Bottleneck Flow Control

JEFFREY M. JAFFE, MEMBER, IEEE

Abstract—**The problem of optimally choosing message rates for users of a store-and-forward network is analyzed. Multiple users sharing the links of the network each attempt to adjust their message rates to achieve an ideal network operating point or an "ideal tradeoff point between high throughput and low delay." Each user has a fixed path or virtual circuit.**

In this environment, a basic definition of "ideal delay–throughput tradeoff" is given and motivated. This definition concentrates on a fair allocation of network resources at network bottlenecks. This "ideal policy" is implemented via a *decentralized* algorithm that achieves the *unique* set of *optimal* throughputs. All sharers constrained by the same bottleneck are treated fairly by being assigned equal throughputs.

A generalized definition of ideal tradeoff is then introduced to provide more flexibility in the choice of message rates. With this definition, the network may accommodate users with different types of message traffic. A transformation technique reduces the problem of optimizing this performance measure to the problem of optimizing the basic measure.

I. INTRODUCTION

VARIOUS store-and-forward packet-switched computer networks have been developed in recent years. The primary function of these networks is to route messages or *packets* from one network location to another. Typically, the source of a message dispatches a packet to a neighboring location or node, which relays the message to another node and so forth, until the message arrives at the destination.

There are a number of disciplines used by networks to funnel a large number of packets from one source to a given destination. For example, ARPANET handles each packet individually [1], trying to find the shortest path for each packet based on changing network characteristics. In this paper we assume a fixed route approach whereby all messages from a given "session" are assigned to a fixed unique route. This approach is currently used in TYMNET [2], [3], IBM's network architecture [4], and various other networks (e.g., [5]). Many sessions may share a given route.

The total time required for transmission of a packet is called its delay. Assuming small nodal processing time, there are two major components to message delay. Since communication links take some time to transmit a message, there is a *transmission* delay component. Also, if a communication link needs to transmit too many packets at once, it temporarily buffers some of them, leading to a *queueing* delay component. The queueing delay clearly depends on the amount of network traffic, and roughly speaking, increases with greater traffic.

Paper appoved by the Editor for Computer Communication of the IEEE Communications Society for publication after presentation at the 5th International Conference on Computer Communication, Atlanta, GA, October 1980. Manuscript received April 25, 1980; revised January 6, 1981. This research was supported in part by the National Science Foundation under Grant EDS-79-25092.

The author is with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598.

Flow control regulates the amount of traffic to maintain good system performance. For example, if the buffers at a link are almost full, some mechanism is needed to slow down the rate of incoming traffic. Otherwise, the buffers would overflow, causing severe queueing delays or even deadlock. Another purpose of flow control is to maintain a good throughput delay tradeoff. If a user is sending a high average message rate (in our studies this is equated with throughput), the resulting delays may be intolerably long. On the other hand, the user would not want to sacrifice too much throughput in order to achieve low delay. Related to this is the notion of *fairly* dividing network resources between competing network users.

In this paper we discuss methods to achieve a well defined notion of system performance which results in fairness to users and a good delay–throughput tradeoff. We concentrate on network access means of flow control [6] where external inputs are throttled based on measurements of internal network congestion. The buffer depletion problem (see [7]) is ignored so that we may concentrate on delay and throughput. Formally, when our model is specified (in Section II), infinite buffers at each link are assumed.

This paper primarily concentrates on the fundamental questions of "what is optimum performance?" and "what notions of optimality are accomplishable in a decentralized environment?". No new method of constraining the input of messages is proposed; it is assumed that message rate is regulated by a simple rate mechanism, i.e., some "black box" at each route which chooses the message rate for that route.

Network access flow control schemes include the isarithmic scheme [8], input buffer limiting [9], and the choke packet scheme [10]. Other schemes are discussed in [6] and [11]. The isarithmic scheme limits the total number of packets allowable in the network. Input buffer limiting locally restricts input traffic in favor of transit traffic.

The "bottleneck flow control" presented here may be viewed as a generalization and abstraction of both the choke packet scheme and certain ideas presented in [9]. Common features with the choke packet scheme are that the decision to decrease message rate is a function of congestion in the bottleneck links. The relationship between the two is further developed throughout this paper. The main difference is that, while optimality is defined in a similar way, the control mechanisms are different. As a result, the choke packet scheme has no *explicit* way of ensuring a specified notion of fairness. On the other hand, bottleneck flow control uses fairness criteria related to those that are described in [9].

In Section III we define and motivate a notion of "optimal tradeoff." An adaptive algorithm is given in Section IV which attempts to achieve this tradeoff in a network that is experiencing changes in traffic patterns and numbers of users. Due

0090-6778/81/0700-0954$00.75

to the changing nature of such a network, it is difficult to state specific "steady-state" properties of the algorithm. We thus restate the problem somewhat to reflect a static network. In that environment it is easier to discuss properties of the "optimal tradeoff" and an algorithm that implements it. In particular, the following is achieved:

- A "decentralized" algorithm is given that always achieves the optimal tradeoff (Sections V and VII).
- The algorithm obtains the tradeoff in linear time [in the number of users (Section VII)].
- The "optimal tradeoff" defines a *unique* set of throughputs that the users of the network must achieve (Section VIII).
- The unique set of optimal throughputs has important "fairness" properties (Section IX).

Section X generalizes these results to the situation where different user classes have different network performance requirements. The main result of Section X is that the techniques developed earlier in the paper may be applied directly to the more general case by a simple transformation technique.

We briefly explain and motivate the notion of a "decentralized" algorithm for flow control. When a user chooses its throughput, the inputs to the process should consist of information locally available to it. The user might be permitted to use information about the interfering traffic on its path, but not about global topology. Basically, in a decentralized algorithm, information not readily available on a user's path should not be usable for throughput determination.

In [12] it is shown that a single user may optimize its power (ratio of throughput to delay) using only such local information. However, in [13] it is shown that, under certain conditions, no decentralized algorithm maximizes power in a multiple user system. Since certain optimality criteria are nondecentralizable, the importance of the decentralizable criterion discussed here is enhanced.

