Frame-counter scheduler: A novel QoS scheduler for real-time traffic

S. Ece (Guran) Schmidt *, Hyong S. Kim

Electrical and Computer Engineering, Carnegie Mellon University, 5000 Forbes Avenue Pittsburgh, PA 15213-3890, USA

Received 7 July 2005; received in revised form 29 January 2006; accepted 14 February 2006
Available online 20 March 2006

Abstract

Real-time traffic communication has Quality of Service (QoS) requirements such as end-to-end bandwidth and delay guarantees. We propose a novel frame-based QoS Scheduler, the frame-counter scheduler, for connection oriented packet switching networks. The frame-counter scheduler significantly reduces the end-to-end delay bound and buffer requirements provided by other frame-based schedulers. A fixed amount of buffer is required per node for no packet-loss operation. There is no need for frame synchronization or inter-node communication. The scheduling complexity of frame-counter scheduler is $O(1)$ which makes it possible to implement it for high-speed networks. The required input traffic shape is not more restrictive than the traffic shapes used by the other schedulers.

In this paper, we present the proof for the end-to-end delay bound and the buffer requirement for the frame-counter scheduler. We also provide simulation results to demonstrate the average performance which show that the average end-to-end delay and delay variation (jitter) of the packets is much lower than the end-to-end bound.

© 2006 Elsevier B.V. All rights reserved.

Keywords: QoS; Real-time traffic; Frame-based scheduler

1. Introduction

The end-to-end Quality of Service (QoS) for the real-time communication includes bandwidth, delay, and jitter guarantees. Connection admission control and traffic shapers at the network edge are required for QoS support. The switches must use traffic scheduling algorithms to serve packets carrying real-time data in the network. Such traffic scheduling algorithms should have low implementation complexity and simple connection admission control to be able to operate at high speed. Processing and scheduling packets at high line speeds need to be completed in the order of nanoseconds. This short time period increases complexity and limits the scalability of switching systems. Thus, providing end-to-end QoS guarantees for real-time traffic in a scalable and low-complexity fashion is an important issue in high-speed communication networks.

Many QoS schedulers that can support different QoS guarantees have been proposed in the literature. These algorithms include Weighted Fair Queuing (WFQ) [1], Self-Clocked-Fair-Queuing (SCFQ) [2], Delay-Earliest-Due-Date (D-EDD) [3], Rate-Controlled-Static-Priority (RCSP) [4], Traffic-Controlled Rate-Monotonic Priority Scheduling (TCRM) [5], Stop-and-Go (S&G) [6], Hierarchical Round-Robin (HRR) [7], Continuous Framing (CF) [8] and Budgeted Weighted Round-Robin (BWRR) [9].

In this paper, we propose a novel frame-based QoS scheduler called the frame-counter scheduler. The frame-counter scheduler can provide tighter end-to-end delay bound than other frame-based schedulers. The amount of buffer required for no-packet-loss operation is also small. The frame-counter scheduler does not need any inter-node frame boundary synchronization or communication yielding a low implementation complexity.

Hence, the proposed frame-counter QoS scheduler can provide end-to-end delay and bandwidth guarantees for
real-time traffic in high-speed networks in a scalable fashion.

Following this introduction, in Section 2, we present background information and a literature survey of the traffic schedulers for QoS support. In Section 3, we describe the problem formulation and explain the design principles. In Section 4, we describe the operation of the frame-counter scheduler. We prove that it satisfies a certain end-to-end delay bound, and works a certain buffer size limit. We also discuss the scheduling complexity of the frame-counter scheduler. In Section 5, we present simulation results to demonstrate the average performance of our algorithm. Finally, we give conclusions in Section 6. The proof of the correct operation and the provided QoS guarantees of the frame-counter scheduler is based on the construction of a state machine model which is presented in Appendix C.

2. Background

In this section, we first introduce the input traffic specification (i.e., packet arrival pattern at the source node), and the operation principles of different QoS schedulers. Then we discuss the performance metrics for these schedulers such as implementation complexity, required buffer space in each switch, and the provided QoS guarantees.

We consider connection-oriented networks for real-time traffic transport. The switches that are on the end-to-end path of a connection can allocate resources to provide performance guarantees. A connection admission process is required to check if it is possible to deliver the required QoS to the new connection without any service degradation for the existing connections. Two main approaches for the QoS Schedulers are sorted-priority and frame-based schemes.

Sorted-priority algorithms compute a timestamp for each arriving packet with respect to the current system state and update the system state accordingly. The scheduler sorts the packets based on their timestamps. The complexity of such algorithms derives from the computation of the timestamp for each packet and from maintaining the priority queues. The required computations have to be performed at the line rate. An increase in the line rate requires faster computation which results in a more expensive implementation. WFQ, SCFQ, D-EDD, RCSP and TCRM are examples for sorted-priority algorithms.

Frame-based approaches provide deterministic delay bounds for the real-time traffic in the packet network. Bandwidth guarantees are provided by splitting time into frames and limiting the amount of traffic that can be transmitted during a frame period [10]. A strict admission policy based on the peak rate of the connections is required to guarantee the bounded end-to-end delay. There might be an additional delay component to smooth the bursts over the frames. Algorithms such as S&G, HRR, CF and BWRR adopt the frame-based approach.

All of the QoS schedulers we mention above provide per-connection end-to-end delay guarantees to traffic streams regulated by a specific traffic model. Traffic specification models for real-time services bound the source traffic so that the number of bits that arrive for a connection during a specified time interval does not exceed a certain amount. Such traffic specifications include $(r, T)$, $(\sigma, \rho)$, and $(X_{\text{min}}, X_{\text{ave}}, I, S_{\text{max}})$ [11]. In the $(r, T)$ model, the traffic is shaped such that no more than $rT$ bits are transmitted in an interval of length $T$ which is called a frame, where $r$ is a measure of the average rate. Similarly, in the $(\sigma, \rho)$ model, $\sigma$ indicates the maximum burst size and $\rho$ indicates the long term bounding rate. In this traffic model, no more than $(\sigma + \rho \tau)$ bits are transmitted during a time interval $\tau$. The $(X_{\text{min}}, X_{\text{ave}}, I, S_{\text{max}})$ model defines the minimum inter-arrival time between packets as $X_{\text{min}}$ and the average inter-arrival time between packets measured over an interval of length $I$ as $X_{\text{ave}}$. The maximum packet size is denoted by $S_{\text{max}}$.

The end-to-end delay bound guaranteed by the QoS schedulers depends on the input traffic specification parameters and the number of hops on the end-to-end path of the connection. The scheduling complexity is either $O(1)$ or depends on the total number of connections to be scheduled ($V$).

WFQ and SCFQ work with the $(\sigma, \rho)$ traffic model. They provide an end-to-end delay bound that grows linearly with the number of hops on the end-to-end path. The end-to-end delay bound for the WFQ algorithm is the time to transmit the burst at the allocated rate in addition to the time spent to transmit the largest packet at each node. The scheduling complexity of the WFQ algorithm is $O(V)$ and the required buffer space for no packet loss operation grows at each switch on the path. SCFQ decreases the scheduling complexity to $O(\log V)$ at the expense of increasing the delay bound.

D-EDD and RCSP work with the $(X_{\text{min}}, X_{\text{ave}}, I, S_{\text{max}})$ traffic model and TCRM works with the $(\sigma, \rho)$ traffic model. They can guarantee bounded link and end-to-end delays which depend on the current state of the connection establishment. They all have a scheduling complexity of $O(\log V)$. TCRM has a connection admission algorithm with a complexity of $O(V)$. Using traffic regulators and assigning static priorities for each connection in these algorithms simplify the implementation. RCSP can operate with fixed buffer size at each node while the required buffer increases at each node on the path for D-EDD.

The proposed frame-counter scheduler is a frame-based QoS scheduler. Next, we explain S&G, HRR, CF and BWRR algorithms which are also frame-based schedulers.

S&G, HRR and CF are frame-based algorithms that are designed for fixed-size packets. They work with the $(r, T)$ traffic model, where $T$ is the used frame size. This traffic shape is maintained throughout the network by these algorithms. There is a difference in the service order of the packets between S&G and HRR. In S&G, packets that are transmitted in the same frame at the network ingress are transmitted in the same frame on all links traversed.
by the connection. In HRR this service rule does not hold, however the traffic shape is still preserved by limiting the number of packets served in a frame. The scheduling complexity for these algorithms is $O(1)$. Hence, they have a significant advantage over sorted priority schedulers. S&G and HRR algorithms guarantee an end-to-end delay bound of $2HT$ where $H$ is the number of hops on the end-to-end path. The required buffer size to operate without any packet loss is also bounded and constant for all switches. If the link rate is $C$, the required buffer size is $3TC$. If S&G and HRR work in synchronized fashion, the guaranteed end-to-end delay bound decreases to $HT$ and the required buffer size decreases to $2TC$. The synchronization sacrifices the flexibility of asynchronous frame boundaries in S&G and HRR and increases the scheduling complexity. The CF algorithm is a frame-based algorithm similar to S&G and HRR. CF synchronizes the frame boundaries of each node to achieve an end-to-end delay bound of $HT$.

