Delay Guarantee of Virtual Clock Server

Geoffrey G. Xie and Simon S. Lam, *Fellow, IEEE*

***Abstract*—We present and prove a delay guarantee for the Virtual Clock service discipline. The guarantee has several desirable properties, including the following *firewall* property: The guarantee to a flow is unaffected by the behavior of other flows sharing the same server. There is no assumption that sources are flow controlled or well behaved. In this paper, we first introduce and define the concept of an active flow. The delay guarantee is then formally stated as a theorem. We show how to obtain delay bounds from the delay guarantee of a single server for different specifications. Derivations of end-to-end delay bounds for various networks and source specifications are presented elsewhere.**

***Index Terms*— Virtual clock, rate-based service discipline, priority queue, throughput guarantee, delay guarantee, packet switching.**

I. Introduction

IN A PACKET switching network, each communication channel is statistically shared among many traffic flows that belong to different end-to-end sessions. Typically, packets are queued and scheduled for transmission on a first come, first served (FCFS) basis. The service received by a particular flow is necessarily impacted by the behavior of other traffic flows that share the same queue. For this reason, it is very difficult for packet switching networks that employ FCFS scheduling to offer the following kinds of guarantees: 1) a session gets a specified throughput rate, 2) end-to-end delays of packets in the session are bounded, and 3) the maximum difference over a set of delays (jitter) is bounded.

Such throughput, delay, and jitter guarantees, however, are precisely the ones needed to support various multimedia applications. To provide some or all of these guarantees without giving up the flexibility of statistical multiplexing, a variety of rate-based service disciplines have been proposed to be used instead of FCFS [13].

In particular, the Virtual Clock service discipline [14], [15], inspired by time division multiplexing (TDM), provides firewalls between individual traffic flows in the following sense: Each traffic flow is allocated a reserved rate of throughput, and such throughput guarantee is independent of the behavior of other traffic flows sharing the same server. Specifically, an aggressive traffic source, one that generates traffic at a rate higher than its reserved rate, may take up idle server capacity, but it cannot affect the throughput rates guaranteed to other flows.

Manuscript received November 9, 1994; revised May 30, 1995; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor L. Zhang. Research supported in part by National Science Foundation Grants NCR-9004464 and NCR-9506048, and in part by the NSA INFOSEC University Research Program. This paper was presented at the 9th IEEE Workshop on Computer Communications, Marathon, Florida, October 1994.

The authors are with the Department of Computer Sciences, the University of Texas, Austin, TX 78712-1188 USA.

IEEE Log Number 9415966.

With TDM, allocating the capacity of a server is constrained by frame and slot sizes. With Virtual Clock, there is no such constraint, i.e., any fraction of a server's capacity can be allocated to a traffic flow. Thus, compared to TDM, Virtual Clock offers the flexibility to support diverse throughput requirements of different applications, in addition to the flexibility of statistical multiplexing.

No delay guarantee was presented for Virtual Clock in [14], [15]. In a recent survey of rate-based service disciplines, Virtual Clock was classified among those that do not provide a delay guarantee [13]. Indeed, Virtual Clock by itself, without any assumption of source control, does not provide a delay bound in the usual sense (namely, a bound on the difference between the departure time and arrival time of each packet). In this paper, we introduce the concept of a delay guarantee to a packet based upon its virtual clock value. The guarantee is conditional and, as such, is actually more useful than a delay bound.

In Section II, we first introduce the virtual clock of a packet flow as a concept that is independent of the Virtual Clock (VC) service discipline.[1] The model of a VC server, statistically shared by a set of traffic flows, is presented. The meaning of an active flow is defined. The delay guarantee provided to packets in a flow is then presented, and formally stated as Theorem 1. Properties of the delay guarantee are discussed in Section III. In Section IV, we show how to derive delay bounds from the delay guarantee for different source specifications. In Section V, we describe a generalization of the delay guarantee, and present a modification of the VC algorithm. In Section VI, we discuss related work, including both extensions of the result in this paper and comparisons with the results of other authors.

II. Delay Guarantee

Consider a number of traffic sources and a service facility. Each source generates a sequence of packets, called a *flow*. Prior to generating packets, the source of flow f requests for a reserved rate from the facility. Let $r(f)$ be the reserved rate, in bits/s, allocated to flow f. Generally, different sources negotiate for different rates depending upon their needs and how much they are willing to pay.

For an arbitrary packet p, its length in bits is denoted by $l(p)$, its arrival time to the facility by $A(p)$ (≥ 0), and its departure time from the facility upon service completion by $L(p)$. The delay of packet p is $L(p) - A(p)$. For all flows, the length of a packet varies from a minimum of $l_{\min}$ to a maximum of $l_{\max}$ bits. The concept of virtual clock is introduced next.

