

Universal Timestamp-Scheduling for Real-Time Networks

Jorge A. Cobb

Department of Computer Science
University of Houston
Houston, TX 77204-3475
cobb@cs.uh.edu

1. Introduction

A *computer network* consists of a set of computers interconnected via point-to-point bi-directional channels. A *flow* in a computer network is a potentially infinite sequence of packets generated by the same source and having the same destination in the network.

When a new flow wishes to join the network, the network finds a path from the source of the flow to the destination of the flow. Then, the network reserves a fraction of the bit rate of each channel along the path, and assigns this rate to the new flow. Finally, the source of the flow is given permission to generate packets at the rate reserved for the source. The chosen path and reserved rate of the flow remain fixed throughout the lifetime of the flow.

Due to the reservation of bandwidth, the network can provide service guarantees to each flow, such as end-to-end packet delays, provided the rate of the flow does not exceed the agreed-upon rate.

Each output channel of a computer is equipped with a scheduler. From the input channels, the scheduler receives packets from flows whose next hop to the destination is the output channel of the scheduler. Whenever its output channel becomes idle, the scheduler chooses a received packet and forwards the packet to the output channel.

The scheduler maintains a separate FIFO queue for the received packets from each flow. We say that a flow is *active* if its queue in the scheduler is non-empty or if a packet of the flow is in the output channel.

Some scheduling protocols assign a timestamp to each incoming data packet, and forward these packets to the output channel in order of increasing timestamp. Examples of these schedulers are Virtual Clock [19,17,6], Weighted Fair Queuing [13, 14], Time-Shift Scheduling [2] and Frame-Based Fair Queuing [16]. The method used to assign timestamps to data packets differs from one protocol to another. However, they all share a similar structure.

In this paper, we reveal the strong relationship among these protocols, by defining a family of scheduling protocols, called *Universal Timestamp-Scheduling*, and by showing that the above protocols are all members of this family.

We show that all members of the protocol family guarantee to each flow an upper bound on the delay similar to that of Virtual Clock scheduling. Since the

protocols mentioned earlier are all members of the protocol family, they all possess these properties.

To show the generality of the protocol family, we show that the above protocols in the literature correspond to one end of the spectrum of the Universal Timestamp-Scheduling family, and that there is another end of the spectrum that is worthy of attention but has not been investigated.

We adopt the following notation and definitions. We say that a packet is *forwarded* to the output channel when the first bit of the packet is transmitted by the output channel. A packet has *exited* the output channel if its last bit has been transmitted by the output channel. A packet is *in* the output channel if it has been forwarded but has not yet exited the output channel.

- clock real-time clock of the scheduler.
- N maximum number of flows allowed by the scheduler.
- C bandwidth (bits/sec) of the output channel.
- R.i reserved rate of flow i (bits/sec).
- queue.i queue of received packets of flow i.
- p.i.n nth packet received from flow i.
- A.i.n time in the scheduler when p.i.n was received (sec).
- L.i.n packet size of p.i.n (bits).
- Lmax upper bound on packet length over all flows.
- P.i.n time when the last bit of packet p.i.n exits the output channel (sec).

Since all N flows share the output channel, the following constraint is necessary.

$$\sum_{f=0}^{N-1} R.f \leq C \quad (1)$$

The timestamp T.i.n assigned to each data packet p.i.n is calculated as follows.

$$T.i.n := \max(\text{base}, T.i.(n-1)) + L.i.n/R.i \quad (2)$$

In the above equation, the value of base is a non-increasing function. This function is protocol dependent, and the choice for base distinguishes one member of the protocol family from another. For example, if base = A.i.n, then the resulting protocol is known as Virtual Clock.

It can be easily shown by induction that in the Virtual Clock protocol, T.i.n equals the time at which p.i.n would exit a dedicated output channel whose sole input flow is flow i and the channel's rate is exactly R.i. Furthermore, it has been shown that in Virtual

Clock, each packet exits the scheduler by the time indicated in its timestamp (plus the small constant L_{\max}/C) [6,17]. Thus, the timestamp in the Virtual Clock protocol may be viewed as an exit deadline, and packets are forwarded in order of increasing deadlines.

The above bound on the exit time is desirable, and we refer to it as *rate-proportional delay*.

4. Universal Timestamp-Scheduling

We next define a family of scheduling protocols. Each protocol in the family assigns a timestamp to each received data packet according to Assignment (2). However, the protocols differ in their choice of base function. We define upper and lower bounds for the base function, such that regardless of which value within these bounds is chosen, the protocol will have rate proportional delay. We first introduce a few definitions.

