Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

Computer Networks 46 (2004) 219–235

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

Optimal sliding-window strategies in networks with long round-trip delays ☆

Lavy Libman [a,*,1], Ariel Orda [b]

[a] National ICT Australia, Bay 15, Australian Technology Park, Eveleigh, NSW 1430, Australia

[b] Department of Electrical Engineering, Technion—Israel Institute of Technology, Haifa 32000, Israel

Received 24 February 2004; accepted 15 March 2004
Available online 11 May 2004

Responsible Editor: S. Low

Abstract

A method commonly used for packet flow control over connections with long round-trip delays is ‘‘sliding windows’’. In general, for a given loss rate, a larger window size achieves a higher average throughput, but also a higher rate of spurious packet transmissions, rejected by the receiver merely for arriving out-of-order. This paper analyzes the problem of optimal flow control quantitatively, for a connection that has a cost per unit time and a cost for every transmitted packet. The optimal strategy is defined as one that minimizes the expected cost/throughput ratio, and is allowed to transmit several copies of a packet within a window. We present an algorithm for computing the optimal strategy and study its properties; in particular, we derive bounds on the optimal strategy cost/throughput performance, and show that it increases merely *logarithmically* with the time price, whereas the cost/throughput of the ‘traditional’ classic window scheme is linear in the time price.

Keywords: Error control; Retransmission; Forward error correction (FEC); Noncooperative flow control; Delay-bandwidth product

1. Introduction

A common method for packet flow control over network connections, used both in the data-link and the transport layers, is *sliding windows* [1]. In this method, the receiver regularly reports to the sender the index of the next-expected packet, thereby acknowledging all the packets up to that index. The sender may transmit up to a certain number of packets, called the *window size*, beyond the last acknowledged packet; if a packet is not acknowledged within a certain ‘timeout’ period (ideally aimed to be the connection round-trip time, or slightly higher), the window is retransmitted from that packet on. In its pure form, this scheme implies that packets must arrive to the destination in-order. While the receiver may

☆ A preliminary version of this study appeared in the Proceedings of IEEE Infocom 2003.

* Corresponding author. Tel.: +61-2-83745506; fax: +61-2-83745531.

E-mail addresses: lavy.libman@nicta.com.au (L. Libman), ariel@ee.technion.ac.il (A. Orda).

1 This research was performed while L. Libman was a Ph.D. student in the Technion—Israel Institute of Technology, and supported in part by the Israeli Ministry of Science.

1389-1286/$ - see front matter
doi:10.1016/j.comnet.2004.03.032

temporarily keep out-of-order packets in a buffer, this does not affect the connection's performance unless the protocol is extended to allow selective, rather than cumulative, acknowledgments [2,3]. Such extensions are not universally implemented, and even when they are, the space allocated to hold such out-of-order packets is, typically, not very large. Therefore, on a coarser level, the packet stream still has to arrive in-order, allowing exceptions only to a limited extent.

Since a lost packet may trigger a retransmission of up to an entire window, its negative effect on throughput is not only due to the loss itself, but due to the time wasted in waiting for the acknowledgment as well. This effect is more severe when the connection's round-trip time (more precisely, the timeout) is long compared to the transmission time of a packet; such a connection is said to have a large *bandwidth-delay product*. A good example is a geostationary satellite link, with a round-trip propagation delay of roughly 0.25 s, used within a high-speed connection where a packet transmission typically takes a fraction of a millisecond; the delay-bandwidth product is then measured in thousands.

Assuming that packet losses are independent (e.g., caused by white noise or a randomized discarding policy along the connection path, such as RED [4,5]), and that transmission of a window takes less than the round-trip delay, the throughput can be improved considerably by retransmitting some or all of the packets several times within the window itself (rather than just after a timeout, as in 'classic' sliding-window schemes), as this increases their initial probability of successful arrival. For the rest of the paper, we extend the definition of the window size to include all such transmissions, counting each one separately whether it is a new packet or a copy of a previous one. We define a *sliding-window strategy* to be a rule that specifies how many copies of each packet, relative to the start of the window, are transmitted and in what order; in particular, it also specifies the window size. We mention at this point that alternative methods, such as forward error correction (FEC), can be used within this framework instead of simple retransmissions; we comment more on this later.

In general, for a given packet loss rate, transmitting more packets in a window—whether new ones or more copies of the same—increases the expected number of successful packets in every round-trip period, and, hence, the long-term throughput (at any rate, so long as the total window transmission time remains below the round-trip time). However, a larger window also increases the average rate of duplicate and out-of-order packets, which needlessly contributes to the network load. Thus, selection of a window size constitutes a tradeoff between these conflicting goals. To quantify this tradeoff, we associate with the connection a 'cost' per unit time and a 'cost' per packet transmission, and define the optimal strategy as one that minimizes the average cost/throughput ratio over time. We point out that these costs can have various interpretations, and should not be taken literally as money charges [6]. For example, the time cost may be associated with the disutility incurred by the application due to increased delay, and the transmission cost may be related to the energy consumption of a mobile device. Similarly, a 'social' (e.g., TCP-friendly) sender that refrains from retransmitting to avoid loading the network for others behaves as if it had a high per-transmission cost.

In 'classic' sliding windows, the sender transmits each packet in the window once, and the optimal strategy computation thus reduces to a trivial optimization of a single parameter (the window size). When each packet may be (re)-transmitted several times within a window, the problem becomes much more interesting. Finding the optimal strategy can then be viewed as being composed of two subproblems: an 'outer' problem of finding the optimal window size N, depending on the time and packet transmission costs; and an 'inner' problem of optimally distributing a total 'budget' of N transmissions among the packets in a window, which, for a given N, no longer depends on the costs. A salient feature of the resulting solution is that not all packets are transmitted an equal number of times: earlier packets in every window get more copies transmitted than later ones, in accordance with their 'importance' (e.g., the loss of the first packet in a window results in the loss of the entire window even if later packets arrive correctly, while the reverse is not true).

In this paper, we present a detailed analysis of optimal sliding-window strategies, following the above decomposition to the 'outer' and 'inner' subproblems. It turns out that the inner problem, of deciding which packet copies to transmit for a given window size N, involves a certain combinatorial optimization problem, and we explore in detail its properties, derive bounds on the solution's performance, and suggest an efficient solution algorithm based on dynamic programming. We then proceed to extend it for the outer problem (of finding the optimal window size) as well, thus establishing an integrated solution algorithm for the strategy optimization problem. Finally, we show that the cost/throughput ratio increases only logarithmically in the time price; this is a significant improvement of the linear dependence achievable by 'classic' sliding windows.

Our current study analyzes optimal strategies limited to simple retransmissions only. A potentially better scheme for increasing the success probability of a group of packets is that of forward error correction (FEC) coding; generally, a (n, k) FEC code encodes a group of k packets into $n > k$ 'copies', so that any k successful ones allow reconstructing the original data. We wish to emphasize that the ideas presented in this paper are not inconsistent with FEC coding, but rather complement it. If the code parameters are fixed (e.g., in a lower layer), our analysis can be readily applied by treating each encoded block as a "super-packet" with the appropriate loss probability. If the code can be controlled, the problem becomes that of finding an optimal *coding strategy*, which, though more complex, is based essentially on the methodology introduced here, except that the number of retransmissions is replaced by the notion of *coding redundancy*. In particular, it is to be expected that the optimal strategy would use higher-redundancy coding for the first packets in every window than for later ones.

The special concerns raised by connections with large delay-bandwidth products in general, and satellite links in particular, have attracted considerable research in recent years. Most of these studies are in the context of the widely-used TCP protocol and propose how to improve its performance, either by tuning the parameters of existing features like extended windows, slow-start, and congestion avoidance [7,8], or by introducing extensions, such as explicit congestion notifications [9]. Considerable attention has also been devoted to FEC coding that is able to adapt to higher-layer protocol requirements, partly in the context of multimedia applications with real-time requirements [10], but mostly, again, in conjunction with TCP [11,12]. None of these works, however, suggested improvements of the sliding-window mechanism itself. In fact, to the best of our knowledge, the idea of basing the number of retransmissions (or the FEC coding redundancy) on the *position* of the packet within a window, which is central to this paper, has not been suggested before. The approach we follow in the paper is *generic*, with the goal of discovering fundamental properties of optimal retransmission strategies, and we do not consider specific implementation issues in existing sliding-window protocols, which may require further work.

The rest of the paper is structured as follows. Section 2 describes the model and formally defines the underlying optimization problems. Section 3 describes basic structural properties of the solution and derives bounds on the optimal strategy performance. The solution algorithm and its properties for the 'inner' problem are analyzed in Section 4 and incorporated into an overall solution algorithm in Section 5. Finally, Section 6 concludes with a discussion of our methodology and its possible extensions, and outlines directions for further research.