[1] We denote the concept by *virtual clock*, and the service discipline by *Virtual Clock* or *VC*.

Let priority(f) denote the virtual clock of flow f. It can be implemented as a variable, which is zero initially and updated as follows, whenever a flow f packet, say p, arrives to the facility [15]

$$\text{priority}(f) := \max\{\text{priority}(f), A(p)\} + \frac{l(p)}{r(f)}. \qquad (1)$$

The new value of priority(f) above is assigned to packet p as its virtual clock value, denoted by $P(p)$. Thus, the virtual clock of flow f is a variable that holds the virtual clock value of the *most recent arrival* of f. Note that the virtual clock values of flow f are determined by the sequence of packet arrival times of f, and are independent of the internal structure and design of the service facility. (In particular, we have not yet specified whether the facility is a single server or a network of servers).

A *VC server* is a priority server that uses the virtual clock value of a packet as a priority. Specifically, whenever the server is ready to serve a new packet, the packet in queue with the smallest virtual clock value is selected for service. Furthermore, the service discipline is work conserving and nonpreemptive.

Consider now a service facility, consisting of a VC server and its queue, for a set F of traffic flows. In what follows, we use the term *system* to refer to the VC server and its queue.

Definition 1: A flow f is active at time t if and only if the value of priority(f) at time t satisfies

$$\text{priority}(f) > t. \qquad (2)$$

Simply stated, a flow is active whenever its virtual clock is running faster than real time. To understand the intuition behind Definition 1, consider a hypothetical server with capacity $r(f)$ bits/s, which is dedicated to flow f (with the same packet arrival times and lengths as those for flow f at the VC server). It is easy to see that the departure time of each packet at the hypothetical server is equal to the virtual clock value of the same packet at the VC server. Therefore, at time t, the condition priority(f) $> t$ holds at the VC server if and only if the hypothetical server for flow f is busy. Note that whenever priority(f) $< t$, the hypothetical server is idle, indicating that flow f is not generating arrivals fast enough to fully utilize its allocated rate at the VC server.[2]

Definition 2: Let C denote the capacity, in bits/s, of a VC server. The server's capacity is not exceeded at time t if and only if the following condition holds:

$$\sum_{f \in a(t)} r(f) \leq C \qquad (3)$$

where $a(t)$, a subset of F, is the set of flows that are active at time t.

Theorem 1: If the capacity of a VC server has not been exceeded for a non-zero duration since the start of a busy period, then the following holds for every packet p served during the busy period:

$$L(p) \leq P(p) + \frac{l_{\max}}{C}. \qquad (4)$$

A proof of Theorem 1 is presented in the Appendix. In practice, a network specifies a maximum size for all packets, which can be used as $l_{\max}$ in (4). However, it is clear from the proof that $l_{\max}$ can be smaller. Specifically, it is sufficient for $l_{\max}$ to be the size of the largest packet among those that have been served in the busy period and do not belong to the same flow as p.

Intuitively, the reason for the term $l_{\max}/C$ in (4) is twofold: 1) for a VC server, it is possible for a packet to be scheduled later than another packet that has a larger virtual clock value (lower priority), and 2) preemption is not allowed.

Our proof is by contradication. The key step is as follows: Consider the sequence of packets served during a busy period. In the proof, each packet is identified by its index in the sequence, 1, 2, $\cdots$, in order of service. (Note: We use i, j, and k for such indices.) For example, $l(i)$ denotes the size of the ith packet served in the busy period. Note that the busy period begins at $A(1)$, arrival time of the first packet in the sequence.

Suppose during the busy period, the kth packet served departs at time $L(k)$ such that

$$L(k) > P(k) + \frac{l_{\max}}{C}.$$

We show that there exists a time $t^* \in [A(1), L(k-1)]$ such that the server's capacity is exceeded at t^* for a non-zero duration, i.e., for $t^* \leq t < t^* + \Delta$, where $\Delta > 0$, the following holds

$$\sum_{f \in a(t)} r(f) > C$$

which contradicts the hypothesis of Theorem 1 (see Appendix).

Theorem 1 is proved without any assumption that sources are flow-controlled or well behaved. Note that the theorem's hypothesis does not depend upon the *actual rates* of flows in F. While the source of a flow, say f, has a reserved rate of $r(f)$, the source can misbehave, i.e., generate traffic at a rate much larger than $r(f)$.[3] The delay guarantee in (4) holds even if the flows in F generate packets at an aggregate rate larger than C.

