Optimal Sliding-Window Strategies in Networks with Long Round-Trip Delays

Lavy Libman and Ariel Orda
Department of Electrical Engineering
Technion – Israel Institute of Technology
Haifa, Israel 32000

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 (these costs can have generic interpretations, not necessarily in terms of money). 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 derive some bounds on the performance of the optimal strategy; in particular, we show that the optimal cost/throughput ratio increases merely logarithmically with the time price. We present a method for computing the optimal strategy; additionally, we demonstrate that a simple and efficient 'greedy' algorithm is sufficient to find a near-optimal solution.

This research was supported by the Israeli Ministry of Science.

I. Introduction

A common method for the flow control of packets over a network connection, used both in the data-link and the transport layers, is *sliding windows* [1]. In this mechanism, the receiver regularly informs the sender of 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 basic form, this scheme implies that packets must arrive to the destination in order. While the receiver may temporarily keep out-of-order packets in a buffer, this, in itself, does not change the connection's behavior 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 size of the buffer 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.

As a consequence, a lost packet may trigger a retransmission of up to an entire window, which obviously causes a reduction in throughput due to the time wasted in waiting for the acknowledgment. This loss is more severe in connections where the round-trip time (more precisely, the timeout) is long compared to the transmission time of a packet; such networks are 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 seconds, used within a high-speed connection in which a packet transmission takes a fraction of a millisecond (a packet of 1000 bits over, say, a 10Mb/s connection takes merely 0.1ms to transmit); in this scenario, the delay-bandwidth product is measured in thousands. Assuming that packet losses are independent (e.g. caused by white noise or a randomized discarding policy along the connection's path, such as RED [4] or a variant thereof), and the transmission time of the window is shorter 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 the initial probability of successful arrival at the destination. 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, a sliding-window strategy specifies the window size. We mention at this point that alternative methods, such as *forward error correction (FEC)*, may be used within this framework instead of simple retransmissions; we comment more on this later.

In general, it is apparent that, for a given packet loss rate, transmitting more packets in the window – be it new packets or more copies of the same ones – increases the average throughput (at any rate, so long as the total transmission time of the packets in a window does not exceed the round-trip time), as it increases the expected number of packets transmitted successfully by the sender each time before it stops to wait for acknowledgments. However, using a larger window size also increases the average number of packets that are not lost but are discarded by the receiver (for arriving out of order or for being copies of packets already received), which needlessly contributes to the network load. Thus, selection of a window size constitutes a tradeoff between these conflicting goals. In order to perform a quantitative analysis of the optimal retransmission strategy, we associate with the connection a 'cost' per unit time and a 'cost' per packet transmission, and define the optimal strategy as one that achieves the lowest average cost/throughput ratio over time. We point out that these costs can have various practical interpretations, and it should not be taken literally that the connection indeed charges money for them [5]. 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, where the sender is limited to transmitting each packet within the window once, computation of the optimal strategy reduces to an optimization of a single parameter (the window size), and can be solved trivially. When each packet is allowed to 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 optimal distribution of a given 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 the same 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 provide a detailed analysis of optimal strategies based on sliding windows with several transmissions per packet. The analysis follows the decomposition pointed above to an 'outer' and 'inner' subproblem, and tackles them separately for the most part. It turns out that the inner problem, of deciding which packet copies to transmit for a given window size, involves nontrivial combinatorial optimization, and a significant portion of the paper is devoted to the study of its properties. We show that, for a window size of N, the expected number of successful (and in-order) packets attained by the optimal solution is $O\left(\frac{N}{\log N}\right)$; we also describe how to compute a more precise tight bound. Additionally, we study several efficient approximation algorithms and compare their performance, finding, in particular, that a simple 'greedy' approach attains nearly-optimal solutions, especially for large N. We then extend this approach for the outer problem, of finding the optimal window size, as well, thus establishing an integrated solution algorithm for the overall strategy optimization problem. Finally, we show that the cost/throughput ratio increases merely logarithmically in the time price; this is a significant improvement of the linear increase rate achievable by 'classic' sliding windows (that do not transmit several copies per packet within a window).

Our current study analyzes optimal strategies limited to simple packet 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', among which any k successfully received allow reconstructing the original ones. We wish to emphasize that the ideas presented in this paper are not inconsistent with FEC coding, but rather complement it. If the network can only employ a fixed encoding (i.e. with fixed n and k), our analysis can be readily applied by treating each encoded block as a "super-packet" with the appropriate loss probability. Performance (i.e. cost/throughput) can be improved even further by employing a flexible *coding strategy*, which may use different codes for different packets. While the optimality analysis for this case is more complex and beyond the scope of this paper, it is based on essentially the same methodology as the one 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 a 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, e.g., [6], [7], [8], [9], [10]. Most of these studies are in the context of the widely-used TCP protocol and study how to improve its performance, either by tuning already-existing features [7], [8], or by introducing new ones, such as explicit congestion notifications [10]. Considerable attention has also been devoted to FEC coding that is able to adapt to higher-layer protocol requirements, and to TCP in particular (e.g. [11], [12], [13]). None of these works, however, suggested improvements to 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. We emphasize that this idea is generic, and can be incorporated in any sliding-window protocol.

The rest of the paper is structured as follows. Section II describes the assumptions made for the optimal strategy analysis and formally states the underlying optimization problem and its decomposition to the two sub-problems. Section III proves some basic properties of the solution and derives some bounds and asymptotic properties of the optimal strategy and the cost/throughput ratio it achieves. The next two sections establish efficient approaches for an approximate solution of the 'inner' combinatorial optimization problem: section IV presents several 'greedy' algorithms, compares their performance, and demonstrates that they generally achieve quite satisfactory approximations, while section V presents an alternative approach of using an analytical solution of an auxiliary problem in continuous variables, more easily amenable to analysis. Section VI shows how these results are incorporated in a search algorithm for the overall optimal strategy and provides a conclusive example of the presented techniques. Finally, section VII concludes with a discussion of our methodology and its possible extensions, and outlines directions for further research.

II. Model and Problem Formulation

A. The model

As explained in the Introduction, we are interested in network connections, either at the data-link or the transport layer, in which the receiver can only accept packets arriving in order (with only a small buffer space, if at all, to handle

a limited number of out-of-order packets), and the round-trip delay is considerably longer than the packet transmission time. In such connections, transmitting several packet copies in advance (even for packets that have not failed yet), and especially of the first packets in every window, achieves a significant improvement over the classic sliding window scheme, namely, of transmitting each packet once and retransmitting only after a timeout (or a negative acknowledgment).

For the model that lies in the basis of our analysis, we shall bring the above 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 means that the size of the window that may be transmitted within a round-trip delay is unlimited. Furthermore, we assume that 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 allow us to simplify the analysis, capturing the essential properties of the resulting strategies without having to deal, from the outset, with details of secondary importance. We shall see later that, though the model requires certain extensions if the above assumptions are alleviated, the optimal strategy solution methodology remains based on the same principles as developed for this, 'idealized' case; a more detailed discussion in this regard appears in the Conclusion.

We denote the loss rate in the network by L, and assume the loss probability of a packet to be independent of other packets (and of its own earlier retransmissions). Thus, the losses may be caused by white noise, or, for instance, a randomizing queue management policy employed on the connection's path (such as RED, which discards packets randomly during times of mild congestion). Our analysis is less applicable if there is a significant correlation between the losses of neighboring packets (e.g. at times of severe congestion), since retransmitting several copies of a packet does not reduce its loss probability significantly then. Furthermore, we assume the loss rate of acknowledgments (generated by the receiver in response to successfully received packets) to be negligible compared to that of the data packets. This is justified by the fact that acknowledgments are, typically, much shorter than data packets, and therefore suffer less from noise and their paths are often less congested. In addition, acknowledgments are capable of mutual substitution, since they only need to carry the index of the next expected packet; hence, a lost acknowledgment has no significance if another one, later in the window, is received successfully. Consequently, we assume that, for each transmitted packet, the sender knows whether it was successfully received or lost after a round-trip time, which we denote by T.

Our analysis is aimed to find the optimal strategy, defined as one that minimizes the cost/throughput ratio over time. We assume that the cost is composed of a 'price' of a per unit of time and b per transmitted packet. As explained in the Introduction, these prices quantify the tradeoff between waiting too long and loading the network too much; properly selected, they can embed the delay or congestion cost (disutility) of the application or the network itself, and need not necessarily be taken literally. Incidentally, we chose to base our analysis on this cost structure, which is linear in the time and number of transmitted packets, reckoning that it is appropriate for a variety of scenarios and cost interpretations [5]; however, any other (nonlinear) cost structure may be used instead, as long as the cost of transmitting a window of packets depends only on its size, and not on the identity of the packets within or the number of packets actually succeeded/lost. This may affect only the analytical results, regarding the asymptotic dependence of the optimal window size on the costs, whereas the algorithm we present for computing the optimal strategy itself, and the line of reasoning that leads to it, remain intact (after substituting the appropriate cost formula).

By showing how to compute the optimal strategy from the connection parameters (L,T,a,b), we implicitly assume that they are known to the entity performing the computation – presumably, the sender machine itself. Therefore, these parameters must either remain constant or change in a quasi-static manner, so that a new optimal strategy is computed after detecting a change (e.g. in the round-trip time). Our model is inadequate if any of the parameters, e.g. the round-trip time, changes quickly and unpredictably; in that case, it should be modeled by a random variable (e.g., as in [5]) rather than a constant value. We point out that this occurence is not typical of the kind of network connections that are the subject of this study: e.g., for satellite links, the propagation delay is the dominant part of the round-trip delay and is constant, up to small fluctuations that can normally be neglected.

In light of the above assumptions, it is apparent that, in the optimal strategy, packets are only sent at times that are multiples of T. There can never be any purpose to send a packet at a time other than a multiple of T, since no extra information is present, and by 'lumping' all such transmissions to the nearest earlier multiple of T (maintaining the order between transmissions), nothing is lost and time is only gained. Furthermore, after having sent a sequence of packets at time nT, it is known by time $(n+1)T$ which packets have been received and which ones lost/discarded, so

the strategy simply restarts ('slides') at the packet subsequent to the last arriving in order. Consequently, the description of a strategy consists simply of a single vector, specifying how many transmissions are to be made 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.

B. Problem formulation

Consider a vector $\langle n_1, \ldots, n_i, \ldots \rangle$, where n_i are whole and non-negative, and define a random variable S, which is 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.† The distribution of S is

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

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

$$\frac{a \cdot T + b \cdot \sum_{i=1}^{\infty} n_i}{\mathrm{E}[S]} = \frac{a \cdot T + b \cdot \sum_{i=1}^{\infty} n_i}{\sum_{j=1}^{\infty} j \cdot P_S(j)}. \tag{2}$$

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.