2. Model and problem formulation

2.1. The model

As explained in the Introduction, we are interested in network connections with a high delay-bandwidth product, in which the receiver accepts packets only in order (with only a small buffer space, if at all, to hold a limited number of out-of-order packets). For our analysis, we shall bring these two characteristics to an extreme. That is, we assume that the receiver is unable to accept out-of-order packets at all, and we take the packet transmission time to be zero, which implies that the size of the window that can be transmitted within a round-trip period is unlimited. Furthermore, we assume there are no other factors that may limit the window size; e.g., the receiving application processes the arriving packets instantly, if necessary, hence no buffer space is consumed by packets arriving in-order. These assumptions simplify the analysis and allow it to concentrate on the essential properties of the resulting strategies, without having to deal, from the outset, with details of secondary importance. Section 6 discusses the extensions required to alleviate these assumptions, and argues that the solution methodology remains similar nonetheless.

We denote the loss rate in the network by L, and assume that losses are independent among packets, as is the case, e.g., for white noise or a random discard policy such as RED. In addition, we neglect the loss rate of acknowledgments, since they are, typically, much shorter than data packets, and therefore suffer less from noise and their paths are often less congested; moreover, since acknowledgments only carry the next-expected packet index, a loss of one has no significance if a later one in that window is received successfully. Consequently, for each packet, the sender knows whether it was successfully received after a round-trip time, which we denote by T.

We introduce a cost composed of a 'price' of a per unit of time and b per transmitted packet, and define an optimal strategy as one that minimizes the cost/throughput ratio over time; as explained in the Introduction, these prices can have generic interpretations. Incidentally, we chose to base our analysis on this cost structure, which is linear in the time and number of packets, reckoning that it is appropriate for a variety of scenarios and cost interpretations [6]. A different (non-linear) cost structure may be used instead, provided that the cost of transmitting a window depends only on its size, and not on the identities of its packets or the actual number successfully received. This may affect only the analytical results, e.g., the asymptotic dependence of the optimal strategy performance on the costs, whereas the actual algorithm for finding it remains intact.

The computation of the optimal strategy from the connection parameters (L, T, a, b) implicitly assumes that they are known; therefore, they must either remain constant or change quasi-statically, allowing the strategy to adapt after a change is detected. If any of the parameters, e.g., the round-trip time, changes quickly and unpredictably, it should be modeled by a random variable (e.g., as in [6]) rather than a constant value. We point out, however, that this is not typical of the kind of network connections that are the subject of this study: e.g., for satellite links, the round-trip time is dominated by the propagation delay, which can be considered essentially constant.

The above assumptions readily imply two fundamental properties. First, in the optimal strategy, packets are transmitted only at multiples of T; sending packets at other times cannot gain, since no extra information is present. Second, once a sequence of packets is sent at time t, the index of the last one to arrive in-order is known by time $t + T$, so the strategy simply restarts ('slides') at the subsequent packet. Consequently, the description of a strategy consists simply of a single vector that specifies the number of copies to be sent of every packet, relative to the next-expected index, at every multiple of T. The purpose of the subsequent analysis will be to find the optimal such vector.

2.2. Problem formulation

Consider a vector $\vec{n} = \langle n_1, \ldots, n_i, \ldots \rangle$, where n_i are whole and non-negative, and define a random variable S to be the number of in-order successful packets at the receiver if the sender transmits n_1 repetitions of packet 1, followed by n_2 repetitions of packet 2, etc. [2] The distribution of S is

$$P_S(j) = \prod_{i=1}^{j} (1 - L^{n_i}) \cdot L^{n_{j+1}}. \tag{1}$$

[2] Obviously, transmitting the same packets in any other order can only decrease the expected number of in-order arrivals.

We define the *score* of $\vec{n}$, denoted by $\phi(\vec{n})$, to be the expected value of S; thus

$$\begin{aligned}\phi(\vec{n}) \triangleq E[S] &= \sum_{j=1}^{\infty} j \cdot \prod_{i=1}^{j} (1 - L^{n_i}) \cdot L^{n_{j+1}} \\ &= \sum_{j=1}^{\infty} j \cdot \left[\prod_{i=1}^{j} (1 - L^{n_i}) - \prod_{i=1}^{j+1} (1 - L^{n_i}) \right] \\ &= \sum_{j=1}^{\infty} \prod_{i=1}^{j} (1 - L^{n_i}). \end{aligned} \tag{2}$$

We seek the vector $\vec{n} = \langle n_1, \ldots, n_i, \ldots \rangle$ that minimizes

$$\frac{a \cdot T + b \cdot \sum_{i=1}^{\infty} n_i}{\phi(\vec{n})} = \frac{a \cdot T + b \cdot \sum_{i=1}^{\infty} n_i}{\sum_{j=1}^{\infty} \prod_{i=1}^{j} (1 - L^{n_i})}. \tag{3}$$

The above expression describes the cost/throughput ratio attained by the strategy $\vec{n}$ over time. The numerator is the fixed cost of a period of T, during which one window is transmitted, and the denumerator is the expected number of packets successfully communicated in that period.

Consider expression (3) more closely. For any N, all the vectors with $\sum_{i=1}^{\infty} n_i = N$, i.e., suggesting the same total window size, attain the same numerator value; hence, the comparison among them is based merely on their score. Consequently, let us define

$$E_L(N) \triangleq \max_{\substack{n_1, n_2, \ldots \\ \text{s.t.} \sum_i n_i = N}} \left\{ \sum_{j=1}^{\infty} \prod_{i=1}^{j} (1 - L^{n_i}) \right\} \tag{4}$$

and rewrite expression (3) accordingly as

$$\frac{a \cdot T + b \cdot N}{E_L(N)}. \tag{5}$$

Then, the problem of finding the strategy vector that minimizes (5) can be separated into the following (sub-)problems:

Inner problem: Computing $E_L(N)$ for a given N.
Outer problem: Searching for N^* that minimizes (5).

This separation is convenient in that it isolates the infinite-dimensional part of the problem to depend solely on L, while the dependence on the other parameters reduces to a one-dimensional optimization only. Furthermore, the vector that actually attains the maximum in (4) is not needed until the final stage, after N^* has been found; during the search of N, it suffices to be able to evaluate $E_L(N)$, without the need to find the maximizing vector explicitly.

To conclude this section, we digress to consider the case of 'classic' sliding windows, where each packet is sent only once in a window; this corresponds to the vector $n_1 = \cdots = n_N = 1$, with a cost/throughput ratio of

$$\frac{a \cdot T + b \cdot N}{\sum_{j=1}^{N} (1 - L)^j} = \frac{L}{1 - L} \cdot \frac{a \cdot T + b \cdot N}{1 - (1 - L)^N}. \tag{6}$$

Maximizing this (e.g., by differentiating with respect to N) yields an optimal window size of

$$N^* = \frac{1}{\log \frac{1}{1-L}} \left[-\operatorname{plog}\left(-\frac{(1-L)^{aT/b}}{\mathrm{e}} \right) - 1 + \frac{aT}{b} \log(1 - L) \right] \underset{(\text{if } aT \gg b)}{\approx} \frac{\log\left(\frac{aT}{b} \log \frac{1}{1-L} + 1 \right)}{\log \frac{1}{1-L}}, \tag{7}$$

where $\operatorname{plog}(\cdot)$ (the product-log function) denotes the inverse function of $f(t) = t \cdot \mathrm{e}^t$, such that $t = -\operatorname{plog}(-y)$ (for $0 < y \leqslant 1/\mathrm{e}$) is the largest positive solution to the equation $y = t \cdot \mathrm{e}^{-t}$; in the final approximation we used the property that $-\operatorname{plog}(-\mathrm{e}^{-x}) \approx x + \log x$ for $x \gg 1$. [3] Thus, as the time cost a increases with respect to the other parameters, the optimal window size increases logarithmically in a. Since the denominator of (6) tends to a finite value as $N \to \infty$, the cost/throughput ratio, overall, increases linearly in a.

3. Basic properties and bounds

In this section, we show some basic structural properties of the optimization problems' solutions, and derive important bounds, in particular, on their asymptotic behavior.

[3] The product-log function is also known elsewhere as Lambert's W-function [13], or, more precisely, as one of its real-valued branches.

3.1. Properties of the inner problem

Our first two lemmas state basic and intuitively obvious structural properties.

Lemma 1. *$E_L(N)$ decreases in L and increases in N.*

Proof. Consider the maximizing vector in (4) for some L and N, and suppose that L is then decreased. The score of that vector then increases; if it is no longer the maximizer for the new L, then, obviously, the maximum value can only be even higher. Therefore, the value of (4) increases.

