# Flow Theory: Verification of Rate-Reservation Protocols

Jorge A. Cobb Mohamed G. Gouda

Department of Computer Sciences*
The University of Texas at Austin
Austin, TX 78712

## Abstract

We develop a simple theory of flows and show how to use this theory in verifying rate-reservation protocols in computing networks. The 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 four types of flow operators: limiters, compactors, expanders, and filters. Many rate-reservation protocols can be represented as linear networks of these flow operators. We prove that, if the input flow to any of these networks is smooth or uniform, then the internal buffer and the delay in each operator in the network are bounded. We use this method to prove that a number of rate-reservation protocols (for example, stop-and-go, hierarchical round-robin, fair queueing and virtual clock) require bounded buffering and introduce bounded delay.

## 1 Introduction

A *rate-reservation protocol* is a connection-oriented protocol that guarantees for each established connection an average transmission rate. The average transmission rate is negotiated during the establishment of the connection, and the connection is rejected if the required resources to guarantee its rate cannot be allocated. For a survey of these protocols, see [9].

Rate-reservation protocols are important for two reasons. First, rate-reservation protocols minimize the possibility of packet loss due to congestion, and hence they reduce the number of round-trip times required to transmit data. Second, rate-reservation protocols are well suited for packet switched networks that are capable of delivering real-time traffic, such as voice and video, which has stringent packet loss and delay requirements.

Reasoning about rate-reservation protocols has been either informal, or based on defining a flow as a real-valued function whose domain is a continuous time interval. This latter approach requires the use of complicated integration techniques from the realm of calculus.

In this paper, we propose an alternative method for reasoning about rate-reservation protocols. This alternative method is based on discrete mathematics in which a flow is represented as an infinite sequence of real numbers. This alternative approach is easier to follow and simplifies the resulting analysis of rate-reservation protocols.

The paper is organized as follows. In Section 2, 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 3. Linear networks of flow operators are presented in Section 4. In Section 5, we prove that several rate-reservation protocols require bounded buffering and introduce bounded delays. Concluding remarks are given in Section 6. Due to space restrictions, we omit all proofs. (The omitted proofs are presented in [1].)

## 2 Flows

A *flow* r is an infinite sequence $r_0, r_1, r_2, \ldots$, 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

*This work is supported in part by an AT&T Bell Laboratories Ph.D. Scholarship.



0-8186-3670-X/93 $03.00 © 1993 IEEE

$j = 0, 1, 2, \ldots,$

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

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

The smoothness property of a flow over a continuous timeline was introduced in [6]. 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 $j = 0, 1, 2, \ldots,$

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

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

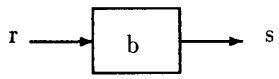
**Theorem 1**
Let r be an (m,R)-smooth flow and r′ be an (m,R)-uniform flow.

1. r is $(m, 2 * R)$-uniform, and
   r′ is (m,R)-smooth.
2. For any n multiple of m,
   r is (n,R)-smooth and r′ is (n,R)-uniform.
3. For any S larger than or equal to R,
   r is (m,S)-smooth and r′ is (m,S)-uniform.

□

## 3 Flow Operators

A *flow operator* has an input flow r and an output flow s. At the ith instant, the operator inputs $r_i$ on its input flow, outputs $s_i$ on its output flow, and stores the remainder in an internal buffer. The content of the internal buffer at the ith instant is denoted $b_i$. The infinite sequence $b_0, b_1, b_2, \ldots$ 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 by the following two equations for every $i = 0, 1, 2, \ldots,$

$$s_i = F.(r_i, b_{i-1})$$
$$b_i = b_{i-1} + r_i - s_i$$

where $b_{-1} = 0$ and F is a nondeterministic function called the *operation* of the flow operator. When invoked, the nondeterministic function F takes a pair of real numbers, computes an interval of real numbers, then returns one real number, chosen nondeterministically, from the computed interval. 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 buffer flow b and output flow s
of the flow operator when its input flow is r,
(For every i, $s_i \leq r_i + b_{i-1}$)
).