Let us look more closely at expression (2). For any N, consider all vectors such that $\sum_{i=1}^{\infty} n_i = N$, i.e. suggesting the same total number of transmissions. These vectors attain the same numerator value in (2); therefore, the comparison among them is based merely on the denominator value, and the best vector is the one that maximizes it. Consequently, let us define

$$\mathrm{E}_L(N) \triangleq \max_{\substack{n_1,n_2,\ldots \\ s.t. \sum_i n_i = N}} \left\{ \sum_{j=1}^{\infty} j \cdot P_S(j) \right\} = \max_{\substack{n_1,n_2,\ldots \\ s.t. \sum_i n_i = N}} \left\{ \sum_{j=1}^{\infty} j \cdot \prod_{i=1}^{j} (1 - L^{n_i}) \cdot L^{n_{j+1}} \right\}, \tag{3}$$

and rewrite expression (2) accordingly as

$$\frac{a \cdot T + b \cdot N}{\mathrm{E}_L(N)}. \tag{4}$$

Then, the problem of finding the strategy vector that minimizes (4) can be divided into the following (sub-)problems:
Inner problem: Computing the function $\mathrm{E}_L(N)$ for a given value of N.
Outer problem: Searching for N^* that minimizes expression (4).

In the rest of the paper, we treat the two problems separately for the most part. It should be noted that, by this separation, the (infinite-dimensional) problem of finding the optimal strategy vector for a combination of parameters (a,b,T,L) is divided into simpler problems, namely, computing a function $\mathrm{E}_L(N)$ that depends only on L, while the dependence on the other parameters is captured in a merely one-dimensional minimization problem. Furthermore, we emphasize that the vector that actually attains the maximum in (3) is required only for the final stage, after N^* has been found; during the search of N, it suffices to be able to compute (or even approximate) $\mathrm{E}_L(N)$, 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 only sent once in a window; this corresponds to a strategy vector of $n_1 = \cdots = n_N = 1$, which brings about a cost/throughput ratio of

$$\frac{a \cdot T + b \cdot N}{\sum_{j=1}^{N-1} j \cdot (1-L)^j \cdot L + N \cdot (1-L)^N} = \frac{a \cdot T + b \cdot N}{\sum_{j=1}^{N} (1-L)^j} = \frac{L}{1-L} \cdot \frac{a \cdot T + b \cdot N}{1 - (1-L)^N}. \tag{5}$$

† It is obvious that this is the best order in which to transmit packets; not transmitting them in-order can only decrease the expected number of in-order arrivals.

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

$$N^* = \frac{1}{\log\frac{1}{1-L}}\left[-1+\frac{aT}{b}\log(1-L)-\operatorname{plog}\left(-\frac{(1-L)^{\frac{aT}{b}}}{e}\right)\right] \underset{(for\, aT\gg b)}{\approx} \frac{\log\left(\frac{aT}{b}\log\frac{1}{1-L}+1\right)}{\log\frac{1}{1-L}}, \tag{6}$$

where $\operatorname{plog}(\cdot)$ (the product-log function) denotes the inverse function of $f(t)=t\cdot e^t$, such that $t=-\operatorname{plog}(-y)$ (for $0<y\leq\frac{1}{e}$) is the largest positive solution to the equation $y=t\cdot e^{-t}$; in the final approximation we used the property that $-\operatorname{plog}(-e^{-x})\approx x+\log x$ for $x\gg 1$.[‡] 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 (5) tends to a finite value as $N\to\infty$, the cost/throughput ratio, overall, increases linearly in a. We shall see later that the ability to use retransmissions within the window enables the cost/throughput to grow much more slowly, namely, logarithmically in a.

III. Basic properties and bounds

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

A. Properties of the inner problem

We begin by introducing a variable change that will make the subsequent analysis 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). Then, the maximized expression of (3) takes the simpler form of

$$\sum_{j=1}^{\infty} j\cdot\prod_{i=1}^{j}(1-L^{n_i})\cdot L^{n_{j+1}} = \sum_{j=1}^{\infty} j\cdot\prod_{i=1}^{j} p_i\cdot(1-p_{j+1}) = 1\cdot p_1(1-p_2)+2\cdot p_1p_2(1-p_3)+3\cdot p_1p_2p_3(1-p_4)+\cdots =$$

$$p_1+p_1p_2+p_1p_2p_3+\cdots = \sum_{j=1}^{\infty}\prod_{i=1}^{j} p_i = \sum_{j=1}^{\infty}\prod_{i=1}^{j}(1-L^{n_i}). \tag{7}$$

We henceforth refer to the rightmost side of (7) as the *score* of the vector $\vec{n}$, and denote it by $\phi(\vec{n})$. Also, we 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 the vector $\vec{p}$.

The next two lemmas describe basic structural properties of the solution.

Lemma 1. *$\mathrm{E}_L(N)$ decreases in L and increases in N.*

Proof. Consider the maximizing vector for a certain value of L and N. Now, suppose that L is decreased; then the score of that vector increases. If it is no longer the maximizer for the new value of L, then, obviously, the maximum value can only be even higher. Therefore, the value of (3) 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 value of N, the value of (3) can only increase further. □

Lemma 2. *For a given N, the elements of the vector that achieves the maximum in* (3) *maintain a non-increasing order, i.e. $n_1\geq n_2\geq\cdots\geq n_i\geq\ldots$.*

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 (7). All the sum elements (products) for $j<i_1$ (which depend on neither n_{i_1} nor n_{i_2}), as well as for $j\geq i_2$ (which contain both $(1-L^{n_{i_1}})$ and $(1-L^{n_{i_2}})$ in the product), remain unchanged. The elements for $i_1\leq j<i_2$, 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 1. *In the maximizing vector, all the elements after the first zero element are also zero.*

[‡] The product-log function is also known elsewhere as Lambert's W-function [14], or, more precisely, as one of its real-valued branches.

Corollary 2. *For a given N, the number of nonzero elements in the maximizing vector is bounded by N.*

Lemma 2 and its corollaries state a fundamental property, which we later use extensively, both in the proof of bounds on the maximum score that can be attained by a vector of a given size N and in the search for the maximizing vector. In particular, as we shall see in section IV, this property significantly reduces the number of 'eligible' vectors that need to be searched, and allows efficient search algorithms to be used for the problem solution.

The next lemma states a basic inequality that relates between the first element of a vector and its score. The following theorem then applies it inductively to derive an important bound on the number of transmissions required to attain a given score.

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

Proof. From Lemma 2, it follows that

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

and the lemma immediately follows by extracting p_1. □

Theorem 1. *For any vector $\vec{n}$, $N = \sum_{i=1}^{\infty} n_i \geq \log_{1/L} \{[\phi(\vec{n}) + 1]!\}$.*[†]

Proof. Obviously, since the factorial and the logarithm are monotonously increasing operations, it suffices to prove the theorem just for the vector $\vec{n}$ that attains the maximum score for a given N. Such a vector is known to satisfy lemmas 2 and 3.

Consider the equivalent vector $\vec{p} = \langle p_1, \ldots, p_M, 0, 0, \ldots \rangle$, 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, $\phi_m = \phi\left(\vec{p}^{(m)}\right) = \sum_{j=m}^{M} \prod_{i=1}^{j} p_i$, for all $1 \leq m \leq M$. Thus, ϕ_1 is the score of the original vector, while ϕ_M simply equals p_M.

Applying Lemma 3 on each of these subvectors in turn, we have $p_m \geq \frac{\phi_m}{\phi_m+1}$, or $1 - p_m \leq \frac{1}{\phi_m+1}$. Consequently, $\left[\prod_{m=1}^{M} (1-p_m)\right]^{-1} \geq \prod_{m=1}^{M} (\phi_m + 1)$.

However, it can be seen (directly from the definition) that $\phi_m = p_m (1 + \phi_{m+1})$, and, therefore, $\phi_{m+1} \geq \phi_m - 1$, for all $1 \leq m < M$. By successive application of this inequality, we get $\phi_m \geq \phi_1 - (m-1)$ for all m. Therefore, we have so far

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

On the other hand, 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 $t \geq 1$, we have

$$(\phi_1 + 1)! = (\phi_1 + 1) \cdot \phi_1 \cdot (\phi_1 - 1) \cdot \ldots \cdot (\phi_1 - \lfloor \phi_1 \rfloor)! = \prod_{m=1}^{M} \max\left[\phi_1 - (m-1) + 1, 1\right] \cdot (\phi_1 - \lfloor \phi_1 \rfloor)! \leq$$
$$\prod_{m=1}^{M} \max\left[\phi_1 - (m-1) + 1, 1\right].$$

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

Combining all the above inequalities, we finally obtain

$$\left[\prod_{m=1}^{M} (1-p_m)\right]^{-1} \geq (\phi_1 + 1)!,$$

[†] Recall that the factorial $t!$, for any $t \geq 0$, is defined by $t! = \int_0^{\infty} x^t e^{-x} dx$; this definition conincides 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 $t \geq 1$.

from which, by taking the logarithm of both sides and noting that $\log_{1/L}(1-p_m) = \log_{1/L}\left(L^{n_m}\right) = -n_m$, we get $\sum_{m=1}^{M} n_m \geq \log_{1/L}\left[(\phi_1+1)!\right]$. □

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. $\mathrm{E}_L(N) = \Theta\left(\frac{N}{\log_{1/L} N}\right)$.†

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

$$N \geq \log_{1/L}\left[\mathrm{E}_L(N)+1\right]! \approx \mathrm{E}_L(N)\cdot\log_{1/L}\left(\frac{\mathrm{E}_L(N)+1}{e}\right) + \log_{1/L}\sqrt{2\pi[\mathrm{E}_L(N)+1]};$$

thus, $N = \Omega\left(\mathrm{E}_L(N)\cdot\log_{1/L}\mathrm{E}_L(N)\right)$. This implies directly that $\mathrm{E}_L(N) = O\left(\frac{N}{\log_{1/L} N}\right)$.