BWRR is also based on a similar idea as the frame-based algorithms. It uses per connection control variables that are modified according to the actual packet interarrivals. BWRR works with periodic packet arrivals. Hence, it restricts the minimum interarrival times with a given period. Because of the mismatches between the periods of the connections and the frame size, the link capacity might not be fully utilized. The scheduling complexity of the BWRR algorithm is $O(1)$. BWRR guarantees an end-to-end delay bound of $HT$, and the required buffer size to operate without any packet loss is bounded by $2TC$.

3. Problem formulation

We consider a connection-oriented network in which the packets have fixed size and the packet times are synchronized. The basic time unit is the time required to transmit one packet and we call this time a slot. We assume that there is a fixed frame duration $T$ that is used by all of the nodes in the network for scheduling and traffic shaping purposes. Let the frame duration be $T$ slots long, hence, $T$ also denotes the number of packets per frame.

In our network model, the real-time traffic is transported in end-to-end traffic pipes, which we call connections. Each connection is assigned a fraction of the total link bandwidth. Let connection $A$ with the allocated rate of $B_A$ bits per second (bps) be established on an end-to-end path.

We express the allocated rate of connections with the unit of packets per frame. Suppose the link rate is $C$ bps and connection $A$ has an allocated rate of $B_A$ bps. Then, connection $A$ has an allocated rate of $\rho_A$ packets/frame where:

$$\rho_A = \left\lfloor \frac{T B_A}{C} \right\rfloor.$$  \hfill (1)

The connection admission control does not oversubscribe the links to provide guaranteed service. Let $K$ be the set of connections established on link $l$ and let $k \in K$ be a connection at the rate of $\rho_k$ packets/frame. Then the following condition holds for all links $l$:

$$\sum_{k \in K} \rho_k \leq T.$$  \hfill (2)

According to Eq. (1), this bandwidth allocation scheme defines the minimum rate a connection can reserve as 1 packet/frame. We define 1 packet/frame as the unit-rate. In this paper, we first describe the operation of the frame-counter scheduler and demonstrate and prove its features and properties for unit-rate connections. We then express multi-rate connections as a superposition of these unit-rate connections and extend the operation of the frame-counter scheduler to multi-rate connections in Section 4.3.

3.1. Traffic shape for packets arriving at the network

The frame-counter scheduler can provide end-to-end QoS guarantees if the incoming traffic for a connection is regulated at the network ingress so that it does not exceed the allocated rate. Let connection $A$ be a unit-rate connection. The required incoming traffic shape for the end-to-end QoS support is as follows:

Suppose that we set the time origin arbitrarily to 0 and we define periodic arrival frames of $T$ slots for connection $A$ (see Fig. 1, with the time origin set arbitrarily to 0.) Let $p_{Ak}$ denote the $k$th packet arriving from connection $A$. Suppose that the first packet of connection $A$, $p_{A1}$, completes its arrival at the network before slot $T$. The second packet completes its arrival after $T$ and before $2T$. Then, the $k$th packet, $p_{Ak}$, completes its arrival after slot $(k - 1)T$ and before slot $kT$.

Our input traffic shape definition does not enforce periodicity or a limit on the minimum interarrival time between two consecutive packets. It is possible that two packets arrive back-to-back with an interarrival time of 1 slot or there are no packet arrivals within a time interval of $T$ and packets arrive with an interarrival time of $2T - 1$ slots (See Fig. 1, maximum and minimum instantaneous interarrival times.). Note that it can be shown that our input traffic shape is equivalent to the $(r, T)$ traffic shape with fixed origin frame times defined in [6].

The traffic regulation described above requires buffering of packets during the shaping process which introduces a delay component. We assume that per connection shaping buffers are used at the network edge. The end-to-end delay guarantees for other scheduling algorithms introduced in Section 1 do not include the traffic shaping delay to achieve the required traffic shape for the algorithm. Similarly, in this paper, we do not include the shaping delay in the end-to-end delay expression.

3.2. Condition for end-to-end delay bound

In this paper, we define the end-to-end delay of a packet as the total time the packet spends in all of the switching nodes on its path to the destination. Hence, we do not include the transmission and propagation delays as well
as the traffic shaping delays at the network edge in our end-to-end delay expression. Lemma 1 below gives a condition for the end-to-end delay bound of the packets. This condition specifies the required service times for each packet from each node on its path. Any switch architecture and QoS scheduler that can guarantee the specified service times by this condition can guarantee the given end-to-end delay bound in Lemma 1. We present the proof for Lemma 1 in Appendix A.

**Lemma 1.** Let \( A \) be a connection established on an \( H \)-hop path. Let the \( k \)th packet of connection \( A \), \( p_{A_k} \), arrive at the network after \( (k - 1)T \) and before \( kT \) complying with the traffic shape described in Section 3.1. Let \( SW_h \) be the \( h \)th switch on connection \( A \)'s path, where \( (h = 1, 2, \ldots, H) \).

**Condition:** Assume that each switch \( SW_h \) can guarantee that if the \( k \)th packet arrives before \( (h + k - 1)T \) then it is served before \( (h + k)T \). Then on an \( H \)-hop path, the end-to-end delay bound for connection \( A \) is \( (H + 1)T \).

Lemma 1 states a local condition for each node \( h \). Hence, the condition can be checked individually for each node independent of the other nodes on the end-to-end path. This property of Lemma 1 makes it possible to consider a single node in proving the statements that describe the behaviour of the frame-counter scheduler. Note that Lemma 1 considers the arrival and the departure times of packets within specified time intervals. The exact arrival and departure instants affect the time that \( p_{A_k} \) spends at node \( SW_h \). The relative displacement of the packet in the arrival and departure frames is included in our end-to-end delay expression. In this paper, we use the end-to-end delay bounds guaranteed by other works as they are presented in the respective papers. However, the end-to-end delay in [6] would be \( (2H + 1)T \) if the relative displacement is taken into account as we do.

We use an example (See Example 1 in Fig. 2) to demonstrate Lemma 1 and packet delays at a node. \( p_{A_1}, p_{A_2} \) and \( p_{A_3} \) arrive at node \( SW_1 \) according to the traffic shape we define above. \( p_{A_2} \) arrives at node \( SW_1 \) at \( tarrp_{A_2} \) which is between \( T \) and \( 2T \). After a time interval \( d_{A_2} \), that is larger than \( T \) slots, \( p_{A_2} \) leaves the node at \( tservp_{A_2} \) which is between \( 2T \) and \( 3T \). Hence, \( p_{A_2} \) complies with Lemma 1 although it is delayed more than \( T \) slots at node \( SW_1 \).

The frame-counter scheduler can satisfy the condition in Lemma 1. Hence, it can guarantee an end-to-end delay bound of \( (H + 1)T \). In the following section, we first explain the operation of the frame-counter scheduler, and then prove that it can satisfy the condition in Lemma 1 for unit-rate connections. For the rest of the paper we consider a single node with its incoming and outgoing links because of the localized condition in Lemma 1.

**4. The frame-counter scheduler**

In our network model, we assume that fixed-size frames of duration \( T \) are defined for each link. We denote the \( j \)th frame by \( F_j \). The frame boundaries of incoming and outgoing links of a switch are not synchronized and they can have offsets \( \theta \) (see Fig. 3), with respect to each other. This is a significant feature of our algorithm which eliminates the internode frame synchronization and simplifies its implementation.

The proposed scheduling algorithm features a set of variables for each connection, which are updated synchronously at packet departures and at the frame boundaries. Let us consider an arbitrary unit-rate connection \( A \). The main counter limits the number of packets served per frame to the allocated rate \( \rho_A \). Thus, each connection has bandwidth guarantee as long as (2) holds for the connection admission. There are two auxiliary variables, the urgent counter and the urgent packet count to provide delay guarantee in addition to the bandwidth guarantee. These auxiliary variables are used to control the service order among the packets from different connections. In addition to these three state variables for each connection, a queue data structure which is called the urgency order list is required per port to maintain a certain service order among the connections.

![Fig. 1. Regulated traffic arrival shape.](image1)

![Fig. 2. Example 1: Node delays and traffic shape.](image2)

![Fig. 3. Example 2: Arrivals from upstream node to node \( h \) and departures from node \( h \).](image3)
4.1. Operation of the frame-counter scheduler

The frame-counter scheduler works with a non-blocking output queuing switch. The packets for connection \( A \) are stored in a per connection queue \( Q_A \) at the output port. The real-time connections reserve the required bandwidth on each link during the connection set-up phase.

The main counter ensures that the number of packets transmitted during a frame, which is the transmission rate for a connection, does not exceed the reserved rate. The main counter for connection \( A \) is denoted by \( C_A \) and \( C_{A_j} \) shows the value of \( C_A \) during a specific frame \( F_j \). At the beginning of each frame, the main counter \( C_A \) is restored to the allocated rate \( \rho_A \). \( C_A \) is decremented when a packet from connection \( A \) gets service. If \( C_A > 0 \), then connection \( A \) is eligible to get service. The output port is not idle as long as there are eligible connections with non-empty queues. By the connection admission condition (2), the total number of eligible packets at an output port never exceeds \( T \). Hence, the capacity of the link is not exceeded.