4.1 Bit Timestamps

Let b be a real number in the interval $[0, L_{i,n}]$. We define the *bit timestamp* [2] of "bit" b of packet $p_{i,n}$ as:

$$T_{i,n} - (L_{i,n} - b)/R_i \quad (3)$$

Thus, we assume bits are of arbitrarily small size, and a timestamp is assigned to each bit. In this way, if $T_{i,n}$ were to correspond to the exit time of packet $p_{i,n}$ from a channel whose bandwidth is R_i , then bit b would exit at the time indicated by (3) above.

Thus, the first bit of the packet has timestamp $T_{i,n} - L_{i,n}/R_i$, and the last bit of the packet has timestamp $T_{i,n}$. For convenience, we refer to the timestamp of the first bit as $S_{i,n}$, that is,

$$S_{i,n} = T_{i,n} - L_{i,n}/R_i$$

We say that bit timestamp t is *contained* by packet $p_{i,n}$ if a bit in the packet has timestamp t , i.e., $S_{i,n} \leq t \leq T_{i,n}$. Flow i contains timestamp t if some packet of i contains t .

4.2 Virtual Server

The possible values for the base are derived from the behavior of a virtual server, which we describe next.

The virtual server receives as input the same set of flows as the scheduler, and has an output channel whose bandwidth is also C . In addition, each packet is assigned a timestamp using the Assignment (2).

However, the virtual server is able to forward a bit at a time, rather than a whole packet at a time as done by the scheduler. The virtual server forwards bits in order of increasing timestamp. Although the virtual server is not implementable, its behavior can be computed by the scheduler, since the input flows of both are the same.

Note that from the way in which timestamps are assigned to bits, if the flows in set B contain all the bit timestamps in the interval $[t, t']$, and no flow outside of B contain any bit timestamps in this interval, then the

time required for the virtual server to forward all the bits in this interval is:

$$\frac{(\sum_{i: i \in B} R_i) \cdot (t' - t)}{C}$$

Note that this is similar to the way in which bits are forwarded by the virtual server used in Weighted Fair Queuing [14]. However, the behavior of both virtual servers is quite different. In Weighted Fair Queuing, timestamps are not assigned to bits. Therefore, if a flow has bits remaining to be forwarded, then bits from this flow will continuously be forwarded until no more bits of the flow remain. This is not the case in our virtual server, since if all the bits of flow i have timestamps that are greater than all the bits of flow j , then all the bits of flow j will be forwarded before any bit of flow i is forwarded.

We define the following:

- $W_{i,n}$ last bit forwarded from flow i
- $T_{i,n}$ largest bit timestamp of flow i , i.e., $T_{i,n} = T_{i,n}$, where $p_{i,n}$ is the last packet received from flow i .
- Z timestamp of the bit currently being served in the virtual server.

Note that if $W_{i,n} < T_{i,n}$, there are still some bits from flow i that remain to be forwarded. Thus, if there is an i such that $W_{i,n} < T_{i,n}$, then Z is the minimum over all these i , that is,

$$Z = (\min_{i: W_{i,n} < T_{i,n}} W_{i,n})$$

On the other hand, if there is no i such that $W_{i,n} < T_{i,n}$, then we assign to Z a value greater than the maximum over all timestamps, that is,

$$Z > (\max_{i: T_{i,n}} T_{i,n})$$

4.3 Computing the Base

We next define how is the value for base chosen. We should choose a range of values that ensures the rate proportional delay property is maintained, while at the same time allowing a great deal of flexibility in the choice of values.

To ensure rate proportional delay, we associate with each packet $p_{i,n}$ a deadline $D_{i,n}$, which is equal to the timestamp the packet would receive under Virtual Clock scheduling. That is,

$$D_{i,n} := \max(A_{i,n}, D_{i,n-1}) + L_{i,n}/R_i$$

We will often need to refer to the deadline of the bit of flow i whose timestamp is t . Thus, $D_{i,t}$, where t is a timestamp, denotes the deadline of the bit of flow i whose timestamp is t . Let timestamp t be contained by packet $p_{i,n}$, and let bit b , $0 \leq b \leq L_{i,n}$, be the bit of $p_{i,n}$ whose timestamp is t . Then, $D_{i,t}$ is defined as follows.

$$D_{i,t} = D_{i,n} - (L_{i,n} - b)/R_i$$

Thus, the deadline of a bit is simply the timestamp the bit would receive in a Virtual Clock scheduler.

Thus, in Virtual Clock, the deadline of a bit and its timestamp are equivalent.

Recall that base is computed immediately before assigning a timestamp to a received packet. We next need to compute a value for base that will ensure rate proportional delay in the virtual server. We later show that the same value for base will also ensure a rate proportional delay in the scheduler.

