Supporting Loss Guarantees in Buffer-Limited Networks

Mahmoud Elhaddad, Rami Melhem, Taieb Znati
Department of Computer Science
University of Pittsburgh
{elhaddad, melhem, znati}@cs.pitt.edu

***Abstract*— We consider the problem of packet scheduling in a network with small router buffers. The objective is to provide a statistical bound on the worst-case packet loss rate for a traffic aggregate (connection) routed along any network path, given a maximum permissible link utilization (load). This problem is argued to be of interest in networks providing statistical loss-rate guarantees to ingress–egress connections with fixed bandwidth demands.**

We introduce a scheduling algorithm for networks using per-packet transmission reservation. Reservations allow loss guarantees at the aggregate level to hold for individual flows within the aggregate. The algorithm employs randomization and traffic regulation at the ingress, and batch local scheduling at the links. It ensures that a large fraction of packets from each connection are consistently subject to small loss probability at every link. These packets are therefore likely to survive long paths.

To obtain the desired loss-rate bound, we analyze the performance of the algorithm under global routing and bandwidth allocation scenarios that maximize the loss rate of a connection routed along an arbitrary network path. We compare the bound to that obtained using the scheduling algorithm that combines the FCFS service discipline and the drop-tail policy. We find that the proposed algorithm significantly improves the constraints on link utilization and path length necessary to achieve strong loss-rate guarantees.

I. Introduction

A. Motivation and Problem Definition

A buffer-limited router is one where the buffer size at link interfaces is constrained due to technological limitations. As a result, the buffer size cannot be arbitrarily set to prevent or minimize the rate of packet loss events. It is now clear that most high-speed Internet routers will be (in the near future) buffer-limited. This is true whether packet buffers will be implemented using electronic or optical components [1]. In routers with integrated optical buffers, the buffer size may be limited to a dozen packets or less [2].

Recent research [3] has shown that loss-sensitive TCP flows traversing a single buffer-limited work-conserving link are able to withstand the high loss rate and achieve good link utilization under assumptions that limit the contribution of each flow to the total link load.[1]

Several questions regarding the performance of networks of buffer-limited routers (buffer-limited networks) remain open. In this paper, we consider only one that is critical to the usability of such networks:

> *What level of statistical guarantees on the packet-loss rate for users (flows and aggregates thereof) can be supported by a buffer-limited network without severely restricting the maximum allowable link utilization or the maximum path length?*

We consider the above question for a network of time-slotted buffer-limited links where incoming packets are classified at their ingress routers into ingress–egress traffic aggregates (connections). Each connection is assigned a fixed route (a path) through the network and has a fixed bandwidth allocation equal to its demand. We do not assume knowledge of the network topology, bandwidth allocations, or routes.

Loss guarantees are essentially statistical bounds on the loss rate experienced by any connection, and by the flows within the connection. The loss rate of a connection is a function of the characteristics of its path. Specifically, the number of hops it traverses, the packet arrival process and the distribution of packet sizes at the links, and the packet scheduling algorithms used at the links. Obvious measures to reduce the frequency of loss events in a buffer-limited network are to regulate incoming traffic at the ingress routers so as to reduce the burstiness of the packet arrival process of individual connections, and to pack incoming packets into time-slot sized packets (data frames) before injecting them into the buffer-limited network.[2] In this paper, we focus on the role played by the scheduling algorithms in determining the loss rate, assuming that these measures are employed.

A link scheduling algorithm consists of a scheduling discipline that specifies the order of serving packets waiting in the link buffer (e.g., FCFS), and a drop policy that specifies the packet to be dropped whenever a packet arrives to a full buffer (for example, the Drop-Tail policy where the arriving packet is dropped).

Given an expression for the loss rate along any network path under worst-case traffic conditions (routing and bandwidth allocation scenarios) and a particular scheduling algorithm, the desired loss guarantee imposes path length and link utilization constraints on the paths that the network may use to route connections. Obviously, these constraints limit the connection-carrying capacity of the network. This approach of admitting connections based on routing constraints derived from an expression for the worst-case loss rate may be argued by some to be too pessimistic. Nevertheless, it is necessary in the

[1] The packet loss rate at a link is the number of lost packets as a fraction of the total number averaged over some interval of time. The packet loss rate of a flow or flow aggregate over the path it traverses is defined in a similar way.

[2] Traffic regulation requires that the ingress routers be equipped with sufficiently large buffers (i.e., edge routers are not buffer-limited).

1-4244-0476-2/06/$20.00 ©2006 IEEE.

absence of measures to prevent the occurrence of the worst-case conditions if the network is to provide loss guarantees.

In this paper, we describe a new link scheduling algorithm which improves the worst-case loss rate along every network path compared to the algorithm combining FCFS and the Drop-Tail policies (FCFS/Drop-Tail). Consequently, it relaxes the constraints on path length and link utilization. The proposed algorithm is simple and uses only local information: From the perspective of each connection, time is divided into epochs of fixed duration. At every time step the link scheduling algorithm assigns (service and drop) priority to connections in the order of starting their current epochs.

The performance gains under the proposed algorithm are attributed to the following advantage over FCFS/Drop-Tail: When the epoch duration is chosen sufficiently large, packets from each connection, except those at the start of each epoch, are subject to a small loss probability at every link even when the average loss rate at the links is high. As a result, a substantial fraction of packets from each connection are likely to survive long paths, even under heavy link loads. This leads to improved link utilization and path length constraints on connection routing.

The implementation of the scheduling algorithm is presented within a reservation-based transmission control framework where the ingress routers inject packets into the buffer-limited network only to acknowledged (per-packet) transmission requests. This implementation is called BATCH. The framework avoids the following limitations in packet networks that do not use reservations: (1) Flows within a connection suffer random packet loss inside the buffer-limited network even when their collective arrival rate is below the connection's effective bandwidth allocation (the difference of the bandwidth allocation and the loss rate), and (2) individual flows within a connection may experience a higher loss rate than the connection's average. The first limitation affects the throughput of high-bandwidth transfers based on TCP or its high-speed variants [1], [4]. The second limitation particularly affects flows that are few packets in length and thus do not have enough random samples to estimate the true average packet loss rate of the connection. This, for example, may significantly increase the completion time of short-lived TCP flows.

The reservation-based transmission control framework transforms the path of each connection into a lossless pipe of capacity equal to the difference between the connection's bandwidth allocation and the request blocking (loss) rate; thus overcoming the first of the limitations described above. In addition, it allows the ingress routers to overcome the second limitation using policies for sharing the bandwidth of the virtual pipe among the flows within a connection. Any reservation-based transmission control scheme has an associated signaling overhead. However, a reasonable overhead is justified if the reservation-based framework enables the network to support loss-sensitive applications.

B. Results and Contributions

In this paper, we focus on specifying the scheduling algorithm and on analyzing its performance; in particular, the quality of loss guarantees it can support. Although we present the algorithm within a reservation-based transmission control framework, evaluation of other elements of the framework, such as reservation signaling and routing, is not the subject of this paper.

In addition to introducing BATCH, the main contribution of this paper is a bounding analysis for the expected request blocking rate of any network connection under the proposed algorithm. The bound is in terms of the buffer capacity and the load at each link the connection traverses. It is obtained under the simplifying assumption that limited buffering does not cause the pattern of requested slots by any connection to violate a certain "envelope pattern" imposed at the ingress. The assumption is necessary to facilitate the analysis of the algorithm over multihop networks [5]. The resulting bound is validated using simulation.

Although BATCH is defined within a reservation-based environment whereas FCFS/Drop-Tail is a scheduling algorithm for networks without reservations, we can compare the two by observing that with reservations, the packet loss rate of a connection is equal to its request blocking rate when the long-term arrival rate of connection packets at the ingress is equal to the connection's bandwidth allocation (the virtual pipe is fully utilized).[3] Thus, a bound on the request blocking rate of any connection is also a bound on its packet loss rate. Similarly, we define connection's loss rate under FCFS/Drop-Tail (when the network is not using reservations) as the loss rate when the connection's rate of packet arrival at the ingress is equal to its bandwidth allocation. We compare the packet loss rates under BATCH and FCFS/Drop-Tail when the ingress routers inject packets (in the case of FCFS/Drop-Tail) and requests (in the case of BATCH) according to identical arrival envelope processes. From now on, we use the terms "loss rate" and "blocking rate" interchangeably–in the context of a reservation-based network they both refer to request blocking rate, and in the context of a traditional packet network they refer to the packet loss rate.