Alternatively, suppose that N is increased, and add the entire amount of the increase to the first element (arbitrarily). Again, this results in an increase of the score; if the resulting vector is not the maximizer for the new N, the value of (4) can only increase further. □

Lemma 2. *For a given N, the elements of the vector that achieves the maximum in* (4) *maintain a non-increasing order, i.e., $n_1 \geqslant n_2 \geqslant \cdots \geqslant n_i \geqslant \cdots$.*

Proof. Suppose, by contradiction, that there exists a pair of indices $i_1 < i_2$ with $n_{i_1} < n_{i_2}$. Consider the score of the vector resulting by swapping n_{i_1}, n_{i_2}, as given by expression (2). All the sum elements (products) for $j < i_1$ (which depend on neither n_{i_1} nor n_{i_2}), as well as for $j \geqslant i_2$ (which contain both $(1 - L^{n_{i_1}})$ and $(1 - L^{n_{i_2}})$ in the product), remain unchanged. The elements for $i_1 \leqslant j < i_2$, which contain only $(1 - L^{n_{i_1}})$ but not $(1 - L^{n_{i_2}})$ in the product, are strictly increased by the swap, thereby increasing the value of the entire sum. Consequently, the original vector cannot be a maximizer. □

Corollary. *In the maximizing vector, all the elements after the first zero element are also zero.*

Corollary. *For a given N, the index of the last non-zero element in the maximizing vector is bounded by N.*

We proceed to derive an important bound on the number of transmissions required to attain a given score. For this purpose, we introduce a variable change that makes the subsequent presentation more convenient. Define $p_i \triangleq 1 - L^{n_i}$ (i.e., p_i is the individual probability of packet i to arrive successfully, regardless of other packets). We shall refer to the vector $\vec{p} = \langle p_1, \ldots, p_i, \ldots \rangle$ as completely equivalent to the vector $\vec{n}$ and interchange them freely for convenience; in particular, with a slight abuse of notation, we refer to $\phi(\vec{p}) = \sum_{j=1}^{\infty} \prod_{i=1}^{j} p_i$ as the score of $\vec{p}$.

Lemma 3. *If $\vec{n}$ is the maximizing vector in* (4), *then $p_1 = 1 - L^{n_1} \geqslant \phi(\vec{n})/(\phi(\vec{n}) + 1)$.*

Proof. Lemma 2 implies that $p_i \leqslant p_1$ for all i; therefore,

$$\phi(\vec{n}) = \sum_{j=1}^{\infty} \prod_{i=1}^{j} p_i \leqslant \sum_{j=1}^{\infty} (p_1)^j = \frac{p_1}{1 - p_1} \tag{8}$$

and the lemma immediately follows by extracting p_1. □

Theorem 1. *For any vector $\vec{n}$, $N = \sum_i n_i \geqslant \log_{1/L} \{[\phi(\vec{n}) + 1]!\}$.* [4]

Proof. Obviously, since the factorial and the logarithm are monotonously increasing operations, it suffices to prove the theorem for the vector with the maximum score for a given N. Such a vector must satisfy Lemmas 2 and 3.

Consider the equivalent vector $\vec{p} = \langle p_1, \ldots, p_M, 0, \ldots \rangle$, where M denotes the index of the last non-zero element. Define the following sequence of subvectors, $\vec{p}^{(m)} \triangleq \langle p_m, p_{m+1}, \ldots, p_M, 0, 0, \ldots \rangle$, and of their corresponding scores, $\phi_m = \phi(\vec{p}^{(m)}) = \sum_{j=m}^{M} \prod_{i=m}^{j} p_i$, for all $1 \leqslant m \leqslant M$; note that ϕ_1 is the score of the original vector. Observe that $\phi_m = p_m(1 + \phi_{m+1})$, and, therefore, $\phi_{m+1} \geqslant \phi_m - 1$, for all $1 \leqslant m < M$; successively applying this inequality, we get $\phi_m \geqslant \phi_1 - (m - 1)$ for all m. On the other hand, applying Lemma 3 on each of the subvectors in turn, we have $p_m \geqslant \phi_m/(\phi_m + 1)$, or $1/(1 - p_m) \geqslant \phi_m + 1$. Consequently,

[4] Recall that the factorial $t!$, for any $t \geqslant 0$, is defined by $t! = \int_0^{\infty} x^t e^{-x}\, dx$; this definition coincides with the more common $t! = 1 \cdot 2 \cdot \ldots \cdot t$ for integer t. A well-known property of the factorial is $t! = t \cdot (t - 1)!$ for any $t \geqslant 1$.

$$\left[\prod_{m=1}^{M}(1-p_m)\right]^{-1} \geqslant \prod_{m=1}^{M}(\phi_m+1) \geqslant \prod_{m=1}^{M}\max\left[\phi_1-(m-1)+1,1\right]. \quad (9)$$

Now, consider the factorial $(\phi_1+1)!$. Denote $\lfloor\phi_1\rfloor$ to be the integer part of ϕ_1 (and, thereby, $(\phi_1-\lfloor\phi_1\rfloor)$ to be its fractional part). Successively applying the factorial property of $t! = t\cdot(t-1)!$ for any $t\geqslant 1$, we have

$$\begin{aligned}(\phi_1+1)! &= (\phi_1+1)\cdot\phi_1\cdot(\phi_1-1)\cdot\cdots\cdot(\phi_1-\lfloor\phi_1\rfloor)! \\ &= \prod_{m=1}^{M}\max\left[\phi_1-(m-1)+1,1\right]\cdot(\phi_1-\lfloor\phi_1\rfloor)! \\ &\leqslant \prod_{m=1}^{M}\max\left[\phi_1-(m-1)+1,1\right]. \quad (10)\end{aligned}$$

Note that we implicitly used the obvious fact that $\phi_1\leqslant M$, and also that $t!\leqslant 1$ for any $0\leqslant t<1$.

Combining inequalities (9) and (10), we obtain $[\prod_m(1-p_m)]^{-1}\geqslant(\phi_1+1)!$. Taking the logarithm of both sides and noting that $\log_{1/L}(1-p_m) = -n_m$, we finally get $\sum_m n_m \geqslant \log_{1/L}[(\phi_1+1)!]$. □

Finally, the following fundamental theorem presents the asymptotic relation between the window size and the maximum score that can be obtained by a vector of that size.

Theorem 2. $E_L(N) = \Theta(N/\log_{1/L}N)$. [5]

Proof. We apply the well-known Stirling's factorial approximation formula, $t! \approx \sqrt{2\pi t}(t/\mathrm{e})^t$ for large t, to the inequality established in Theorem 1, and obtain

$$\begin{aligned}N &\geqslant \log_{1/L}[E_L(N)+1]! \\ &\approx \log_{1/L}\left(\frac{E_L(N)+1}{\mathrm{e}}\right)\cdot[E_L(N)+1] \\ &\quad + \log_{1/L}\sqrt{2\pi[E_L(N)+1]}; \quad (11)\end{aligned}$$

thus, $N = \Omega(E_L(N)\cdot\log_{1/L}E_L(N))$. This implies directly that $E_L(N) = \mathrm{O}(N/\log_{1/L}N)$.

To show that $E_L(N) = \Omega(N/\log_{1/L}N)$ as well, it suffices to find one example of a vector that attains a score of $\Omega(N/\log_{1/L}N)$. Accordingly, consider the vector $\langle n_1,\ldots,n_M,0,\ldots\rangle$, such that $n_1=\cdots=n_M=\log_{1/L}N$ and $M = N/\log_{1/L}N$. Its score is

$$\begin{aligned}&\sum_{j=1}^{M}\prod_{i=1}^{j}(1-L^{n_i}) \\ &= \sum_{j=1}^{N/\log_{1/L}N}\left(1-L^{\log_{1/L}N}\right)^j \\ &= N\left(1-\frac{1}{N}\right)\cdot\left[1-\left(1-\frac{1}{N}\right)^{N/\log_{1/L}N}\right] \\ &\geqslant N\left(1-\frac{1}{N}\right)\left(1-\mathrm{e}^{-1/\log_{1/L}N}\right), \quad (12)\end{aligned}$$

completing the proof, as $\mathrm{e}^{-1/x}\approx 1-1/x$ for large x. [6] □

It is insightful to compare the result of Theorem 2 with the total number of packets received successfully (not necessarily in-order), which is, obviously, $N\cdot(1-L)$, i.e., $\Theta(N)$. Hence, it can be said that discarding out-of-order packets impacts the performance by a logarithmic factor. This theorem can also be used inversely: in-order to have an expected number of ϕ packets arriving successfully and in-order to the destination, the total number of packet copies transmitted by the source must be $\Theta(\phi\cdot\log_{1/L}\phi)$.

[5] Recall that $f(N) = \mathrm{O}(g(N))$, for positive functions $f(N)$, $g(N)$, means that $\lim_{N\to\infty}f(N)/g(N)<\infty$; in addition, $f(N) = \Omega(g(N))$ is equivalent to $g(N) = \mathrm{O}(f(N))$, and $f(N) = \Theta(g(N))$ means that both $f(N) = \mathrm{O}(g(N))$ and $f(N) = \Omega(g(N))$.