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 $i = 0, 1, 2, \ldots, b_i \geq 0$.

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

(For every buffer flow b of the operator
when its input flow is r,
(For every i, $b_i \leq B$)
).

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

(For every buffer flow b and output flow s
of the flow operator when its input flow is r,
(For every i, $b_i \leq s_{i+1} + s_{i+2} + \cdots + s_{i+D}$)
).

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

**Theorem 2**
Consider an arbitrary flow operator whose input is (m,R)-uniform.

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

Then $B \leq \lceil D/m \rceil * m * R$.

□

Note that Theorem 2 remains valid even if the input to the arbitrary flow operator is (m,R/2)-smooth.

The notion of viewing network traffic in a nonprobabilistic way by requiring that a flow satisfy a burstiness constraint was introduced in [2, 3]. However, the model used for a flow was based on a continuous timeline, which differs from our discrete approach. Moreover, the flow conservation property and the nondeterministic behavior of operators were not explicitly stated, and a precise definition of delay was not given.

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 the boundedness of buffers and delays in both operators. (Linear networks are discussed in the next section.)

### 3.1 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 $2 * m * R$, and
3. the delay is at most $2 * m$.

□

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 $m * R$ and the delay becomes at most m.

The limiter is a discrete version of the FIFO operator in [2]. A continuous version of theorem 3 can be inferred from the results in [2].

### 3.2 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 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.

At every instant i, where $i \bmod m \neq 0$, $s_i = 0$ and the input $r_i$ is added to the buffer, $b_i = b_{i-1} + r_i$. At every instant i, where $i \bmod m = 0$,

$$s_i = b_{i-1} = \sum_{j=i-m}^{i-1} r_j$$

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 $m * R$, and
3. the delay is at most m.

□

Theorem 4 can be extended slightly. In particular, it can be shown that if the input flow of an m-compactor is (n,R)-smooth, then its output flow is (n, $(\lceil m/n \rceil + 1) * R$)-smooth and its buffer capacity is at most $(\lceil m/n \rceil + 1) * n * R$.

### 3.3 Expanders

Let m be a positive integer. An m-expander is a flow operator that nondeterministically chooses a value to output at each instant. The chosen value is restricted to satisfy the flow conservation property. To guarantee a bounded delay, the entire buffer content is transferred to the output flow every m instants. Formally, the operation of an m-compactor is defined as follows

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

where r is the input flow, s is the output flow, b is the buffer flow, and each $X_i$ is a real value chosen nondeterministically from 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 $m * R$, and
3. the delay is at most $m - 1$.

□

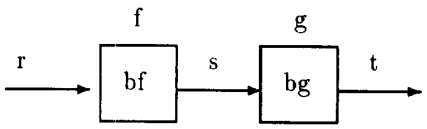
Theorem 5 can be extended slightly. In particular, it can be shown that if the input flow of an m-expander is (n,R)-smooth, then its output flow is $(n,(\lceil (m-1)/n \rceil + 1) * R)$-smooth and its buffer capacity is at most $\lceil (m-1)/n \rceil * n * R$.

## 4 Linear Networks

A finite sequence of flow operators $\langle f_0, f_1, \ldots, f_{n-1} \rangle$ is a *linear network* iff for each i, $i < n-1$, 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 $c_{-1}, c_0, c_1, \ldots$ such that $c_{-1} = 0$ and for every $i = 0, 1, 2, \ldots$,

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

where 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 c be the buffer flow of a linear network $\langle f, g \rangle$ with operators f and g.