To show that $\mathrm{E}_L(N) = \Omega\left(\frac{N}{\log_{1/L} N}\right)$ as well, it suffices to find a strategy vector, not necessary the optimal one, that attains a score of $\Omega\left(\frac{N}{\log_{1/L} N}\right)$. 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 $M = \frac{N}{\log_{1/L} N}$ (i.e. a strategy that transmits $\frac{N}{\log_{1/L} N}$ packets an equal number of times). Its score is

$$\sum_{j=1}^{M}\prod_{i=1}^{j}\left(1-L^{n_i}\right) = \sum_{j=1}^{\frac{N}{\log_{1/L} N}}\left(1-L^{\log_{1/L} N}\right)^j = \sum_{j=1}^{\frac{N}{\log_{1/L} N}}\left(1-\frac{1}{N}\right)^j =$$

$$N\left(1-\frac{1}{N}\right)\left[1-\left(1-\frac{1}{N}\right)^{\frac{N}{\log_{1/L} N}}\right] \geq N\left(1-\frac{1}{N}\right)\left(1-e^{-\frac{1}{\log_{1/L} N}}\right),$$

which completes the proof, since $e^{-\frac{1}{x}} \approx 1-\frac{1}{x}$ for large x. Incidentally, note that the fact that either $\log_{1/L} N$ or $\frac{N}{\log_{1/L} N}$ may not be integers, which was ignored above, is only a minor technical difficulty; to overcome it, simply round either or both of them up to the nearest integer, if necessary. This can only increase the vector's score even further, while the sum of the vector's elements is raised by $\frac{N}{\log_{1/L} N}+1$ at most, which is negligible compared to N and does not change the asymptotic relation. □

It is insightful to compare the result of Theorem 2 with the total number of packets that reach the receiver successfully (not necessarily in order), which is, obviously, $N\cdot(1-L)$, i.e., $\Theta(N)$. Hence, it can be thought that discarding out-of-order packets impacts the performance by a logarithmic factor. Note, incidentally, that this theorem can also be used inversely; that is, 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 $\Theta(\phi\cdot\log\phi)$.

B. Properties of the outer problem

This subsection is concerned with the dependence of the optimal window size on the cost factors. Lemma 4 states the intuitive fact that the optimal window size increases in the time cost and decreases in the packet transmission cost, and that it depends only on the ratio between the two. Theorem 3 states the central result of this section, namely, that with the optimal window size, the cost/throughput ratio increases merely logarithmically in the time cost a, not linearly as was the case with 'classic' sliding windows in the end of section II.

Lemma 4. *The optimal N^* is monotonously increasing in $\frac{aT}{b}$.*

† Recall that, for positive functions $f(x), g(x)$, the notation $f(x) = O(g(x))$ means that there exists a finite constant C such that $f(x) \leq C\cdot g(x)$ for all (sufficiently large) x; a sufficient condition for this is $\lim_{x\to\infty}\frac{f(x)}{g(x)} < \infty$. In addition, $f(x) = \Omega(g(x))$ is equivalent to $g(x) = O(f(x))$, and $f(x) = \Theta(g(x))$ means that both $f(x) = O(g(x))$ and $f(x) = \Omega(g(x))$.

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

$$\frac{a_1 T_1 + b_1 N_2^*}{\mathrm{E}_L\left(N_2^*\right)} \geq \frac{a_1 T_1 + b_1 N_1^*}{\mathrm{E}_L\left(N_1^*\right)},$$
$$\frac{a_2 T_2 + b_2 N_2^*}{\mathrm{E}_L\left(N_2^*\right)} \leq \frac{a_2 T_2 + b_2 N_1^*}{\mathrm{E}_L\left(N_1^*\right)},$$

or

$$\left(\frac{a_1 T_1}{b_1} + N_2^*\right) \cdot \mathrm{E}_L\left(N_1^*\right) \geq \left(\frac{a_1 T_1}{b_1} + N_1^*\right) \cdot \mathrm{E}_L\left(N_2^*\right),$$
$$\left(\frac{a_2 T_2}{b_2} + N_2^*\right) \cdot \mathrm{E}_L\left(N_1^*\right) \leq \left(\frac{a_2 T_2}{b_2} + N_1^*\right) \cdot \mathrm{E}_L\left(N_2^*\right).$$

Subtracting the second inequality from the first and dividing by a common factor of $\left(\frac{a_1 T_1}{b_1} - \frac{a_2 T_2}{b_2}\right)$, which is positive by assumption, we obtain $\mathrm{E}_L\left(N_1^*\right) \geq \mathrm{E}_L\left(N_2^*\right)$. In light of the monotonicity of $\mathrm{E}_L(N)$ (Lemma 1), this implies $N_1^* \geq N_2^*$. □

Theorem 3. *As $a \to \infty$ (for fixed values of T, b, L), the cost/performance ratio attained by the optimal strategy (as given by expression* (4)*) increases logarithmically in a.*

Proof. Consider the expression $h(x) \triangleq \frac{a \cdot T + b \cdot x}{x / \log_{1/L} x}$, as a function of a (continuous) variable x. By differentiation with respect to x, it is easily found that its minimum is attained at $x^* = \frac{aT}{b} \cdot \left[-\operatorname{plog}\left(-e \cdot \frac{b}{aT}\right)\right]$.† Using again the property that $-\operatorname{plog}\left(-e^{-y}\right) \approx y + \log y$ for very 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 (expression (4)) is itself $\Theta(h(N))$, and its minimum value can, therefore, deviate from that of $h(N)$ by a constant factor at most. □

Incidentally, it should be noted that no similar result exists for $b \to \infty$ with the other parameters constant. Indeed, as $\frac{aT}{b} \to 0$, the optimal strategy tends to $\langle 1, 0, 0, \ldots \rangle$ (simple stop-and-wait), and the value of expression (4) simply increases linearly in b. This is true, of course, for the 'classic' case as well.

IV. Approximation algorithms for the inner problem

In this and the next section, we consider approaches for approximation of the function $\mathrm{E}_L(N)$ (i.e. the solution of the inner problem). This section focuses on direct-search algorithms; the next section presents an alternative approach of solving a similar auxiliary problem in continuous variables, that is more easily amenable to analysis.

To begin, we note that the most straightforward approach, arguably, is exhaustive search among all the nonnegative integer vectors with elements in non-increasing order (in light of Lemma 2), that sum up to N. Figure 1 shows how the number of such vectors increases in N. Thus, for $N = 10$ there are only 42 vectors to check, and this number increases to $204\,226$ for $N = 50$ and $190\,569\,292$ for $N = 100$. It can be concluded that this approach is quite viable for small values of N, especially considering that the computation of the optimal sliding-window strategy is only needed occasionally (when the parameters of the problem, e.g. the round-trip delay, are changed), and can be performed offline.

For larger values of N, exhaustive search may not be practical, and we seek an alternative that produces a reasonably approximate solution at a low computational cost. Figures 2, 3, and 4 show three possible algorithms. All these algorithms are 'greedy' in the sense that they proceed in iterations, seeking the best possible improvement in each iteration until unable to find any further improvement. Algorithm 'Greedy-R' begins with the vector $\langle N, 0, 0, \ldots \rangle$ and, at every iteration, seeks the best score that can be obtained by a right-move — namely, by decrementing an element by 1 and incrementing another element further to the right, provided that the resulting vector remains non-increasing. Algorithm 'Greedy-L' operates the other way around; it begins with the vector $\langle \underbrace{1, \ldots, 1}_{N} \rangle$ and repeatedly seeks to improve the score by left-moves. Finally, algorithm 'Greedy-A' makes additions instead of moves; that is, it begins with a vector of all-zeros, and at each iteration finds the element whose increment brings about the highest score, keeping the vector non-increasing. Obviously, Greedy-A always makes exactly N iterations.

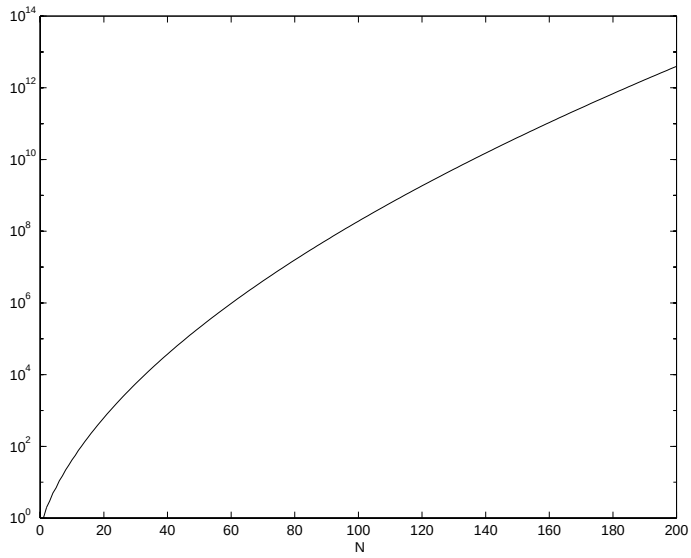

Fig. 1. The number of nonnegative integer vectors summing up to N, with elements in non-increasing order, as a function of N. Note the vertical scale is logarithmic.

Initialization: Set $\vec{n} = \langle N, 0, 0, \dots \rangle$
Repeat:
    For every i, j such that $i < j$, $n_i > n_{i+1}$, and $n_{j-1} > n_j$:
        Temporarily set $n_i \leftarrow n_i - 1$, $n_j \leftarrow n_j + 1$
        Compute the score of $\vec{n}$
        Restore n_i, n_j
    For the i, j pair that got the best score:
        If the score is better than that of the current vector:
            Set $n_i \leftarrow n_i - 1$, $n_j \leftarrow n_j + 1$, go back to *Repeat*
        Else terminate.

Fig. 2. Algorithm Greedy-R.

Initialization: Set $\vec{n} = \langle \underbrace{1, \dots, 1}_{N}, 0, \dots \rangle$
Repeat:
    For every i, j such that $i > j$, $n_i > n_{i+1}$, and either $j = 1$ or $n_{j-1} > n_j$:
        Temporarily set $n_i \leftarrow n_i - 1$, $n_j \leftarrow n_j + 1$
        Compute the score of $\vec{n}$
        Restore n_i, n_j
    For the i, j pair that got the best score:
        If the score is better than that of the current vector:
            Set $n_i \leftarrow n_i - 1$, $n_j \leftarrow n_j + 1$, go back to *Repeat*
        Else terminate.

Fig. 3. Algorithm Greedy-L.

Initialization: Set $\vec{n} = \langle 0, 0, \dots \rangle$
Do N times:
    For every j such that either $j = 1$ or $n_{j-1} > n_j$:
        Temporarily set $n_j \leftarrow n_j + 1$
        Compute the score of $\vec{n}$
        Restore n_j
    For the j that got the best score:
        Set $n_j \leftarrow n_j + 1$

Fig. 4. Algorithm Greedy-A.