[6] The fact that $\log_{1/L}N$ and/or $N/\log_{1/L}N$ may not be integers is insignificant: rounding both expressions up to the nearest integers only increases the vector's score further, with an asymptotically negligible impact on the window size.

3.2. Properties of the outer problem

This subsection is concerned with the dependence of the optimal window size N^* on the cost factors a, b. Theorem 3 states an intuitively evident monotonicity property. Theorem 4 presents the central result of this section, regarding the asymptotic dependence of the cost/throughput ratio on the time cost a.

Theorem 3. *The optimal N^* is non-decreasing in aT/b.*

Proof. Consider two sets of parameters a_1, b_1, T_1 and a_2, b_2, T_2 such that $a_1T_1/b_1 \geqslant a_2T_2/b_2$, and suppose that N_1^*, N_2^* are their corresponding solutions (to the outer problem). This implies, in particular, that

$$\frac{a_1T_1/b_1 + N_2^*}{E_L(N_2^*)} \geqslant \frac{a_1T_1/b_1 + N_1^*}{E_L(N_1^*)}, \tag{13}$$

$$\frac{a_2T_2/b_2 + N_2^*}{E_L(N_2^*)} \leqslant \frac{a_2T_2/b_2 + N_1^*}{E_L(N_1^*)}. \tag{14}$$

Subtracting the second inequality from the first and noting that the common factor $(a_1T_1/b_1 - a_2T_2/b_2)$ is positive, we obtain $E_L(N_1^*) \geqslant E_L(N_2^*)$. In light of the monotonicity of $E_L(N)$ (Lemma 1), this implies $N_1^* \geqslant N_2^*$. □

Theorem 4. *As $a \to \infty$ (for fixed values of T, b, L), the cost/performance ratio attained by the optimal strategy increases logarithmically in a.*

Proof. Consider the expression $h(x) \triangleq (a \cdot T + b \cdot x)/(x/\log_{1/L} x)$, as a function of a (continuous) variable x. By differentiation with respect to x, it is easily found that its minimum is attained at $x^* = aT/b \cdot [-\text{plog}(-\text{e} \cdot b/(aT))]$. [7] Using again the property that $-\text{plog}(-\text{e}^{-y}) \approx y + \log y$ for large y, we obtain $x^* = \Theta(a \cdot \log a)$, and the minimum value of $h(x)$ is therefore $\Theta(\log a)$. This proves the theorem, since, in light of Theorem 2, the cost/throughput ratio is itself $\Theta(h(N))$, and its minimum value can, therefore, deviate from that of $h(N)$ by a constant factor at most. □

Thus, the ability to use retransmissions within the window enables the average cost per successful packet to increase merely logarithmically in a, rather than linearly as in the case of 'classic' sliding windows. Incidentally, note that no similar result exists for $b \to \infty$ with the other parameters constant; indeed, as $aT/b \to 0$, the optimal strategy tends to $\langle 1, 0, 0, \ldots \rangle$ (simple stop-and-wait), and the value of expression (5) simply increases linearly in b. This is true, of course, for the 'classic' case as well.

4. Solution of the inner problem

In this section, we present two approaches to the solution of the inner problem. First, we show how to solve it (i.e., compute the value of the function $E_L(N)$) exactly, using a technique of dynamic programming. The corresponding solution algorithm has a complexity of $\text{O}(N^2)$. However, it does not provide an insight to the structural properties of the solution; therefore, we also consider a similar optimization problem in continuous variables, for which the dependence of the solution on the problem parameters can be demonstrated more easily. Though the solution of the auxiliary problem is only an approximation to that of the original one, we show that it is tight for large values of the window size, and its properties therefore provide a useful insight.

4.1. Exact solution

To present the solution to the inner problem, we consider the score expression (2) and rearrange it as follows:

$$\sum_{j=1}^{\infty} \prod_{i=1}^{j} (1 - L^{n_i}) = (1 - L^{n_1}) \left[1 + \sum_{j=2}^{\infty} \prod_{i=2}^{j} (1 - L^{n_i}) \right]. \tag{15}$$

[7] Recall the definition of the plog function at the end of Section 2.

Therefore, if n_1 is fixed, the dependence of the vector's overall score on the other elements is only through the score of the subvector that begins with the second element. Consequently, the following relation holds:

$$E_L(N) = \max_{\substack{n_1,n_2,\ldots \\ \text{s.t.} \sum_i n_i = N}} \left\{ \sum_{j=1}^{\infty} \prod_{i=1}^{j} (1 - L^{n_i}) \right\}$$
$$= \max_{1 \leqslant n_1 \leqslant N} (1 - L^{n_1})[1 + E_L(N - n_1)]. \quad (16)$$

Relation (16) suggests that the optimal score for a window size of N (and the vector that achieves it) can be found from the scores of smaller window sizes by dynamic programming. The algorithm is described formally in Fig. 1, and is termed DI (for "Dynamic Inner"). We note that, since the computation of $E_L(N)$ requires N score computations once the optimal scores for all window sizes up to $N - 1$ are known, the overall complexity of the algorithm is $O(N^2)$.

Example. The following table summarizes the optimal vectors and their scores for $N = 15$ and selected values of L.

L	Optimal vector	Score
0.1	$\langle 2,2,2,2,1,1,1,1,1,1,1,0,0,\ldots \rangle$	8.41131
0.3	$\langle 3,2,2,2,2,2,1,1,0,0,\ldots \rangle$	5.39436
0.5	$\langle 4,3,3,2,2,1,0,0,\ldots \rangle$	3.61954
0.7	$\langle 6,5,3,1,0,0,\ldots \rangle$	2.24336
0.9	$\langle 11,4,0,0,\ldots \rangle$	0.92217

In accordance with intuition, for low loss rates, it is best to send at least one copy of more individual packets; conversely, when the loss rate is high, the expected number of successful in-order arrivals is maximized by duplicating just the first few packets. In fact, it is obvious that, for any N, the optimal vector tends to

$$\langle \underbrace{1,\ldots,1}_{N}, 0, 0, \ldots \rangle$$

for $L \to 0$ and to $\langle N, 0, 0, \ldots \rangle$ for $L \to 1$.

4.2. Approximation through continuous relaxation

We now analyze the properties of the optimization problem that defines the function $E_L(N)$, expressed by (4), omitting the requirement for the elements of $\vec{n}$ to be integers. This way, we have a relaxed optimization problem in a continuous space, which can be analyzed more easily by 'traditional' methods from optimization theory. Obviously, this technique results in a value that is higher than $E_L(N)$.

To distinguish the relaxed problem from the original one, we denote the maximum score by $\Phi_L(s)$, where s (rather than N) is used to denote the vector size, to emphasize that $\Phi_L(\cdot)$, unlike $E_L(\cdot)$, is well-defined for non-integer arguments. Additionally, we again make the convenient variable change of $p_i \triangleq 1 - L^{n_i}$, after which the score expression is simply $\sum_{j=1}^{\infty} \prod_{i=1}^{j} p_i$, while the constraints on the vector elements become

$$n_i \geqslant 0 \quad \Rightarrow \quad 0 \leqslant p_i \leqslant 1; \quad (17)$$

$$\sum_i n_i = s \quad \Rightarrow \quad \prod_i (1 - p_i) = L^s. \quad (18)$$

We immediately observe that the problem essentially depends on just one parameter, L^s, i.e., $\Phi_L(s) = \Phi(L^s)$ (even though the translation back to the original variables, $n_i = \log_L(1 - p_i)$, involves the specific value of L). This crucial property enables us to solve the problem for a wide range of input parameters and demonstrate the results in a simple one-dimensional plot of $\Phi(L^s)$, which is indeed given in Fig. 3 at the end of this subsection. We point out that the graph (or an equivalent

Initialization: Set $\vec{n}(0) = \langle 0, 0, \ldots \rangle$
For every $N' = 1, 2, \ldots, N$:
  Set $E_L(N') \leftarrow \max_{1 \leq n_1 \leq N'} (1 - L^{n_1}) [1 + E_L(N' - n_1)]$
  Set n_1^* to the argument that achieved the maximum in the previous line
  Set $\vec{n}(N')$ to the concatenation of $\langle n_1^* \rangle$ and $\vec{n}(N' - n_1^*)$

Fig. 1. Algorithm Dynamic-Inner (DI).

table) can be used in any case where a quick estimation of the score is required, without the need to perform a full run of the dynamic programming algorithm. The rest of this subsection describes how the optimization problem in continuous variables is solved.