Numerical and simulation results show that, at small link buffer sizes, BATCH achieves significantly better bounds on the expected loss rate compared to FCFS/Drop-Tail. Equivalently, BATCH results in less stringent constraints on path length and link utilization. For instance, when the buffer size at the links is 5 frames and the load at the links is 60%, the proposed algorithm can support an expected loss rate of 0.02 over paths of length up to 25 hops, compared with less than 5 hops for FCFS/Drop-Tail.

To obtain the loss rate bounds, we first derive a necessary and sufficient condition for maximizing the blocking rate (loss rate in for FCFS/Drop-Tail) at a link given its load. The condition characterizes the "worst-case" routing

[3]The loss rate is the difference between the bandwidth allocation and the virtual pipe capacity.

and bandwidth allocation scenarios w.r.t the blocking rate at every hop along the path of a connection. We then derive the blocking probabilities experienced by a connection routed along an arbitrary network path under the worst-case routing and bandwidth allocation scenarios.

C. Related work

Research related to resource management in networks with limited buffers within the optical packet and burst switching communities focuses primarily on system performance metrics, such as switch and network throughput as opposed to quality of service guarantees (See [6] and references therein).

Work on statistical quality of service guarantees focuses mostly on devising scheduling strategies that support packet delay guarantees assuming that contention resolution buffers are large enough to prevent packet loss (for example, see [7] and reference therein).

Work on statistical loss guarantees typically aims at deriving bounds on the tail probability of the queue size at a link, given an input traffic characterization. The bounds are then used in dimensioning the link buffer size to achieve a desired loss rate [8]. By definition, this approach is not applicable to buffer-limited networks.

The work by Reisslein et al. [5] provides a bufferless-multiplexing framework for supporting statistical delay guarantees in multihop networks. Using traffic regulation at the ingress and bufferless multiplexing at the core, they transform the problem of providing ingress–egress delay guarantees into one of providing loss guarantees. The loss bounds are obtained for a fluid-model of bufferless multiplexing. The fluid model underestimates the packet loss probability as it assumes that flows sharing a link have peak transmission rates that are constant at every time scale. Equivalently, at every time instant a link can serve any number of flows as long as the sum of their peak transmission rates does not exceed the link capacity. This assumption is obviously not satisfied in a packet multiplexer. At the time scale of a packet transmission time, all flows have peak rates at least equal to the link capacity. Consequently, simultaneous arrival of traffic from different flows results in packet loss that the fluid model may fail to capture.

The paper is organized as follows: We describe the reservation-based network environment and the transmission scheduling algorithm in Section II. In Section III, we identify routing and bandwidth allocation scenarios that result in maximal request blocking rate at every link along the path of a connection (loss rate in the case of FCFS/Drop-Tail). These scenarios are used in deriving bounds on the loss rate of a connection under BATCH and FCFS/Drop-Tail. The bound on the expected request blocking rate of a connection under BATCH is obtained in Section IV and the loss rate bound under FCFS/Drop-Tail in Section V. Numerical results characterizing the routing constraints under BATCH are presented in Section VI. The constraints are compared to those achieved by a FCFS/Drop-Tail. Simulation experiments evaluating the accuracy of the bounds are presented in Section VII. Section VIII discusses the computational and storage overhead of the proposed algorithm, followed by concluding remarks in Section IX.

II. The Batch Scheduling Algorithm

We begin with an overview of the reservation-based transmission control framework within which BATCH is defined. In the reservation-based framework, the ingress border router of a connection regularly generates requests for time slots at a rate equal to the connection's bandwidth allocation, and releases time-slot sized data frames (packets)[4] into the network according to the schedule of transmission requests that are successfully acknowledged by the egress.

The transmission requests originate at the ingress router in the form of reservation packets that are processed by core link schedulers as they traverse the connection's path. A link scheduler may block requests due to contention among requests from different connections. As in burst-switched networks [9], [10], reservation traffic is transmitted out-of-band over dedicated control channels to avoid contention between data and reservation traffic. The volume of reservation traffic generated by a connection (reservation overhead) should be made much smaller than the volume of its data traffic by the appropriate choice of the data frame size [11]. We will assume that the capacity of the control channels is dimensioned so that no reservation traffic is lost.

A. Generation of time-slot requests at the ingress

Let μ be the capacity of the first link along the path of a connection c in frames/s. Further, let ρ denote c's bandwidth allocation, also in frames/s. The ingress of connection c chooses the slots to request according to the staircase function $\alpha_{\rho,\mu}(t) = \left\lceil t\frac{\rho}{\mu} \right\rceil$, $t \geq 0$, which specifies the number of requested slots within an interval of length t slots. Precisely, if connection c requests its first slot at τ_0, then for all $\tau \geq \tau_0$, connection c requests slot τ iff $\alpha_{\rho,\mu}(\tau-\tau_0+1) > \alpha_{\rho,\mu}(\tau-\tau_0)$ (cf. Fig. 4). The staircase pattern limits the variation in the number of slots between two consecutively requested slots to ± 1. In the absence of blocking, this minimizes both the maximum and expected inter-frame transmission delay (which contribute to the jitter experienced by arriving packets). It should be clear that the ingress performs an $O(1)$ amount of work per request.

The ingress router periodically sends a reservation packet along the path of the connection. Each reservation packet contains the batch of time slot requests covering a future *reservation epoch* of fixed duration. The time between successive transmissions of reservation packets at the ingress is equal to the duration of a reservation epoch so that a new epoch starts immediately at the end of the preceding one. The duration of time between the transmission of the reservation packet and the start of the corresponding reservation epoch is called the *horizon*. Figure 1 specifies the timing relationships

[4]Recall that packets offered to the network are packed into time-slot sized data frames at the ingress. We use the term "packets" to refer to the data frames when the meaning is clear from the context.

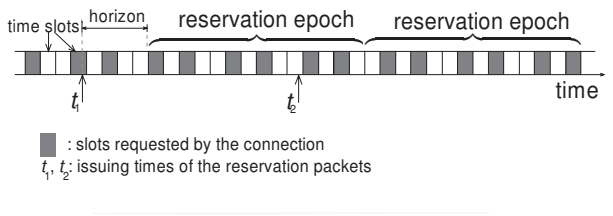

Fig. 1. Timing diagram of the reservation process at the ingress of a connection for two consecutive epochs.

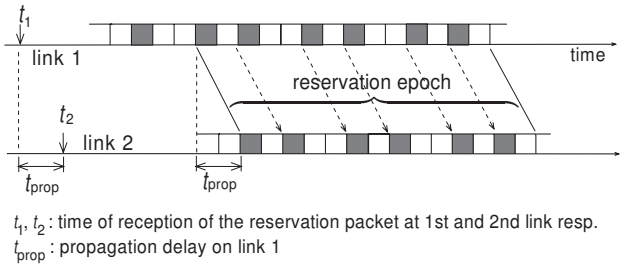

Fig. 2. Timing diagram showing the pattern of requested slots by a connection on two tandem links.

between events in the reservation process for two consecutive reservation epochs at the ingress of a connection.

Beyond the ingress, the start of a reservation epoch and the requested time slots are shifted by each link scheduler along the path of the reservation packet to account for the propagation delay of its upstream neighbor. Figure 2 depicts time slot requests over two tandem links labeled link 1 and link 2 in the order of traversal by the reservation packet.

Upon the initialization of a connection, the ingress picks a phase shift uniformly at random from the length of a reservation epoch. The start time of the first reservation epoch is therefore the sum of the horizon, the phase shift and the time of initialization of the connection. Phase randomization ensures that at each link along the path, the reservation epochs of a connection have a uniform random phase with respect to any duration of length equal to that of a reservation epoch.

B. Contention resolution at core links (service and drop policies)

Core link schedulers process request batches without interleaving in the order of reception of their respective reservation packets. Requests within a batch are sequentially scheduled. Each link scheduler maintains a vector (calendar) of future time slots within a limited horizon indicating whether a slot has been *allocated*, or is *available*. If a connection requests a slot that is not available, the scheduler interchanges the requested slot with the first subsequently available link slot (the scheduler is therefore work-conserving). Since the data frame corresponding to the request will arrive at the link at the beginning of the originally requested slot, it will have to be buffered at the link's interface.

