

Flow Theory

Jorge A. Cobb, *Member, IEEE*, and Mohamed G. Gouda

Abstract— We develop a simple theory of flows to study the flow of data in real-time computing networks. Flow Theory is based on discrete and nondeterministic mathematics, rather than the customary continuous or probabilistic mathematics. The theory features two types of flows: smooth and uniform, and eight types of flow operators. We prove that, if the input flow to any of these operators is smooth or uniform, then both the internal buffer and delay of that operator are bounded. Linear networks of flow operators are introduced, and their internal buffers and delays are derived from the internal buffers and delays of their constituent operators. We extend Flow Theory so that it can be used in analyzing cyclic networks and networks of multiflows. Since many rate-reservation protocols can be represented as linear networks of flow operators, we use Flow Theory to prove that a number of these protocols (Stop-and-Go, Hierarchical Round-Robin, Weighted Fair Queueing, Self-Clocking Fair Queueing, and Virtual Clock) require bounded buffering and introduce bounded delay.

Index Terms— Guaranteed performance protocols, protocol verification, rate reservation protocols.

I. INTRODUCTION

THE main objective of this paper is to study real-time network protocols. These protocols require upper bounds on buffer requirements and packet delays. In order to study such protocols, we introduce a simple theory of flows.

Flow Theory is based on discrete mathematics, in which a flow of data is represented as an infinite sequence of real numbers. This is in contrast to fluid models of data flow, which needlessly complicate the analysis. We define two properties of flows—smoothness and uniformity—and introduce several nondeterministic operators that manipulate flows. We show that each of these operators preserves the flow properties of smoothness or uniformity, while maintaining bounded delay and buffer requirements. We discuss how to combine flow operators to form networks of arbitrary topology.

We also discuss how to apply Flow Theory to the analysis of rate-reservation protocols. (For a survey of rate-reservation protocols, see [20].) A rate-reservation protocol is a connection-oriented protocol that guarantees that the data packets of each established connection are forwarded through the network at a rate no less than the rate reserved for that connection. The rate reserved for each connection is negotiated during the establishment of the connection. A request to establish a connection is rejected if there is a computer in the

path of the connection that does not have enough resources to serve the connection at its desired rate.

Rate-reservation protocols can be represented as linear networks of flow operators [1], and thus Flow Theory can be used to verify their real-time properties. In particular, we show that these protocols have bounded delay and bounded buffer requirements. These properties make rate-reservation protocols appealing for the support of multimedia applications [18].

The paper is organized as follows. In Section II, we introduce the concept of a flow and define the smoothness and uniformity of a flow. Flow operators, their properties, and some operator examples are presented in Section III. Linear networks of flow operators are presented in Section IV. In Section V, we prove that several rate-reservation protocols require bounded buffering and introduce bounded delays. Then, in Section VI, we show how to extend this theory to analyze general networks. In Section VII, we present the concept of multiflows and discuss how to analyze networks with multiflows. Concluding remarks are given in Section VIII.

Due to space restrictions, only the proofs of selected theorems are found in the Appendix. The remaining proofs may be found in [5].

II. FLOWS

A flow r is an infinite sequence r_0, r_1, r_2, \dots , of nonnegative real numbers.

Informally, each r_i represents the number of bits or packets that travel in flow r at instant i .

Let m be a positive integer, and R be a positive real number. A flow r is (m, R) -smooth iff, for every $j = 0, 1, 2, \dots$,

$$\sum_{i=j \cdot m}^{(j+1) \cdot m - 1} r_i \leq m \cdot R.$$

In this definition, if an (m, R) -smooth flow is partitioned into adjacent subsequences of m elements each, then the sum of the elements of each subsequence is at most $m \cdot R$.

The smoothness property of a flow over a continuous timeline was introduced in [11]. The above is an adaptation of this property to our discrete model.

Let m be a positive integer and R be a positive real number. A flow r is (m, R) -uniform iff, for every $j = 0, 1, 2, \dots$,

$$\sum_{i=j}^{j+m-1} r_i \leq m \cdot R.$$

In these definitions of smooth and uniform flows, R can be regarded as an upper bound on the rate of the flow, and m can be regarded as the size of the interval over which the rate is computed. The smoothness (or uniformity) of the flow is measured by m : the smaller the m , the smoother (or more uniform) the flow.

Manuscript received August 29, 1994; revised June 19, 1995, August 22, 1996; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor K. K. Sabnani.

J. A. Cobb is with the Department of Computer Science, University of Houston, Houston, TX 77204-3475 USA (e-mail: cobb@cs.uh.edu).

M. G. Gouda is with the Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712-1188 USA (e-mail: gouda@cs.utexas.edu).

Publisher Item Identifier S 1063-6692(97)06689-2.

We have chosen smoothness as a flow property for two reasons. First, this property intuitively captures the concepts of the rate and burstiness of a flow. Second, it is easy for a data source to ensure that its data flow is (m, R) -smooth using one timer that expires every m seconds. Every time the timer expires, the sender can send up to $m \cdot R$ units of data.

Uniformity is a stronger property than smoothness and it is harder to enforce. However, we include it in our theory because it occurs naturally in some intermediate flows of flow operator networks. See, for example, Sections V-A and V-B, and see also [5].

The relationship between smoothness and uniformity is given by the following theorem.

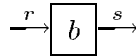
Theorem 1: Let r be an (m, R) -smooth flow, and r' be an (m, R) -uniform flow.

- 1) For any $n, n \geq 1$,
 - a) r is $[n, (\lceil n/m \rceil + 1) \cdot (m/n) \cdot R]$ -uniform.
 - b) r' is $[n, \lceil n/m \rceil \cdot (m/n) \cdot R]$ -uniform.
- 2) For any n multiple of m ,
 - a) r is (n, R) -smooth.
 - b) r' is (n, R) -uniform.
- 3) For any S larger than or equal to R ,
 - a) r is (m, S) -smooth.
 - b) r' is (m, S) -uniform.

Corollary 1: r is $(m, 2 \cdot R)$ -uniform and r' is (m, R) -smooth. ■

III. FLOW OPERATORS

A *flow operator* has an input flow r and an output flow s . At the i th instant, the operator inputs r_i , outputs s_i , and stores the remainder in an internal buffer. The content of the internal buffer at the i th instant is denoted b_i . The infinite sequence b_0, b_1, b_2, \dots , is called the *buffer flow* of the flow operator.



Formally, a flow operator with an input flow r , an output flow s , and a buffer flow b is defined, for every $i = 0, 1, 2, \dots$, as follows:

$$s_i \text{ is a value in the interval } F(r_0, r_1, \dots, r_i, b_{i-1}) \quad (1)$$

$$b_i = b_{i-1} + r_i - s_i \quad (2)$$

where $b_{-1} = 0$, and F is a function, called the *operation* of the flow operator. Function F takes as arguments the sequence r_0, r_1, \dots, r_i and b_{i-1} , and it returns an interval of real numbers. Below, we introduce a number of flow operators and define the operation of each of them.

The operation of a flow operator is required to ensure the following *flow conservation* property:

(For every input flow r , output flow s , and buffer flow b , satisfying (1) and (2),

$$\text{(For every } i, \quad s_i \leq r_i + b_{i-1})$$

).

This property states that a flow operator cannot output at any instant more than the sum of its input at that instant and the

content of its buffer at the last instant. Note that this property guarantees that for every $i = 0, 1, 2, \dots$, $b_i \geq 0$.

The *buffer capacity* B of a flow operator with respect to input flow r is the smallest nonnegative real number that satisfies the following condition:

(For every output flow s and buffer flow b ,
satisfying (1) and (2),

$$\text{(For every } i, \quad b_i \leq B)$$

).

The *delay* D of a flow operator with respect to input flow r is the smallest nonnegative integer that satisfies the following condition:

(For every output flow s and buffer flow b ,
satisfying (1) and (2),

$$\text{(For every } i, \quad b_i \leq s_{i+1} + s_{i+2} + \dots + s_{i+D})$$

).

The relationship between the buffer capacity and the delay of any flow operator is described in the next theorem.

Theorem 2: Consider any flow operator whose input is (m, R) -uniform.