Obviously, the solution vector, which attains the globally maximum score, must, in particular, be also a local maximum; as both the target expression and the constraint are differentiable, this means that it has to satisfy the corresponding Kuhn–Tucker conditions [14]. These can be verified, by a straightforward simplification, to require that there exists a constant λ such that

$$(1-p_m)\sum_{j=m}^{\infty}\prod_{\substack{i=1\\ i\neq m}}^{j} p_i = \lambda \quad (\text{if } 0 < p_m < 1), \tag{19}$$

$$\sum_{j=m}^{\infty}\prod_{\substack{i=1\\ i\neq m}}^{j} p_i \leqslant \lambda \quad (\text{if } p_m = 0).\,^{8} \tag{20}$$

Using (19) and (20), we can prove the following claim about the structure of the solution vector.

Lemma 4. *A vector that satisfies conditions* (17)–(20) *has the form* $\langle p_1, p_2, \ldots, p_M, 0, 0, \ldots\rangle$, *where* M *is finite*, $p_1 > \cdots > p_M > 0$, *and* $p_M \leqslant \frac{1}{2}$.

Proof. First, note that if $p_k = 0$ for some k, then condition (19) cannot be satisfied for any $m > k$, since all the products contain p_k and hence equal 0. Therefore, there are no positive elements after the first zero element.

For any $1 \leqslant m < M$, condition (19) implies that

$$\frac{1-p_m}{p_m}\sum_{j=m}^{\infty}\prod_{i=1}^{j} p_i = \frac{1-p_{m+1}}{p_{m+1}}\sum_{j=m+1}^{\infty}\prod_{i=1}^{j} p_i. \tag{21}$$

Since, obviously, $\sum_{j=m}^{\infty}\prod_{i=1}^{j} p_i > \sum_{j=m+1}^{\infty}\prod_{i=1}^{j} p_i$, it follows that $(1-p_m)/p_m < (1-p_{m+1})/p_{m+1}$, which implies $p_m > p_{m+1}$.

Comparing expression (20) for p_{M+1} with expression (19) for p_M, we have

$$\prod_{i=1}^{M} p_i \leqslant (1-p_M)\prod_{i=1}^{M-1} p_i, \tag{22}$$

and after dividing both sides by the common factor of $\prod_{i=1}^{M-1} p_i$, it reduces to $p_M \leqslant 1-p_M$, hence $p_M \leqslant \frac{1}{2}$.

It remains to show that M is finite, i.e., the vector cannot have infinitely many positive elements. Suppose, by contradiction, that such a vector exists. Then constraint (18) implies that $p_k \underset{k\to\infty}{\longrightarrow} 0$. Consequently, choose an index K such that $p_K < \frac{1}{2}$. We now show that the assumption $p_{K+1} > 0$ leads to a contradiction.

If $p_{K+1} > 0$, then condition (19) implies

$$\frac{1-p_{K+1}}{p_{K+1}}\sum_{j=K+1}^{\infty}\prod_{i=1}^{j} p_i = \frac{1-p_K}{p_K}\sum_{j=K}^{\infty}\prod_{i=1}^{j} p_i, \tag{23}$$

or, dividing both sides by the common factor of $\prod_{i=1}^{K} p_i$,

$$\frac{1-p_{K+1}}{p_{K+1}}\sum_{j=K+1}^{\infty}\prod_{i=K+1}^{j} p_i = \frac{1-p_K}{p_K}\cdot\left(1+\sum_{j=K+1}^{\infty}\prod_{i=K+1}^{j} p_i\right), \tag{24}$$

therefore

$$\sum_{j=K+1}^{\infty}\prod_{i=K+1}^{j} p_i = \frac{(1-p_K)p_{K+1}}{p_K - p_{K+1}}. \tag{25}$$

However, we showed above that the positive elements of $\vec{p}$ must maintain a decreasing order; hence, $p_i < p_{K+1}$ for $i > K+1$, and the following inequality holds:

$$\sum_{j=K+1}^{\infty}\prod_{i=K+1}^{j} p_i \leqslant \sum_{j=K+1}^{\infty}(p_{K+1})^{j-K} = \frac{p_{K+1}}{1-p_{K+1}}. \tag{26}$$

Substituting this inequality into (25), we get

$$\frac{1-p_K}{p_K-p_{K+1}} \leqslant \frac{1}{1-p_{K+1}} \;\Rightarrow\; 1+p_K p_{K+1} \leqslant 2p_K, \tag{27}$$

which contradicts $p_K < \frac{1}{2}$. Thus, condition (19) cannot be satisfied; therefore, $p_{K+1} = 0$. □

[8] In principle, there should also be a condition for $p_m = 1$; however, it is immediately seen that $p_m = 1$ for any m contradicts constraint (18).

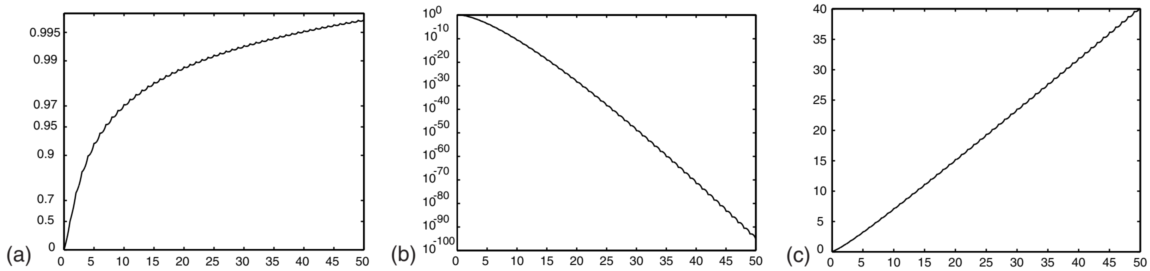

Fig. 2. (a) Plot of $\boldsymbol{p}_1(t)$, (b) plot of $L^s(t) \triangleq \prod_i (1 - \boldsymbol{p}_i(t))$, (c) plot of $\phi(t) \triangleq \sum_{j=1}^{\infty} \prod_{i=1}^{j} \boldsymbol{p}_i(t)$.

Suppose that a vector $\langle p_1, p_2, \ldots, p_M, 0, 0, \ldots \rangle$ is known to satisfy conditions (17), (19) and (20), and that only the value of the last element p_M is known. Then the other elements can be uniquely determined by a procedure of backward iteration. Specifically, once the values of $p_{m+1}, \ldots, p_M$ are known, one can equate expression (19) for p_m and p_{m+1}, divide by the common factor of $\prod_{i=1}^{m-1} p_i$, and extract p_m. This results in the formula

$$p_m = \frac{1 + \sum_{j=m+1}^{M} \prod_{i=m+1}^{j} p_i}{2 + \sum_{j=m+2}^{M} \prod_{i=m+2}^{j} p_i}. \tag{28}$$

Conversely, it is easy to see that, for any M and $0 < p_M \leqslant \frac{1}{2}$, the vector obtained through formula (28) satisfies conditions (19) and (20). For convenience, we define a function $\vec{\boldsymbol{p}} : \mathbb{R}^+ \mapsto \mathbb{R}^\infty$, such that, for any $t > 0$, $\vec{\boldsymbol{p}}(t)$ is the vector corresponding to $M = \lceil t \rceil$ and $p_M = \frac{1}{2}(t + 1 - \lceil t \rceil)$. [9] This puts the set of all vectors that satisfy conditions (19) and (20) (and are therefore "eligible candidates" to be solutions to the optimization problem for the corresponding values of L^s) in one-to-one correspondence with the positive real axis. We also define $\boldsymbol{p}_i : \mathbb{R}^+ \mapsto \mathbb{R}$ to be the ith component of $\vec{\boldsymbol{p}}$.

Lemma 5. *The function $\vec{\boldsymbol{p}}$ is continuous.*

Proof. The continuity of $\vec{\boldsymbol{p}}$ at non-integer points (continuity in p_M only, for a fixed M) is obvious from formula (28), which shows p_m, for any $1 \leqslant m \leqslant M - 1$, to be continuous in $p_{m+1}, \ldots, p_M$, and therefore (applying backward induction from $m = M - 1$ to $m = 1$) to be continuous in p_M.