It should be understood that the three algorithms are not equivalent in terms of complexity. The algorithms Greedy-R and Greedy-L can, in the worst case, require $O\left(N^2\right)$ iterations, and each iteration may potentially require $O\left(N^2\right)$ score evaluations. Algorithm Greedy-A, on the other hand, requires only N iterations, and each iteration requires no more than N score evaluations.[‡] We shall see that the lower complexity of Greedy-A comes at the expense of achieving a lower score more frequently.

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	$\langle 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 0, 0 \rangle$	8.41131
0.3	$\langle 3, 2, 2, 2, 2, 2, 1, 1, 0, 0, 0, 0 \rangle$	5.39436
0.5	$\langle 4, 3, 3, 2, 2, 1, 0, \dots, 0 \rangle$	3.61954
0.7	$\langle 6, 5, 3, 1, 0, \dots, 0 \rangle$	2.24336
0.9	$\langle 11, 4, 0, \dots, 0 \rangle$	0.92217

It can be seen that, in accordance with intuition, the lower the loss probability, the better it is to send at least one copy of more packets; conversely, when the loss probability is high, the highest expected number of successful in-order arrivals is attained by sending just the first few packets many times. In fact, it is obvious that, for any N, the optimal vector tends to $\langle \underbrace{1, \dots, 1}_{N} \rangle$ for $L \to 0$ and to $\langle N, 0, \dots, 0 \rangle$ for $L \to 1$.

The values in the above table were found by exhaustive search. The greedy algorithms did not converge to these vectors in all cases. Specifically, for $L = 0.3$, the Greedy-A algorithm found the vector $\langle 3, 3, 2, 2, 2, 2, 1, 0, \dots, 0 \rangle$, with a somewhat lower score of 5.38234. For $L = 0.5$, only the Greedy-L algorithm converged to the optimal solution, while the other two found the vector $\langle 4, 4, 3, 3, 1, 0, \dots, 0 \rangle$, with a score of 3.59482. In general, it is possible that none of the three algorithms converges to the optimal vector. The smallest N for which an example of this possibility was found is $N = 90$, at $L = 0.505$; there, the value of $\mathrm{E}_L(N)$ (found by exhaustive search) is 14.3278, while Greedy-L attains only 14.2939, and Greedy-R and Greedy-A achieve an even lower score of 14.2535. □

We point out that the good performance of all the algorithms for 'extreme' values of L (close to 0 or to 1), and less-than-optimal results for L around 0.5, as exhibited in the above example, are to be expected. In fact, it can be proved that when the maximizing vector is indeed $\langle \underbrace{1, \dots, 1}_{N} \rangle$ or $\langle N, 0, \dots, 0 \rangle$, all the greedy algorithms converge to the correct vector. (We do not go into the formal details of the proof.) On the other hand, for $L \approx 0.5$, the optimum is much less proclaimed (i.e., there is a large number of vectors with scores that are very close to the optimum), which causes the occasional convergence of the greedy algorithms to nearby vectors.

Nevertheless, we found the algorithms' results to be always very close to each other. We have run them for all values of $N \leq 300$ and L between 0.001 and 0.999 in increments of 0.001; we also ran the exhaustive search on all values of $N \leq 100$ (for values of N higher than that, the search turned out to be too time-consuming). Figure 5 shows the lowest

[†] Recall the definition of the product-log function at the end of section II.

[‡] This complexity estimation counts each score evaluation as a single operation. In fact, since there is a great deal of repetition in the score computations, a lot of elementary operations (multiplications and summations) can be saved by using auxiliary storage for temporary results; however, even with the best optimization, the computational cost of Greedy-R and Greedy-L is still higher by at least an order of N than that of Greedy-A.

score ratio between the worst and the best algorithm, taken over all values of L, as a function of N. It turns out that the worst score ratio is attained at $N = 5$, $L = 0.5$: there, Greedy-A converges to a score of 1.53125, while all the other algorithms reach 1.59375, for a worst-to-best ratio of approximately 0.96; from there, it tends to get closer to 1. In fact, for $N \geq 43$, the worst-to-best score ratio never drops below 0.99, for any value of L.

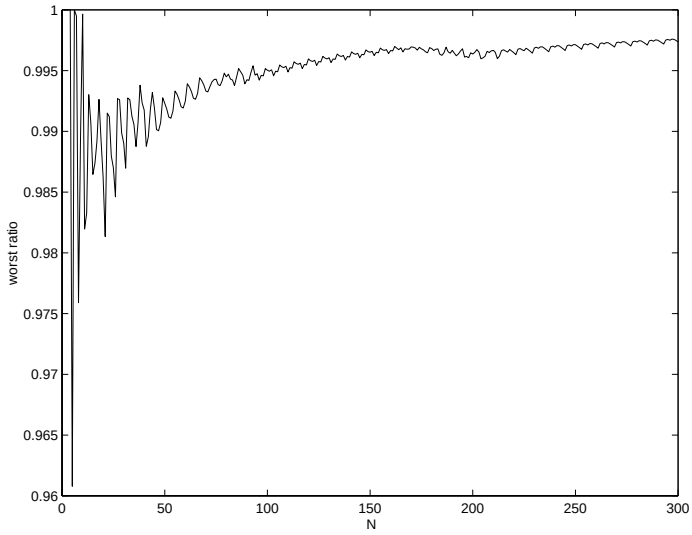

Fig. 5. The lowest ratio between any two algorithms (over all L), as a function of N. The lowest point of the graph is approximately 0.9608 for $N = 5$ (reached for $L = 0.5$). The graph stays above 0.99 for all $N \geq 43$.

It is worthwhile mentioning that, in all the points of Figure 5, the worst algorithm was Greedy-A. This is not to say that it always achieves the worst score; in fact, it frequently reached the best score, with one of the other algorithms lagging behind (though there was not a single instance in which the Greedy-A algorithm was the *single* best among all three). However, in such cases the ratio between that other algorithm's score and Greedy-A's score was never as low as the inverse ratio for some different value of L, for which the case was reversed and Greedy-A had the worst score.

If one is willing to accept the conjecture that the graph of Figure 5 continues similarly for $N > 300$, then the conclusion emerging from the results of this section is that the Greedy-A algorithm, which is computationally the cheapest (fastest) one, can be used to approximate the optimal retransmission strategy with only a small deviation from optimality, probably acceptable for most purposes.

V. Approximation through continuous relaxation

In this section, we analyze the properties of the same optimization problem as the one that defines the function $\mathrm{E}_L(N)$, except that the requirement for the strategy vector elements to be integers is omitted. This way, we have a relaxed optimization problem in a continuous space, which can be analyzed more easily by 'traditional' methods from optimization theory. Unlike the direct-search methods of the previous section, which resulted in vectors with lower scores than $\mathrm{E}_L(N)$, our present technique obtains a value that is higher than $\mathrm{E}_L(N)$. This can be regarded as an improvement to the upper bound derived earlier (in Theorem 1); we shall show in the end of this section that this bound is tight, in a certain sense, and therefore can be used as an alternative approximation method for $\mathrm{E}_L(N)$, especially for large N.

To distinguish the relaxed problem from the original one, we change the notation and refer to the strategy vector as $\langle x_1, x_2, \dots \rangle$, and to the function in the denominator of (4) as $\Phi_L(s)$:

$$\Phi_L(s) \triangleq \max_{\substack{x_1,x_2,\dots \\ s.t.\ x_i \geq 0, \sum_i x_i = s}} \left\{ \sum_{j=1}^{\infty} j \cdot \prod_{i=1}^{j} (1 - L^{x_i}) \cdot L^{x_{j+1}} \right\}. \tag{8}$$

In other words, $\Phi_L(s)$ is defined exactly like $\mathrm{E}_L(N)$ except that the maximum is taken over vectors whose elements are not restricted to be integers. Note that, in particular, $\Phi_L(s)$ is well-defined for non-integer values of s.

We shall again make the convenient variable change of $p_i \triangleq 1 - L^{x_i}$, after which the score expression is simply $\sum_{j=1}^{\infty} \prod_{i=1}^{j} p_i$ (see equation (7)), while the constraints on x_i become

$$x_i \geq 0 \qquad \Longrightarrow \qquad 0 \leq p_i \leq 1; \tag{9}$$

$$\sum_i x_i = \sum_i \log_L (1 - p_i) = s \qquad \Longrightarrow \qquad \prod_i (1 - p_i) = L^s. \tag{10}$$

To summarize, we seek the solution of the following optimization problem, in terms of the vector $\vec{p} = \langle p_1, p_2, \ldots \rangle$:

$$\begin{array}{ll} \text{maximize} & \displaystyle\sum_{j=1}^{\infty} \prod_{i=1}^{j} p_i = p_1 + p_1 p_2 + p_1 p_2 p_3 + \ldots \\ \text{subject to} & \displaystyle 0 \leq p_i \leq 1, \quad \prod_{i=1}^{\infty} (1 - p_i) = L^s. \end{array}$$

It is to be noted at once that this problem essentially depends on just one parameter, L^s (though the translation back to the original variables, $x_i = \log_L (1 - p_i)$, involves the specific value of L).

We proceed to derive the solution to this problem and study its properties. We begin by noting that the solution vector, i.e. one that attains the global maximum must, in particular, be also a local maximum. Consequently, as both the target expression and the constraint are differentiable, we can employ the technique of defining the Lagrangian

$$\mathcal{L} = \sum_{j=1}^{\infty} \prod_{i=1}^{j} p_i + \lambda \left[\prod_{i=1}^{\infty} (1 - p_i) - L^s \right],$$

and requiring that the solution satisfy the corresponding Kuhn-Tucker conditions [15], which require that

$$\frac{\partial \mathcal{L}}{\partial p_m} = 0 \qquad \text{if } 0 < p_m < 1, \tag{11}$$

$$\frac{\partial \mathcal{L}}{\partial p_m} \leq 0 \qquad \text{if } p_m = 0. \tag{12}$$

(In fact, there is also a condition to be fulfilled if $p_m = 1$; however, it is immediately seen that the constraint $p_m \leq 1$ cannot be active, i.e. no element of the solution vector can be equal to 1, since this would immediately contradict the equality constraint (10).)