Notice that all future bits received from any flow k will have a timestamp at least the value of base. Thus, to guarantee that bit $W.j$ of flow j exits by its deadline, we simply count, for each flow k , where $W.k \leq W.j$, the maximum number of bits that could have a timestamp at most $W.j$ that are yet to be transmitted. These bits are:

$$(W.j - \max(W.k, \text{base})) \cdot R.k$$

To obtain the amount of time required to transmit these bits, simply divide by C . The virtual server should have enough time to transmit all these bits from every flow k before the deadline of bit $W.j$ of flow j expires. Thus, the following predicate should always hold in the virtual server:

$$\frac{(\forall j : Z \leq W.j : (\sum_{k : W.k < W.j} (W.j - \max(W.k, Z)) \cdot R.k)}{C} \tag{4}$$

We refer to the above predicated as the *safety predicate*. We will later show that if the above predicate holds, it continues to hold as bits are transmitted. However, when packet $p.i.n$ is received, if $S.i.n < Z$, then Z is updated to $S.i.n$. The change in Z could invalidate the safety predicate. Thus, we should choose a value of base such that the safety predicate holds after $p.i.n$ is assigned a timestamp, ensuring the safety predicate holds at *all* times. We therefore choose a value of base that satisfies the following *base predicate*.

$$\frac{(\forall j : \text{base} \leq W.j : (\sum_{k : W.k < W.j} (W.j - \max(W.k, \text{base})) \cdot R.k)}{C} \tag{5}$$

In addition, we restrict the new value of base to be nondecreasing, and also, if bits remain to be transmitted (i.e., $B \neq \text{empty}$), then base should be at most Z . In the next section, we show that if we choose base in this manner, the safety predicate holds at all times and the virtual server has rate-proportional delay.

5. Upper Delay Bound

In this Section, we prove that both the virtual server and the Universal Timestamp-Scheduler have rate-proportional delay. We begin with a few lemmas. In the

following, the expression "at time t " refers to the state of the virtual server when variable clock has the value t .

Theorem 1 For any j , if $W.j < T.j$, then $D.j(W.j) \geq \text{clock}$.

◆ Theorem 1 states that each bit in the virtual server exits before its deadline expires.

The packet scheduler will choose a value of base equal to a possible value chosen by the virtual server. Thus, both the packet scheduler and the virtual server will assign equal timestamps to the same packet. Furthermore, the deadline of a packet in both the scheduler and the virtual server are also the same. What remains to be shown is that the packet scheduler does have rate proportional delay, that is, that each packet will exit by its deadline plus a small constant, namely, L_{max}/C . To show this, we take advantage of the behavior of the virtual server.

Theorem 2 In the packet scheduler, for all i and n , $P.i.n \leq D.i.n + L_{\text{max}}/C$.

◆ We therefore conclude that the scheduler has rate-proportional delay. Thus, when a new scheduling protocol based on timestamps is designed, the designer needs only to show that the protocol belongs to the Universal Timestamp-Scheduling family of protocols, and as a corollary the protocol has rate-dependent delay.

6. Existing Family Members

In this section, we show the flexibility of our family of Universal Timestamp-Scheduling protocols by showing that many scheduling protocols in the literature belong to the family. For each protocol, we give a proof sketch that their choice of base satisfies the base predicate, and thus they are members of the family.

6.1. Weighted Fair Queuing

It is straightforward to show that the Weighted Fair Queuing (WFQ) is a member of the protocol family. If we simply always choose $\text{base} = Z$, we obtain WFQ as a result. Thus, WFQ satisfies the base predicate (from the proof of Lemma 1) and is a member of the protocol family.

6.2. Virtual Clock

In Virtual Clock scheduling, the deadline of a packet is also its timestamp. It is easy to show that the value of Z is at least the clock, as follows. Assume $Z \geq \text{clock}$. Z increases at least as fast as real-time (Lemma 1), and thus $Z \geq \text{clock}$ is maintained if Z increases. If the value of Z decreases, it is because a packet $p.i.n$ was received, and its $S.i.n \geq \text{clock}$, and thus $Z = S.i.n \geq \text{clock}$ after the packet is timestamped. Thus, we always have that $\text{clock} \leq Z$.

We would like to show that the base predicate holds by choosing $\text{base} = \text{clock}$. Thus, we have $\text{base} = \text{clock}$

$\leq Z$. Furthermore, since the deadline of a packet is also its timestamp, for any j , where $\text{base} \leq j$,

$$D.j.(W.j) - \text{clock} = W.j - \text{clock} \geq (\sum k : W.k \leq W.j : W.j - \max(\text{clock}, W.k) \cdot R.k) / C$$

Thus, the base predicate holds, and Virtual Clock scheduling is a member of the family.