The proof of Theorem 1 is valid for arbitrary arrival times and packet lengths. Note that, over time, each flow alternates between being active and inactive. The set of active flows, $a(t)$, a subset of F, changes dynamically with time. At any time, the server can determine $a(t)$ because it can determine whether or not a flow is active by comparing its virtual clock value with the local clock value (applying Definition 1). The server capacity is exceeded when sum of the reserved rates of *active* flows is larger than C. This condition can always be ensured by a VC server because the allocation of reserved rates is under the server's control.

[2]Without the context of a hypothetical dedicated server, Definition 1 is not very intuitive. During a busy period of the VC server, flow f may be active even when there is no flow f packet in the system. On the other hand, whenever real time t surpasses priority(f), flow f becomes inactive even when there are flow f packets in the system (waiting behind packets of other flows).

[3]It is assumed that each flow is allocated its own buffers, so that if a source misbehaves, it will fill up its own buffers but not those of other flows.

A. *How to Ensure Server Capacity Is Not Exceeded*

There are two obvious ways for a VC server to ensure that its capacity is not exceeded without *a priori* knowledge of source arrival characteristics. The first is by static assignment, namely, every flow is statically assigned a reserved rate, and requiring that sum of the reserved rates not to exceed C, for the entire set of flows. This ensures *a priori* that the channel capacity cannot be exceeded because for all time t

$$\sum_{f \in a(t)} r(f) \leq \sum_{f \in F} r(f) \leq C.$$

The second is by per-session demand assignment, namely, by requiring each source to request for a session and be allocated a reserved rate by the server before the source can begin generating traffic. Subsequently, upon session termination, the source stops generating traffic and the reserved rate is deallocated. Let $D(t)$ denote the set of sessions at time t with allocated reserved rates. The server ensures that the sum of the reserved rates for the set of flows in $D(t)$ does not exceed C by refusing new requests if necessary (admission control). Since a source generates traffic only when it is allocated a reserved rate, $a(t)$ is a subset of $D(t)$ at any time t. The server ensures *a priori* that the channel capacity cannot be exceeded because for all time t

$$\sum_{f \in a(t)} r(f) \leq \sum_{f \in D(t)} r(f) \leq C.$$

If statistical guarantees are acceptable [3], [13], a VC server can exploit the fact that flows are not active all the time and overcommit its capacity. The design of admission control techniques to do so requires *a priori* knowledge of source traffic statistics. Such techniques should be designed in the context of a specific packet network architecture and traffic model, and are beyond the scope of this paper.

III. Properties of Delay Guarantee

There is no assumption that sources are flow controlled or well behaved in proving Theorem 1, because the delay guarantee is not a bound on $L(p) - A(p)$. A delay guarantee of the form, $L(p) \leq P(p) + \beta$ where β is a constant, is a *conditional guarantee*. Note that the deadline for a packet's departure is measured from the packet's virtual clock value, rather than its actual arrival time. Specifically, if a packet arrives early, its deadline is bounded from its expected arrival time based upon the reserved rate of its flow. Thus, if a source generates traffic faster than its reserved rate, its packets arrive earlier than expected and may incur large delays.

In designing a packet switch, on-time packet arrivals are preferable to early packet arrivals. If a flow's packets can arrive very early, the flow is effectively more bursty, and requires more buffer space. (Thus rewarding arrivals that are too early with prompt service is counterproductive.) On the other hand, packets that arrive late do not get better service from a VC server, i.e., the VC service discipline is designed to encourage sources to generate on-time arrivals.

Like the throughput guarantee of a VC server, the delay guarantee in Theorem 1 has a desirable firewall property. This property is obvious from examining the inequality in (4), where $P(p)$ on the right hand side, computed using (1), is a function of the packet arrival times of p's flow. A misbehaving source would degrade the delay performance of its own packets, but would not interfere with performance guarantees offered to other flows.

IV. Delay Bounds

Consider a flow f with reserved rate $r(f)$. Its packets are indexed by $n = 1, 2, \cdots$, in order of arrival. If its source is known to be well behaved, either voluntarily or by source control, such that for packet n in the flow, $P(n) - A(n)$ is bounded by a constant, the delay of packet n is bounded.

Thus the goal of source control is to upper bound $P(n) - A(n)$ for all n, that is, the extent to which the virtual clock of flow f is allowed to run ahead of real time. One example of source control is to ensure the interarrival time between two consecutive packets in the flow is lower bounded, i.e., $A(n+1) - A(n)$ is greater than or equal to $l(n)/r(f)$ for all n. By induction on n, it is easy to show that $P(n) = A(n) + (l(n)/r(f))$. Therefore, a VC server provides the following delay bound to every packet n in the flow

$$L(n) \leq A(n) + \frac{l(n)}{r(f)} + \frac{l_{\max}}{C}. \tag{5}$$

The above observation was first used in [7] to obtain a tight upper bound on end-to-end delays.