Let us develop equation (11) into an explicit form:

$$\frac{\partial \mathcal{L}}{\partial p_m} = \sum_{j=m}^{\infty} \prod_{\substack{i=1 \\ i \neq m}}^{j} p_i - \lambda \prod_{\substack{i=1 \\ i \neq m}}^{\infty} (1 - p_i) = \sum_{j=m}^{\infty} \prod_{\substack{i=1 \\ i \neq m}}^{j} p_i - \frac{\lambda \cdot L^s}{1 - p_m} = 0,$$

consequently

$$(1 - p_m) \sum_{j=m}^{\infty} \prod_{\substack{i=1 \\ i \neq m}}^{j} p_i = \lambda \cdot L^s \triangleq \lambda' \text{ (constant for all } m \text{ such that } p_m > 0), \tag{13}$$

or, in an alternative form that will prove to be more convenient,

$$\frac{1 - p_m}{p_m} \sum_{j=m}^{\infty} \prod_{i=1}^{j} p_i = \lambda' \text{ (constant for all } m \text{ such that } p_m > 0). \tag{13'}$$

Similarly, from inequality (12) we get

$$\sum_{j=m}^{\infty} \prod_{\substack{i=1 \\ i \neq m}}^{j} p_i \leq \lambda' \text{ (for all m such that } p_m = 0). \tag{14}$$

Using (13)–(14), we are now in a position to prove a few structural properties of the solution vector. The next few lemmas show that it has a finite length and its elements are monotonously decreasing; note that these properties are similar to those of the solution vector of the original (non-relaxed) problem, as stated by Lemma 2.

Lemma 5. *If $p_M = 0$ for some index M, then $p_m = 0$ for any $m > M$.*

Proof. Immediate from (13): given that $p_M = 0$ for some $M < m$, the left-hand size of (13) equals 0, thus the equality cannot be satisfied; consequently, it is impossible to have $p_m > 0$. □

Lemma 6. *The positive elements of the solution vector maintain a strictly decreasing order.*

Proof. Choose a pair of indices m_1, m_2, such that $m_1 < m_2$ and $p_{m_1} > 0, p_{m_2} > 0$. Condition (13') requires that $\frac{1-p_{m_1}}{p_{m_1}} \sum_{j=m_1}^{\infty} \prod_{i=1}^{j} p_i = \frac{1-p_{m_2}}{p_{m_2}} \sum_{j=m_2}^{\infty} \prod_{i=1}^{j} p_i$. Since, obviously, $\sum_{j=m_1}^{\infty} \prod_{i=1}^{j} p_i > \sum_{j=m_2}^{\infty} \prod_{i=1}^{j} p_i$, it follows that $\frac{1-p_{m_1}}{p_{m_1}} < \frac{1-p_{m_2}}{p_{m_2}}$. By a straightforward simplification, this implies $p_{m_1} > p_{m_2}$. □

Lemma 7. *There exists a finite M such that $p_m = 0$ for any $m > M$.*

Proof. Suppose, by contradiction, that such M does not exist; in light of Lemma (5), it follows that the solution vector consists of an infinite sequence of positive elements. Choose an index M such that $p_M < \frac{1}{2}$; such M must exist, otherwise it would be impossible to satisfy constraint (10) (the product on its left-hand side would converge to 0). We now show that the assumption $p_{M+1} > 0$ leads to a contradiction.

If $p_{M+1} > 0$, then condition (13') requires

$$\frac{1-p_{M+1}}{p_{M+1}} \sum_{j=M+1}^{\infty} \prod_{i=1}^{j} p_i = \frac{1-p_M}{p_M} \sum_{j=M}^{\infty} \prod_{i=1}^{j} p_i = \frac{1-p_M}{p_M} \left(\prod_{i=1}^{M} p_i + \sum_{j=M+1}^{\infty} \prod_{i=1}^{j} p_i \right),$$

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

$$\frac{1-p_{M+1}}{p_{M+1}} \sum_{j=M+1}^{\infty} \prod_{i=M+1}^{j} p_i = \frac{1-p_M}{p_M} \left(1 + \sum_{j=M+1}^{\infty} \prod_{i=M+1}^{j} p_i \right) \quad \Longrightarrow \quad \sum_{j=M+1}^{\infty} \prod_{i=M+1}^{j} p_i = \frac{(1-p_M)\, p_{M+1}}{p_M - p_{M+1}}.$$

However, as a result of Lemma 6, the following inequality holds:

$$\sum_{j=M+1}^{\infty} \prod_{i=M+1}^{j} p_i \le \sum_{j=M+1}^{\infty} \prod_{i=M+1}^{j} p_{M+1} = \sum_{j=M+1}^{\infty} (p_{M+1})^{j-M} = \frac{p_{M+1}}{1-p_{M+1}}.$$

Substituting this inequality into the previous equation, we therefore get

$$\frac{(1-p_M)\, p_{M+1}}{p_M - p_{M+1}} \le \frac{p_{M+1}}{1-p_{M+1}} \quad \Longrightarrow \quad (1-p_M)\,(1-p_{M+1}) \le p_M - p_{M+1} \quad \Longrightarrow \quad 1 + p_M p_{M+1} \le 2p_M,$$

which contradicts $p_M < \frac{1}{2}$. Thus, condition (13') cannot be satisfied; therefore, $p_{M+1} = 0$. □

Lemma 8. *If a vector $\langle p_1, p_2, \ldots, p_M, 0, 0, \ldots \rangle$, where $p_m > 0$ for all $1 \le m \le M$, satisfies condition* (13)*, then it satisfies condition* (14) *if and only if $p_M \le \frac{1}{2}$.*

Proof. For p_{M+2} and beyond, the left-hand side of (14) equals 0 (as it includes a factor of $p_{M+1} = 0$), and the condition is satisfied trivially. For p_{M+1}, upon substituting the value of λ' evaluated at p_M at the right-hand side, condition (14) becomes

$$\sum_{j=M+1}^{\infty} \prod_{\substack{i=1 \\ i \ne M+1}}^{j} p_i \le (1-p_M) \sum_{j=M}^{\infty} \prod_{\substack{i=1 \\ i \ne M}}^{j} p_i \quad \Longleftrightarrow \quad \prod_{i=1}^{M} p_i \le (1-p_M) \prod_{i=1}^{M-1} p_i,$$

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

It follows from the above lemmas that the solution to the maximization problem is of the form $\langle p_1, p_2, \ldots, p_M, 0, 0, \ldots \rangle$, where $p_m > 0$ for all $1 \leq m \leq M$, and $p_M \leq \frac{1}{2}$. We shall henceforth use M solely to denote the index of the last nonzero element of the solution vector.

Suppose that a vector $\langle p_1, p_2, \ldots, p_M, 0, 0, \ldots \rangle$ is known to satisfy conditions (9) and (13)–(14), 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, p_m can be computed using condition (13), as follows:

$$(1-p_m)\sum_{j=m}^{\infty}\prod_{\substack{i=1\\ i\neq m}}^{j} p_i = (1-p_{m+1})\sum_{j=m+1}^{\infty}\prod_{\substack{i=1\\ i\neq m+1}}^{j} p_i \quad \Longrightarrow$$

$$(1-p_m)\left(1+\sum_{j=m+1}^{M}\prod_{i=m+1}^{j} p_i\right) = (1-p_{m+1})\,\frac{p_m}{p_{m+1}}\sum_{j=m+1}^{M}\prod_{i=m+1}^{j} p_i \quad \Longrightarrow$$

$$p_m = \frac{p_{m+1}\left(1+\sum_{j=m+1}^{M}\prod_{i=m+1}^{j} p_i\right)}{p_{m+1}+\sum_{j=m+1}^{M}\prod_{i=m+1}^{j} p_i} = \frac{1+\sum_{j=m+1}^{M}\prod_{i=m+1}^{j} p_i}{2+\sum_{j=m+2}^{M}\prod_{i=m+2}^{j} p_i}. \quad (15)$$

Having shown that M and $0 < p_M \leq \frac{1}{2}$ uniquely determine a vector that satisfies conditions (13)–(14), we argue it to be more convenient to define a function $\vec{\boldsymbol{p}} : \mathbb{R}^+ \longmapsto \mathbb{R}^\infty$, such that, for any $t > 0$, $\vec{\boldsymbol{p}}(t)$ is the vector obtained through formula (15) for $M = \lceil t \rceil^\dagger$ and $p_M = \frac{1}{2}(t + 1 - \lceil t \rceil)$. This way, any vector that satisfies conditions (13)–(14) (and is therefore an "eligible candidate" to be a solution to the optimization problem for some value of E) corresponds to a positive real number, and the space of all such vectors is in one-to-one correspondence with the positive real axis. We also define $\boldsymbol{p}_i : \mathbb{R}^+ \longmapsto \mathbb{R}$ to be the i-th component of $\vec{\boldsymbol{p}}$.

Lemma 9. *The function $\vec{\boldsymbol{p}}$ is continuous.*

Proof. The continuity of $\vec{\boldsymbol{p}}$ at non-integer points (continuity in p_M only, with M unchanged) is obvious from formula (15), which shows p_m, for any $1 \leq m \leq M-1$, 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 nonzero element of $\vec{\boldsymbol{p}}(t)$; that is, $M = K$ and $\boldsymbol{p}_K(t) = \frac{1}{2}(t + 1 - K)$. Therefore,

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

For $t \to K^+$, $\boldsymbol{p}_K(t)$ is the penultimate nonzero element; that is, $M = K+1$, $p_M = \frac{1}{2}(t-K)$, and $\boldsymbol{p}_K(t)$ can be computed from (15):

$$\lim_{t\to K^+}\boldsymbol{p}_K(t) = \lim_{t\to K^+}\frac{1+\sum_{j=K+1}^{K+1}\prod_{i=K+1}^{j} p_i}{2+\sum_{j=K+2}^{K+1}\prod_{i=K+2}^{j} p_i} = \lim_{t\to K^+}\frac{1+\frac{1}{2}(t-K)}{2+0} = \frac{1}{2}.$$

Hence, the component $\boldsymbol{p}_K(t)$ is continuous at $t = K$. From here, the continuity of components $\boldsymbol{p}_1(t), \ldots, \boldsymbol{p}_{K-1}(t)$ follows from their being continuous in $\boldsymbol{p}_K$, according to formula (15) (again, by applying backward induction from $m = K-1$ to $m = 1$). □

†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.

Figure 6 shows a plot of the function $\boldsymbol{p}_1(t)$. The plot is divided into three separate ranges 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 $\boldsymbol{p}_m(t)$.