Request blocking occurs whenever the corresponding frame requires buffering through a time-slot during which the buffer is fully occupied (i.e., reserved). A blocked request is either removed or indicated as blocked in the reservation packet before it is forwarded so that it is not scheduled downstream. To keep track of buffer usage, the scheduler associates with each slot in the calendar a variable, its *buffer occupancy*, indicating the number of frames that are to be buffered during that slot.

Figure 3 shows a contention resolution example. The diagram on the left depicts the availability of a sequence of time slots at a particular link and the buffer occupancy during those slots prior to processing a batch of requests. The link buffer

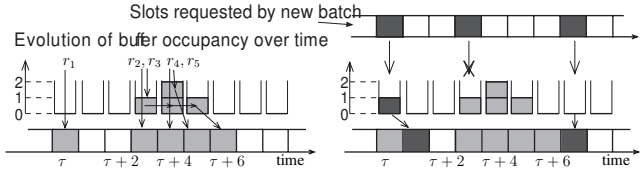

Fig. 3. Contention resolution example.

capacity is taken to be 2 data frames. Shaded slots and buffer places are already allocated to transmission requests. There are two incoming requests (r_2 and r_3) for slot $\tau+3$ and two requests (r_4 and r_5) for $\tau+4$, and there are no requests for slot $\tau+5$ and $\tau+6$. Request r_2 is granted slot $\tau+3$. The data frame corresponding to r_3 will arrive at the link's interface at slot $\tau+3$ and will have to be buffered until the first available slot. The scenario in the figure is one where r_3 is scheduled for transmission after both requests for $\tau+4$. Therefore, the first available slot is $\tau+6$. The requests result in buffer occupancy 1 during $\tau+3$, 2 during $\tau+4$, and 1 during $\tau+5$.

The diagram in the top right specifies the slots requested by the new batch of requests (shown as dark-shaded). Recall that requests within a batch are scheduled sequentially in the order of requested slots. The request for slot τ is granted slot $\tau+1$, thus the corresponding frame has to be buffered during slot τ. The request for slot $\tau+3$ is blocked despite $\tau+7$ being available. This is because the corresponding data frame would have to be buffered during slot $\tau+4$ where the buffer is fully reserved. Finally, the request for slot $\tau+7$ is granted the same slot it requested.

Remarks. The scheduling algorithm is parameterized by a globally defined minimum bandwidth allocation (e.g., in frames/s) and the minimum batch size (number of requests within a batch for a connection having minimum bandwidth allocation). Together, these parameters determine the duration of a reservation epoch, which is common for all connections. The actual batch size for a connection is obtained by multiplying the duration of the reservation epoch by the connection's bandwidth allocation.

The analysis in coming sections will reveal which parameters determine the size of the calendar and the amount of work done to schedule a transmission request at a core link. We therefore defer the discussion of the calendar size and the complexity issues to Section VIII.

At any given link and any time slot, BATCH assigns priority

to connections in the order of starting their current epochs (that is, current w.r.t the time slot): connections with earlier-starting current epochs have higher priority. This is explained as follows. One can see that the order of starting the current epochs is the order of receiving the reservation packets corresponding to these epochs at the link: Consider the connection whose epoch spanning a particular slot τ is earliest-starting among connection epochs spanning τ. If each reservation packets is received before the start of the epoch by a constant duration equal to the horizon, the reservation packet corresponding to the earliest starting epoch is received and processed at the link first among all epochs spanning τ. Thus the connection with earliest starting epoch at τ does not face contention from any connections at that slot. The analysis will reveal that when the epoch duration is chosen large enough, phase randomization ensures that the service priority and hence blocking probability of requests within an epoch gradually improves with time until the beginning of the following epoch.

III. Analysis Framework

In the remainder of the paper, we analyze BATCH and FCFS/Drop-Tail to obtain bounds on the expected blocking rate for a connection routed along an arbitrary network path. The analysis of BATCH is probabilistic due to phase randomization (Section II-A). We analyze FCFS/Drop-Tail assuming that to minimize packet loss, the ingress of a connection performs traffic regulation by injecting packets into the network according to the same deterministic rule used by BATCH to generate time-slot requests. With FCFS/Drop-tail (Section V), randomness arises in the superposition of packet streams of different connections at a link.

The desired bounds are derived in terms of the load at each link along the path of the connection. They are obtained by analyzing the scheduling algorithms under conditions that maximize the blocking (loss) rate at each link, given its load and the regulated arrival process. In this section, we identify a necessary and sufficient condition for maximal blocking rate at a link. We then characterize the network-wide bandwidth allocation and routing scenarios, called **regimes**, under which this condition holds at every link along the connection's path. We begin by stating our modeling assumptions and properties of the pattern of slots requested by a connection.

A. The network model

Let $\mathcal{L} = \{1, 2, \ldots, L\}$ represent the set of links in the network, and let $\mathcal{C}_l$ be the set of connections sharing link $l \in \mathcal{L}$. The capacity of a link l is denoted by μ_l. The bandwidth allocation of a connection c is denoted by ρ_c. Link capacities and bandwidth allocations are expressed as integral numbers of data frames (equivalently, time slots) per unit time (e.g., frames/s). The load at a link is expressed as the sum of the connection bandwidth allocation traversing the link. The analysis is performed under the following assumptions:

(M1) Connection bandwidth allocations and the link capacities are integral multiple of a minimum allocation ρ_0.

(M2) $\sum_{c \in \mathcal{C}_l} \rho_c < \mu_l$ for all $l \in \mathcal{L}$.

(M3) At any link l, the pattern of requested slots by every connection c is constrained by the staircase function α_{ρ_c,μ_l}.

Since ρ_0 can be chosen arbitrarily small, (M1) simplifies the analysis without any loss of generality. Despite (M2), blocking still occurs whenever too many connections contend for a link slot. Assumption (M3) is necessary to facilitate the analysis of the algorithm over multihop networks in the presence of buffers: the pattern of requested slots by a connection c is said to be constrained by α_{ρ_c,μ_l} at l if $\alpha_{\rho_c,\mu_l}(t)$ is the maximum number of slots that can be requested by the connection over any sequence $t \geq 1$ of consecutive slots on l. Let l_1 be the first link along the path of the connection. Since the choice of slots at the ingress is defined by $\alpha_{\rho_c,\mu_{l_1}}$ the request pattern at the ingress satisfies the constraint.

Assumption (M3) holds whenever link traversals may eliminate requests (through blocking) and may increase, but not decrease, the spacing between slots requested by back-to-back surviving requests. In other words, any violation of (M3) must be due to buffering for contention resolution at some link, which may lead to the "bunching up" of transmission requests or packets. Our simulation observations indicate that such violations are infrequent when the buffer capacity used for contention resolution is small, and that the analytical bounds hold in the presence of occasional violations of (M3).

The function α_{ρ_c,μ_l} defines an *envelope pattern* for c starting with its first requested slot at link l.

Definition 1: (Envelope pattern) Let τ_0 be the first slot requested by connection c on a link l along its path. The staircase function α_{ρ_c,μ_l} specifies a sequence of slots $(\tau : \alpha_{\rho_c,\mu_l}(\tau - \tau_0 + 1) - \alpha_{\rho_c,\mu_l}(\tau - \tau_0) = 1)$. We refer to this sequence as the envelope pattern defined by α_{ρ_c,μ_l}.

A connection c constrained by α_{ρ_c,μ_l} requests the maximum allowed number of slots within an interval of consecutive slots of length $t \geq 1$ on link l, only if it requests the same slots specified by the envelope pattern.

B. Properties of the envelope pattern

Suppose that the pattern of slots requested by connection c at some link l is defined by α_{ρ_c,μ_l} (as it is for the choice of slots at the ingress). Then the following properties hold:

(P1) The request pattern defined by α_{ρ_c,μ_l} is periodic.

(P2) Let $T_{0,l}$ be the period size (in slots) of the request pattern defined by ρ_0 (i.e., α_{ρ_0,μ_l}). Then the pattern α_{ρ_c,μ_l} repeats periodically every $T_{0,l}$ slots.