With the delay guarantee in (4), different delay bounds can be derived for different methods of source control. As another example, if the source is (σ, ρ) leaky bucket controlled, then $P(n) - A(n)$ is bounded by σ/ρ for all n [6].

V. Generalization and Algorithm Modification

Theorem 1 can be made more general by weakening its hypothesis, specifically, by reducing the duration of time when a flow is defined to be active. The proof of Theorem 1 in the Appendix is for an arbitrary busy period. From the definition of the Rate$(\cdot)$ function in (24), it is clear that Theorem 1 holds as long as the server capacity is not exceeded by sum of the reserved rates of flows that have had a packet served during the busy period. Therefore, Definition 1 can be modified as follows to reduce the time duration when a flow is defined to be active.

Definition 1': A flow f is active at time t if and only if: i) the system is not empty, ii) a flow f packet has arrived in the current busy period, and iii) the value of priority(f) at time t satisfies[4]

$$\text{priority }(f) > t.$$

We can achieve the same generalization of Theorem 1, without changing the active flow definition (Definition 1), by resetting the virtual clock of each flow to zero at the end of a busy period. Specifically, whenever a packet departs, if the

[4]Theorem 1 holds even if ii) is strengthened to the following: a flow f packet has been served in the current busy period.

system becomes empty, the following reset is performed, for all f in F:

$$\text{priority}\ (f) := 0.$$

Note that the above reset causes flow f to immediately become inactive at the end of a busy period. Subsequently, flow f becomes active again when its source generates a new arrival. The time periods during which flow f is defined to be active are identical to those given by Definition 1′ (without resetting virtual clocks). Thus resetting virtual clocks does not affect Theorem 1.

There is, however, a side effect introduced by resetting virtual clocks at the end of busy periods. Specifically, the VC algorithm has been modified in a nontrivial way. For a flow that has heretofore used up more server capacity than its reserved rate, resetting its virtual clock to zero at the end of a busy period means that its "debt" has been forgiven. Such a side effect was actually considered desirable in [1], where the idea of virtual clock reset was discussed. For a slightly different model, we described how to reset the virtual clock of an active flow whenever there is no packet in the system belonging to another flow [7].

VI. Related Work

A. Extensions

The first application of Theorem 1 was to derive end-to-end delay bounds for a class of packet switching networks, called Burst Scheduling networks. In these networks, each guaranteed flow is modeled as a sequence of bursts (each of which is a sequence of packets). The concept of bursts was introduced to specify two types of jitter bounds; over the delays of packets in a burst, and over the delays of bursts in a flow. Furthermore, a flow can be partitioned into intervals (bursts) that have substantially different average rates. The first packet of a burst carries information on the size and average rate of the burst. Switches are designed to process flows efficiently in bursts. Tight upper bounds on the end-to-end delays of packets and bursts were derived and presented [7]–[9].

The idea of a conditional guarantee to a packet based upon the packet's virtual clock value was subsequently extended to an end-to-end network path [6]. It was shown that such a delay guarantee can be used for deriving upper bounds on end-to-end delays for different source specifications, and for a class of scheduling algorithms including not only Virtual Clock, but also PGPS [10] and SCFQ [5].

B. Comparisons

The Delay–EDD service discipline was designed to offer a delay bound to packet arrivals [3], [13]. When a flow f packet arrives to a Delay–EDD server, it must depart within $d(f)$ seconds of its arrival time, where $d(f)$ is a performance parameter that can be specified. Note that the delay bound offered by a VC server to flow f is coupled to the reserved rate $r(f)$ allocated to the flow, that is, to get a smaller delay bound, the source has to request for a larger reserved rate (and presumably pay more for it). Delay–EDD avoids such direct coupling. Instead, it requires a schedulability test prior to session establishment. The result of the test, i.e., whether or not a new session can be established depends upon the traffic patterns of all flows sharing the server. Furthermore, Delay–EDD requires a stronger source traffic specification, that includes a minimum packet interarrival time in addition to a reserved rate (average packet interarrival time).

The packet-by-packet generalized processor sharing (PGPS) service discipline, also known as weighted fair queueing [2], was described and analyzed by Parekh and Gallager [10], [11]. It has been shown that a network path of VC servers has the same delay upper bound as PGPS servers [6]. Comparing VC and PGPS, note that if a VC server's capacity is not fully allocated to flows, the remaining unallocated capacity is not shared fairly as defined in [10]. However, a VC server does ensure that each flow receives its allocated throughput rate. Moreover, VC is a much more efficient algorithm than PGPS, i.e., the computation of virtual clock values for VC scheduling is much simpler than the computation of virtual time finishing times for PGPS scheduling.