We further remark that the criterion expressed here has other advantages over the power concept. It is shown in [14] that, in some network configurations, optimizing power implies that certain users must choose zero throughput. A corollary of the fairness property of Section IX is that no users are required to have zero throughput at optimal performance. This fact is still true for the generalization of Section X where users are not handled identically in terms of throughput allotment.

II. NETWORK MODEL

We model a data network as a graph (N, L) with vertex (or node) set N and edge (or link) set L. Each link $l \in L$ has a *service rate* of $s(l)$ bits/s. A *path* p in the network is a sequence $p = (n_1, \cdots, n_k)$ with $n_i \in N$ such that for $i = 1, \cdots, k-1$, $l_i = (n_i, n_{i+1}) \in L$. The set $\{l_1, \cdots, l_{k-1}\}$ is denoted $l(p)$, the *links of* p. A path p models a fixed route that is used by one of the "users" of the network.

In order to evaluate the delays on the links, a queueing model is needed which relates throughputs to delay. We use a simple model ([15, Sect. 5.6]) which, as indicated above, has infinite buffers. Specifically, we assume that each link may be modeled as an $M/M/1$ queue, the average message length is b bits/message, there is no nodal processing time, and Kleinrock's independence assumption applies [15].

Define the *capacity* of link l, $c(l)$, by $c(l) = s(l)/b$. Assume that there are K users, all of whose fixed routes use a link l. Let γ_i denote the message rate of the ith user. In that case, the *average* steady-state *delay* for the packets (of each user) that traverse the link at l is $d_l(\gamma) = 1/(c(l) - (\gamma_1 + \cdots + \gamma_K))$. The *average total delay* of packets sent by user i, $D_i(\gamma)$ is the sum of the average delays experienced at the individual links.

III. OPTIMALITY CRITERION

In this section an optimality criterion is presented using several levels of description. First, optimum throughput is defined in terms of link capacity. We explain why our definition might be considered "the optimum operating point of a network." Next, the definition is reformulated to express a tradeoff between user throughput and delay. Section IV gives an adaptive algorithm for optimizing the criterion in a "dynamically changing" network. It is difficult, however, to present any concrete analysis for a rapidly changing network. Starting with Section V we analyze the optimality criterion in a "static" environment.

Recall that $c(l)$ is the capacity of the link l. Let $\gamma(l)$ denote the sum of the throughputs of all users of link l. The maximum value that $\gamma(l)$ can be is $c(l)$ or else messages are generated at a faster rate than they can be transmitted. Certainly, $\gamma(l) > c(l)$ is not a situation we would like to encourage for *any* link. In fact, it is probably not even desirable to have $\gamma(l) = c(l)$ for two reasons. First of all, if $\gamma(l) = c(l)$ the system "never reaches steady state"; the delays of the messages increase over time due to the fact that buffer occupancy approaches infinity. Also, choosing $\gamma(l) = c(l)$ leaves no room for fluctuations in the network. One user may be forced by certain considerations to increase his throughput or new users may attempt to open up new routes sharing link l. For that reason, optimum $\gamma(l)$ is chosen to be somewhat less than $c(l)$, as we proceed to describe. This distance is parameterized by a variable x. This variable permits designers of different systems to choose somewhat different notions of "ideal throughput-delay tradeoff." If they are throughput-oriented, they choose x large; if delay-oriented, then x should be small.

Define the *residual capacity of* l by $r(l) = c(l) - \gamma(l)$. Let γ denote the throughput of a user whose path includes link l. The user *saturates* l if $\gamma = x(r(l))$. The user *overloads* l if $\gamma > x(r(l))$. A user is *overloaded* if it overloads any link on its path. A user is *saturated* if it is not overloaded and it saturates at least one link on its path. These preliminaries prepare us for the following.

Definition: Given a data network (as modeled in Section II) with several paths through the network (corresponding to users of the network), and a rate assigned to each user, the rate assignment is *optimal* if all users are saturated.

Remarks: The way that we keep $\gamma(l)$ somewhat less than $c(l)$ is to guarantee that no user overloads any links. Thus, for each link l, $x(r(l)) \geq \gamma_{\max}$ where $\gamma_{\max}$ is the largest throughput of any user of link l. In addition to keeping $\gamma(l)$ somewhat less than $c(l)$, we also desire a large measure of throughput in the network. Thus, each user must not only prevent

overload—it also must be saturated. Each user would then have the largest possible throughput subject to x and the residual capacities.

To contrast this with the Cyclades choke packet proposal, we remind the reader that optimality in [10] basically requires that no link exceeds a certain threshold of utilization. For instance, $\gamma(l)$ should not exceed (0.8) $(c(l))$ if the threshold equals 0.8.

We feel that it is better to force *saturation* of each user and choose $\gamma(l)$ as a function of γ_{max} for a few reasons. The primary reason is that the choke packet scheme has no regard for the number or types of users of the link, and therefore loses the ability to fairly allocate resources. By fixing the requirement that no link should exceed a certain utilization, one loses the ability to predict transients in future utilization based on current utilization. This is developed further in Section IX. Also assume that $x(r(l)) = \gamma_{max}$. Then, with our definition, if $x = 1$, we can accommodate one new user with throughput γ_{max} without causing $\gamma(l) > c(l)$. Similarly, choosing $r(l) = (\gamma_{max})/x$ protects the network against percentage changes in each user's throughput due to transients. If a user increases his throughput by a factor of $1/x$, the inequality $c(l) \geqslant \gamma(l)$ still applies. Methods of obtaining an optimality criterion similar to "80 percent of utilization," as a limiting case of saturation, are discussed in Section XII.

Next, we motivate saturation as a means of expressing an "optimal delay-throughput tradeoff." Recall that the delay at l is given by $d_l \doteq 1/(c(l) - \gamma(l))$. Thus, saturation for user p is equivalent to

$$\gamma = \min_{l:l\in l(p)} x/d_l(\gamma) \tag{1}$$

From (1) it is evident that saturation is a direct method of expressing a delay-throughput tradeoff for the users of the network. A user may increase its throughput until the delay on its "bottleneck" link is too large. As delay increases, γ is constrained by (1).