1. If the buffer flows of f and g are bf and bg, then for every $i = -1, 0, 1, 2, \ldots$

   $$c_i = bf_i + bg_i$$

2. If the buffer capacities of f and g are Bf and Bg, and the buffer capacity of the network $\langle f, g \rangle$ is C, then

   $$C \leq Bf + Bg$$

3. If the delays of f and g are Df and Dg, and the delay of the network $\langle f, g \rangle$ is D, then

   $$D \leq Df + Dg$$

□

Part 1 in this theorem states that the buffer contents of a network at each instant equals the sum of the buffer contents of the network operators at that instant. Part 2 implies that if each operator in a network has a bounded buffer, then the network has a bounded buffer. Part 3 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.

## 5 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 with a maximum rate of R packets per second 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,R) message along this simple path. Third, when a computer k on the path from i to j receives the reserve(i,j,R) message, it checks whether it can support the transmission of R data packets per second from i to j. If the answer is no, computer k returns a reject(i,j) message along the path towards i. If the answer is yes, computer k forwards the reserve(i,j,R) message to the next computer on the path from i to j. Fourth, when computer j receives the reserve(i,j,R) message, it returns an accept(i,j) along the reverse path towards 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.

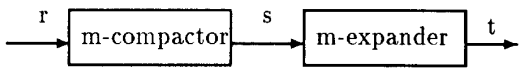
All rate-reservation protocols follow the above procedure in establishing connections. They differ, however, in the rules for receiving, buffering, and later forwarding the data packets along the path from i to j, once a connection from i to j has been established.

In this section, we discuss four rate-reservation protocols: stop-and-go, hierarchical round-robin, 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 four protocols.

### 5.1 Stop-and-Go

In this protocol[6], time is partitioned into consecutive periods called frames. Each computer on the path from i to j buffers all the data packets that it receives in one frame, then forwards 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 also (m,R)-smooth (by Theorem 1). Therefore, flow t is (m,R)-smooth (by Theorem 5). From Theorems 4, 5, and 6, we conclude that the buffer capacity of the network is at most $2 * m * R$ and its delay is at most $(2 * m) - 1$.

If a number n of these linear networks are connected in one big linear network (to represent a path of n computers from i to j), then the buffer capacity along the path is at most $2 * m * n * R$ and the delay is at most $(2 * m - 1) * n$. 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 ⟨m-compactor, m-expander⟩ pairs, one can also view it as a single m-compactor followed by a series of ⟨m-expander, m-compactor⟩ pairs ending with an m-expander as illustrated in Figure 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 ⟨f, g⟩ is the same as the input flow of h, then the output flow of ⟨f, g⟩ is the same as the output flow of h. □

From Theorems 4 and 7, the delay of an (m,R)-smooth flow through the ⟨m-expander, m-compactor⟩ 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 $m - 1$, and at each

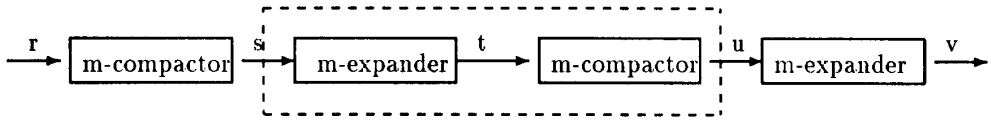



Figure 1: Alternative view of a stop-and-go linear network.

⟨m-expander, m-compactor⟩ pair is at most m. Hence, by Theorem 6, the total delay in the network is at most $(n-1)*m+2*m-1$.

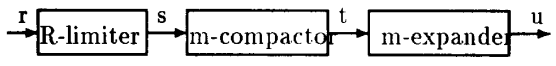
Notice that we cannot obtain a similarly reduced bound on the buffer capacity of the network, since physically, an m-expander is implemented in a different computer than its succeeding m-compactor. Thus, their buffer spaces cannot be combined.

The upper bounds which we obtained on the buffer capacity ($2*m*n*R$) and the delay ($(n-1)*m+2*m-1$) of the stop-and-go protocol match those that were reported in [6].