The delay guarantee in (4) was independently discovered and proved by Figueira and Pasquale [4] and by us [12]. We next discuss some of the differences between the two contributions. First, one significant difference is our active flow definition, missing in [4]. The condition for their delay guarantee is the sum of the reserved rates of all flows sharing the server does not exceed the server capacity [4]. Theorem 1 herein requires that the reserved rates of only those flows that are *active* at time t be summed to determine whether or not the server capacity is exceeded at time t (see Definitions 1 and 2). Second, our proof approach, presented in the Appendix, is very different from the one presented in [4]. Third, we have also shown that Theorem 1 holds for a modified VC algorithm such that virtual clocks are reset to zero at the end of server busy periods.

VII. Conclusions

We discovered and proved a delay guarantee for a VC server with a firewall property, namely, the delay guarantee to a flow is independent of the behavior of other flows sharing the same server. With this property, the impact of source-controller malfunctioning is limited, a significant advantage not found in FIFO and other static priority service disciplines.

With both delay and throughput guarantees, and a relatively simple implementation, Virtual Clock is an attractive algorithm for high speed networks. We have chosen Virtual Clock as the basis of a packet network architecture, called Burst Scheduling, for switching multimedia traffic [7].

Our concept of an active flow in Definition 1 (also Definition 1′) is novel. We are investigating techniques to exploit this concept to increase the number of flows that can share a VC server providing statistical guarantees. The concept of a delay guarantee based upon the virtual clock value of a packet, rather than its actual arrival time, is also novel. It allows the derivation of different delay bounds for different source specifications.

APPENDIX

We first state and prove a lemma, which will be used in our proof of Theorem 1.

Lemma 1: Let t' be the time when a flow f packet arrives at a VC server. If f is not active at time $t > t'$, where t and t' are in the same server busy period, then the following holds:

$$(t - t') \geq \frac{\sum_{p \in N(t',t)} l(p)}{r(f)} \tag{6}$$

where $N(t',t)$ is the set of flow f packets that arrive during $[t',t]$.

Proof: By assumption, time t is within a busy period. Since f is not active at time t and the system is not empty, from Definition 1, the value of priority(f) satisfies

$$\text{priority}(f) \leq t. \tag{7}$$

Let p_0 be the flow f packet that arrives at time t'. From the Virtual Clock algorithm in (1), we have

$$P(p_0) \geq t' + \frac{l(p_0)}{r(f)}. \tag{8}$$

Applying (1) repeatedly, we have, at time t, the value of priority(f) satisfies

$$\text{priority}(f) \geq t' + \frac{\sum_{p \in N(t',t)} l(p)}{r(f)}. \tag{9}$$

Combining (7) and (9), we have

$$t \geq t' + \frac{\sum_{p \in N(t',t)} l(p)}{r(f)}. \tag{10}$$

Therefore, (6) holds

$$(t - t') \geq \frac{\sum_{p \in N(t',t)} l(p)}{r(f)}. \qquad \square$$

A. Proof of Theorem 1

Consider the sequence of packets served during a busy period. Each packet is identified by its index in the sequence, 1, 2, $\cdots$, in order of service. (Notation: We use i, j, and k for such indices.) For example, $l(i)$ denotes the size of the ith packet served in the busy period. Note that the busy period begins at $A(1)$, arrival time of the first packet in the sequence.

We will carry out a proof by contradiction. Specifically, we assume that during the busy period, the kth packet served departs at time $L(k)$ such that

$$L(k) > P(k) + \frac{l_{\max}}{C}. \tag{11}$$

Then we will show, given (11), there exists a time $t^* \in [A(1), L(k-1)]$ such that the server's capacity is exceeded at t^* for a non-zero duration, i.e., for $t^* \leq t < t^* + \Delta$, where $\Delta > 0$, the following holds:

$$\sum_{f \in a(t)} r(f) > C. \tag{12}$$

From (11), we have

$$P(k) < L(k) - \frac{l_{\max}}{C}. \tag{13}$$

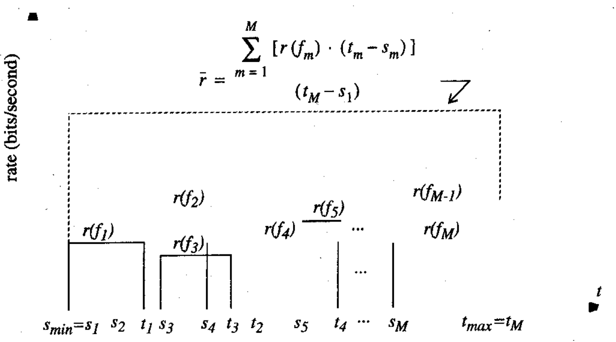

$\bar{r} = \frac{\sum_{m=1}^{M} [r(f_m) \cdot (t_m - s_m)]}{(t_M - s_1)}$
rate (bits/second)
$r(f_1)$
$r(f_2)$
$r(f_3)$
$r(f_4)$
$r(f_5)$
$r(f_{M-1})$
$r(f_M)$
$s_{min}=s_1$ s_2 t_1 s_3 s_4 t_3 t_2 s_5 t_4 $\cdots$ s_M $t_{max}=t_M$
t
system time (seconds)

Fig. 1. Illustration for Theorem 1.