Let B denote the buffer capacity of this operator,
 D denote the delay of this operator.

$$\text{Then } B \leq \lceil D/m \rceil \cdot m \cdot R.$$

Note that Theorem 2 remains valid if the input to the arbitrary flow operator is $(m, R/2)$ -smooth. Also note that this theorem is related to the well-known Little's theorem in Queueing Theory [15].

Theorem 2 does not provide a lower bound on the buffer capacity since each of the elements of the input flow can be arbitrarily small (even zero), which yields an arbitrarily small buffer capacity. However, if we place a lower bound on the elements of a flow, a lower bound for the buffer capacity can be obtained. For example, if the input flow is greedy, i.e., every m consecutive elements sum to exactly $m \cdot R$, then the buffer capacity is at least $\lceil D/m \rceil \cdot m \cdot R$. Thus, for a greedy flow, the buffer capacity differs from $D \cdot R$ by no more than $m \cdot R$.

The notion of viewing network traffic in a nonprobabilistic way by requiring that a flow satisfy a burstiness constraint was introduced in [6] and [7]. Our work is different from the work in [6] and [7] in several ways. First, the model of a flow used in [6] and [7] was based on a continuous timeline, which differs from our discrete approach. Second, we use a different characterization of the burstiness of a flow, and apply different operations to flows. Third, the protocols studied in [6] and [7] were mainly first-come-first-served protocols, while our focus is on real-time protocols.

In the remainder of this section, we present three types of flow operators: limiters, compactors, and expanders. For each operator, we show that if the input flow is smooth, then the output flow is smooth or uniform, and both the buffer capacity and delay of the operator are bounded. Because each uniform flow is also smooth, the output of a flow operator can become the input of another operator in a linear network while maintaining bounded buffer capacities and delays in both operators. (Linear networks are discussed in the next section.)

A. Limiters

Let R be a positive real number. An R -limiter is a flow operator that performs the following steps at each instant: it inputs a flow element, outputs at most R , and buffers the remainder. Formally, the operation of an R -limiter is defined as follows:

$$s_i = \begin{cases} R & \text{if } b_{i-1} + r_i > R \\ b_{i-1} + r_i & \text{if } b_{i-1} + r_i \leq R \end{cases}$$

where r is the input flow, s is the output flow, and b is the buffer flow of the R -limiter.

Theorem 3: For an R -limiter, if the input flow is (m, R) -smooth, then

- 1) the output flow is $(1, R)$ -uniform,
- 2) the buffer capacity is at most $2 \cdot m \cdot R$, and
- 3) the delay is at most $2 \cdot m$. ■

By Theorem 1, a flow that is (m, R) -uniform is also (m, R) -smooth. Therefore, Theorem 3 still holds when the input flow of the R -limiter is (m, R) -uniform. In this case, however, the upper bounds can be lowered: the buffer capacity becomes at most $m \cdot R$, and the delay becomes at most m .

B. Compactors

Let m be a positive integer. An m -compactor is a flow operator that accumulates the input flow in its buffer during an interval of m instants, and then outputs the buffer contents in the first instant of the next interval. Formally, the operation of an m -compactor is defined as follows:

$$s_i = \begin{cases} 0 & \text{if } i \bmod m \neq 0 \\ b_{i-1} & \text{if } i \bmod m = 0 \end{cases}$$

where s is the output flow and b is the buffer flow of the m -compactor.

Note that at every instant i , where $i \bmod m \neq 0$, the input r_i is added to the buffer, i.e., $b_i = b_{i-1} + r_i$.

Also, at every instant i , where $i \bmod m = 0$,

$$s_i = \sum_{j=i-m}^{i-1} r_j \text{ and } b_i = r_i.$$

Theorem 4: For an m -compactor, if the input flow is (m, R) -smooth, then

- 1) the output flow is (m, R) -uniform,
- 2) the buffer capacity is at most $m \cdot R$, and
- 3) the delay is at most m . ■

C. Expanders

Let m be a positive integer. An m -expander is a flow operator that may output any value at each instant provided the flow conservation property is satisfied. In addition, to guarantee a bounded delay, the entire buffer content is transferred to the output flow every m instants. Formally, the operation of an m -expander is defined as follows:

$$s_i = \begin{cases} b_{i-1} + r_i & \text{if } i \bmod m = m - 1 \\ (b_{i-1} + r_i) \cdot X_i & \text{if } i \bmod m \neq m - 1 \end{cases}$$

where r is the input flow, s is the output flow, b is the buffer flow, and each X_i is a real number in the closed interval $[0, 1]$. ■

Theorem 5: For an m -expander, if the input flow is (m, R) -smooth, then

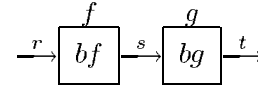
- 1) the output flow is (m, R) -smooth,
- 2) the buffer capacity is at most $m \cdot R$, and
- 3) the delay is at most m . ■

IV. LINEAR NETWORKS

A finite sequence of flow operators $\langle f_0, f_1, \dots, f_{n-1} \rangle$ is a *linear network* iff for each i , $0 \leq i < n-1$, the output flow of operator f_i is the input flow of operator f_{i+1} . The input flow of operator f_0 is the input flow of the network and the output flow of operator f_{n-1} is the output flow of the network.

In this section, we discuss the properties of linear networks. For simplicity, the discussion is limited to networks with two flow operators. However, the results (Theorem 6 below) extend in a straightforward manner to linear networks with any number of flow operators.

Consider a linear network $\langle f, g \rangle$ of two flow operators f and g . For operator f , the input flow is r , the output flow is s , and the buffer flow is bf . For operator g , the input flow is s , the output flow is t , and the buffer flow is bg . The input flow of the network is r and the output flow of the network is t .



The buffer flow c of a linear network $\langle f, g \rangle$ is an infinite sequence c_0, c_1, \dots , such that for every $i = 0, 1, 2, \dots$,

$$c_i = c_{i-1} + r_i - t_i$$

where $c_{-1} = 0$, r is the input flow of network $\langle f, g \rangle$, and t is the output flow of network $\langle f, g \rangle$.

This definition of the buffer flow of a network coincides with the definition of the buffer flow of a flow operator where the input flow is r and the output flow is t . Hence, we define the buffer capacity and delay of a linear network in the same manner as was done previously for flow operators.

Theorem 6: Let r be the input flow and s the intermediate flow of a linear network $\langle f, g \rangle$ with operators f and g .

- 1) If the buffer capacity of f with respect to r is at most Bf , the buffer capacity of g with respect to any intermediate flow s is at most Bg , and the buffer capacity of the network $\langle f, g \rangle$ with respect to r is C , then

$$C \leq Bf + Bg.$$

- 2) If the delay of f with respect to r is at most Df , the delay of g with respect to any intermediate flow s is at most Dg , and the delay of the network $\langle f, g \rangle$ with respect to r is D , then

$$D \leq Df + Dg.$$

■

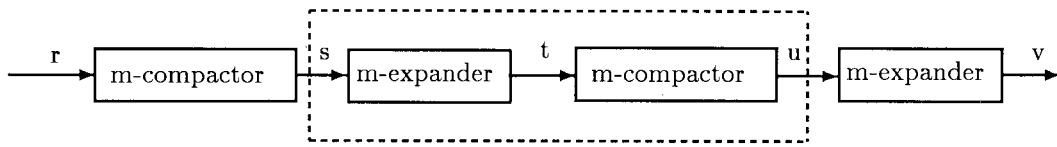


Fig. 1. Alternative view of a Stop-and-Go linear network.

Part 1 in this theorem implies that if each operator in a network has a bounded buffer, then the network has a bounded buffer. Part 2 implies that if each operator in a network has a bounded delay, then the network has a bounded delay. Theorem 6 is used in the next section to prove that a number of rate-reservation protocols, that were proposed earlier, indeed guarantee bounded buffer capacities and delays.

V. RATE-RESERVATION PROTOCOLS

A computer network can be represented by a finite undirected graph. Each node in the graph represents a computer and each edge represents a two-way link for transmitting messages between the two computers at both ends of the link.