Note the role played by the parameter x in all viewpoints of the optimality criterion. From the network point of view, it indicates the amount of traffic fluctuation that is to be protected against. From the user viewpoint, it indicates the amount of effect that increased delay should have on throughput.

There is a third viewpoint of saturation. Using Little's theorem [16], the average number of messages waiting at a link l when the throughput of a user is γ, and the delay is d_l is $\gamma \cdot d_l$. Now if $\gamma \leqslant x/d_l$ for every link l in the path of a given user, the user is willing to tolerate x messages waiting at each link, and a total of x times # (user's links), messages waiting in the system. Thus, the average number of waiting messages that a user will tolerate varies linearly with the length of his path—if the path is longer, the user may have more messages in transit.

To review, the features of optimum network operation based on the use of the saturation measure are

1) protection for the network against changes in users' rates

2) protection for the network from arrivals of new users

3) establishment of delay/throughput tradeoff at the bottleneck link

4) use of the parameter (x) to permit flexibility in the definition of optimum performance

5) protection for the buffers in an average sense

6) fair allocation of resources (Section IX).

In addition to *stating* what optimal performance is (all users saturated), it might be helpful to evaluate how far suboptimal solutions are from optimal. To do this, it is useful to have an objective function which characterizes the quality of a set of throughput assignments. Assume that there are m users with throughputs $\gamma = (\gamma_1, \cdots, \gamma_m)$. Define

$$f(\gamma) = \sum_{i=1}^{m} \left| \gamma_i - \min_{l:l\in l(i)} \frac{x}{d_l(\gamma)} \right|. \tag{2}$$

If each user is saturated at γ, then for all i, $\gamma_i = \min_{l:l\in l(i)} x/d_l(\gamma)$ and $f(\gamma) = 0$. Conversely, if $f(\gamma) = 0$, all users are saturated. Thus, the goal of saturating all users may be conveniently restated as an attempt to minimize f.

IV. AN ADAPTIVE DISTRIBUTED ALGORITHM

An adaptive distributed algorithm which attempts to saturate all paths without overloading any is now given. Each user adjusts its message rate based on information sent to it by the links and nodes on its path. The information needed by a user with path p is

1) its current throughput γ

2) $\min_{l:l\in l(p)} r(l)$.

We do not specify the mechanics of when this information is made available and in what form the information arrives. Each link may know to dispatch information to all users of the link at regular intervals, or alternatively, information gathering may be prompted by a signal from the user. Each link may *compute* $r(l)$ or estimate it based on buffer occupancy. Also, the links may send the throughputs of the individual users of the links, and let user p calculate $r(l)$.

The algorithm executed by user p each time it desires to recalculate its message rate γ' from the old rate γ is

$$\gamma' \leftarrow \min_{l:l\in l(p)} \frac{x(r(l) + \gamma)}{x + 1}. \tag{3}$$

The following explains why we say that the above algorithm attempts to achieve saturation. First, note that after executing one step of the algorithm, the user is saturated. This can be seen as follows. For a link l, the new sum of throughputs $\gamma'(l) = \gamma(l) - \gamma + \gamma'$. Thus, $x(c(l) - \gamma'(l)) = xc(l) - x\gamma'(l) = xc(l) - x\gamma(l) + x\gamma - x\gamma' = \gamma'$ by (3) for the link at which $r(l)$ was minimized. Also, $x(c(l) - \gamma'(l)) \geqslant \gamma'$ for all other links, l, i.e., none is overloaded.

If there were no transients, such as no new users entering the system, *and* each user converged to a steady-state throughput, then those throughputs that are converged to will saturate all users. Any unsaturated or overloaded user must change its throughput! Unfortunately, we are unable to show, even

without transients and new users, that each user does converge. To clearly express an algorithm that saturates all users, we spend the rest of this paper discussing a static case, i.e., no new users.

As a practical matter, the above algorithm would need to be modified in an adaptive situation. Choosing γ' by (3) may cause large deviations in certain user's message rates, leading to instabilities in the system. A better way is to have users slowly change rates in the direction (increase or decrease) implied by (3). The reader is referred to [14] for an algorithm to coordinate user updates, so that many users do not change their rates at once.

V. ALGORITHM TO SATURATE ALL USERS

In this section an algorithm is presented which saturates all users in a static network with a fixed set of users. It is assumed that if a user is assigned by the algorithm to send messages at a rate γ, that indeed its average throughput is γ. (Variations of this are described in Section XI.) The algorithm is decentralized in the sense described above. Each user chooses its throughput based on information provided from its links. In fact, the execution of the algorithm will be presented in a manner which distributes the computation even more—the links (or whatever controls the links) will do some computation in the algorithm. The link computation provides a concise description of the current traffic on the link.

There are a number of idealizations used in this section. It is assumed that each link may accurately calculate message rates of users that use the link. Also, in order to conveniently discuss the convergence time of the algorithm, a synchronous algorithm is assumed (i.e., a clock at each node permits all updates to occur at once). However, the main feature of using "local information," i.e., information accumulated along a user's path, is preserved. In practice, one would probably use a hybrid of the algorithm of Section IV and the algorithm that we proceed to present here.

The algorithm proceeds in iterations. Consider a link l which is shared by a number of users, exactly j of which are not saturated before the ith iteration. Let $\gamma_{\text{sat}}(l, i)$ denote the sum of the throughputs of the users of link l that are saturated before the ith iteration. Then the *saturation allocation of* l at i, denoted $\gamma(l, i)$, is

$$\gamma(l, i) = x\left(\frac{c(l) - \gamma_{\text{sat}}(l, i)}{1 + jx}\right). \tag{4}$$

Intuitively, if each unsaturated user of link l chooses the saturation allocation as its throughput, and each saturated user leaves its throughput unchanged, then all unsaturated users become saturated. This follows from the fact that $r(l)$ in that case would be $(c(l) - \gamma_{sat}(l, i))/1 + jx$.

The following is the algorithm for the ith iteration. Initially, all throughputs are 0 and each link knows how many users have paths which use it.

Saturation Algorithm (ith Iteration)

1) Each link l calculates $\gamma(l, i)$.

2) Each link sends the value $\gamma(l, i)$ to all users of l.

3) Each user sets its new throughput γ to the smallest value of $\gamma(l, i)$ among links l that it uses.