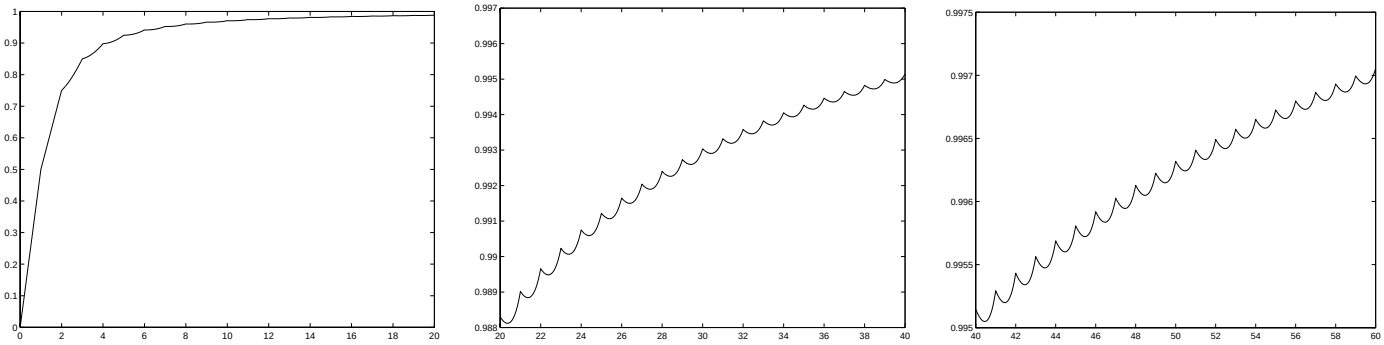

Fig. 6. Plot of the function $\boldsymbol{p}_1(t)$.

Figure 7 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 (10). For convenience, the y-axis is logarithmic.

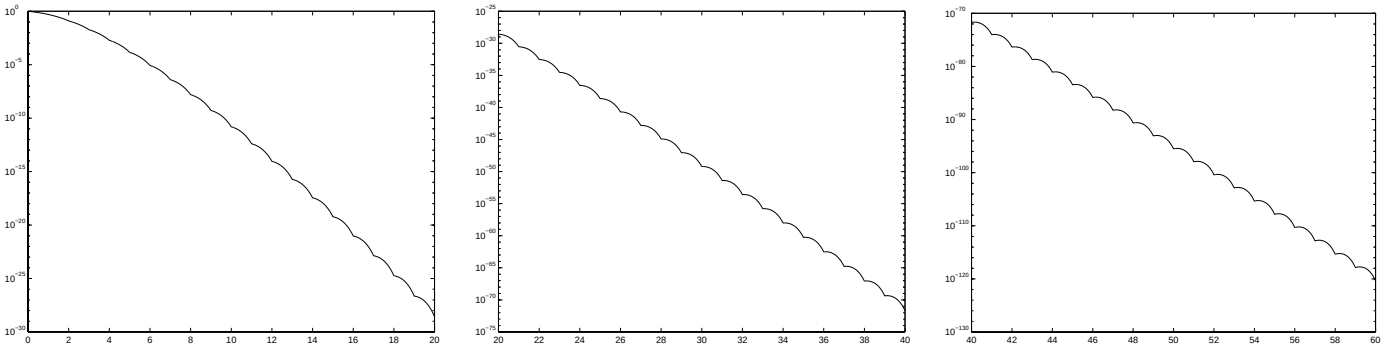

Fig. 7. Plot of the function $L^s(t) \triangleq \prod_i (1-\boldsymbol{p}_i(t))$.

Finally, Figure 8 shows a plot of the score attained by $\vec{\boldsymbol{p}}(t)$.

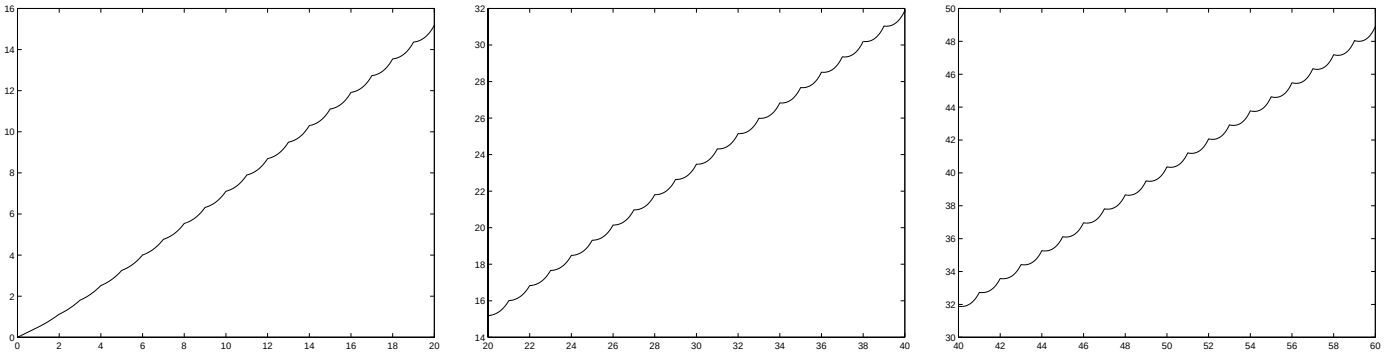

Fig. 8. Plot of $\phi(t) = \sum_{j=1}^{\infty}\prod_{i=1}^{j}\boldsymbol{p}_i(t)$.

Conceptually, the solution of the optimization problem for a given value of L^s is obtained by locating the set of points $\{t\,|\prod_i(1-\boldsymbol{p}_i(t)) = L^s\}$ (e.g., from Figure 7), and selecting the point that attains the maximal value of $\sum_j\prod_{i=1}^{j}\boldsymbol{p}_i(t)$ (e.g., from Figure 8). Note that the set contains more than one point for $L^s \lessapprox 1.586\cdot 10^{-43}$ (the function of Figure 7 ceases to be strictly decreasing after $t = 27$). An algorithm to compute the solution could begin by evaluating $L^s(t)$

at integer points, exploiting the function's continuity to find an initial search range, and then perform 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. The implementation details of such an algorithm are tedious yet entirely straightforward, and we do not consider them here any further.

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 figures 7 and 8 are very nearly "mirror images" of each other (and become ever more so as t gets larger). To illustrate this, Figure 9 shows a parametric plot of $\phi(t)$ versus $E^s(t)$, for t that varies continuously in the same range as above. It can be seen that the plot 'almost' defines a function; it takes a great deal of "zooming in" to notice that the plot actually zig-zags back and forth. Obviously, Figure 9 contains the 'true' $\Phi\left(L^s\right)$ as well; that is, $\Phi\left(L^s\right)$ is described by the topmost point of the plot for any value of L^s. Strictly speaking, therefore, the function $\Phi\left(L^s\right)$ is not continuous; however, its 'jumps' are markedly minuscule.

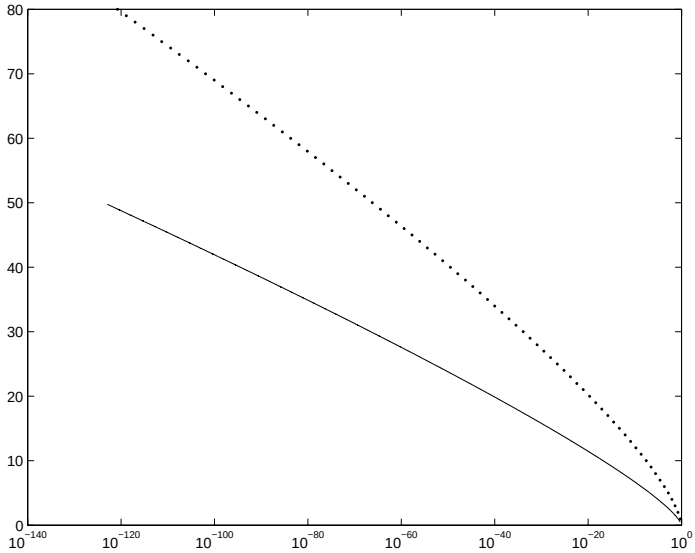

Fig. 9. Parametric plot of $\phi(t)$ vs $L^s(t)$, and comparison to the bound implied by Theorem 1.

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 $L^s \leq \frac{1}{[\Phi(L^s)+1]!}$. This bound is plotted by the dotted line in Figure 9. Thus, it can be seen that the auxiliary function $\Phi\left(L^s\right)$, while not difficult to compute, provides a much tighter bound.

To complete this section, the next lemma provides the asymptotic connection between $\mathrm{E}_L(N)$ and the auxiliary function $\Phi_L(s)$. It states that, in a certain sense, the function $\Phi_L(s)$ provides a close approximation to $\mathrm{E}_L(N)$ for large values of N.

Lemma 10. *For any given value of* L^s, $\lim_{N\to\infty} E_{(L^s)^{1/N}}(N) = \Phi\left(L^s\right)$.

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

Now, denote the maximizing vector of $\Phi\left(L^s\right)$ by $\vec{p^*} = \langle p_1^*, \ldots, p_M^*, 0, 0, \ldots \rangle$, and define the vector $\vec{p_{|N}} \triangleq \langle Y_{\Lambda_N}(p_1^*), \ldots, Y_{\Lambda_N}(p_M^*), 0, 0, \ldots \rangle$, where $\Lambda_N \triangleq (L^s)^{1/N}$. Define also the corresponding vector $\vec{n_{|N}} \triangleq \langle \lfloor \log_{\Lambda_N}(1-p_1^*) \rfloor, \ldots, \lfloor \log_{\Lambda_N}(1-p_M^*) \rfloor, 0, 0, \ldots \rangle$, and denote $N' = \sum \vec{n_{|N}}$. Obviously, $N' \leq \sum_{i=1}^M \log_{\Lambda_N}(1-p_i^*) = \log_{\Lambda_N} \prod_{i=1}^M (1-p_i^*) = \log_{\Lambda_N} L^s = N$.

Now, consider the score of $\vec{p_{|N}}$. It cannot be higher than $\mathrm{E}_{\Lambda_N}(N')$, since $\vec{n_{|N}}$ is just one of the 'eligible' vectors over which $\mathrm{E}_{\Lambda_N}(N')$ is maximized. In light of Lemma 1, it is therefore not higher than $\mathrm{E}_{\Lambda_N}(N)$ as well. Thus, $\mathrm{E}_{\Lambda_N}(N)$ is 'sandwiched' between the scores of $\vec{p_{|N}}$ and $\vec{p^*}$ (the latter, by definition, being simply $\Phi\left(L^s\right)$). However, since

Initialization: Set $N_L = 1; N_H = 4$
Bound estimation:
    While $CTR\,(2N_H) \leq CTR\,(N_H)$
        $N_L \leftarrow 2N_L$, $N_H \leftarrow 2N_H$
Search:
    While $N_L \neq N_H$:
        Set $N_1 \leftarrow N_H - r\,(N_H - N_L)$, $N_2 \leftarrow N_L + r\,(N_H - N_L)$ (rounded to integers)
        If $CTR\,(N_1) > CTR\,(N_2)$:
            Set $N_L \leftarrow N_1$
        Else:
            Set $N_H \leftarrow N_2$

Fig. 10. Fibonacci search algorithm for the optimal N^*; $CTR(N)$ denotes the target expression $\frac{a \cdot T + b \cdot N}{\mathrm{E}_L(N)}$, i.e. the cost/throughput ratio, and $r = \frac{\sqrt{5}-1}{2} \approx 0.618$.