A rate-reservation protocol for transmitting data packets in an (m, R) -smooth flow from a computer i to a computer j proceeds as follows. First, computer i determines a simple path in the network for transmitting the data packets from i to j . Second, computer i sends a reserve (i, j, m, R) message along this simple path. Third, when a computer k on the path from i to j receives the reserve (i, j, m, R) message, it checks whether it can support an (m, R) -smooth flow from i to j . If the answer is no, computer k returns a reject (i, j) message along the path toward i . If the answer is yes, computer k forwards the reserve (i, j, m, R) message to the next computer on the path from i to j . Fourth, when computer j receives the reserve (i, j, m, R) message, it returns an accept (i, j) message along the reverse path toward i . Fifth, when computer i receives the accept (i, j) message, it recognizes that a connection from i to j has been established and starts to transmit its data packets over the path from i to j , ensuring that the resulting flow is (m, R) -smooth.

All rate-reservation protocols follow the above procedure in establishing connections. They differ, however, in the rules for receiving, buffering, and later forwarding the data packets along the path from i to j , once a connection from i to j has been established.

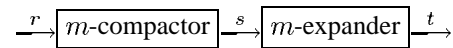
In this section, we discuss five rate-reservation protocols: Stop-and-Go, Hierarchical Round-Robin, Weighted Fair Queueing, Self-Clocking Fair Queueing, and Virtual Clock. For each protocol, we show how to represent each computer on the path from i to j as a linear network of flow operators. Therefore, the whole path from i to j is represented as a linear network of flow operators. We use this path representation to show that the buffer capacity and delay of the path are bounded for each of the five protocols.

A. Stop-and-Go

In this protocol [11], time is partitioned into consecutive periods of equal duration called frames. Each computer on the path from i to j buffers all the data packets that it receives

in one frame, then forward these packets in the next output frame. The protocol makes no guarantees on how the packets are to be arranged in the output frame. Thus, the packets in the output frame are not necessarily arranged as they were in the input frame.

Each computer on the path from i to j can be represented as a linear network of an m -compactor followed by an m -expander as follows:



where m is the number of (discrete) instants in a frame, r is the input flow of the network, s is the internal flow of the network, and t is the output flow of the network.

The m -compactor buffers the contents of each frame and forwards them at the beginning of the next frame. The m -expander is included to nondeterministically rearrange the contents of each output frame.

Assume that flow r is (m, R) -smooth. In this case, flow s is (m, R) -uniform (by Theorem 4) and so it is (m, R) -smooth (by Theorem 1). Therefore, flow t is (m, R) -smooth (by Theorem 5). From Theorems 4–6, we conclude that the buffer capacity of the network is at most $2 \cdot m \cdot R$ and its delay is at most $2 \cdot m$.

If a number n of these linear networks are connected in one large linear network (to represent a path of n computers from i to j), then the buffer capacity along the path is at most $2 \cdot m \cdot n \cdot R$, and the delay is at most $2 \cdot m \cdot n$. Next, we show that the upper bound on the delay can be reduced by a factor of two.

Instead of viewing a Stop-and-Go linear network as a series of $\langle m\text{-compactor}, m\text{-expander} \rangle$ pairs, one can also view it as a single m -compactor followed by a series of $\langle m\text{-expander}, m\text{-compactor} \rangle$ pairs ending with an m -expander as illustrated in Fig. 1. The reason behind this is to take advantage of the following theorem.

Theorem 7: Let f be an m -expander, and g and h be m -compactors. If the input flow of the linear network $\langle f, g \rangle$ is the same as the input flow of h , then the output flow of $\langle f, g \rangle$ is the same as the output flow of h . ■

From Theorems 4 and 7, the delay of an (m, R) -smooth flow through the $\langle m\text{-expander}, m\text{-compactor} \rangle$ pair is at most m , and the output flow of this pair is (m, R) -uniform and hence (m, R) -smooth.

Consider now a Stop-and-Go linear network of n m -compactors alternating with n m -expanders. By Theorems 4, 5, and 7, if the input flow to the network is (m, R) -smooth, then every flow in the network is (m, R) -smooth. By these same theorems, the delay at the first m -compactor is at most m , at the last m -expander is at most m , and at each $\langle m\text{-expander},$

m -compactor} pair is at most m . Hence, by Theorem 6, the total delay in the network is at most $(n + 1) \cdot m$.

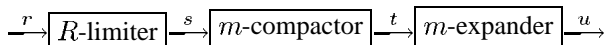
Although we can obtain a similarly reduced bound on the buffer capacity of the linear network, it is not of practical use, because an m -expander is implemented in a different computer than its succeeding m -compactor. Thus, the memory used by their buffers cannot be combined.

The upper bounds which we obtained on the buffer capacity ($2 \cdot m \cdot n \cdot R$) and the delay $[(n + 1) \cdot m]$ of the Stop-and-Go protocol match those that were reported in [11]. For a more detailed analysis of the Stop-and-Go protocol, an analysis of the Rate Controlled Static Priority protocol (RCSP) [21], and the relationship between RCSP and Stop-and-Go, please see [5].

B. Hierarchical Round-Robin

In this protocol [14], when a new connection for transmitting data packets from computer i to computer j is established, each computer k on the selected path from i to j can be involved in several other connections. Thus, computer k starts to serve all the connections in which it is involved, including the new connection from i to j . To serve a connection, computer k forward along the connection one of the data packets that it has received earlier along the connection. To serve all the connections in which it is involved, computer k executes successive rounds according to the following three rules. First, each round takes the same amount of time. Second, within a round, computer k devotes a predetermined amount of time to serving each of the connections in which it is involved. Third, in different rounds, computer k may serve its connections in different orders.

The involvement of one computer along some connection in serving that connection can be represented as a linear network of three flow operators: an R -limiter, an m -compactor, and an m -expander



where m is the number of (discrete) instants in one round, and $m \cdot R$ is the maximum number of data packets that the computer can forward along the connection in one round.

From the second rule of the protocol, there is an upper bound on the number of packets served for the connection at each round. Thus, the first operator in the above network is an R -limiter to ensure that the network outputs no more than $m \cdot R$ packets in every round (i.e., in every m instants). From the third rule of the protocol, the order in which connections are served may change from one round to the next. To emulate this behavior, the output of the R -limiter is fed to an m -compactor that groups together the contents of each round. Then, the output of the m -compactor is fed to an m -expander to nondeterministically rearrange the contents of each round.

Assume that the input flow to the network is (m, R) -smooth. Then, by Theorems 1–5, the output flow of the network is also (m, R) -smooth. Moreover, by Theorems 1–6, the required buffer capacity of the network is at most $4 \cdot m \cdot R$ and its delay is at most $4 \cdot m$.

If n copies of this linear network are connected in a large linear network to represent a connection involving n computers, then the required buffer capacity along the connection is at most $4 \cdot m \cdot n \cdot R$, and the delay along the connection is at most $4 \cdot m \cdot n$.

It is reported in [14] that an upper bound on the delay of Hierarchical Round-Robin in one computer is $2 \cdot m$. However, this delay is only for the first packet of a flow. We have managed to exhibit a scenario where the delay is $3 \cdot m$ if the input flow is (m, R) -smooth [5]. Therefore, our upper bound of $4 \cdot m$ is not far off.

Lastly, it is possible that the round size of one computer is different from the round size of the previous computer in the path of the connection. In this case, let m be the round size of a computer k , and m' be the round size of the previous computer. Computer k can still be represented by the above linear network of three operators, but the input flow to this linear network is (m', R) -smooth. From Theorem 3, the output flow of the R -limiter is $(1, R)$ -uniform, and so it is also (m, R) -smooth. From Theorems 1–6, the delay through this linear network is $2 \cdot m' + 2 \cdot m$, and the buffer capacity is at most $2 \cdot m' \cdot R + 2 \cdot m \cdot R$.

C. Timestamp Protocols

In timestamp protocols, such as Weighted Fair Queueing [8], [17], and Virtual Clock [22], when a computer k that is involved in several connections receives a data packet along one of the connections, computer k assigns the received packet a timestamp and stores it in a buffer. Later, when computer k has an opportunity to forward a data packet, it selects from its buffer the packet with the smallest timestamp and forwards the packet along its connection.

The details for computing the timestamps for incoming packets are slightly different for each timestamp protocol; however, the same principle for computing the timestamps applies to all of them. When a computer k receives a packet along a connection whose reserved rate is R (packets per second), the packet is assigned a timestamp equal to the time at which the packet would have been forwarded if the connection was being served by an exclusive computer at the exact rate of R . It is straightforward to show that under this principle, each packet will be forwarded at the time, or at an earlier time than that, indicated by its timestamp¹ [3], [19].