Figure 4 is an example illustrating the periodicity of the envelope pattern (Property (P1)) when $\rho_c/\mu_l = 2/5$. Property (P2) results from the fact that ρ_c is an integral multiple of ρ_0. It states that the envelope pattern of each connection at link l repeats periodically every $T_{0,l}$ slots. We therefore refer to any sequence of length $T_{0,l}$ of consecutive slots on l as the **link period** for link l.

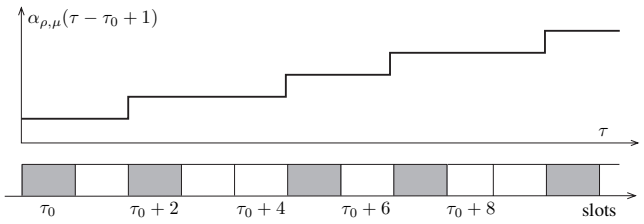

Fig. 4. An example envelope pattern generated by $\alpha_{\rho_c,\mu_l}(\tau-\tau_0+1)$ where $\rho_c/\mu_l = 2/5$. Observe that the pattern repeats periodically every 5 slots.

C. A necessary and sufficient condition for maximal blocking rate at a link

As described in Section II-B, request blocking at a link occurs whenever contention leads to demands for buffers exceeding the buffer capacity at some link slot. Contention at a link maybe affected by blocking at other links in the network, hence we define contention as follows.

Definition 2: (Contention). Let τ be a time slot on some link in the network. Contention at τ, under a regime G (a network-wide routing and bandwidth allocation scenario), denoted χ_τ^G, is a random variable representing the number of requests for slot τ under G.

Higher contention increases the demand for buffering and the rate of request blocking events. Specifically, the probability of demand exceeding the buffer capacity at a slot is at least the probability of contention at the slot exceeding the buffer capacity. If buffer demands at a slot are not entirely caused by contention at that slot, then they must be due to contention at preceding slots.[5] Thus, a regime that maximizes the blocking rate at a link is one that results in maximum contention given the prescribed link load. The relationship between contention and blocking is formally explored in Section IV-A.

Definition 3: (Regimes of Maximal Contention) Let $\mathcal{G}$ be the family of regimes that satisfy a prescribed load at link l. A regime $G^* \in \mathcal{G}$ is a regime of maximal contention at link l if for each slot τ on l, $\Pr\{\chi_\tau^{G^*} > \kappa\} \geq \Pr\{\chi_\tau^G > \kappa\}$ for all $G \in \mathcal{G}$ and $\kappa \geq 1$.

We write that G results in higher contention than G' at a given link as a synonym to $\Pr\{\chi_\tau^G > \kappa\} \geq \Pr\{\chi_\tau^{G'} > \kappa\}$ for all link slots τ and $\kappa \geq 1$.

Theorem 1: Let $\mathcal{G}$ be the family of regimes that satisfy a prescribed load at link l. A regime $G^* \in \mathcal{G}$ is a regime of maximal contention at link l if and only if it satisfies the following condition: For every connection c routed through l under G^*, the pattern of slots requested by c on l is the pattern defined by α_{ρ_0,μ_l}, and slot requests by different connections are statistically independent.

For space considerations, the proof is relegated to [12]. The condition in Theorem 1 can be interpreted as follows: The blocking rate at a link is maximal if only if the connections sharing the link are statistically independent, have minimum bandwidth allocation, and do not experience any blocking at upstream links. Statistical independence implies that no two connections share an upstream links, otherwise contention between these connections would have been resolved upstream.

The blocking rate of a connection traversing a sequence of links is maximal if the blocking rate at each link it traverses is maximal (given the load). Building on the condition of Theorem 1, the blocking rate of a tagged connection traversing a sequence of links is maximal if and only if:

(1) It competes at each link against a different set of "background" connections (statistical independence at every link).
(2) At every link, the background connections do not experience request blocking along the upstream portions of their paths.
(3) The background connections have unit bandwidth allocation and the tagged connection contributes a small portion of the load at every link (i.e., also has unit bandwidth allocation) so that the blocking it experiences upstream does not significantly improve the probability of blocking of its surviving requests.

Together (1), (2) and (3) imply that to obtain an upper-bound on the blocking rate along the path of the tagged connection, each link can be studied in isolation, thus leading to a product-form expression for the overall request blocking probability along the tagged connection's path.[6]

IV. Analysis of Batch

To evaluate the performance of BATCH, we derive a bound on the expected blocking rate of a connection routed along an arbitrary network path. First, for each request within an epoch, we obtain a bound on the blocking probability at any particular link under the condition of Theorem 1. Then, we compute the expected blocking rate of the connection using the consequence of the same theorem (shown above) that the probability of blocking a request at some link along the path has a product-form bound.

A. Blocking at a Link

Consider any particular reservation epoch of the tagged connection c as a *reference* epoch. The reference epoch spans n periods at link l numbered $1, 2, \ldots, n$. Let N be the number of connections sharing link l. Since all reservation epochs have the same duration, the reference epoch overlaps with either one epoch (complete overlap) or two reservation epochs from each of the remaining $N-1$ connections. Complete overlap of reservation epochs occurs when the respective connections start (and end) their epochs at the same link slot (Fig. 5).

Due to phase randomization, reservation epochs from each of $N-1$ connections overlap with c's reference epoch at a uniform random phase. That is, each connection is equally likely to end their current epoch and start a new one at any slot within the reference epoch. To simplify analysis, we make the conservative assumption that epochs starting within the slots of period i of c's epoch, to have started at that period's boundary.

[5] In the example at the end of Section II-B, the buffer demand leading to blocking at slot $\tau+4$ is caused in part by requests for $\tau+3$.

[6] Observe that Theorem 1 validates the common practice of evaluating resource sharing protocols (e.g., congestion control) using the so-called "parking-lot" topology.

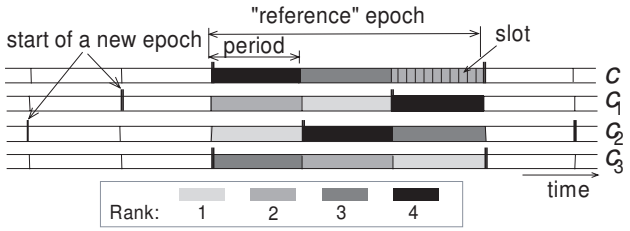

Fig. 5. Illustration of connection ranks.

This implies that each period of the reference epoch belongs to exactly one epoch of each connection, hence covered by one reservation packet (batch of requests) from each of them.

We define the rank of a connection during a period as the order of scheduling of its batch of requests relative to the N request batches covering the period. Recall that request batches are scheduled without interleaving in the order of reception of the corresponding reservation packets at the link's interface. We assume that reservation packets are received in the same temporal order of the starting slots of the corresponding epochs.[7]

Let $R_1, R_2, \ldots, R_n$ be random variables representing the ranks of connection c during the periods of the reference epoch. Over every sample path $R_1, \ldots, R_n$ is a decreasing sequence. This is because, within the reference epoch, each of the connections competing with c has to end an epoch and start a new one, during which it succeeds connection c in rank.

The concept of ranks is illustrated in Figure 5. The diagram shows the reservation epoch of connection c under consideration (the reference epoch) along with the overlapping epochs of the connections competing with c, namely $c_1, c_2,$ and c_3. The duration of a reservation epoch spans $n = 3$ link periods. Connection c's reservation epoch (at the top) overlaps with two epochs from each of c_1 and c_2, and one epoch of c_3. The figure shows the rank of each connection during the periods of the reference epoch. Connection c succeeds c_1 in rank during the first two periods of its reservation epoch and precedes it during the third. This is because the first two periods of c's epoch overlap with an epoch of c_1 that had started earlier, whereas the third period overlaps with the first period of a new epoch of c_1. In this example connection c ranks fourth, third, and second (in that order) in the three periods spanned by its epoch. Connection c_3 precedes c_1 throughout their completely overlapping epochs.