There are two possible cases (that are mutually exclusive and collectively exhaustive):

Case I: There exists j, $1 < j \leq k$, such that for $i = j, \cdots, k$

$$P(i) < L(k) - \frac{l_{\max}}{C} \tag{14}$$

$$P(j-1) \geq L(k) - \frac{l_{\max}}{C}. \tag{15}$$

Note that packet $j-1$ has a larger virtual clock value than packets in $\{j, \cdots, k\}$ but is served earlier.[5]

When a packet arrives, its flow is either already active or becomes active at the instant of the packet's arrival (by Definition 1). Thus, every packet arrives in a time interval during which its flow is continuously active, to be called an *active period.* Let M denote the number of active periods during which packets in $\{j, \cdots, k\}$ arrive, where $1 \leq M \leq k-j+1$. These active periods are indexed by $m = 1, 2, \cdots, M$. For active period m in this set, define the following:

f_m	Active period m is for flow f_m.
t'_m	Time instant when active period m ends.
t_m	$t_m = \min\{t'_m, L(k) - l_{\max}/C\}$.
N_m	Set of f_m packets in $\{j, \cdots, k\}$ that arrive during active period m.
s_m	Earliest arrival time among packets in N_m.

The active periods are illustrated in Fig. 1. Note that two or more active periods in the set may be for the same flow (that is, the f_m's are not necessarily distinct). Also, by definition, $N_1, N_2, \cdots, N_M$ constitute a partition on $\{j, \cdots, k\}$.

Because of (14), the arrival time of every packet in $\{j, \cdots, k\}$ is less than $L(k) - l_{\max}/C$. Hence, every packet in $\{j, \cdots, k\}$ arriving during active period m has an arrival time less than t_m.

For every active period m, one of the following two cases applies:

- $t_m < L(k) - l_{\max}/C$: Recall that $N(s_m, t_m)$ denotes the number of flow f_m packets that arrive during the

[5] Also, from (13) and (15), note that $P(k) < P(j-1)$ which implies packets $j-1$ and k do not belong to the same flow.

time interval $[s_m, t_m]$. Noting that $N_m \subseteq N(s_m, t_m)$, and applying Lemma 1, we have

$$(t_m - s_m) \geq \frac{\sum_{p \in N(s_m,t_m)} l(p)}{r(f_m)} \geq \frac{\sum_{p \in N_m} l(p)}{r(f_m)} \quad (16)$$

$$\sum_{p \in N_m} l(p) \leq r(f_m)(t_m - s_m). \quad (17)$$

- $t_m = L(k) - l_{\max}/C$: Let last$_m$ be the packet in N_m that is served last. Apply the Virtual Clock algorithm in (1) repeatedly, and we have

$$P(\text{last}_m) \geq s_m + \sum_{p \in N_m} \frac{l(p)}{r(f_m)}. \quad (18)$$

From (14), we have

$$P(\text{last}_m) < L(k) - \frac{l_{\max}}{C} = t_m. \quad (19)$$

Combining (18) and (19), we have

$$t_m > s_m + \frac{\sum_{p \in N_m} l(p)}{r(f_m)} \quad (20)$$

$$\sum_{p \in N_m} l(p) < r(f_m)(t_m - s_m). \quad (21)$$

Define

$$s_{\min} = \min_{1 \leq m \leq M}\{s_m\} \quad (22)$$

$$t_{\max} = \max_{1 \leq m \leq M}\{t_m\}. \quad (23)$$

Next let us consider Fig. 1. In the figure, without loss of generality, we have assumed $s_{\min} = s_1$ and $t_{\max} = t_M$.

Let $I(x)$ be a function that has a value of one if condition x is true, and zero, otherwise.