This behavior can be represented by a new flow operator called R -filter, where R is a positive real number. An R -filter may output any value at each instant provided the following two conditions are satisfied. First, the flow conservation property is not violated. Second, the output flow has a rate of at least R . This second property is guaranteed as follows. If an R -limiter and an R -filter have the same input flow, then for any i , $i \geq 0$, the sum of the first i elements of the output flow of the R -filter is at least the sum of the first i elements of the output flow of the R -limiter. This is accomplished by ensuring that the buffer of the R -filter at each instant is not greater than the buffer of the R -limiter at that instant.

¹The exit time of a packet is actually at most the timestamp of the packet plus a small constant. We address this constant below.

Let bf be the buffer flow of an R -limiter with input flow r . The operation of an R -filter is as follows:

$$s_i = \max[(bf_{i-1} + r_i) \cdot X_i, \quad bf_{i-1} + r_i - bf_i]$$

where r is the input flow, s is the output flow, bf is the buffer flow, and each X_i is a real number in the closed interval $[0, 1]$.

Notice that since each X_i is at least zero, then each s_i is also at least zero. Also, since each bf_i is at least zero, then neither term in the max operator can be larger than $bf_{i-1} + r_i$. Hence, the flow conservation property is satisfied. The first term in the max operator allows the output to be any value satisfying the flow conservation property. The second term in the max operator ensures that bf_i is at most bf_i .

We next present some properties of the R -filter.

Theorem 8: For an R -filter, if the input flow is (m, R) -smooth, then

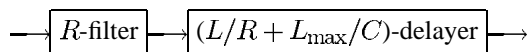
- 1) the buffer capacity is at most $2 \cdot m \cdot R$, and
- 2) the delay is at most $2 \cdot m$. ■

Theorem 9: Let f and g be two R -filters. If the input flow of f is the same as the input flow of the linear network $\langle f, g \rangle$, then every output flow of f is an output flow of $\langle f, g \rangle$ and vice versa. ■

Theorem 9 states that a linear network of two R -filters is identical to a single R -filter. By induction on the number of R -filters, any linear network of R -filters is identical to a single R -filter. Hence, from Theorem 8, the delay of an (m, R) -smooth flow in any linear network of R -filters is at most $2 \cdot m$.

In a timestamp protocol, the timestamp assigned to each packet of an input flow is the time at which the packet exits an R -limiter with the same input flow. From the definition of an R -filter, each packet in its input flow exits no later than the time it would exit an R -limiter and, hence, no later than its timestamp. Thus, an R -filter can be used to represent the behavior of a single computer using a timestamp protocol.

However, in an R -filter, the first bit of a packet of size L may exit up to L/R instants earlier than the last bit of the packet. These bits should exit together, since packets are indivisible units of data and must be transmitted as a whole. In addition, because the transmission of a packet cannot be preempted by the arrival of a packet with a smaller timestamp, the exit time of a packet in a timestamp protocol is actually at most the time it would exit an R -limiter plus L_{\max}/C [3], [19], where L_{\max} is the maximum packet size over all connections, and C is the bandwidth of the output channel of the computer. Therefore, we represent a single computer using a timestamp protocol with the following linear network, where L is the maximum packet size of the connection.



A d -delayer, where d is positive integer, is a flow operator that delays its input flow in an arbitrary manner by at most d instants. More formally, a d -delayer can be defined recursively as a $(d - 1)$ -delayer followed by a 1-delayer. The operation of a 1-delayer is as follows

$$s_i = r_i \cdot X_i + b_{i-1}$$

where r is the input flow, s is the output flow, b is the buffer flow, and each X_i is a real number in the closed interval $[0, 1]$.

The R -filter and d -delayer operators have the following useful properties.

Theorem 10: Let h be a d -delayer, and let f and g be R -filters. If the linear networks $\langle h, g \rangle$ and $\langle f, h \rangle$ have the same input flow, then:

- 1) any output flow of $\langle h, g \rangle$ is also an output flow of $\langle f, h \rangle$;
- 2) the buffer capacity of $\langle f, h \rangle$ is at most the buffer capacity of f plus $d \cdot R$. ■

A connection involving n computers using Weighted Fair Queueing or Virtual Clock is represented by a linear network consisting of n $\langle R$ -filter, $(L/R + L_{\max}/C)$ -delayer \rangle pairs. Using Theorems 9 and 10, it is easy to show that the buffer capacity and delay of this linear network are at most, respectively, the buffer capacity and delay of a single R -filter followed by an $[n \cdot (L/R + L_{\max}/C)]$ -delayer. Note that this property is independent of the characteristics of the input flow. From Theorem 8, if the input to the linear network is (m, R) -smooth, then the delay is at most $2 \cdot m + n \cdot (L/R + L_{\max}/C)$. Furthermore, the buffer capacity of the j th $\langle R$ -filter, $(L/R + L_{\max}/C)$ -delayer \rangle pair is at most $2 \cdot m \cdot R + j \cdot (L + R \cdot L_{\max}/C)$.

The bounds on buffer capacity and delay for Weighted Fair Queueing given above match the bounds derived in [17]. In addition, because both Weighted Fair Queueing and Virtual Clock are instances of a filter followed by a delayer, our analysis yields the new result that Virtual Clock also exhibits the same bounds.

By increasing the delay between R -filters, we have the flexibility to represent protocols other than Weighted Fair Queueing and Virtual Clock. For example, we can represent the Self-Clocking Fair Queueing protocol [12], which is a simplification of the Virtual Clock protocol. The simplification consists of stamping each packet with a value similar to that used in the Virtual Clock protocol, but without the need of a real-time clock. This causes the delay of Self-Clocking Fair Queueing to be somewhat larger than that of Virtual Clock. In particular, let $k + 1$ be the number of connections sharing the output link. A single computer using Self-Clocking Fair Queueing can be represented by the linear network $\langle R$ -filter, $(k \cdot L_{\max}/C + L/R)$ -delayer \rangle . A path of n computers is represented by a linear network of n of these pairs, and its delay and buffer capacity follows from Theorems 8, 9, and 10.

Note that, if each computer in a network path uses a protocol that can be represented by a filter followed by a delayer, then the end-to-end delay and buffer capacities at each computer can be obtained from Theorems 8, 9, and 10. This is the case even though the same protocol is not used in all computers in the path.

We have learned that, concurrently to our work, others have independently proven upper bounds on the end-to-end delay of Virtual Clock and Self-Clocking Fair Queueing that are similar to those presented in this section [10], [13]. Also concurrently, and using a different network model, a result to obtain the end-to-end delay in a network path of computers with different timestamp protocols is presented in [13].

VI. GENERAL NETWORKS

In this section, we extend Flow Theory with three flow operators with multiple inputs or multiple outputs. This extension allows us to construct flow operator networks of arbitrary topology.

A. Mergers

A merger is a flow operator with two input flows and one output flow. It outputs at each instant the sum of its two inputs at that instant. Formally, the operation of the merger is defined as follows:

$$s_i = q_i + r_i$$

where q and r are the input flows, and s is the output flow of the merger.

Theorem 11: Let m and n be positive integers, and k be a common multiple of m and n . For a merger:

- 1) if one input flow is (m, R) -smooth and the other input flow is (n, S) -smooth, then the output flow is $(k, R+S)$ -smooth;
- 2) if one input flow is (m, R) -uniform and the other input flow is (n, S) -uniform, then the output flow is $(k, R + S)$ -uniform. ■

B. Splitters

Let X be a real number in the closed interval $[0, 1]$. An X -splitter is a flow operator with one input flow and an ordered pair of output flows. The X -splitter splits its input between its two outputs with the ratio X to $1 - X$. Formally, the operation of an X -splitter is defined as follows:

$$s_i = X \cdot r_i$$

$$t_i = (1 - X) \cdot r_i$$

where s and t are the output flows and r is the input flow of the X -splitter.

