A Modular Analysis of Network Transmission Protocols

Micah Adler [*] [†] Yair Bartal[*†] John W. Byers[*†]

Michael Luby [‡] Danny Raz[*†]

TR-97-001

April 1997

Abstract

We propose a new model for the analysis of data transmission protocols in lossy communication networks. The overall goal of a data transmission protocol is to successfully transmit a message from the sender to the receiver. We study the performance of protocols in an adversarial setting where the loss pattern and latencies of packets are determined by an adversary.

We advocate the modular decomposition of data transmission protocols into a *time scheduling policy*, which determines *when* packets are to be sent, and a *data selection policy*, which determines *what* data is to be placed in each sent packet.

We concentrate on the data selection policy and require that the protocol will achieve high bandwidth utilization in transmitting any prefix of the transmitted message. The simple and universal data selection policy we introduce is provably close to optimal in the following sense: For *any* time scheduling policy and *any* network behavior, in the worst case prefix measure our data selection policy performs as well as any other data selection policy up to a constant additive term.

Our explicit modular decomposition of a transmission protocol into two policies should be contrasted with existing network protocols such as TCP/IP. Our result

[*]The International Computer Science Institute in Berkeley, California, and the University of California, Berkeley.

[†]Supported in part by NSF operating grants CCR-9304722 and NCR-9416101.

[‡]DEC/SRC, 130 Lytton Avenue, Palo Alto, California 94301. A large portion of this research was done while still at the International Computer Science Institute in Berkeley, California. Research supported in part by National Science Foundation operating grants CCR-9304722 and NCR-9416101, and United States-Israel Binational Science Foundation grant No. 92-00226.

shows that the performance of the overall transmission protocol would not degrade in performance (and could improve dramatically) if it used our universal data selection policy in place of its own. We therefore reduce the problem of designing a data transmission protocol to the task of designing a time scheduling policy.

1 Introduction

TCP/IP traffic currently accounts for over 90% of the throughput on the Internet. The reason for TCP's predominance stems from a the fact that TCP is a robust protocol which has withstood the test of time in achieving both high performance and fairness across connections. Now, however, TCP's resilience is being tested as new environments, such as wireless networks, and new protocols, such as real-time protocols that use UDP, become more prevalent. The natural question then, is whether TCP is likely to continue to succeed, as the environment in which it is used changes in unpredictable ways?

There is a clear difference in the way networking practitioners and the theory research community tend to approach such questions. Theoretical research often introduces restrictions in order to be able to model the problem and to design and analyze suitable modifications to the protocol under this model. These restrictions typically include assumptions about the input distribution on the data to be transmitted, the topology of the network, and the capacity of the links, and other properties of the underlying network. Indeed, under certain assumptions, protocols have been designed and analyzed to have nearly optimal throughput, small latencies, etc.

However, in practice, working networks have become extremely complex and difficult to model, since these networks can have arbitrary topologies, variable edge capacities and can experience bursty periods of abnormally high traffic [LTWW 95, YKT 96]. Furthermore, packets traversing these networks can be dropped by intermediate routers or can experience unpredictable and widely varying round-trip times [Pax 96]. Therefore, drawing conclusions about TCP based on a particular model or interpretation of the network is sharply limited by those assumptions introduced to derive the model.

This considerable and as of today, unquantifiable, discrepancy between conditions under which existing networks operate and conditions under which protocols have been proven to work well motivates our work. One of our main goals is to abstract away these details of the network, so as to avoid the contentious problem of settling on a particular model for the network behavior and data distribution. Our abstraction models the network behavior as arbitrary, i.e., it may drop any sequence of packets and may vary the latency on packet transmissions arbitrarily. This is somewhat similar to the approach of recent works such as the adversarial models of [BKR+ 95] and [AAF+ 96]. In these works the performance of protocols on packet routing networks have been provably analyzed, even when the distribution of packet arrivals and routes is arbitrary, subject to the requirement that edge capacity constraints are not violated.

We advocate and justify in this paper a modular approach to protocol design. This approach has obvious advantages: It encourages the design of simple and easy to understand protocols, while explicitly discouraging consideration of complex interactions between the different parts. This notion is not new – the architects of NETBLT [CLZ 87] and the proponents of forward acknowledgment in TCP/IP [MM 96] both advocate explicit decoupling of congestion control algorithms from other parts of the protocol. As in [MM 96], we propose decomposing a transmission protocol into two parts: A *time scheduling policy* that determines *when* the packets are to be sent, and a *data selection policy* that determines *what* data is to be placed in each sent packet. The simple and universal data selection policy we introduce is provably close to optimal in the following sense: For *any* time scheduling policy and *any* network behavior, our data selection policy provably performs almost as well as *any other* data selection policy when used in conjunction with the given time scheduling policy and network behavior. This is true even when the data selection policy we compare against is specifically designed to work with the given time scheduling policy and network behavior.

The overall goal of a data transmission protocol is to successfully transmit a message from the sender to the receiver. We choose to measure the performance of such a data transmission protocol quite stringently: At each point in time, our measure is the length of the longest message prefix acknowledged as received up to that point in time.