Define

$$\text{Rate}(t) = \sum_{m=1}^{M} [I(s_m \leq t < t_m) r(f_m)] \quad (24)$$

$$\overline{r} = \frac{\sum_{m=1}^{M} [r(f_m)(t_m - s_m)]}{t_{\max} - s_{\min}}. \quad (25)$$

In Fig. 1, each possible pattern of M active periods, represented by the Rate$(\cdot)$ function, is specified by the sequences, $f_1, f_2, \cdots, f_M$; $s_1, s_2, \cdots, s_M$ and $t_1, t_2, \cdots, t_M$, such that $s_m < t_m$ for all $m = 1, 2, \cdots, M$. For each pattern, consider a rectangle of width $t_{\max} - s_{\min}$ and height $\overline{r}$. Compare it to Rate(t) over the interval $[s_{\min}, t_{\max}]$. Note that the area of the rectangle is equal to the area under Rate(t) over the interval $[s_{\min}, t_{\max}]$. Thus if Rate(t) is less than $\overline{r}$ over a non-zero duration, then Rate(t) must be larger than $\overline{r}$ over a non-zero duration. It is easy to see that for all M, $1 \leq M \leq k - j + 1$, and all patterns of M active periods, there exist $t^* \in [s_{\min}, t_{\max}]$ and $\Delta > 0$, such that for $t^* \leq t < t^* + \Delta$

$$\text{Rate}(t) \geq \overline{r}. \quad (26)$$

From (17), (21) and (26), we have for $t^* \leq t < t^* + \Delta$

$$\text{Rate}(t) \geq \frac{\sum_{m=1}^{M} \sum_{p \in N_m} l(p)}{t_{\max} - s_{\min}}. \quad (27)$$

From the definition of t_m, we have

$$t_{\max} \leq L(k) - \frac{l_{\max}}{C}. \quad (28)$$

Furthermore, the following holds for any packet i in $\{j, \cdots, k\}$

$$A(i) > L(j-1) - \frac{l(j-1)}{C}. \quad (29)$$

Otherwise some packet in $\{j, \cdots, k\}$ would have been selected for service before packet $j - 1$, because $P(j-1) > P(i)$ for $i = j, \cdots, k$ from (14) and (15). Therefore we have

$$s_{\min} > L(j-1) - \frac{l(j-1)}{C}. \quad (30)$$

Combining (28) and (30), we have[6]

$$t_{\max} - s_{\min} < L(k) - \frac{l_{\max}}{C} - \left(L(j-1) - \frac{l(j-1)}{C}\right) \quad (31)$$

$$= L(k) - L(j-1) - \frac{l_{\max} - l(j-1)}{C} \quad (32)$$

$$\leq L(k) - L(j-1). \quad (33)$$

From (27) and (33), we have for $t^* \leq t < t^* + \Delta$

$$\text{Rate}(t) > \frac{\sum_{m=1}^{M} \sum_{p \in N_m} l(p)}{L(k) - L(j-1)}. \quad (34)$$

By their definition, $N_1, N_2, \cdots, N_M$ constitute a partition on $\{j, \cdots, k\}$, therefore

$$\{j, \cdots, k\} = \bigcup_{m=1}^{M} N_m \quad (35)$$

$$\sum_{p=j}^{k} l(p) = \sum_{m=1}^{M} \sum_{p \in N_m} l(p). \quad (36)$$

Combining (34) and (36), we have for $t^* \leq t < t^* + \Delta$

$$\text{Rate}(t) > \frac{\sum_{p=j}^{k} l(p)}{L(k) - L(j-1)}. \quad (37)$$

Since

$$L(k) - L(j-1) = \frac{\sum_{p=j}^{k} l(p)}{C} \quad (38)$$

we have from (37) that for $t^* \leq t < t^* + \Delta$

$$\text{Rate}(t) > C. \quad (39)$$

To complete our proof of Theorem 1, the following case remains to be considered.

Case II: There is no j, $1 < j \leq k$, such that for $i = j, \cdots, k$, (14) and (15) are satisfied. That is, for $i = 1, \cdots, k$

$$P(i) < L(k) - \frac{l_{\max}}{C}. \quad (40)$$

The proof is similar to Case I, but (15) in the above proof does not apply here. Equations (16)–(28) remain the same. Equations (29) and (30) become

$$s_{\min} = A(1) \quad (41)$$

[6]Note that Theorem 1 holds if $l_{\max} \geq l(j-1)$, that is sufficient for the inequality in (33).

where $A(1)$ is the start of the busy period. Equations (31)–(33) become

$$t_{\max} - s_{\min} \leq L(k) - \frac{l_{\max}}{C} - A(1) \quad (42)$$
$$< L(k) - A(1). \quad (43)$$

Equations (34)–(39) become, for $t^* \leq t < t^* + \Delta$

$$\text{Rate}(t) > \frac{\sum_{m=1}^{M} \sum_{p \in N_m} l(p)}{L(k) - A(1)} \quad (44)$$
$$= \frac{\sum_{p=1}^{k} l(p)}{L(k) - A(1)} = C. \quad (45)$$

Thus, (12) holds for Case II as well as for Case I. □

REFERENCES

[1] G. M. Bernstein, "Reserved bandwidth and reservationless traffic in rate allocating servers," *Comput. Commun. Rev.*, pp. 6–24, July 1993.