Since the rank during a period is defined by the order of processing reservation packets covering that period, c's requests in period i of the reference epoch, can be blocked only due to contention caused by connections that precedes it in rank. We start by assuming that connection c requests one slot per link period: define the random variables $X_{1,l}, X_{2,l}, \ldots, X_{n,l}$ as

$$X_{i,l} \triangleq \begin{cases} 1 & c\text{'s request within period } i \text{ is blocked} \\ 0 & \text{otherwise.} \end{cases}$$

Since reservation packets are generated periodically at the ingress, the assumption that reservation packets are received at each link's head in the same temporal order of their epochs' starting slots implies that the analysis of $X_{i,l}$, $i = 1, \ldots, n$ applies to every reservation epoch of connection c. Let $X_l = \sum_{i=1}^{n} X_{i,l}$. Then the worst-case expected blocking rate that can be experienced by connection c at link l is given by:

$$\mathbb{E}[X_l/n] = 1/n \sum_{i=1}^{n} \Pr[X_{i,l} = 1], \tag{1}$$

where,

$$\Pr\{X_{i,l} = 1\} = \sum_{k=1}^{N} \Pr\{X_{i,l} = 1 | R_i = k\} \Pr\{R_i = k\}. \tag{2}$$

During period $i > 1$, c faces blocking due to contention among a subset of the connections that preceded it in rank during period $i - 1$. This indicates that $\Pr\{X_{i,l} = 1\}$ is decreasing over the periods of the epoch. In addition, equation (1) is an upper-bound on the c's expected blocking rate when it requests multiple slots per link period. This is because the event that all of c's requests within period i are blocked, occurs with probability no greater than $\Pr\{X_{i,l} = 1\}$, which is obtained under the conditions for maximal contention among the connections preceding c in rank during period i.

In the following subsections, we obtain expressions for the distribution of ranks and bounds on the blocking probabilities, then use the analysis to give some insight about how the algorithm works.

1) Distribution of ranks : For $k = 1, \ldots, N$, $\Pr\{R_i = k\}$, is the probability that for any $r \in \{0, 1, \ldots, k-1\}$, $(k-1)-r$ connections (out of the $N-1$ connections competing with c) start new epochs within the sequence of consecutive periods of length $n - i$ that ends immediately before the first period of c's epoch (the reference epoch), and at least r connections (in addition to c) start new epochs at the beginning of the reference epoch, with c always ranking the $r+1$st among them (see Figure 6). Thus, R_i is stochastically dominated by the number of connections starting new epochs within the $n-i+1$ link periods ending with (and including) period 1 of reference epoch. Precisely, for $1 \le i \le n$, R_i is dominated by a binomial random variable with distribution $\mathrm{Bin}(N-1, 1-\frac{i-1}{n})$.[8] This simple characterization is also a good approximation of the distribution of R_i when n is large enough that the number of connections starting a new epoch during each period is small. In which case we write:

$$\Pr\{R_i = k\} \approx \binom{N-1}{k-1} \left(1 - \frac{i-1}{n}\right)^{k-1} \left(\frac{i-1}{n}\right)^{N-k}. \tag{3}$$

[7]The time between the reception of a reservation packet at a link and the start of the corresponding reservation epoch is equal to the horizon minus the processing delays that the packet encounters upstream. If the processing delays are zero, the assumption about the order of reception of packets holds. Otherwise, the duration of a reservation epoch can be chosen large enough that it becomes unlikely (due to phase randomization) that a large number of connections start new epochs (their reservation packets are received) within the latency resulting from processing delays. Thus making the effect of violating the assumption about the order of reception of packets negligible.

[8]We use the convention that a $\mathrm{Bin}(N-1, 1)$ variable deterministically assumes the value $N-1$.

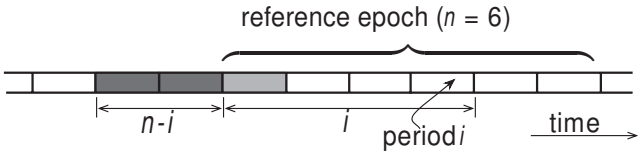

Fig. 6. The Rank of connection c during period i of some reference epoch. Connections that start epochs within the dark-shaded periods precede c in rank during i. Those are in addition to connections starting new epochs during the first period of c's epoch (light-shaded) and whose reservation packets are received before c's packet.

2) Bounding the blocking probabilities : In this section we derive bounds on $\Pr\{X_{i,l} = 1\}$ in terms of expected length of the busy period and the tail probability of buffer occupancy in an $mD/D/1$ queue.

Let the size of the l's period be M slots. Define the backlog at slot t of link l, as the number of frames that will be available at the link's output at the beginning of slot t. This number is the number of frames that are to be buffered during t (if any), in addition to the frame to be transmitted during that slot. Recall that the link scheduler is work-conserving in the sense that if t is not being used to transmit data then the backlog at slot t is zero.

Different connections may see different backlog at t. The backlog perceived by a connection at a slot is due to connections preceding it in rank. That is, the connections whose request batches (reservation packets) covering slot t have already been processed by the link scheduler. Suppose the rank of the tagged connection c during period i of the reference epoch is $R_i = k$ and let $L_{t,k}$ be the backlog at slot t known to the link scheduler at the time it starts processing c's reservation packet covering t.

By assumption (M2) and the periodicity of connection patterns implied by Theorem 1, over any sequence of M slots (any link period) there exist at least one slot with zero backlog after all requests covering the period have been scheduled. This is necessarily true at the time when c's reservation packet covering the period is processed by the link's scheduler. Suppose connection c requests slot t in period i of the reference epoch. Then, by the reasoning above, there is one or more zero-backlog (available) slots in the interval $[t, t + M)$. Let t^* be the first such slot. That is $L_{t^*,k} = 0$ and $L_{\tau,k} > 0$ for $t \le \tau < t^*$. If $L_{t,k} > 0$, $V_k \triangleq t^* - t$ can be interpreted as the residual length of a busy period of the link. Otherwise $V_k \triangleq 0$. The link scheduler assigns t^* to connection c iff $L_{\tau,k} < B$ $(t \le \tau < t^*)$, where B denotes the buffer capacity available at the link's interface. In other words, the request is blocked if c encounters a slot with backlog B within the residual length of the busy period containing t.

The buffer occupancy at the link can never exceed B. However, to derive a bound on $\Pr\{X_{i,l} = 1|R_i = k\}$ it is useful to consider a model with infinite buffer. Clearly, if the buffer is infinite blocking does not occur, and as a result, the tail probability $\Pr\{\tilde{L}_{\tau,k} \ge B\}$, where $\tilde{L}_{\tau,k}$ is the backlog at slot τ as perceived by connection c when the buffer is infinite, becomes an upper bound on $\Pr\{L_{\tau,k} = B\}$. Let $\tilde{t}^*$ be the first slot in $[t, t + M)$ such that $\tilde{L}_{\tilde{t}^*,k} = 0$ and let $\tilde{V}_k = \tilde{t}^* - t + 1$. Then $\tilde{V}_k \ge V_k$ with equality holding only when no blocking occurs in $[t, t^*)$ prior to processing c's reservation packet covering i.

Suppose the link's buffer is infinite. Then the backlog at any slot t can be described as:

$$\tilde{L}_{t,k} = \max\left(\tilde{L}_{t-1,k}, 0\right) + \chi_{t,k}, \tag{4}$$

where $\chi_{t,k}$ is the value of contention at t due to the $k-1$ connections preceding c in rank during i. Consider the M-slots period ending immediately before slot t. There exist a slot $t' : t - M \le t' < t$ such that $\tilde{L}_{t',k} = 0$ and $\tilde{L}_{\tau,k} > 0$ for $t' < \tau < t$. It follows from (4) that $\tilde{L}_{t,k}$ is determined by the vector $(\chi_{t'+1,k}, \chi_{t'+2,k}, \ldots, \chi_{t,k})$. Precisely,

$$\tilde{L}_{t,k} = \sum_{\tau=t'+1}^{t} \chi_{\tau,k} + t - t' - 1. \tag{5}$$

Since the system remains busy in the interval $[t, \tilde{t}^*)$, (5) applies for any τ' in $[t, \tilde{t}^*)$.

Suppose $i > 1$. Eq. (5) implies that t' may fall within period $i-1$. Therefore the backlog at a subset of the slots in $[t, t^*]$ is due to contention at slots in period $i-1$. Since $R_{i-1} \ge R_i$, the backlog at any slot in $[t, t^*]$ is due to contention among at most $R_{i-1} - 1$ connections. For simplicity, we consider the case where n, the number of periods spanned by a reservation epoch, is large enough that the distribution of $\tilde{L}_{\tau,R_{i-1}}$ is approximately equal to that of $\tilde{L}_{\tau,R_i}$.