Theorem 12: For an X -splitter:

- 1) if the input flow is (m, R) -smooth, then the first output flow is $(m, X \cdot R)$ -smooth and the second output flow is $[m, (1 - X) \cdot R]$ -smooth;
- 2) if the input flow is (m, R) -uniform, then the first output flow is $(m, X \cdot R)$ -uniform and the second output flow is $[m, (1 - X) \cdot R]$ -uniform. ■

C. Separators

A separator is a flow operator with one input flow and two output flows. It splits its input between its two outputs with a ratio chosen arbitrarily at each instant. Formally, the operation of a separator is defined as follows:

$$s_i = X_i \cdot r_i$$

$$t_i = (1 - X_i) \cdot r_i$$

where s and t are the output flows, r is the input flow, and each X_i is a real number in the closed interval $[0, 1]$.

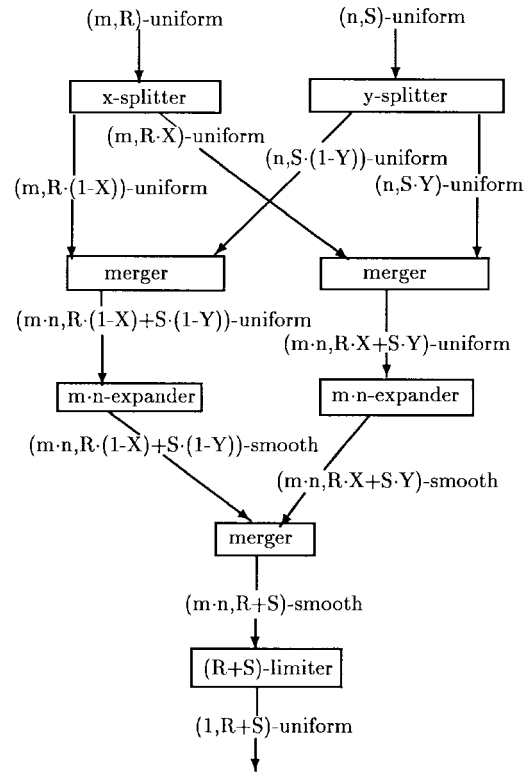


Fig. 2. Acyclic network of flow operators.

Theorem 13: For a separator:

- 1) if the input flow is (m, R) -smooth, then both output flows are (m, R) -smooth;
- 2) if the input flow is (m, R) -uniform, then both output flows are (m, R) -uniform. ■

D. Acyclic Networks

Mergers, splitters, and separators, along with previous flow operators, can be used to construct nonlinear acyclic networks. An *acyclic network* consists of a collection of interconnected flow operators that form an acyclic directed graph.

Fig. 2 shows an example of an acyclic network with two input and one output flows. From the fact that the two input flows are (m, R) -uniform and (n, S) -uniform, one can use the theorems in Sections III and VI to deduce the smoothness or uniformity of every flow in the network as shown in Fig. 2.

Acyclic networks may be used to represent various common scenarios in the routing of flows. For example, an acyclic network with multiple input flows and a single output flow may represent the case of multiple sources generating data to be received at a common destination. Also, an acyclic network with one input flow and one output flow may represent the case when, for purposes of fault tolerance and load balancing [16], the input flow is split into multiple flows. Each flow is routed via a separate path, and then merged with the remaining flows at the destination.

E. Cyclic Networks

A *cyclic network* consists of a collection of interconnected flow operators that form a directed graph with cycles.

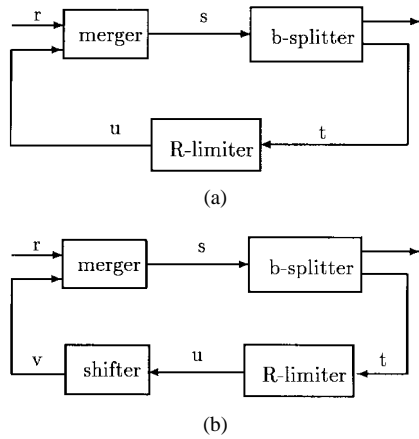


Fig. 3. Cyclic network of flow operators.

Some cyclic networks have an anomaly not present in acyclic networks. This anomaly is explained by an example. Fig. 3(a) shows a cyclic network, where r is the input flow of the network; and flows s , t , and u constitute a cycle. Consider the first element s_0 of flow s . Element s_0 depends on u_0 , which depends on t_0 , which in turn depends on s_0 . Hence, the first elements in the flows in the cycle are not well defined.

This anomaly would not exist if, for one flow operator, the i th element of the output flow is independent of the i th element of the input flow. For example, assume that, for all i , u_i is independent of t_i , and $u_0 = 0$. In this case, s_0 is well defined (namely, $s_0 = r_0 + 0$). Since s_0 is well defined, so is t_0 . This in turn ensures that u_1 is well defined. Hence, we can repeat this argument to show that every element in every internal flow in the network is well defined.

An operator whose i th output flow element is independent of its i th input flow element is the m -compactor. Thus, if each cycle in the network has at least one compactor operator, the cyclic dependency problem mentioned above does not occur. The simplest compactor operator is the 1-compactor, whose behavior is simply to shift the input flow elements by one instant. That is, if u is its input flow and v its output flow, then $v_{i+1} = u_i$ for all $i \geq 0$, and $v_0 = 0$. Due to this behavior, we also refer to the 1-compactor as the shifter.

Consider the cyclic network in Fig. 3(b). It is similar to the one in Fig. 3(a) except that a shifter is added between the R -limiter and the merger. Since v_i is independent of u_i , and $v_0 = 0$, every internal flow in the network is well defined.

Notice that the input to a shifter must be uniform, since if its input is (m, R) -smooth, then its output is not necessarily (m, R) -smooth. However, if its input is (m, R) -uniform, then its output is also (m, R) -uniform.

Next, we consider how to calculate the rate of the internal flows of a cyclic network. Fig. 4 shows a network with two cycles. An m -compactor is common to both cycles, and hence all network flows are well defined. We are given that the input flow of the network is (m, R) -smooth.

We begin by choosing a flow in each cycle and label its rate with a variable. In Fig. 4, we assume that the output of the top m -expander is (m, X) -smooth for some X , and the output of the bottom m -expander is (m, Y) -smooth for some Y . We

would like to obtain values of X and Y given the input rate R and the constants associated with each splitter (i.e., b and c).

The theorems in Sections III and VI can be used to determine the characteristics of each flow in the network in terms of variables X and Y . The results of applying these theorems are shown in the figure. From the input to each m -expander, and from Theorem 5, we have the following two equations:

$$\begin{aligned} X &= (1 - c) \cdot (b \cdot X + Y) + R \\ Y &= (b \cdot X + Y) \cdot c. \end{aligned}$$

Solving these two linear equations, we obtain the following values for X and Y :

$$\begin{aligned} X &= \frac{R}{1 - b} \\ Y &= \frac{R \cdot b \cdot c}{(1 - b) \cdot (1 - c)}. \end{aligned}$$

After X and Y are calculated, we can determine the smoothness or uniformity of each flow in the network.

VII. MULTIFLOWS

In this section, we consider how a number of flows are merged into a single entity, called a multiflow.

A *multiflow* is an ordered pair (r, s) , where both r and s are flows. Flows r and s are the *constituent* flows of the multiflow (r, s) .

A flow t is an *instance* of multiflow (r, s) if and only if there is some j , $j \geq 0$, such that

$$t_i = \begin{cases} r_i & \text{if } i < j \\ r_i + s_{(i-j)} & \text{if } i \geq j. \end{cases}$$

That is, an instance of a multiflow is the element-wise addition of its two constituent flows, after the elements of one constituent flow have been “shifted” by some number of instants. This shift characterizes the addition of a new flow into the network after the processing of an existing flow has begun.

Since a multiflow is a representation of all its instances, a property that is shared by all the instances of a multiflow can be considered a property of the multiflow. Thus, we have the following definitions.

A multiflow (r, s) is (m, R) -smooth if and only if each instance of the multiflow is (m, R) -smooth. Likewise, a multiflow (r, s) is (m, R) -uniform if and only if each instance of the multiflow is (m, R) -uniform.

Theorem 14: Let r and s be the constituent flows of a multiflow (r, s) .