This choice is motivated by applications such as video and audio broadcasting, where only the intact received prefix of the broadcast can be played back in full fidelity. Another justification for our measure is the strategy used by commercial Internet browsers to display pages with graphical content downloaded over the Internet: the browser incrementally updates the image as quickly as possible as the stream arrives. In this example, there may be only a small number of breakpoints in the stream where the image can be updated, and thus it may be argued that a more realistic measure would only compare the progress of policies at these breakpoints. On the other hand, since our data selection policy performs well at any possible breakpoint, it clearly also performs well when comparisons are made only at certain breakpoints. Thus, using our stringent performance measure ensures that our data selection policy works well independent of where the breakpoints are within the stream.

We analyze this performance measure for our data selection policy in the spirit of competitive analysis of on-line algorithms introduced in [ST 85]. The analysis measures the worst-case performance of our data selection policy when compared with the performance of an optimal data selection policy, where comparisons are made at all points in time for each possible time scheduling policy and each specification of the behavior of the network. We use the *competitive ratio*, i.e., the ratio between the profit of the on-line algorithm and the profit of the optimal algorithm (up to a constant additive term), as our performance measure. To achieve a more refined analysis in the randomized case, where the ratio is 1, we focus on the performance measure known as the *minimax regret*; i.e., we bound the worst-case difference between the optimal profit and the on-line profit.

The design of a universal time scheduling policy that is optimal for any given network behavior is beyond the scope of this paper. At this point, this is not even a well-defined question, because the performance of a time scheduling policy for one user depends on the interactions of an arbitrary set of users transmitting over an arbitrary network.

The remainder of the paper is organized as follows. In § 2, we define the model, with particular emphasis on our design decisions, and the measure of performance for the data selection policy. In § 3 we present a deterministic data selection policy and analyze its competitive ratio along with a matching lower bound. In § 4 we present a randomized algorithm and prove nearly matching upper and lower bounds on the minimax regret of randomized data selection policies. Finally, in § 5, we summarize our results and outline future research directions.

2 The Specification of the Model

The specification of the model which we develop is *sender-centric*, meaning that the protocol is run from the sender's perspective, based only on *events* which have already occurred at the sender. The sender receives feedback about the progress of its transmissions based on receiver-driven acknowledgments in the style of TCP, i.e. each packet arrival at the receiver triggers an acknowledgment. However, both packets and acknowledgments may be lost, so the algorithms which we develop must address this possibility. Assuming that packets, acknowledgments and each of the message words can be uniquely identified, the following is a complete list of events that may occur at the sender at a time-step:

- Transmit packet i containing data w_i.
- Receive acknowledgment ACK (i) (or NACK(i)) acknowledging (or NACKing) packet i.
- Terminate transmission.

Our model supports negative acknowledgments (NACKs), when the receiver can infer that a packet has been lost. It can also support the use of acknowledgments which acknowledge multiple packets, similar to the use of selective acknowledgment in TCP (see for example [FF 96, JB 88]), simply by unpacking such a multiple acknowledgment into a sequence of separate acknowledgments. Although the model implicitly permits the receiver to choose *what* to acknowledge at each time-step, our algorithms will use a rather simple mechanism for doing so.

We associate the following quantities with packet i:

- s_i, the packet transmission time
- a_i, the time at which the first (possibly negative) acknowledgment for packet i arrives at the sender
- b_i, a boolean set to 1 iff packet i was positively acknowledged
- $a_i - s_i$, the round-trip time for packet i
- w_i, the data, or payload, contained in packet i

The time-scheduling policy determines when packets are to be transmitted, which in our model corresponds exactly to the choice of the times s_i, based on events which occur prior to this time. The network behavior is captured by the round-trip times, which reflects network latencies, and by the values of the b_i, which specify the pattern of packet loss. We assume that the network does not base its decisions on the contents of the packets, eliminating dependencies between the data selection policy and the network behavior. Finally, the data selection policy, which determines what information should be placed into each packet, is responsible for determining the w_i.

2.1 Transcripts

Our objective is to study the worst-case performance of data transmission algorithms, with any choice of network behavior and any choice of time-scheduling policy. Together, the network behavior and the time-scheduling policy jointly determine the s_i, a_i, and b_i for every packet i, and since we make no restrictions on the dependencies between the network behavior and the time-scheduling policy, our data selection algorithm must perform with any specification of these 4-tuples, which we refer to as a *transcript*. We define the *backlog* of a transcript θ at time t, $B_t(\theta)$, to be the number of transmissions sent prior to time t which have not yet been acknowledged, and we let $B_\theta = \max_t B_t(\theta)$. When the transcript is understood, we frequently use the shorthand B_t to specify the backlog at some time t.