4) Each link l determines which of its users are now saturated at l and informs each such user.

5) Each user that is saturated at *any* link informs *all* of its links that it is saturated.

There are basically two computations done at each iteration. After receiving $\gamma(l, i)$ from each link l on its path, a user readjusts its throughput by taking the minimum allocation [step 3)]. Also, each link must calculate $\gamma(l, i)$. The information needed for this calculation is the number of saturated users [obtained in step 5)] and $\gamma_{\text{sat}}(l, i)$ (obtained in some way by measuring each saturated user's throughput).

One method whereby a link can determine $\gamma_{\text{sat}}(l, i)$ without explicitly finding out which user sent each message is briefly described. Let each saturated user set a bit in the message header to 1 and each unsaturated user to 0. Then $\gamma_{\text{sat}}(l, i)$ is just the average rate of messages arriving with header bit equal to 1. Further elaboration on implementation is omitted.

The key properties of the algorithm (proved in Section VII) follow.

- Any user that is saturated after iteration i, remains saturated after iteration $i + 1$.
- If not all users are saturated at the beginning of an iteration, then at least one becomes saturated at the iteration.

From the above two facts it is immediate that if there are m users, they are all saturated after no more m iterations.

VI. AN EXAMPLE

Consider the network of Fig. 1. The following is a trace of the iterations of the algorithm with $x = 1$. The labels of the links are the capacities.

	Iteration 1	Iteration 2	Iteration 3
γ_1	1/2 (from link D)	1/2	1/2
γ_2	1 1/2 (F)	7/4 (E)	7/4
γ_3	1 1/2 (F)	11/6 (F)	15/8 (F)
γ_4	10 (A)	10	10
γ_5	3 1/3 (C)	19/4 (C)	19/4

User 1 is saturated at link D, 2 at E, 3 at F, 4 at A, and 5 at C.

VII. PROOF OF CORRECTNESS

The main result of this section is the following.

Theorem 1: Fix a network with m paths. Define

$$f(\gamma) = \sum_{i=1}^{m} \left| \gamma_i - \min_{l: l \in l(i)} \frac{x}{d_l(\gamma)} \right|. \tag{5}$$

If the saturation algorithm is executed, then after at most m iterations, the resulting value of γ, satisfies $f(\gamma) = 0$. Furthermore, γ is unchanged by subsequent iterations of the algorithm.

Proof: As mentioned in Section IV, this is proved by showing that saturated users stay saturated—and each iteration produces at least one saturated user. (Recall that $f(\gamma) = 0$ if all users are saturated at γ.) The main technical result

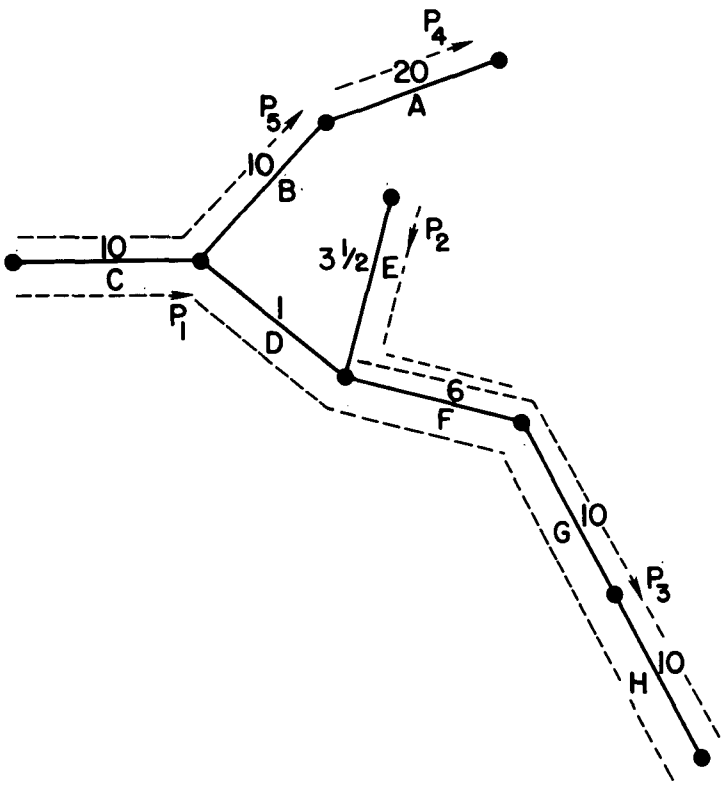


Fig. 1. Example network for execution of algorithm.

needed to prove Theorem 1 may be stated informally as "$\gamma(l, i)$ is a nondecreasing function of i." This fact, and the fact that saturated users stay saturated, are proved inductively in the following lemma.

Lemma 1:

1) For all $l \in L$, all $I \in \mathbb{Z}+$, $\gamma(l, i+1) \geq \gamma(l, i)$.

2) If any user becomes saturated at link l during the ith iteration, then all users of l that were not saturated before the ith iteration become saturated at l during the ith iteration.

3) If a user is saturated after the ith iteration with throughput γ, he remains saturated after the $i+1$st iteration with throughput γ.

The proof of Lemma 1 is given in the Appendix. To complete the proof of Theorem 1 we prove the following.

Lemma 2: At each iteration which starts with some unsaturated users, at least one user becomes saturated.

Proof: For each link at which not all users are saturated at a given iteration, consider the saturation allocation of the link. Some link must have minimal allocation among all such links. All unsaturated users of that link choose that allocation. Since all saturated users of the link do not change their throughputs [3) of Lemma 1], all of the unsaturated users of that link become saturated. ∎

Theorem 1 follows directly from Lemma 2 and 3) of Lemma 1. At each iteration at least one user becomes saturated—and saturated users stay saturated. ∎

Corollary–(Existence): Given any network and set of users of the network, there is a throughput assignment γ, which saturates all of the users.

Note that the saturation algorithm determines the optimal throughputs exactly. In contrast, even when the adaptive algorithm of Section IV converges to an optimal solution, it does not converge exactly. Rather, the sequence of throughputs achieved by the users converge (in a Cauchy sense) to the optimal throughputs.