To show the continuity of $\vec{\boldsymbol{p}}$ at $t = K$ for an integer K, one must prove $\lim_{t \to K^-} \vec{\boldsymbol{p}}(t) = \lim_{t \to K^+} \vec{\boldsymbol{p}}(t)$. Consider first the component $\boldsymbol{p}_K$. For $t \to K^-$, $\boldsymbol{p}_K(t)$ is simply the last non-zero element of $\vec{\boldsymbol{p}}(t)$; that is, $M = K$ and $\boldsymbol{p}_K(t) = \frac{1}{2}(t + 1 - K)$. Therefore,

$$\lim_{t \to K^-} \boldsymbol{p}_K(t) = \lim_{t \to K^-} \tfrac{1}{2}(t + 1 - K) = \tfrac{1}{2}. \tag{29}$$

For $t \to K^+$, $\boldsymbol{p}_K(t)$ is the penultimate non-zero element; that is, $M = K + 1$, $p_M = \frac{1}{2}(t - K)$, and $\boldsymbol{p}_K(t)$ can be computed from (28):

$$\lim_{t \to K^+} \boldsymbol{p}_K(t) = \lim_{t \to K^+} \frac{1 + \frac{1}{2}(t - K)}{2} = \tfrac{1}{2}. \tag{30}$$

Hence, the component $\boldsymbol{p}_K(t)$ is continuous at $t = K$. From here, the continuity of $\boldsymbol{p}_1(t), \ldots, \boldsymbol{p}_{K-1}(t)$ in t follows from their continuity in $\boldsymbol{p}_K$, according to (28) (again, using backward induction from $m = K - 1$ to $m = 1$). □

Fig. 2 shows plots of a few functions derived from the definition of $\vec{\boldsymbol{p}}(t)$. Fig. 2a shows a plot of $\boldsymbol{p}_1(t)$, using a logarithmic vertical axis to emphasize the 'waviness' of the function. Note that, by construction, $\boldsymbol{p}_i(t) = \boldsymbol{p}_{i+k}(t + k)$ for any integer k and any $t > 0$; hence, appropriately shifted, the plot is valid for any component $\boldsymbol{p}_m(t)$. Fig. 2b shows a plot of $L^s(t) \triangleq \prod_{i=1}^{\infty} (1 - \boldsymbol{p}_i(t))$, i.e., the value of L^s for which the vector $\vec{\boldsymbol{p}}(t)$ would satisfy constraint (18); for convenience, the vertical axis is logarithmic here as well. Finally, Fig. 2c shows a plot of the score attained by $\vec{\boldsymbol{p}}(t)$.

[9] The operator $\lceil t \rceil$ denotes the integer obtained by rounding up of t, i.e., the smallest integer that is not less than t.

Conceptually, the solution of the optimization problem for a given L^s is obtained by locating the set of points $\{t|\prod_i [1-\boldsymbol{p}_i(t)] = L^s\}$ (e.g., from Fig. 2b), and selecting the point that attains the maximal value of $\sum_j \prod_{i=1}^{j} \boldsymbol{p}_i(t)$ (e.g., from Fig. 2c). Note that the set contains more than one point for $L^s \lessapprox 1.586 \times 10^{-43}$ (the function of Fig. 2b ceases to be strictly decreasing after $t = 27$). In practical terms, computing the solution begins by evaluating $L^s(t)$ at integer points, exploiting the function's continuity to find an initial search range, and then performing a detailed search, e.g., by evaluation of $L^s(t)$ on a sufficiently dense grid of points (depending on the required precision) and subsequent interpolation.

Our experience from running this computation for various problem instances suggests that different values of t that correspond to the same L^s tend to attain very close values of ϕ as well, hence simply finding any such t is nearly optimal. In graphical terms, this means that the plots in Figs. 2b and 2c are very nearly "mirror images" of each other (and become ever more so as t gets larger). To illustrate this, Fig. 3 shows a parametric plot of $\phi(t)$ versus $L^s(t)$. Observe that the plot is virtually indistinguishable from a function; it takes a great deal of "zooming in" to notice that the plot actually zig-zags back and forth, and that every $L^s \lessapprox 1.586 \times 10^{-43}$ has several corresponding values of ϕ, of which only the topmost one is the 'true' $\Phi(L^s)$. Thus, strictly speaking, the function $\Phi(L^s)$ is not continuous; however, its 'jumps' are markedly minuscule.

Incidentally, it can be observed that the proof of Theorem 1 is easily extended to the continuous version of the problem; it then states that $L^s \leqslant 1/[\Phi(L^s)+1]!$ This bound is plotted by the dotted line in Fig. 3. Thus, it can be seen that the auxiliary function $\Phi(L^s)$ provides a much tighter bound.

We conclude this subsection with a theorem that provides the asymptotic connection between $E_L(N)$ and the auxiliary function $\Phi_L(s)$. It states that, in a certain sense, $\Phi_L(s)$ closely approximates $E_L(N)$ for large values of N.

Theorem 5. *For any L,s,* $E_{(L^s)^{1/N}}(N) \xrightarrow[N\to\infty]{} \Phi_L(s)$.

Proof. Define the following auxiliary function, for $0 < \Lambda < 1$ and $0 \leqslant p < 1$: $Y_\Lambda(p) = 1 - \Lambda^{\lfloor \log_\Lambda (1-p) \rfloor}$, where $\lfloor\cdot\rfloor$ denotes the integer-part operator. Thus,

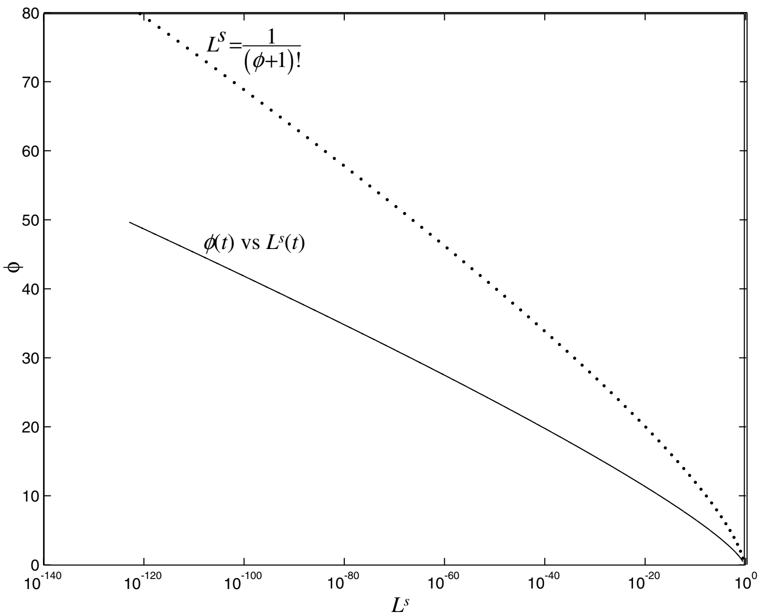

Fig. 3. Parametric plot of $\phi(t)$ versus $L^s(t)$, and comparison to the bound implied by Theorem 1.

$Y_\Lambda(p)$ is the highest number that is no higher than p and can be expressed as $1 - \Lambda^n$, for some integer n. It is obvious that as $\Lambda \to 1$, the set of points expressable as $1 - \Lambda^n$ for some integer n becomes dense in the segment $[0, 1]$, i.e., any $0 \leqslant p < 1$ can be approximated with an arbitrarily small difference by such a point, for Λ sufficiently close to 1. Therefore, $\lim_{\Lambda \to 1} Y_\Lambda(p) = p$ for any $0 \leqslant p < 1$.

Now, denote the maximizing vector of $\Phi_L(s)$ by $\overrightarrow{p^*} = \langle p_1^*, \ldots, p_M^*, 0, 0, \ldots \rangle$, and define the vector $\overrightarrow{p_{|N}} \triangleq \langle Y_{\Lambda_N}(p_1^*), \ldots, Y_{\Lambda_N}(p_M^*), 0, 0, \ldots \rangle$, where $\Lambda_N \triangleq (L^s)^{1/N}$. Define also the corresponding vector $\overrightarrow{n_{|N}} \triangleq \langle \lfloor \log_{\Lambda_N}(1 - p_1^*) \rfloor, \ldots, \lfloor \log_{\Lambda_N}(1 - p_M^*) \rfloor, 0, 0, \ldots \rangle$, and denote $N' = \sum \overrightarrow{n_{|N}}$; observe that $N' \leqslant \sum_i \log_{\Lambda_N}(1 - p_i^*) = \log_{\Lambda_N} \prod_i (1 - p_i^*) = \log_{\Lambda_N} L^s = N$.

Now, consider the score of $\overrightarrow{p_{|N}}$. It cannot be higher than $E_{\Lambda_N}(N')$, since $\overrightarrow{n_{|N}}$ is just one of the 'eligible' vectors over which $E_{\Lambda_N}(N')$ is maximized. In light of Lemma 1, it is therefore not higher than $E_{\Lambda_N}(N)$ as well. Thus, $E_{\Lambda_N}(N)$ is 'sandwiched' between the scores of $\overrightarrow{p_{|N}}$ and $\overrightarrow{p^*}$ (the latter, by definition, being simply $\Phi_L(s)$). However, since $\Lambda_N \to 1$ as $N \to \infty$, we have $\overrightarrow{p_{|N}} \xrightarrow[N \to \infty]{} \overrightarrow{p^*}$, which finally implies $E_{\Lambda_N}(N) \xrightarrow[N \to \infty]{} \Phi_L(s)$. □

5. Finding the optimal window size

We now turn to discuss the solution of the 'outer problem', namely, finding the window size (N) that minimizes the cost/throughput ratio (5). To begin, note that generic search algorithms (e.g., Fibonacci or golden-section search [14]), using algorithm DI as a 'subroutine' for computing $E_L(N)$, are inefficient, as they neglect the internal redundancy between computations for different N. Indeed, for any N, algorithm DI computes the scores for all window sizes up to N anyway. This raises the idea of proceeding with the iterations of that algorithm until the cost/throughput ratio (computed on the fly) ceases to decrease, instead of setting an advance limit.