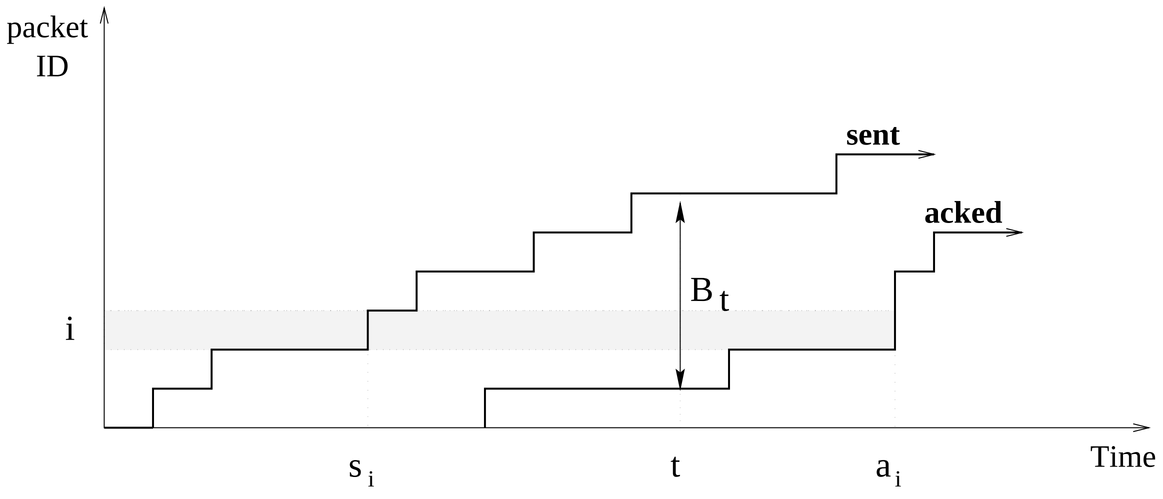

Figure 1: A sample transcript

In Figure 2.1, we provide a pictorial representation of a transcript. As time elapses along the horizontal axis, we separately plot the transmission and acknowledgment times of the packets. Note that in this picture, we do not specify whether the acknowledgment was positive or negative. In this specific transcript the acknowledgments are in-order, and therefore the vertical distance between the two plots at any time t is exactly B_t, the backlog at time t. The backlog may grow (or shrink) due to network factors such as packet loss, acknowledgment loss, and increased network latency, or because of a time-scheduling policy decision resulting in a faster transmission rate. The larger the backlog, the greater the challenge in choosing a data selection policy. For example, if the backlog is always fixed at one, then the trivial policy of selecting the smallest unsent message word is optimal. In fact, in the special case in which the transcript maintains a fixed backlog, a fairly simple algorithm achieves a slightly better result than the more general one we describe.

2.2 Quantifying the Performance of a Data Selection Policy

To describe our performance measure, we first need some more definitions: The *available bandwidth* through time t on a transcript θ is $P^*(t, \theta) = \sum_i b_i$, taken over i such that $a_i \leq t$, i.e. $P^*(t, \theta)$ is the number of distinct transmissions which have been positively acknowledged by time t. The *maximum prefix* of the message with data selection algorithm A and transcript θ at time t, denoted $P^A(t,\theta)$, is the largest integer w such that packets containing message words 1 through w have all been positively acknowledged prior to t.

Intuitively, a successful data transmission algorithm is one which makes effective use of the available bandwidth. On the other hand, we also want to ensure that the buffer sizes required at the receivers are not too large. Therefore, there is a tradeoff involved in sending redundant information; it limits the effectiveness of the bandwidth utilization, but can help keep buffer sizes manageable. The measure we have chosen reflects this intuition. Furthermore, as we described earlier, applications such as Web browsers and lossless real-time multimedia broadcasts can only operate on a continuous prefix of the message.

Now, given any transcript θ, we can define an optimal, omniscient data selection policy, OPT, whose maximum prefix at time t is $P^*(t,\theta)$, i.e. it always manages to use all of the available bandwidth in constructing the prefix. Algorithm OPT achieves this goal simply by transmitting packet j at the time of jth successful send event. This off-line optimum forms the baseline for our comparison. For an algorithm A, let $R^A(t,\theta) = P^*(t,\theta)/P^A(t,\theta)$ denote the ratio, and $\Delta^A(t,\theta) = P^*(t,\theta) - P^A(t,\theta)$ denote the difference between the prefix obtained by A and the prefix obtained by the optimal data selection policy.

2.3 Discussion and Statement of the Main Results

First, we will present a deterministic data selection policy that achieves a worst-case ratio that tends to one as the message length m goes to infinity. We complement this result with a proof that no deterministic strategy can achieve a ratio of 1, and therefore cannot achieve a bounded worst-case difference. These results are stated as the following theorems:

Theorem 1 *There exists a deterministic algorithm A, such that for all transcripts θ and for all times t,*

$$R^A(t,\theta) = 1 + \ O\left(\frac{(B_\theta \log B_\theta)^{3/2}}{\sqrt{P^*(t,\theta)}}\right).$$

Theorem 2 *For any deterministic algorithm A, there exists a transcript θ and a time t, for which*

$$R^A(t,\theta) = 1 + \Omega\left(\frac{1}{\sqrt{P^*(t,\theta)}}\right).$$

For randomized data selection policies, we can do better, but only on transcripts in which successfully transmitted packets arrive in order. This natural assumption is further supported by recent Internet measurements by Paxson [Pax 96] and others which indicate that out-of-order packet arrivals currently occur very rarely. Under this assumption, we can focus on bounding the minimax regret, or competitive difference. In this case, the backlog of the transcript, B_θ, turns out to be of fundamental importance, as it captures an information gap between our algorithm and OPT, measuring the maximum number of transmissions whose fate is unknown to our algorithm at any time. Our results for randomized policies are summarized by the following theorems:

Theorem 3 *There exists a randomized algorithm A, such that for all transcripts θ and for all times t, the following holds with high probability:*

$$\Delta^A(t,\theta) = \tilde{O}(B_\theta \log^2 B_\theta).$$

Theorem 4 *For any randomized algorithm A, and for any sufficiently large B, there exists a transcript θ with $B_\theta = B$ and a time t_A for which*

$$\Delta^A(t_A,\theta) = \Omega(B \log B), \text{ with constant probability.}$$

3 An Efficient Deterministic Data Selection Policy

In this section we focus on deterministic data selection policies. In this case the transcript uniquely determines the content of the packets. Simple algorithms such as sending the smallest unacknowledged message word, or the fully redundant algorithm that keeps on sending a message word until it is positively acknowledged, have poor competitive ratios. We present a simple algorithm and prove that it has a competitive ratio that tends to one as the message size goes to infinity. We then prove that this is the best we can hope for, since the competitive difference for deterministic algorithms cannot be bounded by any function of B_θ.

3.1 A Simple Deterministic Algorithm

Our deterministic algorithm for the problem runs in a sequence of rounds. In round i, it succeeds in transmitting the next k_i message words, where k_i is a parameter to be specified later. Let l_i be the index of the smallest word sent in round i. We say that a word of the message is *completed* once the sender has received a positive acknowledgment for it. For a word which has not been completed, we count the number of packets in the backlog which contain that word as the payload. At each send event in round i, the algorithm sends the uncompleted word in $[l_i, l_i + k_i - 1]$ with the minimum such value, breaking ties arbitrarily.

In each round, the algorithm extends its prefix by the amount k_i. We now prove that the optimal algorithm extends its prefix in this round by at most $k_i + B_\theta \log B_\theta$.

Lemma 5 *Any word which is completed while more than B_θ/j words in the interval are not yet completed is transmitted successfully at most j times.*

Proof: Fix some j, and consider some word w which is first positively acknowledged (and therefore completed) at time a from a transmission at time t. Assume further that at least B_θ/j words have not been completed by time a. We want to show that w could have been retransmitted at most j times after time t but before time a. Indeed, if w was transmitted $j+1$ times then just after the time of the $j+1$st transmission, there must be at least $j \times (B_\theta/j) + 1 > B_\theta$ packets in the backlog, which is a contradiction. ■

Using the lemma above, we have that all but the last B_θ words are transmitted successfully exactly once. Then of the last B_θ words, half of them are transmitted at most twice, a quarter of them are transmitted at most four times, etc., so that in total, the last B_θ words are transmitted successfully at most $B_\theta(\log B_\theta + 1)$ times. Therefore, the total number of successful transmission slots in round i is at most $k_i + B_\theta \log B_\theta$, which bounds the increase in prefix length by the optimal algorithm in round i.

To achieve the best worst-case ratio, we must upper bound the ratio at all times t. Choosing t be a time in phase $i+1$, we must now specify the k_i so as to minimize:

$$R^A(t,\theta) \le \frac{\sum_{j \le i+1}(k_j + B_\theta \log B_\theta)}{\sum_{j \le i} k_j} = 1 + \frac{k_{i+1} + (i+1)B_\theta \log B_\theta}{\sum_{j \le i} k_j}.$$

Choosing $k_i = i$ we get

$$\begin{aligned} R^A(t,\theta) &\le 1 + \frac{(i+1)(B_\theta \log B_\theta + 1)}{i(i+1)/2} \\ &= 1 + \frac{2(B_\theta \log B_\theta + 1)}{i}. \end{aligned}$$

Now, by using $P^*(t,\theta) \le \frac{(i+1)(i+2)}{2} + (i+1)B_\theta \log B_\theta$ we can bound the competitive ratio, and prove Theorem 1:

$$R^A(t,\theta) \le 1 + \frac{4(B_\theta \log B_\theta)^{3/2}}{\sqrt{P^*(t,\theta)}}.$$

3.2 The Deterministic Lower Bound

If the algorithm uses a deterministic strategy to place the data into the packets, then the following adversarial strategy guarantees that $\Delta(t,\theta)$ grows without bound. Repeatedly allow the algorithm to transmit two packets into the system – if they contain identical words, admit both copies; otherwise admit only the packet containing the word with larger index (we henceforth refer to it as the larger packet). Clearly, the backlog of the transcript resulting from this adversarial strategy with any time-scheduling policy is at most two. The following claim directly implies Theorem 2:

Claim 6 *For any deterministic data selection policy A, and any transcript θ resulting from the adversarial strategy described above, there exists a time t in which $R^A(t,\theta) \ge 1 + \left(\frac{1}{2\sqrt{P^*(t,\theta)}}\right)$.*

Proof: From the definitions, $R^A(t,\theta) = 1 + \frac{\Delta^A(t,\theta)}{P^A(t,\theta)} \ge 1 + \frac{\Delta^A(t,\theta)}{P^*(t,\theta)}$. If we then fix τ so that $P^*(\tau,\theta) = 2n(n+1)$, it is enough to show that $\Delta^A(t,\theta) \ge n$ for some time $t \le \tau$. Assume that this claim is not true and $\Delta^A(t,\theta) < n$ for all times t. Then consider an interval of time $[u,v]$ in which the algorithm transmits $2n$ packets. By the adversarial strategy, either the smallest of those packets does not get through, or two copies of the smallest packet are transmitted and both are admitted. If the smallest packet is not admitted, $\Delta^A(v,\theta) = n$, giving a contradiction. So a duplicate packet must be transmitted in every sequence of $2n$ packets, and hence, after admitting $2n \times (n+1)$ packets, $\Delta^A(t,\theta) \ge n$; giving a contradiction and proving the result. ■

4 An Almost Optimal Randomized Data Selection Policy

We next want to develop an even better data selection policy given a transcript θ in which successfully transmitted packets arrive in order. Since our lower bound for deterministic data selection policies still applies for this case, we use randomization to achieve better performance. Our goal is to define a randomized policy A which for all times t and transcripts θ, has a bounded value of $\Delta^A(t,\theta)$ with high probability.