[2] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queuing algorithm," in *Proc. ACM SIGCOMM*, 1989, pp. 3–12.

[3] D. Ferrari and D. Verma, "A scheme for real-time channel establishment in wide-area networks," *IEEE J. Select. Areas Commun.*, pp. 368–379, Apr. 1990.

[4] N. R. Figueira and J. Pasquale, "An upper bound on delay for the virtual clock service discipline," *IEEE/ACM Trans. Networking*, vol. 3 no. 4, Aug. 1995.

[5] S. J. Golestani, "A self-clocked fair queueing scheme for high speed applications," in *Proc. IEEE INFOCOM*, 1994.

[6] P. Goyal, S. S. Lam, and H. M. Vin, "Determining end-to-end delay bounds in heterogeneous networks," in *Proc. Workshop Network and OS Support for Digital Audio Video*, April 1995.

[7] S. S. Lam and G. G. Xie, "Burst Scheduling: architecture and algorithm for switching packet video," Department of Computer Sciences, University of Texas, Austin, TX, Technical Rep. TR-94-20, July 1994. Abbreviated version in *Proc. IEEE INFOCOM*, 1995.

[8] ——, "Burst scheduling networks: flow specification and performance guarantees," in *Proc. Workshop on Network and OS Support for Digital Audio Video*, April 1995.

[9] ——, "Group priority scheduling," Department of Computer Sciences, University of Texas at Austin, Tech. rep. TR-95-28, 1995; available `http://www.cs.utexas.edu/users/lam/NRL`. An abbreviated version to appear in *Proc. IEEE INFOCOM '96*.

[10] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single node case," *IEEE/ACM Trans. Networking*, vol. 1, no. 3, pp. 344–357, June 1993.

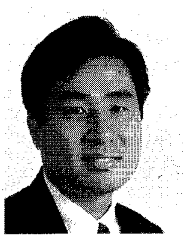
[11] ——, "A generalized processor sharing approach to flow control in integrated services networks: the multiple node case," *IEEE/ACM Trans. Networking*, vol. 2, no. 2, pp. 137–150, Apr. 1994.

[12] G. G. Xie and S. S. Lam, "Delay guarantee of virtual clock server," Department of Computer Sciences, University of Texas, Austin, TX, Technical Report TR-94-24, 1994, presented at 9th IEEE Workshop on Computer Communications, October 1994.

[13] H. Zhang and S. Keshav, "Comparison of rated-based service disciplines," in *Proc. ACM SIGCOMM*, 1991, pp. 113–121.

[14] L. Zhang, "A new architecture for packet-switched network protocols," Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, 1989.

[15] ——, "Virtual clock: a new traffic control algorithm for packet switching networks," in *Proc. ACM SIGCOMM*, Aug. 1990, pp. 19–29.



Geoffrey G. Xie received the B.S. degree in computer science from Fudan University, Shanghai, China, in 1986. He received the M.S. degree in computer science and the M.A. degree in mathematics from Bowling Green State University, Ohio, in 1988. He has been working toward the Ph.D. degree in computer science at the University of Texas, Austin, TX, since 1988 and holds an Intel Graduate Fellowship.

From 1991 to 1993, he worked as a full-time project engineer in Schlumberger Austin Systems Center. His research interests are computer networking, multi-media communications, and distributed computing.

Simon S. Lam (S'71–M'74–SM'80–F'85) received the BSEE degree, with Distinction, from Washington State University in 1969, and the M.S. and Ph.D. degrees in engineering from the University of Califonia, Los Angeles, CA, in 1970 and 1974, respectively.

From 1971 to 1974, he was a postgraduate research engineer at the ARPA Network Measurement Center (UCLA). From 1974 to 1977, he was a research staff member at the IBM T. J. Watson Research Center, Yorktown Heights, NY. Since 1977, he has been on the faculty of the University of Texas at Austin, where he is a Professor of Computer Sciences. He holds two anonymously endowed professorships, and served as department chair from 1992 to 1994. His research interests are in network protocol design, performance analysis, formal verification, network security, and multi-media.

Dr. Lam's paper on packet switching in a multi-access broadcast channel, derived from his doctoral dissertation, received the 1975 Leonard G. Abraham Prize Paper Award from the IEEE Communications Society. He was elected an IEEE Fellow in 1985 and served on the editorial boards of IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, IEEE TRANSACTIONS ON COMMUNICATIONS, PERFORMANCE EVALUATION, and PROCEEDINGS OF THE IEEE. He organized and was program chair of the first ACM SIGCOMM Symposium, held at the University of Texas, Austin, in 1983. He presently serves as Editor-in-Chief of the IEEE/ACM TRANSACTIONS ON NETWORKING.