82] Schwartz, M., "Performance Analysis of SNA Virtual Route Pacing Control," *IEEE Transactions on Communications*, Vol. 30, No. 1, Jan. 1982, pp. 172-184.

[Schw79] Schweitzer, P., "Approximate analysis of multiclass closed networks of queues," in *Proceedings Int'l Stochastic Control and Optimization*, Amsterdam, Holland: 1979.

[Silv84] Silva, Edmundo de, "Algorithms for Queueing Network Analysis of Distributed Systems," UCLA, Los Angeles, CA, Tech. Rep. CSD-840040, 1984.

[Tucc85] Tucci, S. and C.H. Sauer, "The Tree MVA algorithm," *Performance Evaluation*, Vol. 5, No. 3, Aug. 1985, pp. 187-196.

[Whit72] Whitney, V. M., "Lagrangian Optimization of Stochastic Communication Systems Models," in *Proceedings MRI Symposium on Computer-Communication Networks*, Brooklyn, New York: April 1972, pp. 332-342.

[Wong] Wong, F. and J. Roberto B. de Marca, "Fairness in Window Flow Controlled Computer Networks," *sumbitted for publication in the IEEE Transactions on Communications,*.

[Wong82a] Wong, J., J. Sauve, and J. Field, "A Study of Fairness in Packet-Switching Networks," *IEEE Transactions on Communications*, Vol. 30, No. 2, Feb. 1982, pp. 346-353.

[Wong82b] Wong, J.W. and S.S. Lam, "Queueing Network Models of Packet Switching Networks Part 1: Open Networks," *Performance Evaluation*, Vol. 2, 1982, pp. 9-21.