$\tilde{L}_{\tau,k}$ is backlog at a slot τ in period i due to the superposition of requests from $k-1$ periodic connections over the M-slot period ending with τ, assuming infinite link buffer. Each of the $k-1$ connections has a random is equally likely to request any slot within the period. The distribution of $\tilde{L}_{\tau,k}$ is therefore the same as the backlog witnessed by an external observer at any slot in an mD/D/1 queue where $m = k - 1$, and the period $D = M$.[9] Therefore, we have $\Pr\{\tilde{L}_{\tau,k} \ge B\} = S_{k-1}(B-1)$ where $S_m(r)$ is the probability that the backlog at any particular slot of an $mD/D/1$ queue exceeds r. Furthermore, if t has non-zero backlog, $\tilde{V}_k$ has the same distribution as the residual length of a busy period in the $(k-1)D/D/1$ queue for which the period size $D = M$ slots.

Suppose we inspect random slots in a $mD/D/1$ queue. Let $\tilde{\theta}_m$ be an indicator random variable of the event that the inspected slot falls in a busy period and that within the residual length of the busy period there is at least one slot with backlog no less than B. Then, for $i = 1, \ldots, n$:

$$\Pr\{X_{i,l} = 1|R_1 = k\} \le \Pr\{\tilde{\theta}_k = 1\}.$$

[9] An $mD/D/1$ queue [13] is a FCFS multiplexer of m periodic streams of constant-size (1-slot) "packets". The streams have equal transmission rates of 1 packet every period. The period of each connection is D (D slots between consecutive packets arrivals from the same stream). Taking any D consecutive link slots as a reference, a stream's packet is equally likely to arrive at the link in any of the D slots in that period. Note that the specific service discipline does not affect the queue backlog at a given slot as long as it is work-conserving. The backlog in an $mD/D/1$ is driven by the same rule in (4), (5).

Observe that $\Pr\{X_{i,l} = 1|R_1 = k\}$ is independent of the period index i. Substituting into (2), we get

$$\Pr\{X_{i,l} = 1\} \leq \sum_{k=1}^{N} \Pr\{\tilde{\theta}_k = 1\} \Pr\{R_i = k\}. \quad (6)$$

Let $\mathcal{E}$ denote the conditional event $\{\tilde{\theta}_k = 1|\tilde{V}_k = v\}$ and, $\mathcal{E}_j$, the event $\{\tilde{L}_{t+j,k} \geq B|\tilde{L}_{t+j,k} > 0\}$, $j = 0, 1, \ldots$. Then $\mathcal{E} = \cup_{j=0}^{v} \mathcal{E}_j$. By the subadditivity of probabilities, we have

$$\begin{aligned}\Pr\{\tilde{\theta}_k = 1|\tilde{V}_k = v\} &\leq \sum_{j=0}^{v} \Pr\{\tilde{L}_{t+j,k} \geq B|\tilde{L}_{t+j,k} > 0\} \\ &= v \cdot \frac{S_{k-1}(B-1)}{S_{k-1}(0)}.\end{aligned}$$

Let $\sigma = \frac{S_{k-1}(B-1)}{S_{k-1}(0)}$. Unconditioning by $\tilde{V}_k$, we get

$$\begin{aligned}\Pr\{\tilde{\theta}_k = 1\} &\leq \sigma \sum_{v=0}^{\infty} v \Pr\{\tilde{V}_k = v\} = \sigma \mathbb{E}[\tilde{V}_k] \\ &= S_{k-1}(B-1)\mathbb{E}[\tilde{V}_k|\ t \text{ has non-zero backlog}]\end{aligned} \quad (7)$$

This completes the characterization of the probability bounds in terms of the tail probability of backlog and the expected residual length of a busy period in an $mD/D/1$ queue.

The backlog in an $mD/D/1$ queue is derived in [13] as:

$$\begin{aligned}S_m(r) = 0, \quad &\text{for } r \geq m, \\ &\sum_{j=1}^{m-r} \binom{m}{r+j} \left(\frac{j}{D}\right)^{r+j} \left(1 - \frac{j}{D}\right)^{m-r-j} \frac{D-m+j}{D-j} \\ &\text{for } 0 \leq r < m.\end{aligned} \quad (8)$$

Let Y_k be a random variable representing the length of a busy period in a $(k-1)D/D/1$ queue. Then the pmf of the residual length of a busy period for the same queue $\widehat{Y}_k$ can be expressed in terms of the pmf of Y_k as:

$$\Pr\{\widehat{Y}_k = \hat{y}\} = \sum_{y=\hat{y}}^{D-1} \frac{\Pr\{Y_k = y\}}{\mathbb{E}[Y_k]} \quad (9)$$

Since $\mathbb{E}[\tilde{V}_k|\ t \text{ has non-zero backlog}] \triangleq \mathbb{E}[\widehat{Y}_k]$, it can be obtained from (9) using the pmf of Y_k. The pmf of Y_k is computed in [14]).

Remarks. Because (7) uses the backlog and the residual length of busy period in an infinite-buffer link model to bound the backlog at a link with small finite buffers, the bound is not tight when the load approaches the link capacity. Nevertheless, it allows us to derive practically useful bounds on the overall (ingress–egress) expected blocking rate of a connection and bounds on its tail probability under normal load.

B. Multihop Paths

In this section, we derive bounds on the expected blocking rate experienced by a connection routed along an arbitrary path through the network. Since all link capacities are integral multiples of minimum bandwidth allocation capacity, the number of periods spanned by a reservation epoch at every link is equal to the minimum batch size. Consider the case where a reservation epoch spans n periods on each link, and let $\mathcal{P}_c$ denote the path of the tagged connection, c. Furthermore, let $X_1, X_2, \ldots, X_n$ indicate whether the request in period i of an epoch of connection c has been blocked at any link along the path. In Section III, we concluded that under the conditions of Theorem 1, the blocking probability of a request $\Pr\{X_i = 1\}$ has the following product-form:

$$1 - \Pr\{X_i = 1\} = \prod_{l \in \mathcal{P}_c} (1 - \Pr\{X_{i,l} = 1\}), \quad (10)$$

Let X be the number of blocked requests in the scheduling interval, i.e., $X \triangleq \sum_{i=1}^{n} X_i$. Then $\mathbb{E}[X/n]$ is the expected blocking rate for connection c.

$$\mathbb{E}[X/n] = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}[X_i] = \frac{1}{n} \sum_{i=1}^{n} \Pr\{X_i = 1\} \quad (11)$$

Substituting from (10), we get

$$\mathbb{E}[X/n] = 1 - \frac{1}{n} \sum_{i=1}^{n} \prod_{l \in \mathcal{P}_c} (1 - \Pr\{X_{i,l} = 1\}) \quad (12)$$

C. Why the expected blocking rate is small under BATCH

Now we give some insight about why the expected blocking rate is small under BATCH. Suppose that the number of link periods per epoch is n. In Section IV-A.1 we argued that the rank of a tagged connection during every period of a reservation epoch can be well approximated by a binomial random variable. The mean values of the binomial variables drop linearly across the periods of an epoch. Because the binomial distribution is concentrated around its mean, with high probability, the rank during a period does not exceed the mean of the binomial variable by a significant percentage.

The rank of the connection during a period is, in effect, the number of connections that have higher priority. These are the only connections that contribute to the load as perceived by the tagged connection. Thus we can say that the load perceived by any connection drops linearly throughout each reservation epoch. On the other hand, observe that the loss (blocking) probability at a link is typically a convex function that increases superlinearly with load. That is a small reduction in load typically results in much larger reduction in the blocking probability. Thus except for a few periods at the beginning of every epoch, each connection is subject to a small blocking probability at every link it traverses. Clearly, the value of n should be chosen larger than 2 so that the blocking probability is small in more than half the periods in every epoch. As a guideline, n should be between 10% and 20% of the maximum number of connections that may share a link (based on the unit allocation and the maximum link capacity in the network). This ensures that the load perceived by a connection drops, on average, between 10% and 5% per period throughout every reservation epoch (this drop is a percentage of the rank during the first period of the epoch).

V. Performance of FCFS/Drop-Tail

In this section, we evaluate the performance of FCFS/Drop-Tail by bounding the loss rate of a tagged connection, c, routed along a arbitrary network path. The bounding analysis is performed within the framework of Section III–specifically, under assumptions (M1)–(M3) and the condition specified in Theorem 1 where each link is a multiplexer of periodic streams of unit (minimum) bandwidth allocation. That is each link is an $ND/D/1$ queue where N is the number of connections sharing the link and D is the length of the link period.