4.1 Preliminaries

To precisely define such an algorithm, we must specify how it chooses to place data in each of the packets and in each of the acknowledgments. First, we require that in a packet sent at time t the sender must transmit the size of the current backlog, B_t, in a field of the header. Then, when a packet sent at time t arrives at the receiver, we require that it acknowledge *all* packets which were in the backlog at time t. This is made possible by the fact that successfully transmitted packets are guaranteed to arrive in order, and we incur a per acknowledgment overhead of at most B_θ bits. This acknowledgment style is similar to the proposal to add selective acknowledgment to TCP (see for example [JB 88] [FF 96]), designed to improve TCP performance over lossy networks. Furthermore, this technique handles the potentially tricky problem of dealing with lost acknowledgments by guaranteeing that our algorithm has the following property on any transcript:

Claim 7

$$\forall i,j : s_i < s_j \textit{ implies } a_i \leq a_j$$

Proof: Omitted. ■

4.2 The Randomized Algorithm

To specify how our algorithm places message words into packets, we use the following terminology. Recall that a word of the message is *completed* once the sender has received a positive acknowledgment for it. We say that it is *locked* for the intervals of time during which a packet containing the word is in transit (sent but not yet acknowledged). Otherwise, the word is *available*. Note that the size of the backlog at time t, B_t, is exactly equivalent to the number of locked words.

We next develop a randomized data selection policy with competitive ratio 1, and which satisfies the conditions of Theorem 3. Again, this policy requires the assumption that those packets which are not lost arrive at the receiver in order. A formal description of Algorithm A is given in Figure 2. It can be summarized by the following simple rule: At the time t of any send event, it sends one of the $\mathcal{S}(B_t)$ smallest available words, chosen uniformly at random. We refer to this set of words as V_t, the set of *valid* words at time t, choosing the function $\mathcal{S}(x)$ so that $\sum_i \frac{1}{\mathcal{S}(i)} \leq \frac{1}{2}$, e.g. $\mathcal{S}(x) = \lceil dx \log(x+1) \log\log^{1+\epsilon}(x+1) \rceil + 1$, where d and ϵ are appropriately chosen constants.

We can now precisely state the technical claim from which Theorem 3 immediately follows.

Algorithm A

```
$L = \{\}$;  /* Locked words */
$C = \{\}$;  /* Completed words */
$R = M$;  /* Other words */

While $C \neq M$ do:

  Case 'Send':
    – $V$ = set of $\mathcal{S}(|L|)$ smallest indices of $R$.
    – Choose $e$ at random from $V$ and send($m_e$).
    – $R = R \setminus \{e\}$;   $L = L \cup \{e\}$;

  Case 'ACK($e$)':
    – $C = C \cup \{e\}$;  $L = L \setminus \{e\}$;

  Case 'NACK($e$)':
    – $R = R \cup \{e\}$;  $L = L \setminus \{e\}$;
```

Figure 2: A near optimal data selection policy

Claim 8 *For any transcript θ, any constant c, and any time τ, let $\widehat{B}_\tau = \max_{t \leq \tau} B_t$ and define $\delta = \lceil 4c\mathcal{S}(\widehat{B}_\tau)\ln 2\mathcal{S}(\widehat{B}_\tau)\rceil$. Then,*

$$\Pr[\Delta^A(\tau,\theta) \geq \delta] \leq (2S(\widehat{B}_\tau))^{-2c+2}.$$

Proof: Fix a transcript θ and a time τ. Now define τ_δ be the first time (prior to τ) at which $P^*(\tau_\delta,\theta) = P^*(\tau,\theta) - \delta$, and let $x = P^*(\tau_\delta,\theta)$. By these definitions, the optimal algorithm delivers a prefix through x by time τ_δ; our goal is to show that our algorithm delivers a prefix through x by time τ with high probability. This would imply that $P^A(\tau,\theta) \geq P^*(\tau,\theta) - \delta$, proving the claim.

We define word i as *eligible* at time t if it is either valid or locked at time t, and we denote the set of eligible words $E_t = V_t \bigcup L_t$. By the fact that our algorithm does not use redundancy, some word $y \geq x$ is transmitted by time τ_δ. At the instant this word y is transmitted, there are at most $\widehat{B}_\tau$ locked words and at most $\mathcal{S}(\widehat{B}_\tau)$ valid words. Therefore, if we let Ψ denote the set of eligible words smaller than x at this same instant, we have that $|\Psi| \leq \widehat{B}_\tau + \mathcal{S}(\widehat{B}_\tau) \leq 2\mathcal{S}(\widehat{B}_\tau)$. We now want to prove that all elements of Ψ are delivered by time τ with high probability, which implies that an intact prefix through x is delivered by time τ. The first step is to show a lower bound on the probability that an eligible member of this set is transmitted in a given time step in the interval $[\tau_\delta, \tau]$.