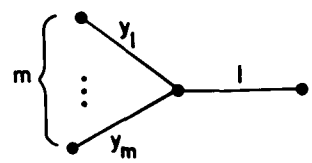


Fig. 2. Worst case network (in terms of number of steps).

The fact that linear time is actually required by our algorithm in the worst case is proved by the example of Fig. 2. Basically, the y_i may be chosen so that each user converges at a different step. See [17] for details.

VIII. UNIQUENESS

In this section it is shown that for any network and any set of users there is a unique way to saturate all users. This is a "well-defined" result for the saturation measure: two different throughput assignments cannot both be optimal for the same network configuration. We first separate out a simple lemma which we refer to later.

Lemma 3: Assume user i is saturated at link l at an optimum solution γ, with throughput γ_i, and user j uses link l and has throughput γ_j. Then $\gamma_i \geq \gamma_j$.

Proof: Since user i is saturated, $\gamma_i = x(r(l))$. Since user j is not overloaded, $\gamma_j \leq x(r(l)) = \gamma_i$. ∎

Theorem 2: The value γ obtained from the saturation algorithm *uniquely* minimizes the objective function f.

Proof: We prove by induction on the iteration number that all users saturated at step i must obtain the same throughput assignment in any optimal solution. The basis step is similar to the inductive step and is left to the reader.

Consider all users saturated at the ith step. By Lemma 1, part 2), a user may only be saturated if it takes the saturation allocation at some link l, and all other not previously saturated users also take their saturation allocations at l (and get saturated). Thus, we may study *all* users that are saturated at the ith step by looking at *all* links at which *all* nonsaturated users take the saturation allocation.

Assume, contrary to the hypothesis, that it is possible for the users saturated at step i to get different assignments in some optimal assignment γ^*. Consider a link l, which is saturated at the ith iteration and has some of its saturated users with different assignments in γ^*. By induction, recall that all users that share l, and are saturated before the ith iteration must receive the same throughputs in any optimal solution.

We first claim that at least one user saturated at l at the ith iteration must obtain less than $\gamma(l, i)$ in γ^*. For if all of them receive $\gamma(l, i)$ or more, and the users saturated before iteration i receive the same amounts, then $x(c(l) - \gamma^*(l)) < \gamma(l, i)$. But then, all those users that receive $\gamma(l, i)$ or more are overloaded at l in γ^*, and thus γ^* is not optimal.

Thus, one may consider a user which obtains throughput γ^* in γ^* where $\gamma^* < \gamma(l, i)$. Assume that the user is satur-

ated in γ^* at link l'. Note that $\gamma(l', i) > \gamma^*$ since $\gamma(l', i) \geq \gamma = \gamma(l, i) > \gamma$. Consider the sharers of l'. Those saturated before iteration i may not change their throughput in γ^* by induction. The r other users must have throughputs in γ^* of at most γ^*, each by Lemma 3. Thus, $x(c(l') - \gamma^*(l')) \geq x(c(l') - \gamma_{sat}(l', i) - r\gamma^*)$. But $< \gamma(l', i)$ implies that $\gamma^* < x(c(l') - \gamma_{sat}(l', i))/(1 + rx)$. Thus,

$$x(c(l') - \gamma^*(l')) > x\,(c\,(l') - \gamma_{sat}(l', i) - \left(\frac{rx}{1+rx}\right)(c(l') - \gamma_{sat}(l', i)))$$

$$= \frac{x}{1+rx}\,(c(l') - \gamma_{sat}(l', i)) > \gamma^*. \tag{6}$$

This contradicts the fact that the user is saturated at l' in γ^*.

IX. FAIRNESS

One aspect of a flow control optimality criterion which is difficult to evaluate is the elusive notion of fairness. One version of fairness is to insist that all users obtain equal throughputs. In a network with different users, using links of different capacities, it is unlikely that such a policy would be desirable.

Recall that flow control is instituted not only to protect a user against high delay due to traffic, but also to equitably divide network resources among competing users. The notion of fairness provided by saturation relates to the equitable division of resources. Briefly, saturation is "fair" because

- each user's throughput is at least as large as all other users that share its bottleneck link (Lemma 3)
- the only factor that prevents a user from obtaining higher throughput is the bottleneck link (which essentially divides resources equally).

X. GENERALIZATIONS

The fact that our algorithm saturates all users is interesting in a network with a homogenous user set, but suffers in that it provides too restrictive a notion of fairness. The property that "all users are treated equally" may not be desirable in practical networks. One user may be more important and thus deserving of a higher message rate. Alternatively, a user that interferes with *many* other users would probably deserve special treatment.

This is only one deficiency that results from the definition of saturation. A different problem arises if many (n) users share a single link. If the link is the bottleneck link for each, then (at $x = 1$) they each choose a message rate of $c(l)/(n + 1)$. As $n \to \infty$, the total rate approaches $c(l)$; thus, there is excessive throughput and disastrous delay. (This particular problem is dealt with both here and in Section XII.)

A final problem with the definition of saturation is that it may not be desirable to have a network-wide value of x as defined. Recall that one reason to choose $\gamma = x \min_l r(l)$ was to protect the network against transients in a user's message rate which were as large as a factor of $1/x$. Clearly, the variability in rates of different users is different. A user that has large variability would need a larger relative amount of residual capacity on its links.

This section solves the above problems by reformulating the definition of saturation. With user p, one associates a number x_p, the *throughput priority of user p*. User p's throughput priority expresses the desired message rate of user p as compared to the rates of interfering users. In particular, user p is saturated at l if $\gamma_p = x_p\, r(l)$. If users p and q are both saturated at l, the ratio of their throughputs is x_p/x_q. This generalization clearly treats users differently. Optimum performance is again equated with rate assignments that saturate all users.

In practice, some higher level protocol would decide what the relative values of x_p should be. If x_p were chosen as a function of the number of interfering users, some network manager could prevent the excessive use of an n user bottleneck. Similarly, a network manager could decide how to appropriately allocate relative priorities to competing users. In some network environments, each user might make a local decision choosing x_p based on the expected variability of its message rate to protect the network. A network manager is not needed if some convention is adopted by network users for determination of their throughput priorities.