6.3. Time-Shift Scheduling

In Time-Shift Scheduling, base is defined as follows. Assume we receive a packet at time t , and the last packet received previous to t was received at time t' , $t' < t$. Let base' be the value of base at time t' . At time t , base is assigned the value $\max(\text{base}' + t - t', S_{\min})$, where S_{\min} is the minimum of all $S_{i,n}$ for all packets $p_{i,n}$ that have not been forwarded to the output channel and also the packet currently in the output channel. Thus, base increases with real time, and may be thought of as an adjustable clock, that on occasions "jumps ahead" to the value S_{\min} when possible.

It is easy to show that at any time t , $S_{\min} \leq Z$. This is because the virtual server always forwards the smallest bit first. Furthermore, Z increases at least as fast as real-time (Lemma 1). Thus, $\text{base} \leq Z$ always holds.

Thus, we always have that base increases at least as fast as real-time and is always at most Z . Any scheduler that chooses a base that satisfies these two properties is called a rate-proportional scheduler (RPS) [15,16]. Thus, the Time-Shift scheduler is a RPS. We next show that all RPS's belong to our protocol family, by showing that the value chosen for base satisfies the base predicate.

If we think of base as an adjustable clock, think of the timestamp of a packet as a pseudo-deadline that is measured with respect to the value of base. It is easy to verify that this is at most the deadline as originally defined. I.e., if a received packet is given a true deadline D seconds greater than the real time clock, then the pseudo-deadline (i.e., the timestamp) given to the packet is at most D seconds greater than the value of base.

In the virtual server, for any flow j with $W.j > Z \geq \text{base}$, the pseudo deadline is $W.j$, and thus the sum of all bits of all flows with timestamps from base up to $W.j$ is at most $(W.j - \text{base}) \cdot C$ (from (1)), and this divided by C is at most $W.j - \text{base}$, and hence the base predicate holds with respect to the pseudo-deadline. Since the real deadline is at least the pseudo-deadline, the base predicate holds, and all RPS schedulers belong to the Universal Timestamp-Scheduling family.

6.4. Frame-Based Fair Queuing

Frame-Based Fair Queuing was proposed in [16], and shown to be a RPS. Therefore, it also belongs to our family of Universal Timestamp-Scheduling protocols.

7. New Family Members

All timestamp protocols in the literature ensure that base increases at least as fast as real-time. It is believed in the literature that increasing the base at least as fast as real-time is necessary to provide rate-proportional delay [19,15]. However, we have shown in this paper that this is not necessarily the case. In addition, protocols in which the base does not increase as fast as real-time have not been investigated in the literature, and we believe they are worthy of investigation.

In the above protocols, if a flow has not generated packets for a certain amount of time, the flow will try to reclaim the bandwidth it has lost during its period of inactivity. However, in doing so *it will not infringe upon the basic deadline guarantees of other flows*.

For example, consider a protocol in which the value chosen for base is always the smallest value possible that satisfies the base predicate. In this case, if a flow has not generated packets for a certain amount of time, the flow will try to reclaim the bandwidth it has lost during its period of inactivity. However, in doing so *it will not infringe upon the basic deadline guarantees of other flows*.

If we assume that the source of flow has paid money for its reserved bandwidth, it is sensible to assume that the source would like, if possible, to recapture some of the bandwidth it did not use during a period of idleness. Other protocols, such as [20,14,2,16], distribute the unused bandwidth among the flows which still have packets to be transmitted, and do not allow a source that has been idle for some time to recuperate its unused bandwidth. Which of these two choices is more appropriate will depend on the nature of the applications supported.

Computing the minimum value of base that satisfies the base predicate takes $O(n)$ time. It would be interesting to obtain protocols which approximate the value of base with lower complexity, such as $O(\log(n))$ time, in the same way that approximations to Weighted Fair Queuing [2,16] compute the base in $O(\log(n))$ or $O(1)$ rather than $O(n)$.

8. End-to-End Delay Bound

Consider a network path from a source to its destination, and let the number of hops in this path be K . Let R be the rate reserved for the flow. Furthermore, assume each of the computers uses a scheduling protocol that has the rate-proportional delay property. In [1,3], it has been shown that, for a source whose flow of packets is constrained by a leaky bucket of size B and rate R , the end-to-end delay of a packet of this source has the following upper bound.

$$B/R + (K-1) \cdot (L/R + L_{\max}/C)$$

In the above, L is the maximum packet size of the flow, and L_{\max} is the maximum packet size allowed along the path.