**Property 1.** \( 2\rho_A \) packets can arrive from an upstream node within frame \( F_j \) (see Example 2 in Fig. 3 for a unit-rate connection) because of the offset between the frame borders of the consecutive nodes. \( \rho_A \) packets get service in \( F_j \) because of the main counter.

Unlike a TDM scheduler, the frame-counter scheduler does not restrict the packets to occupy the same time slots within the frame. Hence, an eligible packet arriving from upstream node in \( F_j \) might not get served in \( F_j \) because of the conflict with other eligible packets. The main counters are restored to the allocated rates at the beginning of each frame for all connections. Hence, if an eligible connection cannot get service in the previous frame, this information is lost in the current frame if main counters are used only. Thus, we cannot determine if the allocated rate is utilized or not. This lack of information leads to instantaneous failures to maintain the bandwidth guarantee and consequently longer end-to-end delays.

We can demonstrate this problem with an example (see Example 3 in Fig. 4). \( T \) unit-rate connections including connection \( A \) arrive at node \( h \) on different incoming lines to be scheduled to the same outgoing line. Suppose that there are no arrivals during the first slot of \( F_j \), \( T - 1 \) eligible packets from connections numbered from 1 to \( T - 1 \) arrive at the switch at slot \( \theta + (h + k - 2)T + 1 \) and a second group of \( T - 1 \) packets from the same set of connections arrive next slot at \( \theta + (h + k - 1)T + 2 \). These second group of packets are not eligible to get service in \( F_j \) (See Property 1). An eligible packet \( pA_k \) (on connection numbered \( T \)) arrives at the switch at slot \( (h + k - 1)T + 3 \) after these two groups of packets. Note that total \( T \) eligible packets arrive in \( F_j \) but since the first slot was idle there are \( T - 1 \) slots left to serve these packets. Assume that the eligible packets are served in First-Come-First-Served (FCFS) manner and first group of \( T - 1 \) packets get served in \( F_j \) while \( pA_k \) is not served.

The main counters of all connections are reset to their allocated rates at the beginning of \( F_{(j+1)} \) hence, the second group of packets become eligible to get service. \( pA_k \) gets served the last at the slot \( (h + k)T - 1 \) following the FCFS order. Assuming that \( \theta > 1 \) \( pA_k \) leaves node \( h \) after \( (h + k)T \) violating the condition in Lemma 1.

In this example, we can identify two groups of eligible packets for \( F_{(j+1)} \). The packets which were eligible but did not get service in \( F_j \) (such as \( pA_k \)) are called urgent packets. The rest of the eligible packets to get service in \( F_{(j+1)} \) which are not urgent packets are called normal packets. Initially, the number of urgent packets is 0, and the number of normal packets that can get service in a frame is \( \rho_A \) for connection \( A \).

The problem in this example is serving the normal packets before \( pA_k \) in \( F_{(j+1)} \). In this case, the instantaneous bandwidth that connection \( A \) can use is less than the allo-

![Fig. 4. Example 3: Instantaneous failures to maintain the bandwidth guarantee.](image-url)
cated rate leading to the violation of the condition in Lemma 1.

The solution to this problem is to serve such urgent packets before normal packets. Suppose that before frame \(F_j\), all eligible packets for connection \(A\) get service in the frame that they arrive. If \(Q_A\) is non-empty and \(C_{Aj}\) is non-zero at the end of \(F_j\), this indicates that an eligible packet from connection \(A\) could not get service in \(F_j\).

Hence, there is an urgent packet for connection \(A\) for \(F_{j+1}\). The urgent packets for connection \(A\) in \(F_{j+1}\) are maintained by the urgent packet count and the urgent counter denoted by \(p_{AU(j+1)}\) and \(C_{AU(j+1)}\), respectively.

The urgent packet count is computed at the frame border and is not modified until end of the frame. At the frame border between \(F_j\) and \(F_{j+1}\), the initial number of urgent packets of connection \(A\) to get served in \(F_{j+1}\) is computed and stored in \(p_{AU(j+1)}\). Later, we show that we need to compare the initial number of urgent packets of the current frame with that of the previous frame. Hence, we store this value in \(p_{AU(j+1)}\) separately.

The urgent packet counter shows the remaining number of urgent packets to get service in a frame. \(C_{AU(j+1)}\) is set to \(p_{AU(j+1)}\) at the beginning of \(F_{j+1}\) and decremented when an urgent packet gets service. The connections with urgent packets are identified with the non-zero urgent counters and are served first in \(F_{j+1}\).

We define the urgent period \((T_{Uj})\) as the time that starts at the beginning of the \(F_j\) and ends when all urgent packets get service. During \(T_{Uj}\), the connections with non-zero urgent counters get service. \([T_{Uj}]\) shows the length of the urgent period. We define the rest of the frame as normal period.

If we generally refer to the value of urgent packet count and the urgent counter we show the values with \(p_{AU}\) and \(C_{AU}\).

When an urgent packet from connection \(A\) gets service, both \(C_A\) and \(C_{AU}\) are decremented. Note that denoting some packets urgent only determines the service order. The number of packets served from connection \(A\) is still controlled by \(C_A\). The connections with normal packets get service after the urgent packets without any specific ordering among these connections.

The urgent packets transferred from \(F_j\) use up the allocated bandwidth in \(F_{j+1}\). Hence, the packets from connection \(A\) that would have been served as normal packets in \(F_{j+1}\), cannot get service and they become the urgent packets for \(F_{j+2}\) shifting the service to the next frame.

For any frame \(F_j\), the number of normal packets left to be served and the size of \(Q_A\) at the end of \(F_j\) (denoted by \(|Q_A(j)|\)) are used to compute the initial number of urgent packets of connection \(A\) for \(F_{j+1}\). Assume that the value of the main counter is \(C_{Aj}\) at the end of \(F_j\). We can compute the number of normal packets that did not get service in \(F_j\) as follows:

The total number of packets that get service in \(F_j\) is \(p_A − C_{Aj}\). Out of these, \(p_{AU}\) are urgent packets. Thus, the total number of normal packets that get service in \(F_j\) is \((\rho_A − C_{Aj}) − p_{AU}\). Let \(C_{ANj}\) represent the maximum number of normal packets (assuming that all connections utilize their allocated bandwidth) that did not get service in \(F_j\) which represents the maximum number of urgent packets to get service in \(F_{j+1}\). Then,

\[
C_{ANj} = \rho_A - ((\rho_A - C_{Aj}) - p_{AU}) = C_{Aj} + p_{AU}.
\]

This computation assumes that at least \(\rho_A\) new eligible packets arrive during \(F_j\). We need to check if that is the case with the number of packets buffered at the end of \(F_j\).

Hence, \(p_{AU(j+1)} = \min(C_{ANj}, |Q_A(j)|) = \min((C_{Aj} + p_{AU}), |Q_A(j)|)\).

Note that \(p_{AU(j+1)}\) is non-zero if there is at least one packet in the queue \(|Q_A(j)|\) is non-zero. Hence, if \(p_{AU(j+1)}\) is 1 at the beginning of \(F_{j+1}\), and \(p_{AU}\) is the first packet in \(Q_A\) then \(p_{AU}\) is an urgent packet in \(F_{j+1}\).

**Property 2.** The maximum total number of urgent packets from all connections during any frame \(F_j\) is \(T - 1\).

If there is at least one eligible packet during \(F_j\), at least one packet gets served. Hence, out of the maximum number of \(T\) eligible packets for \(F_j\), at most \(T - 1\) can be transferred as urgent packets to \(F_{j+1}\). Consider a unit-rate connection \(A\). Note that \(p_{AU}\) for \(F_j\) can change from 0 to \(\rho_A\). Hence, it is possible that all packets of the connection to get served as normal packets can be transferred to the next frame as urgent packets. It is obvious that \(p_{AU}\) cannot exceed \(\rho_A\) because the maximum number of packets to get service in \(F_j\) is limited by \(C_{Aj}\) which is set to \(\rho_A\) at the beginning of \(F_j\).

**Lemma 2.** Let connection \(A\) be a unit-rate connection, \(p_{AU}\) be 1 at the beginning of \(F_j\), and \(p_{Ak}\) be the first packet in \(Q_A\) \(p_{Ak}\) gets served in \(F_j\) as an urgent packet.

**Proof.** There can be at most \(T - 1\) urgent packets at the beginning of \(F_j\) (see Property 2) transferred from \(F_{j+1}\). Urgent packets get served before normal packets and there are \(T\) slots in \(F_j\) which is enough to serve all urgent packets including \(p_{Ak}\).

Next, we provide 2 example cases to demonstrate the operation of the frame-counter scheduler. In both of the cases, initially, the buffer is empty and the urgent packet count is 0. \(p_{Ak}\) arrives at node \(h\) between \((h + k - 1)T\) and \((h + k)T\) during frame \(F_j\).