We proceed to explain how the variable throughput priority case may be effectively reduced to the equal throughput priority case. In particular, the following questions are addressed:

- Is there a static algorithm to saturate all users?
- Is there a unique way to saturate all users?
- Is there an appropriate adaptive, distributed algorithm such as the one described in Section IV?
- What delay/throughput tradeoff is implied by the new definition of saturation?
- What fairness properties are implied?

First, consider the case that x_p is an integer for all p. We assume that each link knows the value of x_p for each user of the link. In this case, the variable x_p case is reduced to the $x = 1$ case as follows. A user with priority x_p is treated as x_p users each with $x = 1$ and identical paths. Initially, if there are j users of l with priorities $x_1, \cdots, x_j$, then

$$\gamma(l, 1) = \frac{c(l)}{1 + S} \tag{7}$$

where $S = \Sigma_{i=1}^{j} x_i$. If $\gamma(l, 1)$ is the minimal allocation for user k (with priority x_k), then user k chooses $\gamma = x_k\, \gamma(l, 1)$. In subsequent steps, γ_{sat} is measured as before, and

$$\gamma(l, i) = \frac{c(l) - \gamma_{sat}(l, i)}{1 + S(l, i)} \tag{8}$$

where $S(l, i)$ is the sum of the x_p's for users of link l that are not saturated before iteration i.

It can be shown that with this modified algorithm, the value of $\gamma(l, i)$ for every l and every i is identical here to the case where each user with priority x_p were replaced by x_p users with priority 1. Also, the message rate γ of a user with

priority x_p after iteration i equals the sums of the rates of the x_p users with $x = 1$. These facts are proved trivially by induction on i. From this it follows that there is a static algorithm to saturate all users, and that saturation is unique.

Actually, using (7) and (8) uniquely saturates all users even if x_p is not an integer. The proof of this follows in a manner similar to the proof of Section VII.

Continuing with the aforementioned questions, the appropriate adaptive algorithm remains roughly the same as in Section IV; each user saturates itself based on current conditions (perhaps changing message rate slowly for stability reasons). The delay-throughput tradeoff defined for user p is

$$\gamma_p = \min_{l:l\in l(p)} \frac{x_p}{d_l(\gamma)}. \tag{9}$$

The relevant fairness statements are as follows.

- Each user's throughput is only constrained by its bottleneck link.
- At its bottleneck link a user gets at least "its share of capacity" based on its throughput priority. That is, the rate γ_p of user p satisfies $\gamma_p \geqslant (x_p/x_q)\,\gamma_q$ if q shares p's bottleneck link.

XI. LOW THROUGHPUT USERS

The saturation algorithm provides each user with an "optimum" throughput, but requires one special assumption to do so. It is assumed that each user has a throughput equal to that assigned in the algorithm. In practice, however, a user may not have enough data to send at the high rate. In this section we briefly discuss the required modifications to handle this case.

Assume that γ is the maximum possible rate for a user based on incoming data rate considerations. Then the user "pretends" that on its path is a "virtual link" of capacity $(\gamma)(1 + x)/x$, which is shared with no one. If all other links have saturation allocation larger than γ, then the rate chosen on the basis of the virtual link is γ. For example, if $x = 1$, the virtual link has capacity 2γ and the user is saturated if its rate is γ. Thus, by slightly modifying the network, the inherent throughput constraints of each user are taken into account, without changing the algorithms and their properties.

XII. RESTRICTING THE PERCENTAGE UTILIZATION OF A LINK

Assume that it was desired that no link exceed a fraction y of its capacity. This might be used to prevent $\gamma(l) \to c(l)$ as $n \to \infty$ in the case of n users sharing a bottleneck link. Section XI prevents $\gamma(l) \to c(l)$ by suggesting that the values x_p should be chosen as a function of n. In this section a more direct approach is used. This approach leads to a derivation of the "optimum Cyclades performance" as a limiting case of saturation.

Define the *effective capacity of* l, $e(l) = yc(l)$. This is the largest amount of capacity of l that should be used. If $e(l)$ is used instead of $c(l)$ in the algorithms to saturate all users, then the capacity of any link utilized is restricted to be at most $e(l)$.

This is not quite the Cyclades notion of optimality—they require that $e(l)$ not be exceeded, but place no other restrictions on the message rates (such as $\gamma \leqslant r(l)$). To effectively remove the restriction $\gamma \leqslant r(l)$, let $x \to \infty$; $\gamma \leqslant xr(l)$ is then trivially accomplished.

To review, a utilization of y at bottleneck links is accomplished by using $e(l)$ instead of $c(l)$, letting $x \to \infty$, and saturating all users. This accomplishes the desired utilization of bottlenecks, and also provides fairness not usually provided by just restricting link utilization. In this case, letting $x \to \infty$ does not strongly degrade delay at the cost of throughput, since the rates are all chosen based on $e(l)$, not $c(l)$.

XIII. CONCLUSIONS

We have presented a "fair" motivatable network performance criterion. Two algorithms have been presented to optimize performance, one of which is guaranteed to find the unique optimal throughput assignments in a static environment.

APPENDIX

PROOF OF LEMMA 1 (BY INDUCTION ON i)

$i = 1$:

1) $\gamma(l, 1) = x \cdot c(l)/(1 + jx)$ where j is the number of users that share l. $\gamma(l, 2) = x(c(l) - \gamma_{\text{sat}}(l, 2))/(1 + rx)$ where r is the number of users of l not saturated at the first iteration. By the way that throughputs are assigned, $\gamma_{\text{sat}}(l, 2) \leqslant (j - r)\,\gamma(l, 1) = (j - r)(x \cdot c(l))/(1 + jx)$. Thus,

$$\gamma(l, 2) = \frac{xc(l) - xy_{\text{sat}}(l, 2)}{1 + rx} \geqslant \frac{xc(l) - \dfrac{x(j-r)(x \cdot c(l))}{1 + jx}}{1 + rx}$$

$$= \frac{x(c(l))\left(1 - \dfrac{x(j-r)}{1 + jx}\right)}{1 + rx}$$

$$= \frac{xc(l)\left(\dfrac{1 + jx - jx + rx}{1 + rx}\right)}{1 + rx}$$

$$= \frac{xc(l)}{1 + jx}$$

$$= \gamma(l, 1). \tag{A1}$$

2) Recall that the saturation allocation is designed to guarantee saturation if all unsaturated users of a link choose the saturation allocation and all saturated users keep the same throughput. Before the first iteration, there are no saturated users, and each user chooses at most the saturation allocation. From this, 2) follows immediately.