$\Lambda_N \to 1$ as $N \to \infty$, we have $\lim_{N\to\infty} \vec{p_{|N}} = \vec{p^*}$; in light of the continuity of the score expression, this finally implies $\lim_{N\to\infty} \mathrm{E}_{\Lambda_N}(N) = \Phi(L^s)$. □

VI. Approaches for finding the optimal window size

The previous two sections provided a detailed discussion on efficient methods for approximate computation of the function $\mathrm{E}_L(N)$, which we termed the 'inner problem'. To complete the picture, we now recall that this computation is only a part of the bigger task of finding the strategy with the minimal cost/throughput ratio; accordingly, we now turn to discuss the outer part of the problem, namely, finding the optimal value of N (the window size).

The most straightforward approach to finding the optimal N, arguably, is that of a direct search, using any of the methods outlined in the previous two sections as a 'subroutine' for computing $\mathrm{E}_L(N)$. A popular search method is that of the Fibonacci or golden-section search [15]. This method begins with a search interval $[N_L, N_H]$ and then reduces this interval by a factor of r in every iteration, by proceeding to either $[N_L, N_L + r(N_H - N_L)]$ or $[N_H - r(N_H - N_L), N_H]$, according to which of the two intermediate points attains a smaller value of the target expression. The value of r is usually taken to be $\frac{\sqrt{5}-1}{2} \approx 0.618$, due to the property that $r^2 = 1 - r$, which causes one of the intermediate points in each iteration to coincide always with one of the intermediate points in the previous iteration, thereby halving the number of computations of the target expression.

An algorithm for minimizing the cost/throughput ratio that is based on the Fibonacci search method is presented in Figure 10.† It begins by estimating an upper bound for the search by doubling the initial N_H as long as it reduces the target expression, and follows with a Fibonacci search within that bound. It is easy to see that, if the eventual result is N^*, the computational complexity of this algorithm is $(\log N^*)$ times the complexity of a single computation of the target expression, which depends on the method chosen to evaluate $\mathrm{E}_L(N)$. Possible minor modifications to the algorithm, such as increasing the initial bound by a factor other than 2, or starting the search in $[1, N_H]$ rather than $[N_L, N_H]$, do not change this complexity significantly.

The drawback of the direct search method is that the computations of $\mathrm{E}_L(N)$ for different values of N are made independently, and neglect their internal redundancy. For example, suppose that the evaluation of $\mathrm{E}_L(N)$ is performed with the Greedy-A algorithm. For any N, this algorithm starts with a zero vector and performs a series of increments that are, by themselves, independent of N (that is, N only sets the total number of increments). In other words, evaluation of $\mathrm{E}_L(N)$ by the Greedy-A algorithm for any N produces, as a free by-product, the corresponding values of $\mathrm{E}_L(N')$ for all $N' \leq N$. Thus, by the time the Fibonacci algorithm completed the initialization of the upper bound, the cost/throughput ratios are, in fact, readily available for all N up to that bound, and the ensuing search is superfluous.

An alternative method, which exploits the redundancy mentioned above, is shown in Figure 11, and we call it 'Greedy-N'. It works identically to Greedy-A, except that the number of steps is not set in advance; rather, it proceeds to ever-increasing N, computing the cost/throughput ratio on the fly, and stops as soon as the ratio ceases to decrease (i.e. improve).

†Figure 10 omits some details about the way the interval is rounded to integers, which pose no major interest.

Initialization: Set $\vec{n} = \langle 0, 0, \dots \rangle$
Loop:
For every j such that either $j = 1$ or $n_{j-1} > n_j$:
Temporarily set $n_j \leftarrow n_j + 1$
Compute the score of $\vec{n}$
Restore n_j
For the j that got the best score:
Set $n_j \leftarrow n_j + 1$
Compute the cost/throughput ratio for $\vec{n}$
If ratio improved from last iteration, go back to *'Loop'*;
else terminate.

Fig. 11. Algorithm Greedy-N.

Both methods described above would converge to the optimal solution (up to the approximation involved in the evaluation of $\mathrm{E}_L(N)$) if the target expression, i.e. the cost/throughput ratio, had a single local minimum in N. Unfortunately, this is not the actual case. Figure 12 shows a typical plot of the target ratio as a function of N. It can be seen that the function decreases steeply and monotonously at first but quickly becomes quite 'flat', eventually increasing rather slowly amid a somewhat noise-resembling behavior.[‡] This shape is indeed expected in light of the ratio expression being $\Theta\left(\frac{a \cdot T + b \cdot N}{N/\log_{1/L} N}\right)$ (recall Theorem 2): for small N, where the numerator is dominantly $a \cdot T$, the ratio decreases at a rate of $\frac{1}{N/\log_{1/L} N}$, and for larger N, where the numerator is dominantly $b \cdot N$, the ratio increases at a rate of $\log_{1/L} N$, i.e., much more slowly. The 'noise' (i.e. non-monotonocity), which is especially apparent around the minimum point, is due to combinatorial effects which we do not go into further; however, it may cause a potentially large number of 'false' local minima, whereas the Greedy-N algorithm naively stops after finding the first one.

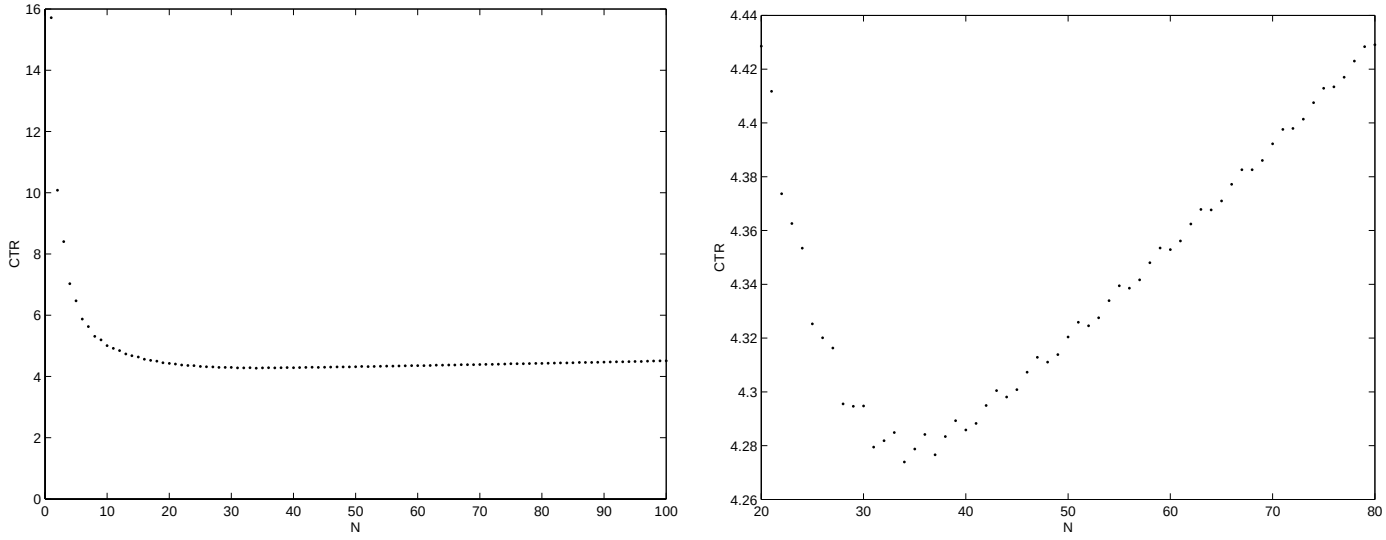

Fig. 12. The cost/throughput ratio $\left(\frac{a \cdot T + b \cdot N}{\mathrm{E}_L(N)}\right)$ as a function of N, for $a = 10$, $T = 1$, $b = 1$, $L = 0.3$. The second graph is simply a 'zoom-in' of the first, for $20 \leq N \leq 80$.

To overcome this problem, the termination condition of Greedy-N can be enhanced so as to stop not at the first local minimum, but only after it is apparent that no further improvement is possible, that is, after the ratio function has reached the steadily increasing region. It turns out, however, that translating this into an exact termination condition is not trivial. We have tested the algorithm with a variety of termination criteria, for all L between 0.001 and 0.999 in

[‡]The plot in Figure 12 was obtained with the exact values of $\mathrm{E}_L(N)$, i.e. those found by an exhaustive search; however, the plot shape is essentially similar if any of the approximate evaluation methods is used instead.

Initialization: Set $\vec{n} = \langle 0, 0, \dots \rangle$, $N \leftarrow 0$, $Best\_CTR \leftarrow \infty$
Loop:
    $N \leftarrow N + 1$
    For every j such that either $j = 1$ or $n_{j-1} > n_j$:
        Temporarily set $n_j \leftarrow n_j + 1$
        Compute the score of $\vec{n}$
        Restore n_j
    For the j that got the best score:
        Set $n_j \leftarrow n_j + 1$
    Set $CTR \leftarrow$ the cost/throughput ratio for $\vec{n}$
    If $CTR < Best\_CTR$
        Set $Best\_CTR \leftarrow CTR$, $N^* \leftarrow N$
    If $N = 2N^*$, terminate; else go back to *Loop*.

Fig. 13. Algorithm Greedy-N with an enhanced termination condition.

increments of 0.001, with $b = 1$, $T = 1$, and $a \in \{1, 2, \dots, 10, 20, \dots, 100, 200, \dots, 1000\}$ (recall that, for a given L, the optimal N^* depends only on $\frac{aT}{b}$). We point out that this range covers all the practically interesting cases: for $\frac{aT}{b} < 1$, the optimal window size rarely gets above 1, while for $\frac{aT}{b} = 1000$ the typical window sizes already reach several thousands, and any higher values are not likely to be implemented in practice in a way that keeps the window transmission time a negligible part of the round-trip delay. It turns out that termination criteria based on fixed numbers (such as "stop if there has been no improvement for 10 iterations") are bound to fail for large enough values of a, while those based on the target value itself (like "stop when the current target value has risen to 5% above the optimum so far") may lead to exponential complexity, due to the logarithmic increase rate of the target expression for large N.