Since all protocols in the Universal Timestamp-Scheduling family have the rate-proportional delay property, then a source whose network path consists of schedulers of this family has the above upper bound on end-to-end packet delay.

9. Related Work

For a more detailed description of other work, see [ISDN???

The family of Rate Proportional Servers, which we briefly mentioned in Section 6.3 above, was originally defined in [15]. It defines a spectrum of scheduling protocols, with Virtual Clock in one end of the spectrum and Weighted Fair Queuing in the other end of the spectrum. In this paper, we have shown that the spectrum is much broader than this, and that the property of the base value increasing at least as fast as real-time, which is called a "fundamental" property in [15], is not necessary for rate-dependent delay.

In addition, in [15] it is claimed that a packet in a Rate Proportional Scheduler will exit by the time the same packet will exit the corresponding virtual server plus L_{\max}/C , i.e., $P_i.n \leq F_i.n + L_{\max}/C$. This is definitely the case for the case of Weighted Fair Queuing, as shown in [14], but we next give a counterexample for the case of Virtual Clock. Thus, the claim in [15] is incorrect.

10. Summary and Concluding Remarks

We have defined a new family of Universal Timestamp-Scheduling protocols, where each member of the protocol family is guaranteed to satisfy the rate-proportional delay property. Thus, when a protocol designer wishes to show that a new timestamp protocol satisfies the rate-proportional delay property, he/she needs only to show that the protocol belongs to the family of protocols. This is done by showing that the base value satisfies the base predicate.

We have so far consider only the case of rate-proportional delay. It would be interesting to determine if a similar family of protocols may be obtained that have more flexible delay assignments. That is, the contribution to the end-to-end packet delay from each hop in the path to the destination is not forced to be L/R . This is known as rate-independent delay. Examples of protocols with rate independent delay include [8,5].

In addition, more investigation is needed on protocols which allow flows to recuperate part of their unused bandwidth during short periods of inactivity. Some issues to investigate are practical applications and efficient ways to implement these protocols.

References

[1] Cobb J., Gouda M., "Flow Theory", *IEEE/ACM Transactions on Communications*, October 1997.

- [2] Cobb J., Gouda M., El-Nahas A., "Time-Shift Scheduling: Fair Scheduling of Flows in High-Speed Networks", *Proc. of the IEEE Int'l Conf. on Network Protocols*, 1996.
- [3] Cobb, J., "Flow Theory and The Analysis of Timed-Flow Protocols", Ph.D. Thesis, The University of Texas at Austin, May 1996.
- [5] Figueira N., Pasquale J., "Leave-in-Time: A New Service Discipline for Real-Time Communications in a Packet-Switching Data Network", *Proceedings of the 1995 SIGCOMM Conference*, p. 207.
- [6] Figueira N. R., Pasquale J., "An Upper Bound on Delay for the Virtual Clock Service Discipline", *IEEE/ACM Transactions on Networking*, Vol. 3, No. 4, Aug. 1995.
- [8] Ferrari D., Verma D., "A Scheme for Real-Time Channel Establishment in Wide-Area Networks", *IEEE Journal of Selected Areas in Communication*, 8(4):368-379, April 1990.
- [13] Keshav S., "A Control Theoretic Approach to Flow Control", *Proceedings of the 1991 ACM SIGCOMM Conference*.
- [14] Parekh A. K. J., Gallager R., "A generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case", *IEEE/ACM Transactions on Networking*, 1(3):344-357, June 1993.
- [15] Stiliadis D., Varma A., "Rate Proportional Servers: A Design Methodology for Fair Queueing Algorithms", University of California at Santa Cruz, Computer Science Department technical report number UCSC-CRL-95-58.
- [16] Stiliadis D., Varma A., "Design and Analysis of Frame-Based Fair Queueing: A New Traffic Scheduling Algorithm for Packet Switched Network", *ACM SIGMETRICS*, 1996.
- [17] Xie G., Lam S., "Delay Guarantee of Virtual Clock Server", *IEEE/ACM Transactions on Networking*, December 1995.
- [18] Zhang H., "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks", *Proceedings of the IEEE*, Vol. 83, No. 10, Oct. 1995.
- [19] Zhang L., "Virtual Clock: A New Traffic Control Algorithm for Packet-Switched Networks", *ACM Transactions on Computer Systems*, Vol. 9, No. 2, May 1991.
- [20] Zhang H., Keshav S., "Comparison of Rate-Based Service Disciplines", *ACM SIGCOMM Conference*, 1991.