3) Similar to the inductive step (below).

Inductive Step: Assume 1), 2), and 3) for $k < i$ and prove 1) and 2) for $k = i$. Then, using 1) and 2) for $k = i$ and 3) for $k < i$, prove 3) for $k = i$ as follows.

1) $\gamma(l, i+1) = x(c(l) - \gamma_{sat}(l, i+1))/(1+rx)$, $\gamma(l, i) = x(c(l) - \gamma_{sat}(l, i))/(1+sx)$ where there are r nonsaturated users of l before the $i+1$st iteration and s before the ith. By induction on 3), any user saturated before the ith iteration remains saturated before the $i+1$st (i.e., after the ith) with the same throughput. Thus, $\gamma_{sat}(l, i+1) = \gamma_{sat}(l, i) + \gamma_{new}$ where γ_{new} is the sum of the throughputs of the $s - r$ users that become saturated at the ith iteration. Note that $\gamma_{new} \leq (s-r)\,\gamma(l, i)$ since each newly saturated user has message rate at most $\gamma(l, i)$. Thus,

$$\begin{aligned}\gamma(l, i+1) &= \frac{x(c(l) - \gamma_{sat}(l, i+1))}{1+rx} \\ &= \frac{x(c(l) - \gamma_{sat}(l, i))}{1+rx} - \frac{x\gamma_{new}}{1+rx} \\ &\geq \left(\frac{sx+1}{rx+1}\right)\gamma(l, i) - \frac{x(s-r)}{1+rx}\gamma(l, i) \\ &= \gamma(l, i)\frac{sx+1-sx+rx}{1+rx} \\ &= \gamma(l, i) \qquad \text{(A2)}\end{aligned}$$

2) By induction on 3), all users saturated before the ith iteration choose the same throughput at the ith iteration. Since each unsaturated user chooses, at most, the saturation allocation at l, by the definition of $\gamma(l, i)$, a user becomes saturated at l at iteration i only if *all* other unsaturated users choose $\gamma(l, i)$ and become saturated.

3) Fix a user that is saturated after the ith iteration with throughput γ. We must show that at the $i+1$st iteration, it chooses the same throughput and remains saturated. Consider a link l at which the user is saturated after the ith iteration. Using 2) for the iteration number k at which the user was first saturated at l, ($k \leq i$), all users that share l are either saturated before the kth iteration or become saturated at the kth iteration. By induction on 3), it follows that all are saturated after the kth iteration. Also, the ones that were previously saturated use the same throughput as before the kth iteration. This continues through iteration i. Since the user is saturated at l, its throughput γ satisfies $\gamma = x(r(l))$. Also, since all users of l are saturated, $\gamma_{sat}(l, i+1) = \gamma(l)$ and $\gamma(l, i+1) = x(c(l) - \gamma(l)) = \gamma$. Thus, due to the saturation allocation at l, the user chooses a throughput of at most γ at iteration $i+1$. Since for every link l' in the user's path $\gamma(l', i+1) \geq \gamma(l', i)$ [by (1)], the user chooses exactly γ.

The above argument may be repeated for each user saturated after the ith iteration. Returning to the user fixed above, it is apparent that the user is saturated at l at the $i+1$st iteration, since all users that share l do not change their throughputs. Thus, $\gamma(l)$ is unchanged and $\gamma = x(r(l))$ still holds. To prove that the user is still saturated after the $i+1$st iteration, it suffices to show that it is not overloaded on any other link on its path.

To prove that the user is not overloaded at a link l', it suffices to show $\gamma \leq x(c(l') - \gamma(l'))$ where $\gamma(l')$ is the sum of throughputs of users of l' after the $i+1$st iteration. Consider the iteration (iteration k) at which the user became saturated (with rate γ). If l' is on its path, $\gamma(l', k) \geq \gamma$ by the way γ is chosen. By induction on (1), $\gamma(l', i+1) \geq \gamma$. Recall that $\gamma(l', i+1) = (x(c(l') - \gamma_{sat}(l', i+1)))/(1+jx)$ (if j users are not saturated before the $i+1$st iteration). Note, the value of $\gamma(l')$ after iteration $i+1$ is given by

$$\gamma(l') \leq \gamma_{sat}(l', i+1) + j\gamma(l', i+1).$$

Thus,

$$\begin{aligned}x(c(l') - \gamma(l')) &\geq x(c(l') - \gamma_{sat}(l', i+1) - j\gamma(l', i+1)) \\ &= x(c(l') - \gamma_{sat}(l', i+1) - \frac{jx}{1+jx}(c(l') \qquad \text{(A3)} \\ &\quad - \gamma_{sat}(l', i+1))) \\ &= x(c(l') - \gamma_{sat}(l', i+1))\left(\frac{1}{1+jx}\right) \\ &= \gamma(l', i+1) \geq \gamma.\end{aligned}$$

ACKNOWLEDGMENT

The author acknowledges helpful conversations with K. Bharath-Kumar, F. H. Moss, and M. Schwartz.

REFERENCES

[1] J. M. McQuillan, "Adaptive routing algorithms for distributed computer networks," Bolt Beranek and Newman Rep. 2831, NTISAD 781467, May 1974.

[2] L. Tymes, "TYMNET—A terminal-oriented communication network," in *AFIPS Conf. Proc., Spring Joint Comput. Conf.*, vol. 38, 1971, pp. 211–216.

[3] M. Schwartz, *Computer Communication Network Design and Analysis*. Englewood Cliffs, NJ: Prentice-Hall.

[4] J. P. Gray and T. B. McNeill, "SNA multiple-system networking," *IBM Syst. J.*, vol. 18, no. 2, 1979.