In the first example case (see Example 4 in Fig. 5), \(p_{Ak}\) gets served as a normal packet and \(p_{Ak}\) arrives at \((h + k)T\) and \((h + k + 1)T\) also during frame \(F_j\) (see Property 1). \(p_{Ak}\) is a packet which has an early arrival. \(p_{Ak}\) is in the buffer at the end of \(F_j\) so \(|Q_A| = 1\) \(C_{ANj} = 0\) at the end of \(F_j\) which shows that there is no normal packet that did not get service in \(F_j\). \(p_{Ak}\) is not eligible in \(F_j\) and becomes eligible in the next frame. Hence, \(p_{Ak}\) gets service as a normal packet in \(F_{j+1}\). \(p_{Ak}\) does not get served during \(F_{j+2}\) and gets service as an urgent packet in \(F_{j+3}\).
Lemma 3. Let \( p_k \) be a normal packet that arrives in \( F_j \) and let \( |T_{Uj}| \geq 0 \). If \( p_k \) arrives before slot number \( |T_{Uj}| + 1 \), it is served within \( F_j \) without becoming an urgent packet for \( F_{j+1} \).

Proof is given in Appendix B.

In the second example case (see Example 5 in Fig. 6), \( p_A(k) \) gets served as a normal packet and \( p_A(k+1) \) and \( p_A(k+2) \) get served as an urgent packet. Although all packets arrive complying to the required traffic shape, the inter-arrival times between \( p_A(k+2) \) and \( p_A(k+3) \) is large and there is a frame border at \( \theta + (h + k + 2)T \) between the service of \( p_A(k+2) \) and arrival of \( p_A(k+3) \). Hence, \( \rho_{AU(j+3)} = 0 \) at the beginning of \( F_{j+3} \). The number of urgent packets can change according to the arrival and service pattern of connection \( A \). If the interarrival times of the packets of connection \( A \) from the upstream node are large enough or connection \( A \) does not utilize its allocated rate 100% all the time, the number of urgent packets can decrease, otherwise, packets of connection \( A \) are continuously transferred to the next frame to be served as urgent packets.

In Fig. 6, although all packets arrive complying to the required traffic shape, the interarrival times between \( p_A(k+1) \) and \( p_A(k+2) \) is large. Hence, \( \rho_{AU(j+3)} = 0 \) at the beginning of \( F_{j+3} \).

The urgent packets get service before the normal packets and the normal packets can get service in any order among each other. Up to now we did not specify an order among the urgent packets.

Consider the following Example 6 in Fig. 7. \( |T_{Uj}| \) shows the length of the urgent period. In this example, \( p_A(k) \) arrives at node \( h \) between \( (h + k - 1)T \) and \( (h + k)T \) during frame \( F_j \). \( p_A(k-1) \) gets service as a normal packet and \( p_A(k) \) is the first urgent packet to get service. At the end of \( F_j \), the total number of urgent packets transferred to \( F_{j+1} \) is \( |T_{Uj+1}| \).

Suppose that \( p_A(k) \) is the last urgent packet to get service. \( p_A(k) \) gets service at the end of \( T_{U(j+1)} \) between \( (h + k)T \) and \( (h + k + 1)T \) and leaves the switch in \( T - 1 \) slots (see the following Lemma 4).

Proof is given in Appendix B.

However this service guarantee in Lemma 4 might not be maintained for the following urgent packets. Assume \( p_A(k+1) \) is also an urgent packet for \( F_{j+1} \) (see Fig. 7). However, for \( F_{j+2} \), more packets are transferred as urgent from \( F_{j+1} \) making \( |T_{U(j+2)}| \) larger than \( |T_{U(j+1)}| \). Suppose that \( p_A(k+1) \) is again the last urgent packet to get service. \( p_A(k+1) \) gets service at the end of \( T_{U(j+2)} \) and cannot satisfy the condition in Lemma 1.

The solution to this problem is to serve the consecutive urgent packets from connection \( A \) at most \( T \) slots apart from each other. The first urgent packet from connection \( A \) satisfies the condition in Lemma 1. If the following urgent packets are served at most \( T \) slots apart from each other, all of them satisfy the condition in Lemma 1. We use the urgency order list to create such service for the urgent packets.

The urgency order list contains tokens that hold the connection IDs that have urgent packets. Initially the urgency order list is empty because none of the connections have urgent packets. For connection \( A \), we compute \( p_{AU(j+1)} \) at the end of \( F_j \) and compare this value with \( p_{AU(j)} \). If \( p_{AU(j+1)} \) is larger than \( p_{AU(j)} \) this implies that connection \( A \) has more urgent packets to get service in \( F_{j+1} \). We insert the number

\[
(h+k+1)T \quad (h+k)T \quad (h+k+1)T \quad (h+k+2)T
\]

Arrivals to node \( h \)

\[
F_1 \quad F_2 \quad F_3 \quad F_4 \quad F_5 \quad F_6
\]

<table>
<thead>
<tr>
<th>(h+k)T</th>
<th>(h+k+1)T</th>
<th>(h+k+2)T</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p_A(k) )</td>
<td>( p_A(k-1) )</td>
<td>( p_A(k-2) )</td>
</tr>
</tbody>
</table>

Fig. 6. Example 5.
of tokens for connection \( A \) that is equal to the difference between \( \rho_{AU(j+1)} \) and \( \rho_{AUj} \). Only one token is inserted for a connection as all connections are unit-rate.

At the beginning of each frame, the connections with urgent packets get service in the order of the tokens stored in the urgency order list. Corresponding urgent counters for these connections are decremented. During \( F_{(j+1)} \), if connection \( A \) has a token in the urgency order list but \( C_{AU(j+1)} = 0 \), this indicates that \( \rho_{AU(j+1)} \) is smaller than \( \rho_{AUj} \). Hence, connection \( A \) has no more urgent packets and the corresponding token should be removed from the urgency order list.

**Property 3.** All urgent packets following the first urgent packet get service with a maximum inter-departure time of \( T \).

The tokens are inserted at the tail of the urgency order list. Once a token is inserted for connection \( A \), no other connections that have urgent packets after connection \( A \) can get service before connection \( A \) in the urgent period.

We can identify two cases, in the first case the position of the token for connection \( A \) does not change and the urgent packets for connection \( A \) get service periodically every \( T \) slots. In the second case, a token which is in front of connection \( A \)'s token is removed. The position of connection \( A \)'s token moves forward and during this frame the urgent packet from connection \( A \) gets service earlier than \( T \) slots from the previous urgent packet of connection \( A \). In the following frames the urgent packets get service periodically in this new position in the urgent period.

We present an example to demonstrate the full operation of the frame-counter scheduler (see Example 7 in Fig. 8). Initially all connections have urgent packet counts of 0 and the buffers are empty. First connection \( C \) has an urgent packet and a token with ID \( C \) is inserted in the urgency order list at the end of \( F_j \). In the following frame, connection \( D \) also has an urgent packet and the second token with ID \( D \) is inserted in the urgency order list at the end of \( F_{(j+1)} \). There is no change in the number of urgent packets at the end of \( F_{(j+2)} \), hence, the urgency order list stays the same. Connection \( C \) does not utilize its allocated bandwidth between \( (h+k+1)T \) and \( (h+k+1)T \), \( \rho_{CU(j+1)} \) becomes 0 at end of \( F_{(j+3)} \) and the token with ID \( C \) is removed from the urgency order list.

A pseudocode for maintaining the variables and the urgency order list for connection \( A \) is illustrated in Fig. 9.

At the end of each frame, the urgent packet count for each connection is computed and if the number of urgent packets increase tokens are inserted at the tail of the urgency order list accordingly. Note that if the number of urgent packets decreased, no updates are performed at this stage. At the beginning of each frame, first, the normal and urgent counters for each connection are initialized.

The frame-counter scheduler, then, starts to scan the urgency order list from the head of the list. If there is a token, the scheduler first checks if this connection still has urgent packets by checking the current value of the urgent counter which was updated at the beginning of the frame with the most recent value. If urgent counter is non-zero, then the token is valid and the scheduler serves the connection.

If the urgent counter is zero then the token is invalid. This situation happens when urgency packet count for the current frame decreased from the previous value. The invalid token is removed from the urgency order list and the scan continues. Normal packets are served following the urgent packets.

### 4.2. End-to-end delay and buffer occupancy per node guarantees for the frame-counter scheduler

In this section, we state lemmas and theorems for the QoS guarantees provided by the frame-counter scheduler. The lemmas, theorems and proofs are for unit-rate connec-
tions. In the next section, we show that the multi-rate connections can be decomposed into unit-rate connections and using the same algorithm we can serve the multi-rate connections with the same QoS guarantees as unit-rate connections. We give sketches for the proofs in this section, but we provide complete formal proofs in Appendix E using a complete state machine model for the frame-counter scheduler for unit-rate connections that we develop in Appendix C and D.

Lemma 5. The maximum number of packets queued for a unit-rate connection in any node is 2.

Proof is given in Appendix D for all possible packet arrival patterns. It is easy to see the result in Lemma 1 if we consider the worst case for the statement in Lemma 5 where a connection is utilizing the allocated rate 100% all the time. From Property 1, we can see that during any $m$ consecutive frames, at most $m + 1$ packets can arrive from

```plaintext
Fig. 9. Pseudocode for the operation of the frame-counter scheduler.
```
the upstream node. Assume that during the first frame no packet gets service. Hence, all packets are served as urgent packets. After the first frame there is a packet served in each frame. Hence, at least \( m - 1 \) packets get served. Then during these \( m \) frames, the maximum buffer occupancy at anytime is \( (m + 1) - (m - 1) = 2 \) packets.