Lemma 9 *Let t be the time of a successful* **'send'** *event between τ_δ and τ, and let y be an arbitrary element of Ψ.*

$$\Pr[\ y \textit{ sent at } t \mid y \textit{ is eligible at } t] \geq \frac{1}{2\mathcal{S}(B_t)}.$$

Proof: Fix a time t satisfying the conditions of the lemma and an element $y \in \Psi$ that is eligible at t. If y is unlocked at t, it is sent with probability $\frac{1}{\mathcal{S}(B_t)}$. Let $t_1, t_2, \ldots t_{B_t}$ be the times of the B_t most recent (unacknowledged) transmissions prior to t. The probability that y is locked at t is bounded by the sum of the probabilities that y was sent in one of those B_t most recent transmissions. Since the probability of y being chosen at t_i is $\frac{1}{\mathcal{S}(B_{t_i})}$, we have the following:

$$\Pr[y \text{ locked at } t] \leq \sum_{i=1}^{B_t} \frac{1}{\mathcal{S}(B_{t_i})}.$$

For all i, the transmission at time t_i is not acknowledged until after time t, and hence the backlog at t_i is at least i. Since $\mathcal{S}$ is monotone, $\frac{1}{\mathcal{S}(B_{t_i})} \leq \frac{1}{\mathcal{S}(i)}$. Therefore by the definition of $\mathcal{S}$ we conclude:

$$\Pr[y \text{ locked at } t] \leq \sum_{i=1}^{B_t} \frac{1}{\mathcal{S}(B_{t_i})} \leq \sum_{i=1}^{B_t} \frac{1}{\mathcal{S}(i)} \leq \frac{1}{2}.$$

Therefore, the probability of y being sent at t is at least $\frac{1}{2} \times \frac{1}{\mathcal{S}(B_t)}$. ∎

Now consider an element $y \in \Psi$. If y is neither completed nor eligible at a successful send event in $[\tau_\delta, \tau]$, then it must be the case that some other member of Ψ (with index $< y$) is successfully delivered at that time. By our earlier bound on $|\Psi|$, this situation can occur at most $2\mathcal{S}(\widehat{B}_\tau)$ times. Since there are δ successful transmissions in $[\tau_\delta, \tau]$, y must therefore be eligible in at least $\delta - 2\mathcal{S}(\widehat{B}_\tau)$ of them, or must have been successfully delivered by τ. By applying Lemma 6, the probability that y is not successfully delivered at time t in $[\tau_\delta, \tau]$ in which it was eligible is at most $\left(1 - \frac{1}{2\mathcal{S}(B_t)}\right) \leq \left(1 - \frac{1}{2\mathcal{S}(\widehat{B}_\tau)}\right)$, so the probability that y is not delivered by time τ is bounded by

$$\left(1 - \frac{1}{2\mathcal{S}(\widehat{B}_\tau)}\right)^{\delta - 2\mathcal{S}(\widehat{B}_\tau)} \leq e^{-(2c \ln 2\mathcal{S}(\widehat{B}_\tau) - 1)}.$$

Now by taking a union bound over the at most $2\mathcal{S}(\widehat{B}_\tau)$ elements of Ψ, the probability that the algorithm does not complete the prefix through word x by time τ is bounded by $(2\mathcal{S}(\widehat{B}_\tau))^{-2c+2}$. ■

4.3 A Lower Bound for Randomized Data Selection Policies

We now prove a nearly matching lower bound on $\Delta(t, \theta)$ for randomized policies. The adversary we use is *oblivious*, meaning that it fixes its strategy in advance of the execution of the algorithm, and does not adapt its strategy to the random choices made by the algorithm. We feel that in the context of our problem, oblivious adversaries are the correct choice, since an adaptive adversary which inspects the content of packets can be thwarted using encryption. Of course, the lower bounds we prove with oblivious adversaries also hold for adaptive adversaries. The simple strategy our adversary employs is to *admit* each packet independently with probability $\frac{1}{2}$, where admitting a packet is defined as successfully routing the packet to its destination. Using this adversary, we prove the following lemma, which implies Theorem 4 directly.

Lemma 10 *For any algorithm $\mathcal{A}$,*

$$\exists \theta \ \exists t \textit{ such that } \Pr\left[\Delta^A(t, \theta) \geq \frac{B_\theta \log B_\theta}{22}\right] \geq \frac{1}{16}$$

Proof: Fix an algorithm A and a sufficiently long message size m. The adversary then chooses B_θ such that $B_\theta^2 \log^2 B_\theta = m$ and uses transcripts in which each *round* of B_θ send events is followed by an acknowledgment for the entire round. The adversary admits each packet into the network independently with probability $\frac{1}{2}$. We define a *phase* of the transcript to be a sequence of $\frac{\log B_\theta}{2}$ consecutive rounds. and we let $k = 2B_\theta \log B_\theta$ be the number of phases. If in a run of the algorithm, A transmits two or more copies of some message word in any round of phase i of the transcript, we say that A uses *redundancy* in phase i. Then we let r_i be the probability over all coin tosses of A and the adversary that A uses redundancy in phase i.