- 1) If r is (m, R) -smooth and s is (m, S) -uniform, then (r, s) is $(m, R + S)$ -smooth.
- 2) If r is (m, R) -uniform and s is (m, S) -uniform, then (r, s) is $(m, R + S)$ -uniform.
- 3) If (r, s) is (m, R) -smooth, then r is (m, R) -smooth and s is (m, R) -uniform.
- 4) If (r, s) is (m, R) -uniform, then each of r and s is (m, R) -uniform. ■

Proof of Theorem 2

Let the input flow of the flow operator be r , the output flow be s , and the buffer flow be b .

From the definition of buffer capacity, we are required to show that for any buffer flow b and any $i \geq 0$

$$b_i \leq \left\lceil \frac{D}{m} \right\rceil \cdot m \cdot R.$$

From the definition of delay, we are given that

$$b_i \leq \sum_{j=i+1}^{i+D} s_j.$$

Since i and D are natural numbers, we rewrite the right-hand side

$$b_i \leq \sum_{j=0}^{i+D} s_j - \sum_{j=0}^i s_j.$$

Applying (3) to both sums over s

$$b_i \leq \sum_{j=0}^{i+D} r_j - b_{i+D} - \left(\sum_{j=0}^i r_j - b_i \right).$$

From arithmetic

$$b_{i+D} \leq \sum_{j=i+1}^{i+D} r_j.$$

From the uniformity of r , the sum of any D consecutive instants of r is at most $\lceil D/m \rceil \cdot m \cdot R$. Hence

$$b_{i+D} \leq \left\lceil \frac{D}{m} \right\rceil \cdot m \cdot R.$$

Thus, since $i \geq 0$, the desired property holds for any b_j where $j \geq D$. For the case of $j < D$, from (3), knowing that every s instance is at least 0, and from the uniformity of r , it follows that

$$b_j \leq \sum_{k=0}^j r_k \leq \left\lceil \frac{D}{m} \right\rceil \cdot m \cdot R.$$

Proof of Theorem 6

Let r be the input flow, s an output flow, and bf the resulting buffer flow of operator f . Let t be the input flow, g an output flow, and bg the resulting buffer flow of operator g . Hence, r is the input flow and t is an output flow of the linear network $\langle f, g \rangle$.

We first show that $c_i = bf_i + bg_i$ by induction on i .

Base Case: We show that $c_{-1} = bf_{-1} + bg_{-1}$.

By the definition of a buffer flow of a network and a buffer flow of a flow operator, c_{-1} , bf_{-1} , and bg_{-1} are all equal to zero. Thus, $c_{-1} = bf_{-1} + bg_{-1}$.

Inductive Step: Assuming $c_{i-1} = bf_{i-1} + bg_{i-1}$, where $i \geq 0$, we show that $c_i = bf_i + bg_i$.

From the definition of a buffer flow of a linear network, we have

$$c_i = c_{i-1} + r_i - t_i.$$

Substituting c_{i-1} by the induction hypothesis we obtain

$$c_i = bf_{i-1} + bg_{i-1} + r_i - t_i.$$

Recall that from the definition of a buffer flow of a flow operator, bf_i and bg_i are defined as

$$bf_i = bf_{i-1} + r_i - s_i \quad bg_i = bg_{i-1} + s_i - t_i.$$

We solve for r_i and t_i in the above two equations and substitute the result into the equation for c_i , and we obtain

$$c_i = bf_{i-1} + bg_{i-1} + (bf_i - bf_{i-1} + s_i) - (-bg_i + bg_{i-1} + s_i).$$

Simplifying, we obtain the desired result

$$c_i = bf_i + bg_i. \quad (4)$$

We now prove the two parts of the theorem.

Part 1: From the definition of buffer capacity, for any buffer flows bf and bg of operators f and g when the input to $\langle f, g \rangle$ is r , we obtain the following for every $i, i \geq 0$

$$bf_i \leq Bf \quad bg_i \leq Bg.$$

Therefore, from (4) shown above we conclude that

$$c_i = bf_i + bg_i \leq Bf + Bg.$$

Since C is the maximum value of any element c_i of any buffer flow c of $\langle f, g \rangle$, then

$$C \leq Bf + Bg.$$

Thus, Part 1 of the theorem holds.

Part 2: We begin by noting that, since the delay of f is Df , the following must hold for every $i, i \geq 0$, where s is the output flow of f

$$bf_i \leq \sum_{j=i+1}^{i+Df} s_j.$$

We add and subtract the sum $s_0 + \dots + s_i$ from the right-hand side and obtain

$$bf_i \leq \sum_{j=0}^{i+Df} s_j - \sum_{j=0}^i s_j.$$

Using (3), we replace each of the sums over s (the input to g) with the buffer of g and a sum over t (the output of g). We also add bg_i to each side

$$bf_i + bg_i \leq \left(bg_{i+Df} + \sum_{j=0}^{i+Df} t_j \right) - \left(bg_i + \sum_{j=0}^i t_j \right) + bg_i.$$

Note that the bg_i terms on the right-hand side cancel each other. We combine the sums over t into one sum and obtain

$$bf_i + bg_i \leq bg_{i+Df} + \sum_{j=i+1}^{i+Df} t_j.$$

Since the maximum delay of g is Dg , we have

$$bf_i + bg_i \leq \sum_{j=i+Df+1}^{i+Df+Dg} t_j + \sum_{j=i+1}^{i+Df} t_j.$$

Joining the two sums over t , we obtain

$$bf_i + bg_i \leq \sum_{j=i+1}^{i+Df+Dg} t_j.$$

From (4) proven above, if c is the buffer flow of the network, then

$$c_i \leq \sum_{j=i+1}^{i+Df+Dg} t_j.$$

Since D is the smallest integer such that for any output flow t and its corresponding buffer flow c

$$c_i \leq \sum_{j=i+1}^{i+D} t_j$$

we conclude that $D \leq Df + Dg$. Hence, Part 2 of the theorem holds. ■

Proof of Theorem 8

Part 1: Since the buffer of an R -filter is always at most the buffer of an R -limiter, then by Part 2 of Theorem 3, the buffer capacity of an R -filter is at most $2 \cdot m \cdot R$. Thus, Part 1 of Theorem 8 holds.

Part 2: Let r be the input flow, s the output flow, and bf the buffer flow of an R -filter.

Let r be the input flow, t the output flow, bl the buffer flow, and D the delay of an R -limiter.

By property (3) (with respect to buffer flow bl), we have the following:

$$bl_j = \sum_{i=0}^j r_i - \sum_{i=0}^j t_i.$$

Since the delay of the R -limiter is D , and by the definition of delay we have

$$\sum_{i=0}^j r_i - \sum_{i=0}^j t_i \leq \sum_{i=j+1}^{j+D} t_i.$$

By algebra, we combine the summations on t , and also by the definition of an R -filter, it follows that

$$\sum_{i=0}^j r_i \leq \sum_{i=0}^{j+D} t_i \leq \sum_{i=0}^{j+D} s_i.$$

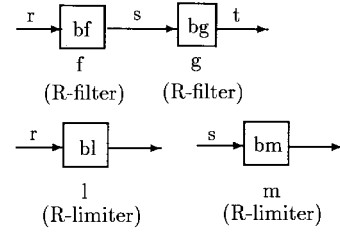


Fig. 5. Composition of two R -filters.

Rearranging the terms, we obtain

$$0 \leq \sum_{i=0}^{j+D} s_i - \sum_{i=0}^j r_i.$$

We add bf_j to both sides and obtain

$$bf_j \leq \sum_{i=0}^{j+D} s_i - \sum_{i=0}^j r_i + bf_j.$$

By (3) (with respect to bf), it follows that

$$bf_j \leq \sum_{i=0}^{j+D} s_i - \sum_{i=0}^j s_i.$$

We combine the two sums over s and finally obtain

$$bf_j \leq \sum_{i=j+1}^{j+D} s_i.$$

Hence, the delay of an R -filter is at most D . Since D is at most $2 \cdot m$ (Theorem 3, Part 3), then the R -filter's delay is also at most $2 \cdot m$. ■

Proof of Theorem 9

Fig. 5 shows the linear network $\langle f, g \rangle$ where f and g are R -filters. The figure also shows the input flow, output flow, and buffer flow of both f and g . Finally, Fig. 5 shows the buffer flow of an R -limiter l whose input flow is r , and the buffer flow of an R -limiter m whose input flow is s .