**Lemma 1.** Of these urgent packets keep satisfying the condition in Theorem 1. Consider the unit-rate connection \( A \) established on an \( H \)-hop path. The frame-counter scheduler can satisfy the condition in Lemma 1. The urgency order list ensures that the urgent packet following a normal packet departs within one slot of its arrival (see Lemma 4) satisfying the condition in Lemma 1. The urgency order list ensures that the following urgent packets depart in a periodic fashion at most \( T \) slots apart from each other, and the departures of these urgent packets keep satisfying the condition in Lemma 1.

**Theorem 1.** Consider the unit-rate connection \( A \) established on an \( H \)-hop path. The frame-counter scheduler guarantees a maximum end-to-end delay of \( (H + 1)T \) for connection \( A \).

**Proof.** Follows from Lemma 6 and Lemma 1. \( \Box \)

4.3. Operation of the frame-counter scheduler for the multi-rate connections

In this section, we first introduce the decomposition idea and expressing multi-rate connections as unit-rate connections. Next we describe how the multi-rate connections can be served in decomposed form without changing the arrival order of the packets.

Suppose that connection \( A \) has allocated rate of \( \rho_A = m \) such that \( 1 \leq m \leq T \). Connection \( A \) can be decomposed into \( \rho_A \) unit-rate components as \( A_1, A_2, \ldots, A_m \). In each switch, the consecutive packets that arrive from connection \( A \) are logically distributed to the components one by one and treated as packets from distinct unit-rate connections. Let \( p_{Ak} \) be the \( k \)th packet to arrive from connection \( A \). Then \( p_{Ak} \) is assigned to component \( c = k \mod (m) \). We define a batch for connection \( A \) as one group of \( m \) consecutive packets. First batch includes packets \( p_{A1} \) to \( p_{Am} \) and batch \( b \) includes packets \( p_{A_{(b-1)m+1}} \) to \( p_{Am} \). \( p_{Ak} \) belongs to batch \( b = \lfloor k/m \rfloor \). \( p_{Ak} \) can be numbered as \( p_{A_{bc}} \).

We extend the traffic arrival shape defined in Section 3.1 to multi-rate connections such that the packets of connection \( A \) that belong to the each component \( A_i, i = 1, \ldots, m \) comply with the traffic shape individually. The first packet to arrive for this component \( A_i \) is \( p_{A_{1i}} \), the second packet is \( p_{A_{2i}} \), and \( k \)th packet is \( p_{A_{ki}} \). In the extended traffic shape, suppose that \( p_{A_{1i}} \) completes its arrival at the network before slot \( T \), \( p_{A_{2i}} \) completes its arrival after \( T \) and before \( 2T \). Then, the \( k \)th packet, \( p_{A_{ki}} \), completes its arrival after slot \( (k - 1)T \) and before slot \( kT \).

The components of connection \( A \) have their own state variables which are updated individually. The arrival order of the packets of connection \( A \) have to be preserved from node to node.

In the previous sections of this paper, all of the connections are unit-rate. For unit-connections we did not specify any service order among the connections which have normal packets. We also did not specify any order of inserting tokens to the urgency order list if there are multiple tokens to insert at the end of a frame.

Different from independent unit-rate connections, the unit-rate components of a multi-rate connection have to get service in round-robin order to preserve the arrival order of packets. The round-robin order has to be maintained for both the components that have normal packets and the components that have urgent packets. If all unit-rate components of a multi-rate connection have normal packets they are served in round-robin fashion. Suppose that a group of normal packets that belong to the unit-rate components of the same multi-rate connection \( A \) did not get service in \( F_j \). Hence, the urgent packet counts of these components become 1 at the end of \( F_j \) and tokens should be inserted at the end of the urgency order list. The tokens for these components are inserted following the round-robin order that they would get service in \( F_j \). Hence, the round-robin service order continues for connection \( A \) in \( F_{(j+1)} \) from where it stopped in \( F_j \). These urgent packets from connection \( A \) are served before the normal packets of the unit-rate components of other multi-rate connections but the internal FCFS order of the packets that belong to connection \( A \) is always preserved. Example 8 in Fig. 10 demonstrates the service for multi-rate connections. Note that, only the service order for the unit-rate connections that are the components of the same multi-rate connection is specified. The order among the unit-rate connections that are components of different multi-rate connections is arbitrary.

First batch of connection \( A \) arrives within frame \( F_1 \). This batch contains packets from \( p_{A11} \) to \( p_{A1m} \). Suppose that all of the packets in the first batch get served as normal packets. The unit-rate component connections are served in round-robin fashion and arrival order of packets is preserved. The last component to get served is \( A_{1m} \). Second batch of connection \( A \) arrives within frame \( F_2 \). This batch contains packets from \( p_{A21} \) to \( p_{A2m} \). Round-robin service resumes from component \( A_1 \) and packets from \( p_{A21} \) to \( p_{A2m} \) are served as normal packets in \( F_2 \) with \( q < m \). Rest of the second batch packets are transferred to \( F_3 \) as urgent packets.
At the end of $F_2$, the urgency packet counts for components $A_{(q+1)}$ to $A_m$ are 1. Tokens are inserted in the urgency order list in order starting from token for $A_{(q+1)}$ to token for $A_m$. The urgent packets $pA_{2(q+1)}$ to $pA_{2m}$ get served in the urgent period in $F_3$ preserving the internal order. The round-robin service continues from $A_1$ with $pA_{31}$, $pA_{31}$ to $pA_{3q}$ are served in $F_3$ as normal packets. Only packets $pA_{41}$ to $pA_{4r}$ where $r < m$ arrive in $F_4$. Hence, only packets $pA_{4(q+1)}$ to token for $pA_{4r}$ are transferred to $F_5$ as urgent packets. During urgent period in $F_5$ these packets get service and the rest of the tokens starting from $A_{(r+1)}$ to $A_m$ are deleted from the urgency order list.

4.4. Implementation issues and scheduling complexity

Generalized implementation of the frame-counter scheduler which can work with multi-rate connections requires four variables per connection, $\rho_A$, $\rho_{AU}$, $C_A$ and $C_{AU}$. The scheduling process has three frequencies of updating these variables. $\rho_A$ is computed during connection admission process according to the bandwidth demand of connection $A$. $\rho_{AU}$ is computed every frame and tokens are inserted into the urgency order list accordingly. Urgency order list is scanned and $C_{AU}$, and $C_A$ are updated every slot. The scheduling complexity of the frame-counter scheduler is $O(1)$. There is no sort or search operation required unlike other scheduling algorithms. We can implement the urgency order list with a simple linked list. The maximum length of the urgency order list is $T - 1$ which is the maximum number of urgent packets buffered at an output port.

Frame-counter scheduler can work with the multi-rate connections without decomposing the connections and support the same QoS guarantees. In this case $\rho_A$, $\rho_{AU}$, $C_A$ and $C_{AU}$ can take values between 0 and $T - 1$. The algorithm presented in the pseudocode in Fig. 9 can directly be applied.

The implementation details of the frame-counter scheduler that can work with multi-rate connections without decomposition is beyond the scope of this paper.

5. Performance evaluation

In this section, we present the simulation results for a network consisting of nodes which use frame-counter scheduler. We use our own simulator. There are 10 nodes connected in a linear fashion ($H = 10$). We assume that all links have the same speed. Hence, a frame time is $T$ slots. Traffic sources are attached to connections set up in parking-lot fashion through these nodes. (See Fig. 11.)

The switching nodes have $T$ input ports and $T$ output ports. There is a unit-rate connection set-up for each input, output port pair. Hence, each output port receives packets from $T$ distinct input ports.

The connections set-up from source node 1 to destination node 10 are foreground connections. In our experiments, we observe the behaviour of these foreground connections. The foreground connections compete with the background connections which start at each node and terminate at the following node. One of these connections is a foreground connection and the rest are background connections.

The incoming traffic in our experiment is generated according to the traffic shape defined in Section 3. In our experiments, the traffic sources attached to foreground connections always generate one packet per frame time. Thus, the allocated rate for these connections are always 100% utilized. We increase the utilization for the background connections from 10% to 100% in 10% increments and examine the change in the performance of foreground connections regarding end-to-end delay and delay variation (jitter) over the 10-hop network. We also demonstrate
the average buffer occupation at the output ports. When background connections are 100% utilized, the entire network is fully utilized representing the worst case. The experiments are performed with enough number of packets so that the confidence interval for the mean delay is 4% with 99% confidence.

5.1. Simulation results: end-to-end delay

We test the performance of the frame-counter scheduler with three different frame sizes of 200, 400 and 800 slots. Note that, if the packet size is kept constant as \( T \) is increased, the end-to-end delay bound increases. If the end-to-end delay bound is kept constant, then the packet size should be scaled accordingly. The end-to-end delay bound guaranteed by the proposed frame-counter scheduler in the experiment set-up described above is \((H+1)T\) which is equal to 2200, 4400 and 8800 slots for the three frame sizes that we test. Fig. 12 shows the average observed end-to-end packet delays in the network normalized by the respective delay bound. We see that at 100% background connection utilization, the average delay observed is less than 50% of the theoretical bound. We also observe that the average end-to-end delay measurements normalized with respect to the delay bound remains fairly constant over three different frame sizes.