Fig. 4 shows a typical plot of the target ratio as a function of N. Observe that the function decreases steeply at first but quickly becomes quite 'flat', eventually rising slowly amid a somewhat noise-resembling behavior. This shape is indeed expected, considering that the ratio expression is $\Theta((a \cdot T + b \cdot N)/(N/\log_{1/L} N))$ (recall Theorem 2): thus, for small N ($N \ll aT/b$), it decreases at a rate of $1/(N/\log_{1/L} N)$, while for $N \gg aT/b$, it

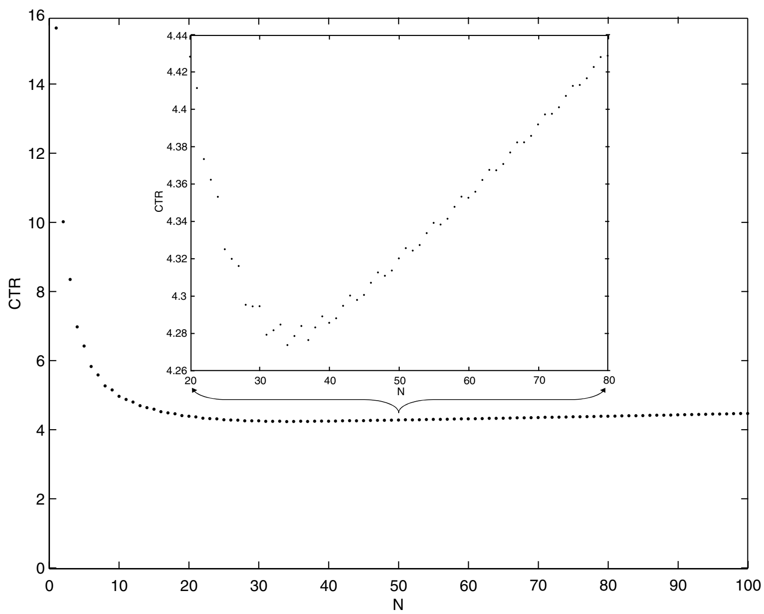

Fig. 4. The ratio $(a \cdot T + b \cdot N)/E_L(N)$ as a function of N, for $a = 10$, $T = 1$, $b = 1$, $L = 0.3$. The inset 'zooms in' on $20 \leqslant N \leqslant 80$.

```
Initialization: Set n⃗(0) = ⟨0, 0, ...⟩, N' ← 0, Best_CTR ← ∞
Loop:
    N' ← N' + 1
    Set E_L(N') = max_{1≤n_1≤N'} (1 − L^{n_1}) [1 + E_L(N' − n_1)]
    Set n_1^* to the argument that achieved the maximum in the previous line
    Set n⃗(N') to the concatenation of ⟨n_1^*⟩ and n⃗(N' − n_1^*)
    Set CTR ← the cost/throughput ratio for n⃗(N')
    If CTR < Best_CTR
        Set Best_CTR ← CTR, N^* ← N'
    If N' = 2N^*, terminate; else go back to Loop.
```

Fig. 5. Algorithm Dynamic-Outer (DO).

increases at a rate of $\log_{1/L} N$, i.e., much more slowly. The noise-like non-monotonocity, especially apparent around the minimum point, is due to combinatorial effects that we do not go into further; however, it may cause a potentially large number of 'false' local minima (e.g., $N = \{29, 31, 34, 37, 40, 44, 48, \ldots\}$ in Fig. 4), requiring care to avoid terminating the search prematurely.

To decide on an appropriate termination condition, we tested the algorithm for all L between 0.001 and 0.999 in increments of 0.001, with $b = 1$, $T = 1$, and $a \in \{1, 2, \ldots, 10, 20, \ldots, 100, 200, \ldots, 1000\}$ (recall that, for a given L, the optimal N^* depends only on aT/b). We point out that this range covers all the practically interesting cases: for $aT/b < 1$, the optimal window size rarely gets above 1, while for $aT/b = 1000$ the search already reaches window sizes of many thousands of packets. In all these runs, we found that, similarly to Fig. 4, the local minima indexes formed nearly arithmetic sequences with periods much smaller than N^* itself (in a few cases, there were two separate regions of local minima sequences with different periods, both much smaller than the corresponding N^*). A simple termination condition that is based on the above observation is $N = 2N^*$, i.e., stop the search after completing twice the iteration number in which the optimum was found. Fig. 5 describes the algorithm with this condition employed; this algorithm, termed DO (for "Dynamic Outer"), did not fail to find the global minimum even in a single instance. Admittedly, this condition is quite conservative; however, considering that the best strategy found so far can be employed even before the search is completed, perfecting the termination condition to reduce the computation by a constant factor at most does not seem to be of major importance. [10]

Finally, Fig. 6 plots the optimal cost/throughput ratio as a function of a, for a few select values of the loss rate; note that the horizontal axis is logarithmic. These plots clearly demonstrate the property predicted by Theorem 4, namely, that the ratio increases logarithmically in a.

We close this section with a conclusive example that demonstrates the performance of the DO algorithm.

Example. For the parameter values depicted in Fig. 4 (namely, $L = 0.3$, $T = 1$, $b = 1$, $a = 10$), algorithm DO finds the strategy $\langle 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 1, 1, 0, 0, \ldots \rangle$, at $N^* = 34$. It has a score of 10.295, which leads to a cost/throughput ratio (i.e., average cost per successfully communicated packet) of 4.2739.

For comparison, the optimal window size with 'classic' sliding windows, found by formula (7), is 5 (i.e., the strategy is $\langle 1, 1, 1, 1, 1, 0, 0, \ldots \rangle$ in our terms), with a corresponding cost/throughput value of 7.7273. Thus, using a strategy with advance retransmissions nearly halves the average cost per packet.

Let us now try $a = 100$, with the other parameters as before. This time, DO finds $N^* = 529$, with the strategy

[10] We point out that termination criteria based on the target value itself, rather than the window size (such as "stop when the current target value has risen to 5% above the optimum so far"), also work, but may lead to exponential complexity, due to the logarithmic increase rate of the target expression for large N. On the other hand, the complexity of algorithm DO is only $O(N^{*2})$.

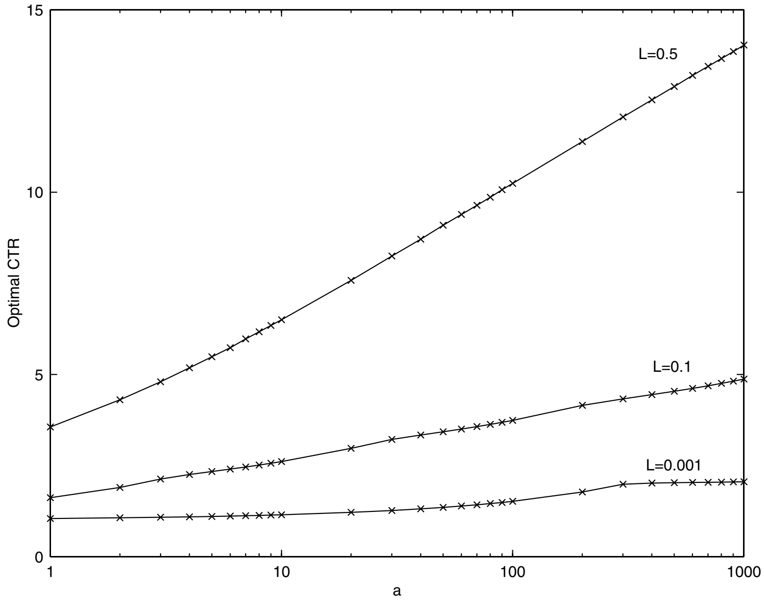

Fig. 6. Optimal cost/throughput versus a for several loss rates.

$$\langle \underbrace{6,\ldots,6}_{7}, \underbrace{5,\ldots,5}_{70}, \underbrace{4,\ldots,4}_{25}, \underbrace{3,\ldots,3}_{9}, 2,2,2,2,1,1,0,0,\ldots\rangle.$$

Its score is 97.3684, and the corresponding cost/throughput value is 6.46. The 'classic' optimal window size here is 10, yielding a cost/throughput value of 48.513; thus, in this case, the advantage of using a strategy with retransmissions is much greater. In fact, it can be seen that the cost/throughput increased only mildly from the previous case, despite the tenfold raise of the time cost, due to using a significantly larger window; this resulted in a nearly-tenfold increase in the throughput as well, which, therefore, nearly canceled the extra time cost.