The termination conditions that were found to work best were those that exploited the fact that the 'noise' tends to exhibit a certain periodicity, with a period that is much smaller than N^* itself. For example, in Figure 12, the local minima points $N = \{29, 31, 34, 37, 40, 44, 48, \dots\}$ form a nearly arithmetic sequence, with a difference much smaller than $N^* = 34$; a similar phenomenon was found to occur in all the above-mentioned runs (in a few cases, two separate regions of local minima sequences with different periods were found, yet both still had periods much smaller than the corresponding N^*). A simple termination condition that is based on the above observation is $N = 2N^*$, i.e., the search stops after the number of iterations completed is twice the number in which the optimum was found. Figure 13 describes the algorithm with this condition employed; this algorithm did not fail to find the global minimum even in a single instance. Admittedly, this condition is quite conservative; however, considering that the computation of the optimal strategy is performed off-line and infrequently, and that the best strategy found so far can begin to be employed even before the search is completed, perfecting the termination condition to reduce the computation time by a constant factor at most does not seem to be of major importance.

Finally, Figure 14 plots the optimal cost/throughput ratio as a function of a, for a few select values of the loss probability; note that the horizontal axis is logarithmic. These plots clearly demonstrate the property predicted by Theorem 3, namely, that the optimal ratio value increases logarithmically in a.

We close this section with a conclusive example that demonstrates the performance of the Greedy-N algorithm, as well as other techniques presented earlier.

Example: For the parameter values depicted in figure 12 (namely, $L = 0.3$, $T = 1$, $b = 1$, $a = 10$), algorithm Greedy-N finds an optimal strategy of $\langle 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 1, 1, 0, 0, \dots \rangle$ 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. Incidentally, the greedy search happens to find the optimal strategy in this case; no further improvement can be gained by exhaustive search.

For comparison, the optimal window size with 'classic' sliding windows, found either by direct search or with the help of formula (6), is 5 (i.e., the strategy is $\langle 1, 1, 1, 1, 1, 0, 0, \dots \rangle$ in our terms), with a corresponding cost/throughput ratio 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, Greedy-N finds $N^* = 531$, with

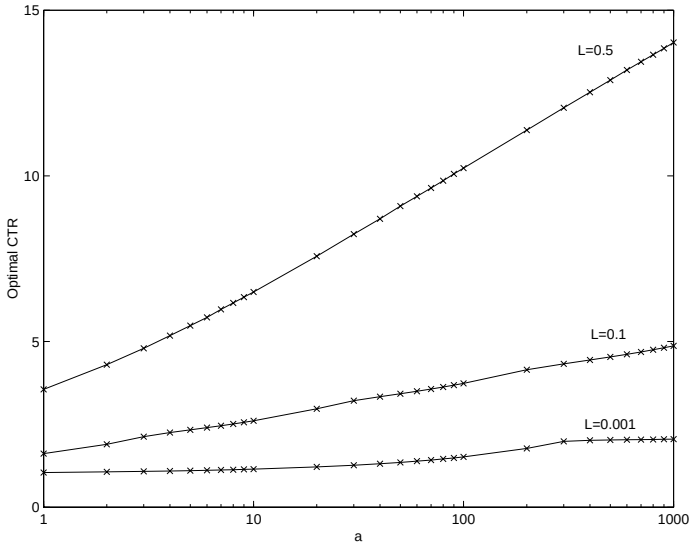

Fig. 14. The optimal cost/throughput ratio as a function of a for several loss probabilities.

the strategy $\langle \underbrace{6,\ldots,6}_{12}, \underbrace{5,\ldots,5}_{66}, \underbrace{4,\ldots,4}_{23}, \underbrace{3,\ldots,3}_{9}, 2,2,2,2,1,1,0,0,\ldots \rangle$. Its score is 97.6449, and the corresponding cost/throughput ratio value is 6.4622. The 'classic' optimal window size here is 10, reaching 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 increase 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.

It is interesting to note that, this time, the greedy search did not find the optimal strategy. While exhaustive search for the inner problem is unfeasible for $N = 531$, it can be verified that both the Greedy-R and Greedy-L algorithms attain higher scores; indeed, Greedy-R for $L = 0.3$ and $N = 531$ obtains the strategy $\langle \underbrace{6,\ldots,6}_{10}, \underbrace{5,\ldots,5}_{68}, \underbrace{4,\ldots,4}_{24}, \underbrace{3,\ldots,3}_{9}, 2,2,2,1,1,0,0,\ldots \rangle$, with a slightly higher score of 97.651. The continuous relaxation method in this case results in an upper bound of 98.6863 for the score; therefore, the strategy obtained by greedy search, with a score of 97.6449, cannot be off by more than about 1% from the 'truly' optimal one. □

VII. Conclusion

We have investigated the properties of optimal sliding-window strategies in network connections where the packet transmission time is negligible compared to the round-trip delay. We assumed a cost per unit of time and per packet transmission associated with the connection, and defined the optimal strategy as one that minimizes the cost/throughput ratio over time. We derived several important bounds on the performance of the optimal strategy: specifically, we showed that, for a window size of N, the number of successful in-order packets is $\Theta\left(\frac{N}{\log N}\right)$, and used this result to further show that the cost/throughput ratio increases logarithmically in the per-time cost. We then studied several practical solution algorithms. We found that, for a given window size, a strategy that is at most only a few percent worse than the optimal can be computed by a very efficient 'greedy' algorithm, and extended this algorithm to find the optimal window size as well, without any further loss in performance. It was demonstrated that such strategies attain a significantly smaller cost/throughput than 'classic' sliding windows, where a packet is retransmitted only after a timeout or negative acknowledgment; in general, the relative improvement is greater as the cost of waiting a round-trip time gets higher compared to the marginal cost of transmitting a packet.

The analysis was based on the assumption that the receiver accepts packets only in-order, and that the sender strategy is limited to simple retransmissions. However, the methodology can be extended to cover other cases as well. For example, as mentioned in the Introduction, if the sender is capable of forward error-correction (FEC) coding, it may employ a coding strategy that uses different codes for different packets. While the computation of the optimal strategy is

somewhat more complex (it involves an extra parameter, namely the size of the coding block), it is based on essentially the same ideas underlying in the basis of this paper's results, except that the strategy score expression is based on the coding redundancy rather than number of retransmissions. In particular, it is to be expected that the optimal strategy uses a higher-redundancy coding for the first packets in every window than for later ones.

Similarly, the technique established in this paper can be extended to the case that the receiver has a buffer capable of accepting a certain number of packets out of order, and reports the state of its buffer, e.g., by using negative acknowledgments in addition to simply reporting the index of the next-expected packet (this is known as a receiver capable of *selective repeat*). Instead of a single vector of non-negative numbers 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 packet retranmissions for each state. Still, the computation of these vectors involves the optimization of essentially the same score expression. Furthermore, the strategy remains invariant to the next-expected packet index and thus can be considered to be of the sliding-windows type.

The strategies discussed in this paper were assumed to wait for the acknowledgments of all packets from a window before beginning the transmission of the next one. We explained, while presenting the general model, why the optimal strategy must fall into this category 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 within our framework, merely by replacing the packet transmission cost coefficient b with $b + a \cdot t_x$, i.e. including the extra per-packet cost due to the time it takes to transmit it; the rest of the strategy computation can then remain unaltered. This produces a strategy that is adequate as long as the round-trip delay is significantly longer than the packet transmission time. However, if a packet transmission lasts for a significant fraction of the round-trip time, it may be worthwhile to begin the transmission of a window even before all acknowledgments from the previous window have been gathered, i.e., with only a partial information on the successes and losses in the previous window. Then, in general, a strategy is no longer described by a vector that is applied at every multiple of the round-trip time, but, rather, by a rule that is applied after every packet transmission and specifies which packet is most worthwhile to be transmitted next (or none at all), according to the information available from acknowledgments up to that moment. The investigation of optimal strategies and their properties in this framework, as well as further development of the other extensions outlined above, form a subject for future work.

REFERENCES

[1] A.S. Tanenbaum. *Computer Networks*. Prentice-Hall, Upper Saddle River, NJ, 3rd edition, 1996.

[2] ISO/IEC standard 13239:2000 (HDLC procedures), February 2002.

[3] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. RFC 2018: TCP selective acknowledgment options, October 1996. Internet RFC.

[4] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[5] L. Libman and A. Orda. Optimal timeout and retransmission strategies for accessing network resources. *IEEE/ACM Transactions on Networking*. To appear (scheduled August 2002); may be obtained from http://tiger.technion.ac.il/~libman/papers/ToN02.ps.gz.

[6] M. Allman, S. Dawkins, D. Glover, J. Griner, D. Tran, T. Henderson, J. Heidemann, J. Touch, H. Kruse, S. Ostermann, K. Scott, and J. Semke. RFC 2760: Ongoing TCP research related to satellites, February 2000. Internet RFC.

[7] M. Allman, C. Hayes, H. Kruse, and S. Ostermann. TCP performance over satellite links. In *Proc. 5th International Conference on Telecommunication Systems*, pages 456–469, Nashville, TN, March 1997.

[8] C. Barakat, N. Chaher, W. Dabbous, and E. Altman. Improving TCP performance over geostationary satellite links. In *Proc. IEEE Globecom*, December 1999.

[9] E. Altman, K. Avrachenkov, and C. Barakat. TCP network calculus: The case of large delay-bandwidth product. In *Proc. IEEE Infocom*, New York, NY, June 2002.

[10] D. Katabi, M. Handley, and C. Rohrs. Internet congestion control for future high bandwidth-delay product environments. In *Proc. ACM SIGCOMM*, Pittsburgh, PA, August 2002.

[11] K. Park and 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)*, pages 196–205, Lafayette, LA, October 1998.

[12] C. Barakat and E. Altman. Bandwidth tradeoff between TCP and link-level FEC. In *Proc. IEEE International Conference on Networking*, pages 97–107, Colmar, France, July 2001.

[13] B. Liu, D.L. Goeckel, and D. Towsley. TCP-cognizant adaptive forward error correction in wireless networks. In *Proc. IEEE Infocom*, New York, NY, June 2002.

[14] R.M. Corless, G.H. Gonnet, D.E.G. Hare, D.J. Jeffrey, and D.E. Knuth. On the Lambert W function. *Advances in Computational Mathematics*, 5:329–359, 1996.

[15] D.G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, Reading, MA, 1984.