Fig. 13 shows the distribution of the end-to-end packet delays in the network for 80% and 100% background connection utilization which represent heavily loaded network conditions. The maximum observed end-to-end delay does not exceed 50% of the delay bound. Note that the packet delay in the frame-counter scheduler is not just queuing delay. There is an additional delay component that is due to packets arriving too frequent and are not eligible to get service as soon as they arrive.

Jitter is also an important performance metric for real-time traffic transport. We define the end-to-end jitter as the variation between the minimum and maximum observed end-to-end delay values. It is possible that a packet is sent out as soon as it arrives at a node without any delay. Hence, the maximum end-to-end jitter is bounded by the end-to-end delay bound. Frame-based S&G scheduler assumes that the \((r, T)\) traffic model is defined according to a time origin similar to our input traffic specification. Hence, we find it reasonable to compare our results with S&G. In [6] it is stated that the end-to-end packet delay of S&G algorithm is distributed between \(HT\) and \(2HT\). In Fig. 13, we observe that for 80% and 100% background connection utilization, the end-to-end packet delay distribution for the frame-counter scheduler spans about 20% and 30% of the entire delay bound (which is the jitter bound at the same time), respectively.

The probabilities are expressed in log scale on y-axis. The observed end-to-end jitter is also much lower than the bound. The frame-counter scheduler yields a much more predictable network behaviour compared to S&G. Thus, the frame-counter scheduler provides a lower average end-to-end packet delay and a smaller delay variance in addition to the delay bound that is half of the delay bound provided by S&G.

5.2. Simulation results: output buffer occupancy

We collect the buffer occupancy statistics at the output ports. Fig. 14 demonstrates the average and maximum output buffer occupancy measured in packets observed in the experiment. The average figures show the output buffer occupancy value averaged over time and all output ports. The average buffer occupancy does not exceed 30% of the limit value for different frame sizes. The maximum buffer occupancy figures show the maximum value of packets observed in any output port during the whole experiment. Similarly, the maximum buffer occupancy does not exceed 50% of the limit value.

6. Conclusion

In this paper, we have proposed a novel frame-based QoS scheduler called the frame-counter scheduler. The frame-counter scheduler provides end-to-end bandwidth guarantee and significantly reduces the end-to-end delay bound and buffer requirement for real-time traffic provided by the other frame-based schedulers such as S&G and HRR. The frame-counter scheduler can guarantee an end-to-end delay bound of \((H+1)T\) on an \(H\)-hop path using a frame duration \(T\); while S&G and HRR can guarantee an end-to-end delay bound of \(2HT\) excluding the delay-jitter due to the position of the packet in the frame. The proposed frame-counter scheduler achieves this performance without any synchronization of the frame boundaries or any communication among the nodes. We assume that the incoming packets that carry real-time data are shaped at the network ingress as described in Section 3. The frame-counter scheduler preserves this traffic shape as packets are switched through the network towards their destination.
The frame-counter scheduler works with fixed size packets and a fixed frame duration. However we have a relaxed framing scheme. In our architecture, each node defines its own frame borders. These frame borders might have offsets with respect to each other. The packets for a particular stream can arrive at any time slot within a frame. Using a fixed frame duration in our proposed scheduler simplifies the connection admission tests compared to other QoS schedulers such as RSCP, TCRM and D-EDD. The connection admission for the frame-counter scheduler is only based on bandwidth availability. The end-to-end delay bound is guaranteed as long as (1) is satisfied. Using fixed size frames results in limited bandwidth allocation granularity and the number of connections that can be supported.
on a particular line. A large frame size supports more number of connections and a finer bandwidth granularity. However, if the packet size is constant over different frame sizes, the end-to-end delay bound increases as the frame size increases. The trade-offs in choosing the suitable frame size are discussed in [6].

There are three variables per connection. One of them is computed every frame without any required information about the states of other connections. The other variables are initialized at the frame boundaries and simply decremented every slot. The frame-counter scheduler does not require any sort or search operation leading to a scheduling complexity of $O(1)$.

BWRR algorithm achieves similar end-to-end delay guarantees, buffer requirements and scheduling complexity as in our algorithm. However, we follow a completely different approach. BWRR algorithm requires a periodic input traffic shape which is more constrained than our required traffic shape described in Section 3. BWRR algorithm uses per connection state variables that is storing and tracking the real-time clock instants for scheduling purposes while in our algorithm we use simple integer variables that take at most the value of $T$.

Simulation results show that the average observed end-to-end delay, jitter and buffer occupancy are much less than the guaranteed bounds.

Our proposed scheduler can provide end-to-end QoS guarantees for high-speed real-time communication in a scalable fashion [12].

Appendix A.

Lemma 2 Let $A$ be a connection established on an $H$-hop path. Let the $k$th packet of connection $A$, $p_{Ak}$, arrive at the network after $(k-1)T$ and before $kT$ complying with the traffic shape described in Section 3.1. Let $SW_h$ be the $h$th switch on connection $A$’s path, where $(h = 1, 2, \ldots, H)$.

Condition: Assume that each switch $SW_h$ can guarantee that if the $k$th packet arrives before $(h+k-1)T$ then it is served before $(h+k)T$. Then on an $H$-hop path, the end-to-end delay bound for connection $A$ is $(H+1)T$.

Proof. By the traffic shape defined in Section 3.1, $p_{Ak}$ arrives at $SW_1$ before slot $kT$. Assume that for $SW_h$, $p_{Ak}$ arrives before $(h+k-1)T$ then because of the condition $p_{Ak}$ leaves before $(h+k)T$. Hence, it arrives at node $SW_{(h+1)}$ before $((h+1)+k-1)T$. Because of induction, $p_{Ak}$ arrives at node $SW_H$ before slot $(H+k-1)T$. Because of the condition $p_{Ak}$ leaves node $SW_H$ before slot $(H+k)T$. (See Fig. 15.)

Let $p_{Ak}$ complete its arrival at $SW_1$ at slot $kT - \Delta_a$ and complete its departure from node $SW'_H$ at slot $(H+k)T - \Delta_d$, where;

$$0 < \Delta_a \leq T \text{ and } 0 < \Delta_d \leq T. \quad (A1)$$

The end-to-end delay for $p_{Ak}$ is $Dp_{Ak} = ((H+k)T - \Delta_d) - (kT - \Delta_a) = HT + (\Delta_a - \Delta_d)$.

By (A1) the maximum value for $(\Delta_a - \Delta_d)$ is $T-1$. Thus;

$$Dp_{Ak} = HT + T - 1 < (H + 1)T. \quad \square$$

Note that in [6], the term $(\Delta_a - \Delta_d)$ is defined as the displacement of the packet within the frame and it is not included in the end-to-end delay bound expression. The end-to-end delay in [6] would be $(2H+1)T$ according to our definition of end-to-end delay.

Appendix B.

We internally number the slots in any frame from 0 to $T-1$.

Lemma 3 Let $p_k$ be a normal packet that arrives in $F_j$ and let $|T_{lj}| \geq 0$. If $p_k$ arrives before slot number $|T_{lj}| + 1$, it is served within $F_j$ without becoming an urgent packet for $F_{(j+1)}$.

Proof.

Case 1: Suppose, all normal packets arrive at the same time during $T_{lj}$ to represent the worst case. The number of remaining slots after the urgent period is equal to the number of normal packets; hence, all normal packets can get service within the remaining slots of $F_j$.

Case 2: Suppose that $|T_{lj}| = 0$. If $p_k$ arrives or is present at the node at the first slot of $F_j$, this packet gets served during $F_j$. If all packets are normal packets there can be at most $T$ normal packets. $p_k$ leaves the node before $F_j$ ends even it is served as the last packet at the last slot. A packet becomes urgent when the total number of packets that are queued in the switch are more than the number of slots left until the end of the frame. $\square$

Lemma 4 Let connection $A$ be a unit-rate connection. Assume that $p_{A_{k-1}}$ is served from node $h$ as a normal packet and $p_{Ak}$ becomes an urgent packet in node $h$. $p_{Ak}$ leaves node $h$ within $T-1$ slots after its arrival.

Proof. Let $p_{Ak}$ be an eligible packet for the unit-rate connection $A$ that arrives at node $h$ after $(h+k-2)T$ and before $(h+k-1)T$ in $F_j$. $p_{Ak}$ has to arrive during the normal period of $F_j$ to become an urgent packet for $F_{(j+1)}$ (see Lemma 3).
Consider Fig. 16. There is an offset of \( \theta \) slots between the frame boundaries of the incoming link and the outgoing link.

Let \( pA_k \) arrive at time instant \( t = tarrpA_k \) during the normal period of \( F_j \); \( pA_k \) does not get service in \( F_j \) and becomes an urgent packet. The total time elapsed until the beginning of the next frame \( F_{j+1} \) is denoted by \( r \).

\[
tarrpA_k = ((h + k - 1)T + \theta) - r \tag{B1}
\]

\( pA_k \) does not get service during last \( r \) slots of \( F_j \) because the output port is busy serving other normal packets. Those \( r \) normal packets are not included in the urgent period of \( F_{j+1} \) (denoted as \( T_{U(j+1)} \)). However, \( pA_k \) becomes an urgent packet and is served in \( F_{j+1} \) during \( T_{U(j+1)} \).