6. Conclusion

We have investigated optimal sliding-window strategies in network connections where the packet transmission time is negligible compared to the round-trip delay. We associated a cost per unit of time and per packet transmission with the connection, and defined the optimal strategy as one that minimizes the expected cost/throughput ratio. We derived several important bounds on the optimal strategy performance; specifically, for a window size of N, we showed the number of successful in-order packets to be $\Theta(N/\log N)$, and used this result to prove that the cost/throughput ratio increases logarithmically in the time price. We then proposed an exact solution algorithm based on dynamic programming, as well as an approximate solution based on a relaxation of the problem to continuous variables, which was used to demonstrate the dependence of the strategy on the input parameters. Our approach was demonstrated to attain a significantly lower cost/throughput ratio than 'classic' sliding windows, where a packet is retransmitted only after a timeout or negative acknowledgment.

The prime importance of this paper is in establishing a theoretical foundation for the study of optimal error-control strategies. However, the analysis was based on some simplifying model assumptions, and further work is therefore called for in order to extend the applicability of our results. Several possible directions are outlined below.

One direction is to consider the case that the receiver has a buffer capable of accepting packets out-of-order, and reports its state in the acknowledgments (i.e., the protocol is capable of *selective repeat*). Instead of a single vector specifying the number of retransmissions for each packet, the optimal strategy in this case is described by a set of such vectors, corresponding to the possible buffer states and specifying the optimal sequence of tranmissions for each state (applied from the next-expected packet index). In principle, the computation of these vectors involves the optimization of an essentially similar score expression. However, due to the exponential number of possible buffer states, an advance computation of the strategy may be unfeasible, whereas in-time computation of the optimal retransmission vector after every acknowldgement is unlikely to be practical. Consequently, there is a need to explore suboptimal strategies, in which only a small number of retransmission vectors is precomputed, and one of them is selected after every round-trip time, according to a certain rule that depends on the buffer state. Studying the performance of such strategies, and proposing a proper rule for selecting the retransmission vector, is left as a subject for further research.

Another simplifying assumption made in the paper is the independence among packet losses. This immediately led to the conclusion that packet repetitions should immediately follow the original packet (see footnote 2). However, if errors are bursty, this may no longer be true: a single error burst can destroy several adjacent copies of the packet, whereas by spacing the copies apart, a higher resilience may be gained. Finding the vector that maximizes the score of expected in-order arrivals under bursty errors, or, in other words, extending the analysis of the "inner problem" to account for the distribution of the error burst length, remains for further investigation.

The strategies discussed in this paper were assumed to wait for all the acknowledgments from a window before setting out to transmit the next one. We explained, while presenting the general model, why such behavior is optimal if the packet transmission time is neglected. In reality, of course, a packet transmission takes a certain time $t_x > 0$. This can be simply catered to by replacing the packet transmission price b with $b + a \cdot t_x$, i.e., including the extra per-packet cost due to the time it takes to transmit it, with no further changes in the solution algorithm. Strategies thus computed are adequate when the connection's delay-bandwidth product is large (and, hence, $t_x \ll T$), such as, e.g., over satellite or very high-bandwidth terrestrial links. Otherwise, i.e., if a packet transmission takes a significant fraction of the round-trip time, it may be better not to wait for all acknowledgments from the previous window, and proceed with transmission with only a partial information on previous successes and losses. Then, a strategy is no longer described by a vector applied at every multiple of the round-trip time, but, rather, by a rule applied after every packet transmission and specifies the packet most worthwhile to transmit next (if at all), according to the information available up to that moment. The investigation of optimal strategies and their properties in this framework is the subject of ongoing work.

Finally, our attention in this paper was limited to strategies that use simple retransmissions only; however, as explained in the Introduction, the methodology can be extended for general FEC coding as well. The optimization problem in that case is more complex (it involves an extra parameter, namely the size of the coding block), yet its solution follows essentially the same approach. This extension is studied in detail in [15].

References

[1] A. Tanenbaum, Computer Networks, fourth ed., Prentice-Hall, Upper Saddle River, NJ, 2003.
[2] ISO/IEC standard 13239:2000 (HDLC procedures), February 2002.
[3] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, RFC 2018: TCP selective acknowledgment options, October 1996.
[4] S. Floyd, V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Transactions on Networking 1 (4) (1993) 397–413.
[5] D. Lin, R. Morris, Dynamics of random early detection, in: Proc. ACM SIGCOMM, Cannes, France, 1997, pp. 127–137.
[6] L. Libman, A. Orda, Optimal timeout and retransmission strategies for accessing network resources, IEEE/ACM Transactions on Networking 10 (4) (2002) 551–564.
[7] M. Allman, C. Hayes, H. Kruse, S. Ostermann, TCP performance over satellite links, in: Proc. 5th International

Conference on Telecommunication Systems, Nashville, TN, 1997, pp. 456–469.
[8] C. Barakat, N. Chaher, W. Dabbous, E. Altman, Improving TCP/IP over geostationary satellite links, in: Proc. IEEE Globecom, Rio de Janeiro, Brasil, 1999, pp. 781–785.
[9] D. Katabi, M. Handley, C. Rohrs, Internet congestion control for future high bandwidth-delay product environments, in: Proc. ACM SIGCOMM, Pittsburgh, PA, 2002.
[10] K. Park, W. Wang, AFEC: an adaptive forward error correction protocol for end-to-end transport of real-time traffic, in: Proc. 7th International Conference on Computer Communications and Networks (ICCCN), Lafayette, LA, 1998, pp. 196–205.
[11] C. Barakat, E. Altman, Bandwidth tradeoff between TCP and link-level FEC, Computer Networks 39 (2) (2002) 133–150.
[12] B. Liu, D. Goeckel, D. Towsley, TCP-cognizant adaptive forward error correction in wireless networks, in: Proc. IEEE Globecom, Taipei, Taiwan, 2002, pp. 2128–2132.
[13] R. Corless, G. Gonnet, D. Hare, D. Jeffrey, D. Knuth, On the Lambert W function, Advances in Computational Mathematics 5 (1996) 329–359.
[14] D. Luenberger, Linear and Nonlinear Programming, Addison-Wesley, Reading, MA, 1984.
[15] L. Libman, A. Orda, Optimal FEC strategies in connections with large delay-bandwidth products, in: Proc. IEEE International Conference on Communications (ICC), Paris, France, 2004.

Lavy Libman received the B.Sc. degrees (summa cum laude) in Electrical Engineering and in Computer Engineering, and the M.Sc. and Ph.D. degrees in Electrical Engineering, from the Technion—Israel Institute of Technology, Haifa, Israel, in 1992, 1997, and 2003, respectively.

He has joined the Networks and Pervasive Computing research program at National ICT Australia, Sydney, in September 2003. He has previously held several visiting and consulting positions, including with Bell Laboratories, NJ, in summer 2002, and with Millimetrix Broadband Networks, Israel, in summer 2000. Between 1993 and 1999, he served as a computer engineer in the Israel Defence Forces.

His research interests include protocols and algorithms for mobile networks, including error control, power control, and QoS, as well as application of game theory to the analysis, design, control and management of large-scale networks with heterogeneous and non-cooperative users. He received the Wolff prize for distinguished Ph.D. students.

He was a member of the Executive Committee of IEEE Infocom'2002, and is currently serving on the Technical Program Committees of IEEE LCN'2004 and ACM SIGCOMM Asia 2005 workshop.

Ariel Orda received the B.Sc. (summa cum laude), M.Sc., and D.Sc. degrees in Electrical Engineering from the Technion—Israel Institute of Technology, Haifa, Israel, in 1983, 1985, and 1991, respectively.

Since 1994, he has been with the Department of Electrical Engineering at the Technion, where he is currently an Associate Professor and the Academic Head of the Computer Networking Laboratory. He has held visiting and research positions at the Center for Telecommunication Research, Columbia University, New York, NY, Bell Laboratories, NJ, and IBM Watson Research Center, NY. In addition, he has held several consulting positions with Israeli industry.

His current research interests include network routing, QoS provisioning, wireless networks, the application of game theory to computer networking and network pricing.

He received the Award of the Chief Scientist in the Ministry of Communication in Israel, a Gutwirth Award for Outstanding Distinction, the Research Award of the Association of Computer and Electronic Industries in Israel, and the Jacknow Award for Excellence in Teaching.

He served as Technical Program co-chair of IEEE Infocom'2002. He is an Editor of Computer Networks and of the IEEE/ACM Transactions on Networking.