By the operation of an R -filter and the definition of a buffer flow, the following hold, where X_i and Y_i are chosen nondeterministically at instant i in the range $[0, 1]$.

$$s_i = \max[(bf_{i-1} + r_i) \cdot X_i, bf_{i-1} + r_i - bl_i] \quad (5)$$

$$bf_i = bf_{i-1} + r_i - s_i \quad (6)$$

$$t_i = \max[(bg_{i-1} + s_i) \cdot Y_i, bg_{i-1} + s_i - bm_i] \quad (7)$$

$$bg_i = bg_{i-1} + s_i - t_i. \quad (8)$$

We first show that any output flow of an R -filter with input r is also an output flow of $\langle f, g \rangle$ with input r . This is easily seen since f can always choose to transfer each input flow element directly to the output flow, i.e., X_i is always chosen to be one. Hence, the input to g would be r , and since g is an R -filter, any output flow of an R -filter with input r can also be an output flow of $\langle f, g \rangle$ with input r .

We next show that any output flow of $\langle f, g \rangle$ is also an output flow of an R -filter with input r .

The buffer flow of the linear network $\langle f, g \rangle$ at instant i is $bf_i + bg_i$. If $\langle f, g \rangle$ is to behave like a single R -filter, its output flow must satisfy the following relationship:

$$t_i = \max[(bf_{i-1} + bg_{i-1} + r_i) \cdot Z_i, bf_{i-1} + bg_{i-1} + r_i - bl_i] \quad (9)$$

for appropriately chosen Z_i in the range $[0, 1]$.

To show the above, we first substitute the definition of s_i given in (5) into the first term of the max operator in (7) and we obtain the following:

$$\{bg_{i-1} + \max[(bf_{i-1} + r_i) \cdot X_i, bf_{i-1} + r_i - bl_i]\} \cdot Y_i.$$

This expression yields only values in the range $0 \dots bf_{i-1} + bg_{i-1} + r_i$. Hence, the possible values of this expression are a subset of the possible values of the first term of the max operator in (9).

We now show that the second term of the max operator in (7) is equivalent to the second term of the max operator in (9). We are required to show the following:

$$bg_{i-1} + s_i - bm_i = bf_{i-1} + bg_{i-1} + r_i - bl_i.$$

Cancelling bg_{i-1} from both sides and substituting $bf_{i-1} + r_i$ with $bf_i + s_i$ [see (6)], the above is equivalent to

$$s_i - bm_i = bf_i + s_i - bl_i.$$

Equivalently, we must show that $bl_i = bm_i + bf_i$ for all i . We do this with induction.

Base Case: $bl_{-1} = bm_{-1} + bf_{-1}$, since by definition bl_{-1} , bm_{-1} , and bf_{-1} are all zero.

Inductive Step: Assuming that $bl_{i-1} = bm_{i-1} + bf_{i-1}$ holds for some $i \geq 0$, we show that $bl_i = bm_i + bf_i$ also holds.

Let the unary operator $^+$ be defined as follows:

$$(x)^+ = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0. \end{cases}$$

It follows from the operation of an R -limiter that

$$bl_i = (bl_{i-1} + r_i - R)^+ \quad (10)$$

$$bm_i = (bm_{i-1} + s_i - R)^+. \quad (11)$$

From the induction hypothesis and (10), we obtain

$$bl_i = (bf_{i-1} + bm_{i-1} + r_i - R)^+. \quad (12)$$

Solving for s in (6) and substituting into (11) yields

$$bm_i = (bf_{i-1} + bm_{i-1} + r_i - bf_i - R)^+. \quad (13)$$

We perform a case analysis on the right-hand side of (12).

- 1) If $bf_{i-1} + bm_{i-1} + r_i - R \leq 0$, then $bl_i = 0$, and since by the definition of an R -filter $bf_i \leq bl_i$, we have $bf_i = 0$. Furthermore, from (13), $bm_i = 0$ since it is always the case that $bf_i \geq 0$. Hence, $bl_i = bm_i + bf_i$.
- 2) If $bf_{i-1} + bm_{i-1} + r_i - R > 0$ then,

$$bl_i = bf_{i-1} + bm_{i-1} + r_i - R. \quad (14)$$

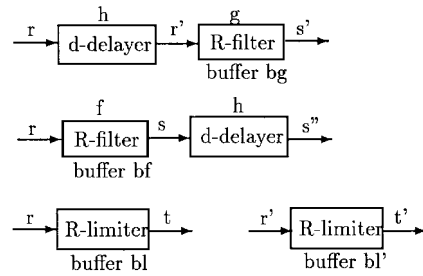


Fig. 6. Exchange of an R -filter and d -delayer.

We have the following two cases for the right-hand side of (13).

- a) If $bf_{i-1} + bm_{i-1} + r_i - bf_i - R \geq 0$ then

$$bm_i = bf_{i-1} + bm_{i-1} + r_i - bf_i - R.$$

The above and (14) yields

$$bl_i = bm_i + bf_i.$$

- b) The case $bf_{i-1} + bm_{i-1} + r_i - bf_i - R < 0$ along with (14) imply that $bl_i - bf_i < 0$, which we know cannot occur, since due to the definition of an R -filter it is the case that $bf_i \leq bl_i$. ■

Proof of Theorem 10

Part 1: Consider the scenario of Fig. 6, where h is a d -delayer, and both f and g are R -filters. To prove the theorem, given the behavior of a d -delayer, it is only necessary to show that, for all i

$$\sum_{j=0}^i s'_j \leq \sum_{j=0}^i s_j \leq \sum_{j=0}^{i+d} s'_j. \quad (15)$$

This allows operator h to turn s into s' . Therefore, given the output flow s' of the first linear network, we must show that there is a nondeterministic choice that operator f can take at each instance such that (15) holds.

First, from the definition of a d -delayer, it follows for all i

$$\sum_{j=0}^i r'_j \leq \sum_{j=0}^i r_j \leq \sum_{j=0}^{i+d} r'_j. \quad (16)$$

Also, from the flow-conservation principle and (16), we have for all i

$$\sum_{j=0}^i s'_j \leq \sum_{j=0}^i r'_j \leq \sum_{j=0}^i r_j. \quad (17)$$

We show by induction that (15) always holds.

Base Case: $i = -1$.

The first two sums of (15) are zero (empty range), and the third is nonnegative, so (15) holds.

Induction Case: $i \geq 0$.

Assume the following relation holds:

$$\sum_{j=0}^{i-1} s'_j \leq \sum_{j=0}^{i-1} s_j \leq \sum_{j=0}^{i-1+d} s'_j.$$

We must show that the following also holds:

$$\sum_{j=0}^i s'_j \leq \sum_{j=0}^i s_j \leq \sum_{j=0}^{i+d} s'_j. \quad (18)$$

Let bf be the buffer flow of filter f . At instance i , f chooses a fraction of $bf_{i-1} + r_i$ as its output s_i . If $s_i = bf_{i-1} + r_i$, then, from the definition of a buffer flow,

$$\sum_{j=0}^i r_j = \sum_{j=0}^i s_j.$$

The above and (17) indicate that f can ensure that the left-hand side of (18) holds. In order not to violate the right-hand side, f should choose for s_i the least fraction of $bf_{i-1} + r_i$ such that the left-hand side holds. Consider first the following case:

$$\sum_{j=0}^i s'_j \leq \sum_{j=0}^{i-1} s_j.$$

The value chosen for s_i is zero, and (18) holds by the induction hypothesis. For the opposite case, s_i is chosen such that

$$\sum_{j=0}^i s'_j = \sum_{j=0}^i s_j$$

holds.

However, the definition of a filter places a lower bound on s_i . If this bound is greater than the value chosen above, then we must check that the right-hand side of (18) still holds. When s_i equals the lower bound, from the definition of the filter, it implies the following:

$$\sum_{j=0}^i s_j = \sum_{j=0}^i t_j.$$

From the definition of a filter, the next relation holds:

$$\sum_{j=0}^{i+d} t'_j \leq \sum_{j=0}^{i+d} s'_j.$$

Thus, we are required only to show the following:

$$\sum_{j=0}^i t_j \leq \sum_{j=0}^{i+d} t'_j.$$

The above follows from Lemma 1 given below.