### 5.2 Hierarchical Round-Robin

In this protocol[7], 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 forwards 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 $m*R$ 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 $m*R$ 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, 3, 4 and 5, the output flow of the network is also (m,R)-smooth. Moreover, by these same theorems and Theorem 6, the required buffer capacity of the network is at most $4*m*R$ and its delay is at most $4*m$.

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 $4*m*n*R$ and the delay along the connection is at most $4*m*n$.

It is reported in [7] that an upper bound on the delay of hierarchical round-robin in one computer is $2*m$, compared with $4*m$ in our analysis. However, we managed to prove that the $2*m$ upper bound is incorrect by exhibiting a scenario that introduces a delay of $3*m$. This scenario illustrates that our upper bound of $4*m$ on the delay is not very far from the actual delay.

### 5.3 Fair Queueing and Virtual Clock

In fair queueing[4][8] and virtual clock[10] protocols, 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 connec-

tion.

The details for computing timestamps for incoming packets are different for the two protocols of fair queueing and virtual clock; however, the same principle for computing the timestamps applies for the two protocols. When a computer k receives a packet along a connection with an average rate R (packets per second), the packet is assigned a timestamp equals 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 (packets per second). 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 timestamp.

This behavior can be represented by a new flow operator called R-filter, where R is a positive real number. An R-filter is a flow operator that nondeterministically choses a value to output at each instant. The nondeterminism is bounded in two ways: the flow conservation property is satisfied, and the output flow has an average rate of at least R. In other words, if an R-limiter and an R-filter have the same input flow, then the sum of any prefix of the output flow of the R-filter is no less than the sum of the same length prefix 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.

Let bl and bf denote the buffer flows of an R-limiter and R-filter, respectively. The operation of an R-filter is as follows

$$s_i = \max((bf_{i-1} + r_i) * X_i,\ \ bf_{i-1} + r_i - bl_i)$$

where r is the input flow to both the R-filter and the R-limiter, s is the output flow of the R-filter, and each $X_i$ is a real value chosen nondeterministically from the closed interval $[0, 1]$.

The first term of the max operator nondeterministically chooses an output value that satisfies the flow conservation property. The second term ensures that $bf_i$ is at most $bl_i$ as follows.

From the definition of the max operator, we have that

$$s_i \geq bf_{i-1} + r_i - bl_i.$$

Rearranging the above we obtain

$$bf_{i-1} + r_i - s_i \leq bl_i.$$

By the definition of a buffer flow, we conclude that

$$bf_i \leq bl_i.$$

Notice that although the second term of the max operator can be negative, the first term is always at least zero, and thus $s_i$ is also at least zero. Also, since $bl_i \geq 0$, the flow conservation property is satisfied, since neither term of the max operator can yield a value larger than $bf_{i-1} + r_i$.

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

1. the buffer capacity is at most $2 * m * R$, and
2. the delay is at most $2 * m$.

□

**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 $2 * m$.

It was shown in [8] that a flow traversing a linear network of computers using fair queueing cannot be delayed more than the maximum delay of the same flow traversing a single computer. The proof of this property was restricted to flows that met a specific burstiness constraint. In Theorem 9, we have shown that this property is insensitive to the characteristics of the input flow, and thus holds for any input flow. Theorem 8 is added simply to show that the delay and buffer requirements in the linear network do have an upper bound. Moreover, because both fair queueing and virtual clock are instances of a filter, our analysis yields the new result that virtual clock also exhibits the same property.

## 6 Concluding Remarks

In this paper, we have presented a theory of flows and showed how to use this theory in verifying rate-reservation protocols. Flow theory is analogous to queueing theory in the sense that it provides a framework for the analysis of delays and buffer requirements in a network. These theories differ, however, in that flow theory's objective is to obtain upper bounds on

these values in a real-time network, while queueing theory's objective is to obtain the average of these values in a probabilistic network.