The r_i probabilities are used to select the value for t in the lemma. Intuitively, if r_i is large for all phases, the redundancy used by the algorithm will waste bandwidth, while if some r_i is small, the algorithm will have difficulty making sure that each of its transmissions contributes to $P^A(t, \theta)$ in phase i. In particular, if $r_i \geq \frac{3}{8}$ for all i then we choose t to be the time of the last send event ending phase k. Otherwise, there must exist a phase $j \in \{1 \ldots k\}$

such that $r_j < \frac{3}{8}$, and we choose t to be the time of the send event ending the first such phase j.

It is left to show that the lemma holds in both of these cases for our choice of t and θ. We make frequent use of the following fact in our analysis.

Fact 11 *Consider a random variable X which takes on values $\leq D$. Then for any α and γ such that $0 \leq \alpha, \gamma \leq 1$,*

$$\mathbf{E}[X] \geq \alpha D \Longrightarrow \Pr\left[X \geq (\alpha D)\gamma\right] \geq \frac{1-\gamma}{\frac{1}{\alpha}-\gamma}$$

Case I: $r_i \geq \frac{3}{8}$ for all $i \in \{1 \ldots k\}$.

In each phase in which A uses redundancy, the probability that two copies of the same message word are both admitted into the network is at least $\frac{1}{4}$, since each copy is admitted independently with probability $\frac{1}{2}$. We refer to such a phase as one in which the algorithm is *caught*. Each such phase increases the redundancy and hence $\Delta^A(t,\theta)$ is at least the number of such phases in $\{1 \ldots k\}$, which we denote by C. The expected value of C, $\mathbf{E}[C]$, is at least $\frac{1}{4} \times \frac{3k}{8}$, and C is bounded above by k, so by applying Fact 11 with $\alpha = \frac{3}{32}$ and $\gamma = \frac{1}{4}$:

$$\Pr\left[C \geq \frac{3k}{32} \times \frac{1}{4}\right] \geq \frac{9}{125} > \frac{1}{16}$$

Using C as a lower bound on the value of $\Delta^A(t,\theta)$, and plugging in for k yields:

$$\Pr\left[\Delta^A(t,\theta) \geq \frac{B_\theta \log B_\theta}{22}\right] \geq \frac{1}{16}$$

which completes the proof for Case I.

Case II: There exists a $j \in \{1 \ldots k\}$ such that $r_j < \frac{3}{8}$.

Recall that t is set to be the time ending the first phase j in which the algorithm is unlikely (i.e., with probability $< \frac{3}{8}$) to use redundancy. To show the result, we assume first that the algorithm will not use redundancy in phase j and let the algorithm run on a transcript θ up until time s, the beginning of phase j.

Let $N_j(\theta,\rho)$ denote the event that algorithm A does not use redundancy on transcript θ with random tape ρ while executing phase j. We use N_j when θ and ρ are understood. Let $Y = P^*(t,\theta) - P^*(s,\theta)$, the random variable which measures the available bandwidth in phase j. We first show that the available bandwidth in phase j must be large with constant probability, even when conditioning on the event that our algorithm chooses not to use redundancy. By our choice of the length of a phase and since each packet is admitted with probability $\frac{1}{2}$, $\mathbf{E}[Y] = \frac{B_\theta \log B_\theta}{4}$. Now, since $\mathbf{E}[Y] = \Pr[N_j]\ \mathbf{E}[Y|N_j] + (1-\Pr[N_j])\ \mathbf{E}[Y|\bar{N}_j]$, and Y (and $\mathbf{E}[Y|\bar{N}_j]$) is bounded above by $\frac{B_\theta \log B_\theta}{2}$, we have:

$$\mathbf{E}[Y|N_j] \geq \frac{1}{\Pr[N_j]}\left(\ \mathbf{E}[Y] - (1-\Pr[N_j])\ \mathbf{E}[Y|\bar{N}_j]\right)$$

$$\geq \frac{B_\theta \log B_\theta}{4}\left(2 - \frac{1}{\Pr[N_j]}\right) \geq \frac{B_\theta \log B_\theta}{10} \tag{1}$$

The next step is to show that any algorithm which does not use redundancy in phase j makes far less progress than the optimal algorithm with very high probability. For an algorithm G let $\overline{P}^G(t,\theta) = \max\{P^G(t,\theta), P^*(s,\theta)\}$. We will show that

$$\max_G \mathbf{E}[\overline{P}^G(t,\theta) - P^*(s,\theta)] < B_\theta$$

where the max is taken over algorithms G which do not use redundancy in phase j. Let C denote the algorithm achieving the maximum above. Now consider a word y not admitted prior to the beginning of the phase by C. By the fact that C does not use redundancy, y is transmitted at most $\frac{\log B_\theta}{2}$ times in round j. And since the adversary chooses whether to admit each transmission independently with probability $\frac{1}{2}$, the probability that y is admitted in the phase is $x = 1 - (\frac{1}{2})^{\log B_\theta/2} = 1 - \frac{1}{\sqrt{B_\theta}}$. Therefore,

$$\mathbf{E}[\overline{P}^C(t,\theta) - P^*(s,\theta)] = \sum_{i=0}^{\infty} i x^i$$

$$= \frac{x}{(1-x)^2} = B_\theta\left(1 - \frac{1}{\sqrt{B_\theta}}\right) < B_\theta.$$