Part 2: Let c denote the buffer flow of the linear network $\langle f, h \rangle$

$$\begin{aligned} & c_i \\ &= \{\text{definition of a buffer flow}\} \\ & \sum_{j=0}^i r_j - \sum_{j=0}^i s'_j \\ & \leq \{\text{shown later below}\} \\ & \sum_{j=0}^i r_j - \sum_{j=0}^{i-d} t_j \end{aligned}$$

$$\begin{aligned} & \leq \{\text{definition of an } R\text{-limiter}\} \\ & \sum_{j=0}^i r_j - \left(\sum_{j=0}^i t_j - d \cdot R \right) \\ & = \{\text{definition of buffer flow}\} \\ & bl_i + d \cdot R. \end{aligned}$$

Thus, since the buffer capacity of f is the same as that of an R -limiter, and each buffer flow element of $\langle f, h \rangle$ is at most the corresponding buffer flow element of an R -limiter plus $d \cdot R$, Part 2 holds. Finally, the relation that we omitted above is the following:

$$\sum_{j=0}^i s'_j \geq \sum_{j=0}^{i-d} s_j \geq \sum_{j=0}^{i-d} t_j.$$

This follows from the definition of a d -delayer and the definition of an R -filter. ■

Lemma 1: For all i

$$\sum_{j=0}^i t_j \leq \sum_{j=0}^{i+d} t'_j.$$

Proof: Let k be the largest index at most $i+d$ for which $bl'_{k=0}$. From the definition of a buffer flow, the following holds:

$$\sum_{j=0}^k r'_j = \sum_{j=0}^k t'_j. \quad (19)$$

In addition, we require the following:

$$\begin{aligned} & \sum_{j=0}^i t_j - \sum_{j=0}^k t'_j \\ &= \{\text{from (19)}\} \sum_{j=0}^i t_j - \sum_{j=0}^k r'_j \\ & \leq \{\text{from (16) if } k \geq d, \text{ empty range if } k < d\} \\ & \sum_{j=0}^i t_j - \sum_{j=0}^{k-d} r_j \\ & \leq \{\text{from flow conservation if } k \geq d, \\ & \text{empty range if } k < d\} \\ & \sum_{j=0}^i t_j - \sum_{j=0}^{k-d} t_j \\ & \leq \{t_j \leq R \text{ for all } j, \text{ and empty range if } k < d\} \\ & [i - (k - d)] \cdot R. \end{aligned}$$

Thus

$$\begin{aligned} & \sum_{j=0}^i t_j - \sum_{j=0}^{i+d} t'_j \\ & \leq \{\text{from the above}\} \\ & [i - (k - d)] \cdot R - \sum_{j=k+1}^{i+d} t'_j \end{aligned}$$

= {if $k = i + d$, both terms are zero, if $k < i + d$,
 from the definition of k , t'_{k+1} up to t'_{i+d}
 are each equal to R }
 0.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their comments and suggestions in improving the quality of this paper.

REFERENCES

- [1] J. Cobb and M. Gouda, "Flow theory: Verification of rate-reservation protocols," in *Proc. 1st IEEE Int. Conf. Network Protocols*, 1993, p. 198.
- [2] ———, "Neighbor-to-neighbor vs. end-to-end: Schemes for error recovery," AT&T Tech. Memo. 11384-931118-43TM.
- [3] ———, "Flow timestamps," in *Proc. Annu. Joint Conf. Inform. Sci.*, 1995.
- [4] J. Cobb, M. Gouda, and A. El-Nahas, "Time-shift scheduling: Fair scheduling of flows in high speed networks," in *Proc. IEEE Int. Conf. Network Protocols*, 1996.
- [5] J. Cobb, "Flow theory and the analysis of timed-flow protocols," Ph.D. dissertation, Univ. Texas Austin, May 1996.
- [6] R. L. Cruz, "A calculus for network delay, Part I: Network elements in isolation," *IEEE Trans. Inform. Theory*, vol. 37, Jan. 1991.
- [7] ———, "A calculus for network delay, Part II: Network analysis," *IEEE Trans. Inform. Theory*, vol. 37, Jan. 1991.
- [8] A. Demers, S. Keshav, and S. Shenkar, "Analysis and simulation of a fair queueing algorithm," in *Proc. ACM SIGCOMM*, 1989.
- [9] A. El-Nahas and K. Ahmed, "The slow-and-go method for congestion control," in *Proc. Annu. Joint Conf. Inform. Sci.*, 1995.
- [10] N. R. Figueira and J. Pasquale, "An upper bound on delay for the virtual clock service discipline," *IEEE/ACM Trans. Networking*, vol. 3, Aug. 1995.
- [11] S. J. Golestani, "A framing strategy for congestion management," *IEEE J. Select. Areas Commun.*, pp. 1064–1077, Sept. 1991.
- [12] ———, "A self-clocking fair-queueing scheme for broadband applications," in *Proc. IEEE INFOCOM*, 1994.
- [13] P. Goyal, S. Lam, and H. Vin, "Determining end-to-end delay bounds in heterogeneous networks," in *Proc. NOSSDAV Workshop*, 1995.
- [14] C. R. Kalmanek, H. Kanakia, and S. Keshav, "Rate controlled servers for very high-speed networks," in *Proc. IEEE GLOBECOM*, 1990.
- [15] D. Little, "A proof for the queueing formula $L = \lambda W$," *Oper. Res.*, vol. 9, pp. 383–387, May 1961.
- [16] K. Maxenchuck, "Dispersivity routing in high-speed networks," *Computer Networks ISDN Syst.*, vol. 25, pp. 645–661, 1993.
- [17] A. K. J. Parekh, "A generalized processor sharing approach to flow control in integrated services networks," Tech. Rep. LIDS-TH-2089, Lab. Inform. Decision Syst., Mass. Inst. Technol.
- [18] C. J. Sreenan, "A service oriented approach to continuous media synchronization," in *Proc. IEEE INFOCOM*, 1994.
- [19] G. Xie and S. Lam, "Delay guarantee of virtual clock server," *IEEE/ACM Trans. Networking*, Dec. 1995.
- [20] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proc. IEEE*, vol. 83, Oct. 1995.
- [21] H. Zhang and D. Ferrari, "Rate controlled static priority queueing," in *Proc. 1993 IEEE INFOCOM Conf.*
- [22] L. Zhang, "Virtual clock: A new traffic control algorithm for packet-switched networks," *ACM Trans. Comput. Syst.*, vol. 9, no. 2, May 1991.



Jorge A. Cobb (M'97) received the B.S. degree with highest honors from the University of Texas at El Paso in 1987, the M.A. degree from the University of Texas at Austin in 1989, and the Ph.D. degree from the University of Texas at Austin in 1996.

He has worked for AT&T Information Systems in Denver, CO. He joined the Faculty of the Department of Computer Science at the University of Houston in 1995, where he is currently an Assistant Professor. His main research interest is computer networking, with an emphasis on real-time scheduling and mobile computing. His research interests also include parallel and distributed computing.

Dr. Cobb was awarded the AT&T Bell Laboratories Scholarship.



Mohamed G. Gouda received the bachelor's degree in engineering and in mathematics from Cairo University, Egypt, the M.A. degree in mathematics from York University, and the M.S. and Ph.D. degrees in computing science from the University of Waterloo.

He moved to the United States where he worked for the Honeywell Corporate Technology Center for three years. In 1980, he moved to the University of Texas at Austin, where he currently holds the Mike A. Myers Centennial Professorship in Computing Science. He spent one summer at Bell Labs in Murray Hill, NJ, one summer at MCC in Austin, and one winter at the Eindhoven Technical University in The Netherlands. His primary research area is distributed and concurrent computing. In this area, he has been working on abstraction, nondeterminism, atomicity, convergence, stability, formality, correctness, efficiency, scientific elegance, and technical beauty. He is the author of the textbook *Elements of Network Protocol Design* (Wiley, 1998).

Dr. Gouda was the founding Editor-in-Chief of the journal *Distributed Computing* (Springer-Verlag). He was the first Program Committee Chairman for the International Conference on Network Protocols, established by IEEE in 1993. He was the first Program Committee Chairman for the Symposium on Advances in Computers and Communications, established by IEEE and held in Egypt in 1995. He is an original member of the Austin Tuesday Afternoon Club. He was the 1993 winner of the Kuwait Award in Basic Sciences.