Excluding the normal packets that are served during time period \( r \), we get:

\[
|T_{U(j+1)}| \leq T - r.
\]

The latest service completion time for \( pA_k \) is at the end of \( T_{U(j+1)} \); \( T_{U(j+1)} \) starts at \( t = (h + k - 1)T + \theta \) and finishes at \( t = (h + k - 1)T + \theta + |T_{U(j+1)}| \). The latest service start time, \( tserv \), for \( pA_k \) is the beginning of the last slot included in \( T_{U(j+1)} \), then:

\[
tservpA_k = (h + k - 1)T + \theta + |T_{U(j+1)}| - 1
\]

\[
\leq (h + k - 1)T + \theta + (T - r) - 1. \tag{B2}
\]

We find the maximum delay for \( pA_k \) using the arrival and service completion times as:

\[
DpA_k = tservpA_k - tarrpA_k. \tag{B3}
\]

Inserting (B1) and (B2) into (B3) we get:

\[
DpA_k \leq ((h + k - 1)T + \theta + (T - r) - 1
- ((h + k - 1)T + \theta) - r \leq T - 1. \tag{\square}
\]

**Appendix C. State machine model of the frame-counter scheduler**

In this section, we model the behaviour of the frame-counter scheduler as a state machine for unit-rate connections. We define the states with respect to a connection and show the changes in buffer size of the connection and the delay of the packets in the node. A state for connection \( A \) is defined as a 3-tuple \( (|Q_A|, C_A, \rho_{AU}) \), where \( |Q_A| \) is the size of the buffer for connection \( A \), \( C_A \) is value of the main counter for connection \( A \), and \( \rho_{AU} \) is the value of the urgent packet count as defined in Section 4. Since states are defined for any connection, we drop the subscript \( A \) in this section and represent the states as \( (|Q|, C, \rho_U) \).

Without any loss of generality, we consider the first node in the network \( h = 1 \) and the first packet is numbered \( 1 \). \( tarrp_k \) shows the arrival time of \( p_k \) and \( tservp_k \) shows the departure time of \( p_k \). We assume that the packet arrivals are governed by the traffic shape defined in Section 3. Hence, \( tarrp_1 \) is before \( T \) and \( tarrp_2 \) is between \( T \) and \( 2T \) and \( tarrp_k \) is between \( (k - 1)T \) and \( kT \). This assumption corresponds to having an upstream node whose frame boundaries are at \( kT \) coinciding with the arrival frame borders.

The state variables change when there is a frame border (\( F \)-transition), when a packet arrives (\( A \)-transition) and when a packet gets served (\( S \)-transition). We can list the rules that govern the states and state transitions based on the operation principles of the frame-counter scheduler and the traffic arrival pattern.

**C.1. Rules for constructing the state machine**

1. States transitions occur at the frame borders, when packets arrive and when packets get served (\( F \), \( A \) and \( S \), respectively).
2. Initial state: \( (|Q| = 0, C = 1, \rho_U = 0) \). There are no packets in the queue, there are no urgent packets and the main counter is set to 1.
3. At most 2 \( A \)-transitions and 1 \( S \)-transition can happen between 2 \( F \)-transitions. See Property 1 for \( A \)-transitions. The main counter ensures that at most 1 packet is served from a unit-rate connection per frame.
4. \( C \) is restored to 1 and \( \rho_U \) is set to \( \min(|Q|, (\rho_U + C)) \) after an \( F \)-transition. These updates are defined in the operation of the scheduler.
5. If \( \rho_U \) is 1 there must be an \( S \)-transition before the following \( F \)-transition. If there is an urgent packet at the beginning of a frame, it must be served during this frame (see Lemma 2).
6. \( |Q| \) is incremented after an \( A \)-transition.
7. \( |Q| \) is decremented and \( C \) is decremented to 0 after an \( S \)-transition. These updates are defined in the operation of the scheduler.
8. There is no \( S \)-transition if \( |Q| = 0 \) or \( C = 0 \). A packet is served if it exists and if the main counter is greater than 0.
9. There must be an \( S \)-transition after \( (|Q| = 1, C = 1, \rho_U = 0, E) \) before the following \( F \)-transition. Verification of this rule is in the next section.
10. There are no \( A \)-transitions which are possible from states \( (|Q| = 2, C = 1, \rho_U = 1), \ (|Q| = 2, C = 1, \rho_U = 1) \).
\( \rho_U = 0, E \}, \{ |Q| = 1, C = 0, \rho_U = 0, E \}. \) Verification of this rule is in the next section.

The state machine built according to these rules is shown in Fig. 17.

C.2. Verifying the state machine

**Lemma C1.** The state machine in Fig. 17 is the state machine that defines the complete behaviour of the frame-counter scheduler.

**Proof.** All rules except for rules 9 and 10 are directly drawn from the operation of the frame-counter scheduler. Next, we verify rules 9 and 10.

**Verification of Rule 9:** If an \( F \)-transition leads to a state \( \{ |Q| = 1, C = 1, \rho_U = 0 \} \), this indicates an early arrival as introduced in Example 4 (See Fig. 5) in Section 4.1. The early packet is a normal packet present at the node at the beginning of the frame which gets service before the end of the frame. Hence, there must be an \( S \)-transition before the following \( F \)-transition. We define new states, to differentiate early arrivals using a 4th state variable \( E \). If there is an early arrival \( E \) is set to 1. If \( E = 0 \), we drop it from the state representation.

**Verification of Rule 10:** We follow all of the transitions (without loops in them) in the state machine starting from the initial state and leading to states \( \{ |Q| = 2, C = 1, \rho_U = 1 \} \), \( \{ |Q| = 2, C = 1, \rho_U = 0, E \} \), \( \{ |Q| = 1, C = 0, \rho_U = 0, E \} \). Then we show that an \( A \)-transition is impossible from this state and the state space is closed as represented in Fig. 17. Arrival times of packets \( p_1, p_2, p_3 \) and \( p_4 \) are denoted by \( a, b, c, \) and \( d \), respectively, and service times of these packets are denoted by \( a', b', c' \) and \( d' \).

**Case 1.** State transition sequence: \((0,1,0)(A) \rightarrow (1,1,0)(A) \rightarrow (2,1,0)(F) \rightarrow (2,1,1)\) (see Fig. 18).

\[ a \in [0, T - 1], b \in [T, \theta + T], \] next possible arrival is in \([2T, 3T - 1] \).

\( p_1 \) does not get served in \( F_1 \) and transferred as an urgent packet to \( F_2 \), \( p_1 \) gets served as the first urgent packet, where \( a' \leq a + T - 1 < 2T - 1 \) (by Lemma 4). \( p_3 \) arrives at \( tar,p_3 = c \), where \( c \in [2T, 3T - 1] \). \( p_1 \) leaves before \( p_3 \) arrives. Hence, there is no arrival transition after \((2,1,1)\).

**Case 2.** State transition sequence: \((0,1,0)(A) \rightarrow (1,1,0)(F) \rightarrow (1,1,1)(S) \rightarrow (0,0,1)(A) \rightarrow (1,0,1)(A) \rightarrow (2,0,1)(F) \rightarrow (2,1,1)\) (see Fig. 19).

\[ a \in [0, T - 1], b \in [\theta + T, 2T - 1], c \in [2T, \theta + 2T - 1] \]

and \( d \in [3T, 4T - 1] \), \( p_4 \) does not get served in \( F_1 \) and transferred as an urgent packet to \( F_2 \), \( p_1 \) gets served as the first urgent packet, where \( a' \leq a + T - 1 < 2T - 1 \) (by Lemma 4). \( p_2 \) does not get served in \( F_2 \) and transferred as the second urgent packet to \( F_3 \), \( p_2 \) gets service at most \( T \) slots after \( p_1 \) gets service (See Property 3). \( p_2 \) gets service at \( tserv,p_2 = b' \), where \( b' \leq a' + T \leq 3T - 1 \) which is before the arrival time of \( p_4 \). Hence, there is no arrival transition after \((2,1,1)\).

**Case 3.** State transition sequence: \((0,1,0)(A) \rightarrow (1,1,0)(F) \rightarrow (1,1,1)(A) \rightarrow (2,1,1)\) (see Fig. 20).

\[ a \in [0, T - 1] \] and \( b \in [\theta + T, 2T - 1] \). \( p_1 \) does not get served in \( F_1 \) and transferred as an urgent packet to \( F_2 \), \( p_1 \) gets served during \( F_2 \) at \( tserv,p_1 = a' \) as the first urgent packet, where \( a' \leq a + T - 1 < 2T - 1 \) (by Lemma 4). \( p_3 \) arrives at \( tar,p_3 = c \), where \( c \in [2T, 3T - 1] \). \( p_1 \) leaves before \( p_3 \) arrives. Hence, there is no arrival transition after \((2,1,1)\).
Case 4. State transition sequence: \((0,1,0,\text{A})(1,1,0,\text{E})\) in Fig. 21. Case 4 in verification of Rule 10.