Flow theory is both rich and effective. The richness of this theory is demonstrated by the many theorems that we were able to prove from a small number of concepts using relatively simple proofs. The effectiveness of the theory is demonstrated by our ability to verify many existing rate-reservation protocols using the theory. The richness and effectiveness of this theory make it a prime candidate for designing future rate-reservation protocols.

In using flow theory to study rate-reservation protocols, we produced a number of surprising results. First, we have matched the known upper bounds [6] on the buffer and delay requirements in the stop-and-go protocol. Second, we have discovered that the rough analysis of the hierarchical round-robin protocol in [7] has yielded an erroneously optimistic delay bound. By contrast, our analysis of the same protocol yields a little pessimistic upper bounds on the buffer and delay requirements. Third, our analysis of the fair queueing protocol has strengthened earlier results [8] by showing that these results are insensitive to the characteristics of the input flow. Fourth, we have showed that the same analysis can be applied to the virtual clock protocol.

We have developed our flow theory around two types of flows, smooth and uniform. Both these types are necessary for the development of the theory for the following reasons. On one hand, the expander operator preserves the smoothness of a flow but does not preserve its uniformity. On the other hand, the need for uniform flows arises when cyclic networks are introduced [1]. Therefore, in order to include cyclic networks and the important expander operator in one theory, both smooth and uniform flows have to be included in the theory.

A different characterization of flows named $(\sigma, \rho)$ is given in [2], where $\sigma$ is a measure of the flow burstiness, and $\rho$ is the flow rate. We chose not to adopt this property into our framework since, if the input to an m-compactor or m-expander is $(\sigma, \rho)$, then the output is $(\sigma + \rho * \mathrm{m}, \rho)$. Thus, in a linear network of these operators, the burstiness parameter increases with each hop in the network, which leads to higher estimates of buffer requirements and delay bounds for some rate-reservation protocols.

Several directions are possible for future work on flow theory. First, the theory can be applyed in designing new rate-reservation protocols. Second, non-linear acyclic networks and cyclic networks may be included in the theory. Third, the theory may be enhanced by relaxing the flow conservation property. This would allow the modeling of broadcast capabilities and the effect of data compression/decompression algorithms applied to a flow as it traverses a network. We examine this issues in a future journal paper[1].

## References


[1] J.A. Cobb, M.G. Gouda, "Flow Theory", in preparation.

[2] R.L. Cruz, "A Calculus for Network Delay, Part I: Network Elements in Isolation", *IEEE Transactions on Information Theory*, Vol. 37, No. 1, January 1991.

[3] R.L. Cruz, "A Calculus for Network Delay, Part II: Network Analysis", *IEEE Transactions on Information Theory*, Vol. 37, No. 1, January 1991.

[4] A. Demers, S. Keshav, S. Shenkar, "Analysis and Simulation of a Fair Queueing Algorithm", *Proceedings of the ACM SIGCOMM*, 1989.

[5] T. Faber, L. H. Landweber, and A. Mukherjee, "Dynamic Time Windows: Packet Admission Control With Feedback", *Procceedings of the ACM SIGCOMM*, 1992.

[6] S. J. Golestani, "A Stop-and-Go Queueing Framework for Congestion Management", *Proceedings of the ACM SIGCOMM*, 1990.

[7] C. R. Kalmanek, H. Kanakia, and S. Keshav, "Rate Controlled Servers for Very High-Speed Networks", *Proceedings of the IEEE GLOBCOMM*, 1990.

[8] A. K. J. Parekh, "A generalized Processor Sharing Approach to Flow Control in Integrated Services Networks", Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Technical Report Number LIDS-TH-2089.

[9] H. Zhang and Srinivasan Keshav, "Comparison of Rate-Based Service Disciplines", *Proceedings of the ACM SIGCOMM*, 1991.

[10] L. Zhang, "Virtual Clock: A New Traffic Control Algorithm for Packet-Switched Networks", *ACM Transactions on Computer Systems*, Vol. 9, No. 2, May 1991.