[5] A. Danet, R. Despres, A. LaRest, G. Pichon, and S. Ritzentheler, "The French public packet switching service: The transpac network," in *Proc. 3rd Int. Conf. Comput. Commun.*, Toronto, Ont., Canada, Aug. 1976, pp. 251–260.

[6] M. Gerla and L. Kleinrock, "Flow control: A comparative survey," *IEEE Trans. Commun.*, vol. COM-28, pp. 553–575, Apr. 1980.

[7] V. Ahuja, "Routing and flow control in systems network architecture," *IBM Syst. J.*, vol. 18, no. 2, 1979.

[8] D. W. Davies, "The control of congestion in packet switching networks," *IEEE Trans. Commun.*, vol. COM-20, June 1972.

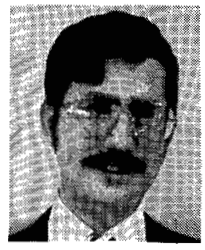
[9] E. Raubold and J. Haenle, "A method of deadlock-free resource allocation and flow control in packet networks," in *Proc. 3rd Int. Conf. Comput. Commun.*, Toronto, Ont., Canada, Aug. 1976.

[10] J. C. Majithia *et al.*, "Experiments in congestion control techniques," in *Proc. Int. Symp. Flow Contr. Comput. Networks*, Versailles, France, Feb. 1979.

[11] *Proc. Int. Symp. Flow Contr. Comput. Networks*, Versailles, France, Feb. 1979.

[12] K. Bharath-Kumar, "Optimum end-to-end flow control in net-

works," in *Proc. Int. Conf. Commun.*, Seattle, WA, June 1980.
[13] J. M. Jaffe, "Flow control power is non-decentralizable," IBM Res. Rep. RC8343, July 1980; also to be published *IEEE Trans. Commun.*, 1981.
[14] K. Bharath-Kumar and J. M. Jaffe, "A new approach to performance oriented flow control," IBM Res. Rep. RC8307, May 1980; also, *IEEE Trans. Commun.*, vol. COM-29, pp. 427–435, Apr. 1981.
[15] L. Kleinrock, *Queueing Systems*, vol. 2. New York: Wiley, 1976.
[16] D. C. Little, "A proof of the queueing formula: $L = \lambda W$," *Oper. Res.*, vol. 9, pp. 383–387, 1961.
[17] J. M. Jaffe, "A decentralized, 'optimal,' multiple user flow control algorithm," in *Proc. 5th Int. Conf. Comput. Commun.*, Oct. 1980.



Jeffrey M. Jaffe (M'80) received the B.S. degree in mathematics and the M.S. and Ph.D. degrees in computer science from the Massachusetts Institute of Technology, Cambridge, in 1976, 1977, and 1979, respectively.

He is currently employed by IBM Research, Yorktown Heights, NY, where he is engaged in research on network algorithms and combinatorial optimization.

Dr. Jaffe is a member of ACM and Phi Beta Kappa, and was a National Science Foundation Fellow while a graduate student.

On the Dynamic Control of the Urn Scheme for Multiple Access Broadcast Communication Systems

KUMUD K. MITTAL, STUDENT MEMBER, IEEE, AND ANASTASIOS N. VENETSANOPOULOS, SENIOR MEMBER, IEEE

Abstract—**The Urn scheme is known to perform better than optimal ALOHA and TDMA for all ranges of traffic rates. In this paper we discuss the dynamic behavior of the Urn scheme to show that it possesses bistable behavior in a manner similar to ALOHA schemes and that dynamic control procedures can be applied to improve the system performance effectively. In particular, an input control procedure (ICP) is presented that gives a delay–throughput characteristic very close to optimal (perfect scheduling) for a wide range of throughput rates. The improvement is obtained at no extra cost in terms of information acquisition and the complexity introduced is minimal. An analytical method is described to calculate the expected delay, throughput, and the probability of packet rejection. Numerical results are shown for various values of user population and compared with corresponding results for other schemes.**

I. INTRODUCTION

PACKET broadcasting systems combine the advantages of packet communication with those of broadcast communication systems. Unlike circuit switching, packet communication does not dedicate circuits or tie them up to establish connections, and hence provides a powerful means of sharing the communication channel among large numbers of users. Among the advantages of a broadcast communication system are multidestination or conferencing capability, absence of topological and routing problems, system modularity, and overall system simplicity.

In packet broadcasting systems, the problem of designing an efficient multiple access scheme is of prime importance. Various schemes have been devised and studied. These can be categorized mainly into the fixed assignment schemes, the polling schemes, and the random access schemes. The random access schemes are particularly suitable for systems in which the number of users is large and the users are characterized by a high ratio of peak to average data rates [1]. However, these schemes generally suffer from system instability and have low obtainable channel capacities (e.g., only 37 percent for slotted ALOHA; see [2] for a brief summary of random access schemes). Recently an adaptive asymmetric scheme, called the Urn scheme [3], has been proposed that is not unstable in the strict sense (elaborated later in the paper) and can achieve a channel utilization of nearly 100 percent.

It is well known that the asymmetric variations of slotted ALOHA perform better than the symmetric ones. For example, Abramson has shown [4] that the channel capacity of an ALOHA system is higher if traffic rates at the users are unequal ("excess capacity"). Metzner [5] has considered the use of unequal transmission power levels for improving channel utilization ("capture effect"). In the Urn scheme, the asymmetry is incorporated in the transmission probabilities. Some users will try to access the system with probability 1 and others with probability 0. However, there is a need for coordination in decision making among the users as to which particular user employs which probability. This coordination is achieved by using the same seed for random number generators at each user site and following a preprogrammed priority mechanism. The information used for decision making is the same as in the case of optimal ALOHA, namely, the number of busy terminals. The scheme behaves like optimal ALOHA for low traffic rates and adapts smoothly to TDMA for heavy traffic, performing better than both throughout the range of traffic intensities.

Paper approved by the Editor for Computer Communication of the IEEE Communications Society for publication after presentation at the National Telecommunications Conference, Washington, DC, November 1979. Manuscript received November 11, 1979; revised April 18, 1980. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada under Grant A-7397.

The authors are with the Department of Electrical Engineering, University of Toronto, Toronto, Ont., Canada.

0090-6778/81/0700-0962$00.75