Case 5. State transition sequence: \((0,1,0,\text{A})(1,1,0,\text{E})\) in Fig. 22. Case 5 in verification of Rule 10.

Early arrivals: Next, we give an example in Fig. 23, to demonstrate the early arrivals on the completed and verified state machine.

Case 4. State transition sequence: \((0,1,0,\text{A})(1,1,0,\text{E})\) in Fig. 21. Case 4 in verification of Rule 10.

Case 5. State transition sequence: \((0,1,0,\text{A})(1,1,0,\text{E})\) in Fig. 22. Case 5 in verification of Rule 10.

C.3. Packet service times and delays

In this section, we trace all of the packet arrivals (A-transitions) to their corresponding S-transitions on the
state machine developed in the previous sections (See Fig. 17). We investigate the service time and status of these packets to compute the departure times and maximum delay in the node for all $p_k$. From the state machine, we see that the maximum queue size is 2. Hence, an arrival can be the first ($|Q| = 0$ before arrival) or second packet ($|Q| = 1$ before arrival) to get service. We differentiate the arrivals accordingly.

**Property C1.** Suppose that $p_k$ arrives as the first packet. Arrival of the following packet before service of $p_k$ does not affect service time of $p_k$.

Without any loss of generality, we consider the first node in the network ($h = 1$) and the first packet is numbered 1. We assume that the packet arrivals are governed by the traffic shape defined in Section 3 as in Appendix C.

**Lemma C2.** If $p_k$ arrives before $kT$ and after $(k - 1)T$, $p_k$ departs before $(k + 1)T$. The maximum delay a packet has in a node is limited to $2T - 1$.

**Proof.** There are seven arrival transitions on the state machine in Fig. 17. Next, we list the possible state transitions for these arrivals until their respective $S$-transitions.

**Arrival 1:** $(0, 1, 0)(A) \rightarrow (1, 1, 0): p_k$ arrives before $kT$ and after $(k - 1)T$. $p_k$ is the first packet to get service. $p_k$ leaves before $(k + 1)T$. See the cases below:

- **Case 1.1:** $(0, 1, 0)(A) \rightarrow (1, 1, 0)(S)$: The packet is served in the same frame: $D_{pk} \leq T - 1$.
- **Case 1.2:** $(0, 1, 0)(A) \rightarrow (1, 1, 0)(F) \rightarrow (1, 1, 1)(S)$: One frame border between arrival and service, $D_{pk} \leq 2T - 1$. Tighter bound: The packet is served in the next frame as the first urgent packet: $D_{pk} \leq T - 1$.
- **Case 1.3:** $(0, 1, 0)(A) \rightarrow (1, 1, 0)(F) \rightarrow (2, 1, 0)(F) \rightarrow (2, 1, 1)(S)$: One frame border between arrival and service, $D_{pk} \leq 2T - 1$. Tighter bound: The packet is served in the next frame as the first urgent packet: $D_{pk} \leq T - 1$.
- **Case 1.4:** $(0, 1, 0)(A) \rightarrow (1, 1, 0)(F) \rightarrow (2, 1, 0)(S)$: The packet is served in the same frame: $D_{pk} \leq T - 1$.
- **Case 1.5:** $(0, 1, 0)(A) \rightarrow (1, 1, 0)(F) \rightarrow (1, 1, 1)(F) \rightarrow (2, 1, 1)(S)$: One frame border between arrival and service, $D_{pk} \leq 2T - 1$. Tighter bound: The packet is served in the next frame as the first urgent packet: $D_{pk} \leq T - 1$.

**Arrival 2:** $(0, 0, 0)(A) \rightarrow (1, 0, 0): p_k$ arrives before $kT$ and after $(k - 1)T$. $p_k$ is the first packet to get service. $p_k$ leaves before $(k + 1)T$. See the cases below:

- **Case 2.1:** $(0, 0, 0)(A) \rightarrow (1, 0, 0)(F) \rightarrow (1, 1, 0)(S)$: One frame border between arrival and service, $D_{pk} \leq 2T - 1$.
- **Case 2.2:** $(0, 0, 0)(A) \rightarrow (1, 0, 0)(F) \rightarrow (1, 1, 0)(E)$: One frame border between arrival and service, $D_{pk} \leq 2T - 1$. Case 2.1 and Case 2.2 are identical for $p_k$ (See Property C1). See Fig. 23. $p_k$ is mapped to early arrival packet $p_2$ in Fig. 23, which arrives before $2T$ and after $T$ and gets service before $3T$. Hence, $p_k$ leaves before $(k + 1)T$.

**Arrival 3:** $(0, 0, 1)(A) \rightarrow (1, 0, 1): p_k$ arrives before $kT$ and after $(k - 1)T$. $p_k$ is the first packet to get service. $p_k$ leaves before $(k + 1)T$. See the cases below:

- **Case 3.1:** $(0, 0, 1)(A) \rightarrow (1, 0, 1)(F) \rightarrow (2, 1, 1)(S)$: One frame border between arrival and service, $D_{pk} \leq 2T - 1$.
- **Case 3.2:** $(0, 0, 1)(A) \rightarrow (1, 0, 1)(F) \rightarrow (1, 1, 1)(S)$: One frame border between arrival and service, $D_{pk} \leq 2T - 1$. Case 3.1 and Case 3.2 are identical for $p_k$ (See Property C1).

**Arrival 4:** $(1, 1, 0)(A) \rightarrow (2, 1, 0): p_k$ arrives before $kT$ and after $(k - 1)T$. $p_k$ is the second packet to get service, $p_k$ leaves before $(k + 1)T$. See the cases below:

- **Case 4.1:** $(1, 1, 0)(A) \rightarrow (2, 1, 0)(E)$: One frame border between arrival and service, $D_{pk} \leq 2T - 1$. Case 4.1 is mapped to packet $p_2$ in Fig. 18, which arrives before $2T$ and after $T$ and gets service before $4T$. Hence, $p_k$ leaves before $(k + 1)T$.

**Arrival 5:** $(1, 1, 0)(A) \rightarrow (2, 1, 0): p_k$ arrives before $kT$ and after $(k - 1)T$. $p_k$ is the second packet to get service, $p_k$ leaves before $(k + 1)T$ and $D_{pk} \leq 2T - 1$.

**Arrival 6:** $(1, 0, 1)(A) \rightarrow (2, 0, 1): p_k$ arrives before $kT$ and after $(k - 1)T$. $p_k$ is the second packet to get service, $p_k$ leaves before $(k + 1)T$. See the cases below:

- **Case 6.1:** $(1, 0, 1)(A) \rightarrow (2, 0, 1)(F) \rightarrow (2, 1, 1)(S)$: One frame border between arrival and service, $D_{pk} \leq 2T - 1$. Case 6.1 and Case 6.3 are identical for $p_k$ (See Property C1).

See Fig. 23. $p_k$ is mapped to packet $p_2$ in Fig. 19, which arrives before $2T$ and after $T$ and gets service before $3T$. Hence, $p_k$ leaves before $(k + 1)T$.
Lemma 5 The maximum number of packets queued for a unit-rate connection in any node is 2.

Proof. See the state machine developed and verified in Appendix C (Fig. 17). The number of packets in the queue never exceeds 2.

Corollary 5 The maximum delay that a packet can stay in a node is $2T - 1$.

Proof. See the state machine developed and verified in Appendix C (Fig. 17) and all possible packet arrival and service sequences in this Appendix. The node delay of a packet number of packets in the queue never exceeds $2T - 1$. □

Lemma 6 Consider the unit-rate connection $A$ established on an $H$-hop path. The frame-counter scheduler can satisfy the condition in Lemma 1.

Proof. We can look at the arrival shape in Lemma 1 in two parts.

Case 7.1: $(1, 1, 1)(A) \rightarrow (2, 1, 1)(S)$-previous packet $\rightarrow (1, 0, 1)(A)$-following packet $\rightarrow (2, 0, 1)(F) \rightarrow (2, 1, 1)(S)$: one frame border between arrival and service, $D_{pk} \leq 2T - 1$. See Fig. 20. $p_k$ is mapped to packet $p_2$ in Fig. 20, which arrives before 2T and after T and gets service before 3T. Hence, $p_k$ leaves before $(k + 1)T$.

Case 7.2: $(1, 1, 1)(A) \rightarrow (2, 1, 1)(S)$-previous packet $\rightarrow (1, 0, 1)(F) \rightarrow (1, 1, 1)(S)$: one frame border between arrival and service, $D_{pk} \leq 2T - 1$.

Case 7.3: $(1, 1, 1)(A) \rightarrow (2, 1, 1)(S)$-previous packet $\rightarrow (1, 0, 1)(F) \rightarrow (1, 1, 1)(A) \rightarrow (2, 1, 1)(S)$: one frame border between arrival and service, $D_{pk} \leq 2T - 1$. Case 7.2 and Case 7.3 are identical for $p_k$ (See Property C1). See Fig. 23 which is modified version of Fig. 20 with a change in the relative arrival time of $p_3$. $p_k$ is mapped to packet $p_2$ in Fig. 23, which arrives before 2T and after T and gets service before 3T. Hence, $p_k$ leaves before $(k + 1)T$ and $D_{pk} \leq 2T - 1$ (Fig. 24). □

References