Since the FCFS/Drop-Tail strategy does not distinguish between different packets from the same connection, we assume that at each link, packets from any given connection are subjected to iid Bernoulli trials of a given success (loss) probability. Let β_l be the packet loss probability for the tagged connection at link l and let $\mathcal{P}_c$ denote the path of the tagged connection. Then the loss probability over the entire path, which is also the expected loss rate of the connection, is given by

$$\beta = 1 - \prod_{l \in \mathcal{P}_c} (1 - \beta_l). \tag{13}$$

The loss probability, β_l, is only bounded by the overflow probability in a finite $ND/D/1$ queue. Using reasoning similar to Section IV-A.2, this overflow probability is in-turn bounded by $S_N(B)$ (Eq. (8)).

VI. Numerical Results

This section has three objectives: first, to present a method for calculating tradeoffs between the maximum allowable link utilization (U_{max}) and the maximum allowable path length (L_{max}) so that the blocking rate of every successfully routed connections satisfies an expected blocking rate guarantee. These tradeoffs can be used to define constraints on connection routing (for example by fixing L_{max} and deriving the corresponding value for U_{max}, or vice-versa), thus are of practical value. Second, to demonstrate using numerical results that Batch is able to achieve better tradeoffs compared to FCFS/Drop-Tail. Third, to demonstrate the effect of the duration of the reservation epoch (i.e., the number of link periods per reservation epoch, n) on the utilization–path length tradeoffs.

We are interested in tradeoffs where the expected blocking rate is in the order of 10^{-2} when the buffer size at every link is $B \leq 10$ as will be the case in routers with integrated optical packet buffers [3].

The contour plots in Figure 7 are obtained for Batch from Eq. (11) using the bounds in Equations (6) and (7) when the link period is $M = 100$ slots at each link and the number of periods per epoch is $n = 10$. The plots show the tradeoffs between U_{max} and L_{max} at the given buffer sizes. For instance, to guarantee an expected blocking rate of 0.02 when $B = 5$ at each link and L_{max} is 25 hops, the load at each link should not exceed 60%. The same guarantee can be supported when $L_{max} = 25$ and $B = 10$ using $U_{max} \approx 0.81$.

Figure 8(a) shows the tradeoffs supported by FCFS/Drop-Tail at a buffer size $B = 5$ and $M = 100$. The tradeoffs are obtained using Eq. (13). Figures 8(b) and 8(c) show the tradeoffs for Batch at $n = 5$ and $n = 20$, resp., given the same values for B and M. Comparing Figures 8(b) and 8(c), we find that larger values of n yield better tradeoffs, as we expect based on the discussion in Section IV-C. On the other hand, comparing Figures 8(a) and 8(c), shows that Batch improves the tradeoffs compared to FCFS/Drop-Tail. For instance, whereas FCFS/Drop-Tail cannot satisfy a blocking rate bound of 0.02 over paths longer than 8 hops at 60% link utilization, Batch with $n = 10$ and above, can provide the same guarantee over 25-hop paths (from Figure 7(a)). That is, Batch allows the network to route traffic over paths that are unusable under FCFS/Drop-Tail. In the next section we show using simulation that FCFS/Drop-Tail can result in even worse tradeoffs in real settings.

VII. Simulation Results

In this section, we report simulation experiments that validate the derived bounds for Batch, and compare the worst-case blocking rates under Batch and FCFS/Drop-Tail. The simulation experiments are used only to support and validate the analytical/numerical results in previous sections.

A. Accuracy of bounds

In Section IV we obtained an upper bound on the expected blocking rate of a connection routed along an arbitrary path under conditions that maximize the expected blocking rate at each link (conditions (1)–(3), Section III-C). In this section, we compare the bound to the average path blocking rate observed in simulation scenarios satisfying the same conditions. Although almost all of our modeling assumptions were on the conservative side, this validation is required because of assumption (M3) of Section III, which optimistically assumes that buffering does not introduce burstiness in the pattern of transmission requests or packet arrivals as they progress through the network. The assumption was needed to facilitate the analysis over multihop paths.

The above-mentioned conditions are satisfied in the so-called "parking-lot" topology (Fig. 9), where a tagged (foreground) connection traverses a path of identical links. At each link, the tagged connection competes with a different set of background connections. These connections do not face any contention, except at the link shared with the tagged connection. All network connections have identical bandwidth allocations equal to the minimum allocation. Each link along that path is shared by the same number of connections, and thus have equal load.

We conducted experiments at different values of n (number of link periods per epoch), link capacities, and buffer sizes. All results validated the accuracy of the expected blocking rate bound. Figure 10(a) is a contour plot of the observed tradeoff between load and path length to achieve a desired blocking rate. In these experiments, the link period size was set to 100 slots, the minimum bandwidth allocation to 1 slot per period,

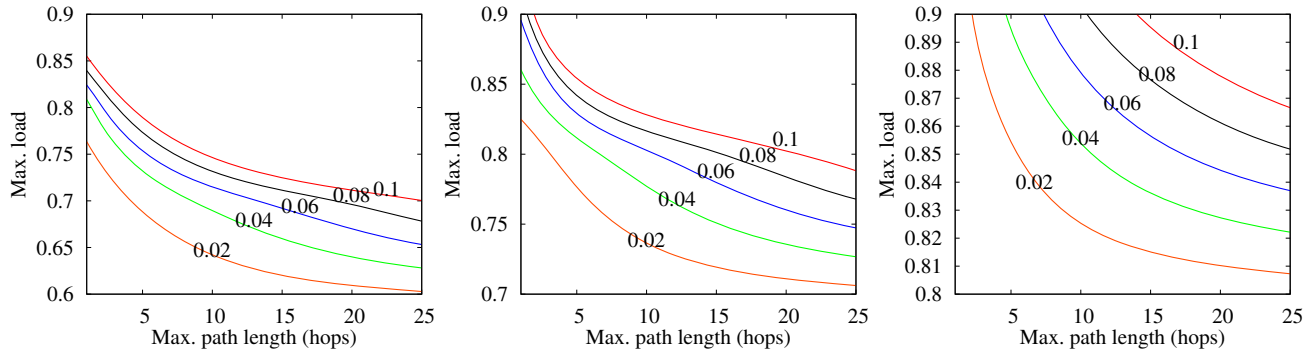

Fig. 7. Path length vs. utilization tradeoffs at constant blocking rate using BATCH with $n = 10$. Left: (a) $B = 5$, middle: (b) $B = 7$, right: (c) $B = 10$.

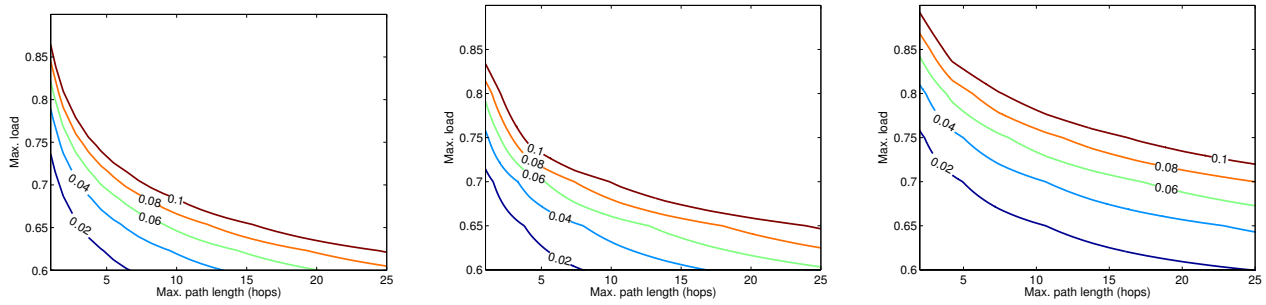

Fig. 8. Path length vs. utilization tradeoffs at constant blocking rate and $B = 5$, Left: (a) FCFS/Drop-Tail, middle: (b) BATCH with $n = 5$, right: (c) BATCH with $n = 20$.

and the buffer size to 5 data frames at every link. The number of link periods per epoch was set to $n = 10$. The plot was obtained by running a set of 100 experiments for each (load, path length) pair and averaging the blocking rate of the tagged connection.