It follows from the fact that C achieves the maximum expectation that

$$\mathbf{E}[\overline{P}^A(t,\theta) - P^*(s,\theta)|N_j]$$

$$\leq \frac{1}{\Pr[N_j]}\ \mathbf{E}[\overline{P}^C(t,\theta) - P^*(s,\theta)] \ < \ \frac{8B_\theta}{5}. \tag{2}$$

Subtracting (2) from (1) we get:

$$\mathbf{E}[P^*(t,\theta) - \overline{P}^A(t,\theta)|N_j] > \frac{B_\theta \log B_\theta}{10} - \frac{8B_\theta}{5} \geq \frac{B_\theta \log B_\theta}{11}.$$

Since $P^*(t,\theta) - \overline{P}^A(t,\theta) \leq \frac{B_\theta \log B_\theta}{2}$, we may use Fact 11 with $\alpha = \frac{2}{11}$ and $\gamma = 1/2$ to obtain

$$\Pr\left[P^*(t,\theta) - P^A(t,\theta) \geq \frac{B_\theta \log B_\theta}{22}|N_j\right] \ \geq$$

$$\Pr\left[P^*(t,\theta) - \overline{P}^A(t,\theta) \geq \frac{B_\theta \log B_\theta}{22}|N_j\right] \ \geq \ \frac{1}{10}.$$

Hence,

$$\Pr\left[P^*(t,\theta) - P^A(t,\theta) \geq \frac{B_\theta \log B_\theta}{22}\right] \geq \frac{5}{8}\cdot\frac{1}{10} > \frac{1}{16}.$$

■

5 Conclusions

Our objective in this work was to provide a theoretical analysis of TCP-like protocols in a model which strives to make as few assumptions about the network as possible. The model we chose enables an adversary to regulate the speed at which a protocol may transmit, the pattern with which the transmitted packets are lost, and the latency with which feedback about transmissions propagates back to the sender. We then measure our algorithm's performance on this transcript in comparison to the theoretically optimum performance, to determine the algorithm's regret on the worst case transcript.

One of our findings is that there is substantial benefit to be derived from decoupling a TCP-style protocol into component policies; in particular, separating out the decision of what to transmit from other policies, such as congestion control. Our main result develops a universal data selection policy, which has a near-optimum performance guarantee on *every* transcript, i.e. for any specification of the behavior of the network.

The most natural open question our paper poses is that of designing an optimal time-scheduling policy, since such a policy can be designed with the knowledge that our data selection policy is guaranteed to work well in conjunction with it. Among the difficulties are that existing protocols (such as TCP) have fairness requirements, and act conservatively in periods of high congestion to avoid overflowing the queues of routers. Neither of these objectives are captured in our current model.

References

[AAF+ 96] D.M. Andrews, B. Awerbuch, A. Fernandez, J. Kleinberg, F.T. Leighton, Z. Liu. Universal stability results for greedy contention-resolution protocols. In *Proc. 37th IEEE Symposium on Foundations of Computer Science*, 1996.

[BPSK 96] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz. A Comparison of Mechanisms for Improving TCP Performance over Wireless Links. In *Proc. ACM SIGCOMM '96*, pp. 256-269, 1996.

[BKR+ 95] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, D. Williamson. Adversarial queueing theory. In *Proc. 28th ACM Symposium on Theory of Computing '96*, 1995.

[BOM 94] L. S. Brakmo, S. W. O'Malley and L. L. Peterson. TCP Vegas: New Techniques for Congestion Detection and Avoidance. In *Proc. ACM SIGCOMM '94*, pp. 24-35, 1994.

[CLZ 87] D. D. Clark, M. L. Lambert, and L. Zhang. NETBLT: A High Throughput Transport Protocol. *Computer Communications Review*, 17(5):353-359, 1987.

[FF 96] K. Fall and S. Floyd. Simulation-based Comparisons of Tahoe, Reno and SACK TCP. *Computer Communications Review*, 26(3), July 1996.

[Jac 88] V. Jacobson. Congestion Avoidance and Control. In *Proc. ACM SIGCOMM '88*, pp. 314-329, 1988.

[JB 88] V. Jacobson and R. Braden. TCP Extensions for Long-Delay Paths, October 1988. RFC 1072.

[LTWW 95] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the Self-Similar Nature of Ethernet Traffic (Extended Version). *IEEE/ACM Transactions on Networking, Vol. 2, No. 1*, pp. 1-15, Feb. 1995.

[MM 96] M. Mathis and J. Mahdavi. Forward Acknowledgment: Refining TCP Congestion Control. In *Proc. ACM SIGCOMM '96*, pp. 281-291, 1996.

[Pax 96] V. Paxson. End-to-End Routing Behavior in the Internet. In *Proc. ACM SIGCOMM '96*, pp. 25-38, 1996.

[ST 85] D. Sleator and R. Tarjan. Amortized Efficiency of List Update and Paging Rules. In *Communications of the ACM*, 28(2):202-208, 1985.

[YKT 96] M. Yajnik, J. Kurose, and D. Towsley. Packet Loss Correlation in the MBone Multicast Network, In *Proceedings of IEEE Global Internet '96,* November, 1996.