Comparing Figures 10(a) and 7(a), we can see that the tradeoff obtained using the analytical bounds is reasonably close to that obtained from simulation for the purpose of identifying constraints on connection routing. The difference is smaller over long paths where the load has to be significantly less than 1 to achieve acceptable path blocking rate. Over shorter path, acceptable path blocking rate can be achieved at load above 0.8. At such high load, the difference tends to increase because the analytical blocking bounds are obtained using an infinite-buffered link model. Naturally, the analytical tradeoffs are intended for characterizing the blocking performance over long paths.

B. Comparison to FCFS/Drop-Tail packet scheduling

In this section, we compare the tradeoff between load and path length to achieve a desired blocking rate under BATCH to the tradeoff observed in a packet network using FCFS/Drop-Tail. We used the same parking-lot topology above and the same routing and bandwidth allocation scenario. Other than FCFS/Drop-Tail packet scheduling, the links are identical to those described above. Each connection is represented by a constant bit rate (CBR) flow. Thus emulating the pattern of ingress-shaped traffic.

Comparing Figures 10(b) and 10(a) we find that the packet loss rate under FCFS/Drop-Tail is much more sensitive to path length compared to the BATCH. In particular, whereas a blocking rate of 0.02 can be maintained for up to 20 hops at 60% load under the proposed scheme, it can only be maintained for only for less than 5 hops using FCFS/Drop-Tail. Observe that the tradeoff under FCFS/Drop-Tail is slightly worse than the bounds obtained in the previous section. This is due to allowing the CBR sources to randomly perturb the interval between consecutive packet in order to avoid traffic phase effects leading to the starvation of some connections (all transmission requests from those connections are blocked). Traffic phase effects can occur as well in real networks if traffic is shaped into periodic streams.[10] In practical settings however, shapers (such as leaky buckets) allow a degree of burstiness that is similar to the random perturbation. The simulation results are therefore more representative of the performance of FCFS/Drop-Tail than the analytical bounds.

VIII. COMPUTATIONAL AND STORAGE OVERHEAD

In this section, we comment on the computational and storage cost of request scheduling in BATCH. Recall that in Section II, we mentioned that the reservation signaling overhead can be minimized by appropriate choice of the data frame size.

The expected number of inspected calendar slots per request is the expected residual length of the busy period within

[10] See for example [15]

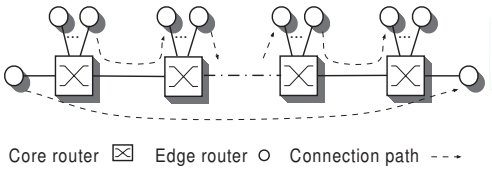

Fig. 9. Simulation topology

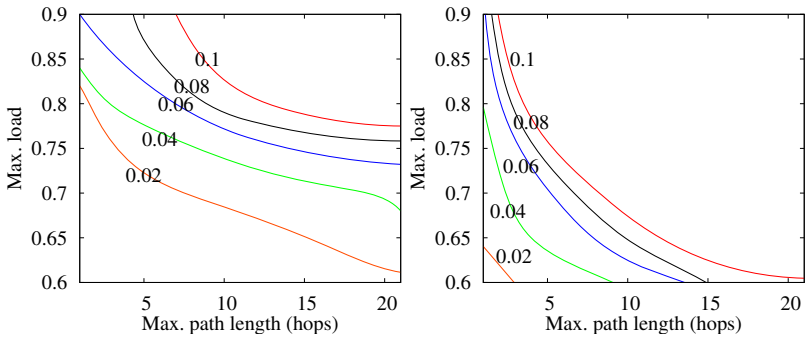

Fig. 10. Observed tradeoff between load and path length to achieve a desired blocking rate when $B = 5$. Left: (a) BATCH ($n = 10$), and right: (b) FCFS/Drop-Tail packet scheduling.

which the requested slot falls (cf. Section IV-A.2) which is typically much smaller than the length of the period, except for connections at the early periods of their epoch.

At any given time, the calendar data structure (Section II) should represent a number of slots equal to two reservation epochs. According to the guidelines in Section IV-C, if the maximum number of connections sharing a link is around 100, then n should be between 10 and 20. When the link period is hundreds of slots, the calendar occupies tens of kilobits of memory per link.

IX. Concluding Remarks

We presented a new link scheduling algorithm for efficiently supporting loss guarantees in buffer-limited networks. The algorithm is simple and uses only local information. The implementation of the algorithm, called BATCH, was presented within a reservation-based transmission control framework. The reservation-based framework transforms the path of each connection into a virtual lossless pipe, thus supporting loss guarantees to individual flows within a connection. This is important to loss-sensitive traffic such as high bandwidth-delay product transfers using TCP.

We probabilistically analyzed the performance of BATCH to obtain a bound on the expected loss rate of a connection routed along an arbitrary network path. The bound was validated using simulation. To demonstrate the potential of BATCH, we used numerical and simulation examples to compare the loss guarantees supported by BATCH to those achieved by the link scheduling algorithm combining the FCFS and Drop-Tail policies in a non-reservation based environment.

There are several avenues for further research. One question that naturally arises is whether there are algorithms that outperform BATCH in a reservation-based environment.[11] We are pursuing a similar question with respect to an implementation of the algorithm in a packet network (without reservations).

[11]It is worth noting that the case where each epoch spans one link period subsumes the standard scheduling algorithm used in burst-switching networks where requests are not batched. BATCH supports better loss guarante since increasing the number of periods per epoch improves performance.

References

[1] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," in *ACM SIGCOMM '04*, August /September 2004.

[2] N. McKeown and D. Wischik, "Hot Topic: Making router buffers much smaller," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 3, pp. 73–74, 2005.

[3] M. Enachescu, Y. Ganjali, A. Goel, N. McKewon, and T. Roughgarden, "Routers with very small buffers," in *IEEE Infocom*, 2006.

[4] D. Barma, G. Smaragdakis, and I. Matta, "The effect of router buffer size on HighSpeed TCP performance," in *GLOBECOM 2004*.

[5] M. Reisslein, K. W. Ross, and S. Rajagopal, "A framework for guaranteeing statistical QoS," *IEEE/ACM Trans. Netw.*, vol. 10, no. 1, pp. 27–42, 2002.

[6] Y. Chen, C. Qiao, and X. Yu, "Optical burst switching: A new area in optical networking research," *IEEE Network*, vol. 18, no. 3, pp. 16–23, May 2004.

[7] C. Li and E. Knightly, "Coordinated multihop scheduling: A framework for end-to-end services," *IEEE/ACM Trans. Netw.*, vol. 10, no. 6, December 2002.

[8] Han S. Kim and Ness B. Shroff, "Loss probability calculations and asymptotic analysis for finite buffer multiplexers," *IEEE/ACM Trans. Netw.*, vol. 9, no. 6, pp. 755–768, 2001.

[9] J. Ramamirtham and J. Turner, "Time sliced optical burst switching," in *IEEE INFOCOM*, 2003.

[10] M. Yoo, M. Jeong, and C. Qiao, "A high speed protocol for bursty traffic in optical networks," in *SPIE'97 Conf. For All-Optical Networking: Architecture, Control, and Management Issues*, 1997, pp. 79–90.

[11] Y. Xiong, M. Vandenhoute, and H. Cankaya, "Control architecture in optical burst-switched WDM networks," *IEEE journal on selected areas in communications*, vol. 18, no. 10, pp. 1838–1851, October 2000.

[12] M. Elhaddad, R. Melhem, and T. Znati, "Supporting bandwidth guarantees in buffer-limited networks," Tech. Rep., University of Pittsburgh, `http://www.cs.pitt.edu/~elhaddad/BCN/`.

[13] J. W. Roberts and J. T. Virtamo, "The superposition of periodic cell arrival streams in an ATM multiplexer," *IEEE Trans. Commun.*, vol. 39, no. 2, pp. 298–303, Feb. 1991.

[14] F. Hübner, "Discrete-time analysis of the output process of an ATM multiplexer with periodic input," Tech. Rep. 36, University of Würzburg, November 1991.

[15] Sally Floyd and Van Jacobson, "Traffic phase effects in packet-switched gateways," *Journal of Internetworking:Practice and Experience*, vol. 3, no. 3, pp. 115–156, September, 1992.