An Optimal Solution to Head-of-Line Blocking

Sanjeev Mehrotra

ABSTRACT

Head-of-Line (HOL) Blocking is a well known cause of performance degradation due to missing packets resulting in latency not only for themselves but also to dependent packets which have already arrived. Many existing approaches attempt to alleviate the issue by removing dependencies between packets so that packets which have already arrived have a lower chance of being delayed due to other packets which are missing. Other complimentary approaches use adhoc proactive retransmission or error correction to lower the probability of loss itself. In this paper, we show how application dependent requirements can be used in conjunction with network parameter estimates to provide an *optimal* approach to proactive retransmissions or error correction. In this approach, using network parameter estimates of bandwidth, delay, and loss, we first mathematically derive a probability distribution of the per packet delay which is then used to optimize parameters in a forward error correction framework given application requirements.

1. INTRODUCTION

In an application where all packets from a network flow are required in order for the receiver to utilize the information, the network effect on application performance can be measured using the *per-flow latency*, which we define as the following.

DEFINITION 1. *The* **per-flow latency** *is the time difference between when the first packet is sent to the time that all packets in the flow have been received.*

Since a rough estimate of the per-flow latency is given by $\delta + \frac{M}{R}$, where δ is the network latency (e.g. in seconds), M is the total size of the flow (e.g. in bits), and R is the effective transmission rate (accounting for losses) (e.g. in bits/second), the best that can be done is to make the effective transmission rate as close to the capacity of the channel. If acknowledgments regarding which packets have been received are available to the sender, then it is known that retransmissions are the best method for recovering from any losses [3]. In applications such as file transfer (e.g. FTP), this is the case.

However, in many applications the receiver can utilize partial information from a network flow and thus individual packet latencies also affect performance. Some examples of these include the following.

1. An application which muxes several different streams or messages together into one flow [1, 15]. In such an application, partial information can be easily utilized by extracting a single stream from the flow.
2. An application where a subset of packets can be rendered and utilized by the receiver with the quality of the rendering being a function of the subset. Typical examples include interactive software and media applications such as web or cloud based software applications, online games, and video conferencing.

In such cases, the *per-packet sequential latency* (PPSL) is also important in determining the application performance and is defined as the following.

DEFINITION 2. *The* **per-packet sequential latency** *(PPSL) is defined to be the time difference between when a particular packet is sent to the time that it and all of the packets upon which it depends have been received. The term* **sequential** *is used since all dependent packets must also be received.*

For example, if a flow consists of the M packets indexed $0, \ldots, M-1$, and s_n is the send time of packet n and r_n is the receive time, we can define the *per-flow latency* as

$$\tau_{flow} = \max(\{r_i, i \in 0, \ldots, M-1\}) - s_0 \qquad (1)$$

and the *per-packet sequential latency* (PPSL) of packet n as

$$\tau_{packet,n} = \max(\{r_i, i \in \mathcal{D}_n\}) - s_n, \qquad (2)$$

where $\mathcal{D}_n$ is the set of packets including packet n and those packets upon which packet n depends. Although for simplicity we use the term *received* throughout our discussion, by *received*, we mean that either the packet is actually received or there is sufficient information (via the use of FEC packets) to recover the packet.

A special case in protocols such as TCP and others which deliver packets losslessly and in-order gives the following expression for per-packet sequential latency,

$$\tau_{packet,n} = \max(\{r_i, i \in 0, \ldots, n\}) - s_n. \qquad (3)$$

This is because packet n is dependent on all prior packets. Thus if packet k is missing, then it is having an effect on the per-packet sequential latency for all received packets $n \geq k$ resulting in what is commonly known as the *Head-of-Line (HOL)* blocking effect. We refer to lossless and in-order delivery as *reliable*.

In cases such as the muxing of several streams into one flow, the use of TCP can result in unnecessary HOL blocking as a missing packet from one stream can cause HOL blocking on other streams which are not missing any packets up to that point [6, 14]. However, in such cases, this can be easily avoided by realizing that the packet dependency structure imposed by TCP is not needed. For example, the underlying transport can simply deliver a packet to the application so long as all prior packets from a given *stream* have been received instead of waiting for all prior packets from the *flow* [6]. In addition, if an intelligent acknowledgment (e.g. selective ACK) and retransmission scheme is used, the protocol can perform at close to capacity [10]).

However, in other cases, the packet dependency structure is truly an application required dependency and not simply a protocol imposed dependency. For example, in interactive software applications and online games, the packets being sent may be used to show the current rendering of the user interface (UI) in response to user actions. The coding of this rendering would likely employ techniques to remove redundancy and thus have dependencies between packets. In addition, the effect of missing packets may severely affect the rendering and the application may in fact require lossless, in-order delivery in order to be usable.

For applications such as video-conferencing, packet dependency is often imposed by commonly used techniques such as the use of P-frames and also intra-frame prediction techniques [18]. Although it is commonly assumed that video decoders are capable of recovering from lost packets, the effect of packet losses is often more severe than the redundancy incurred in imposing close to lossless delivery (via the use of proactive forward error correction) [20, 21].

Thus, we see that even though there are several applications where protocol imposed packet dependencies can be easily removed, there are other applications where HOL blocking can affect performance severely. Although we can use ad-hoc techniques such as in [13] to proactively retransmit and send forward error correction packets, it may result in under-protection or over-protection depending on application requirements and network characteristics.

Therefore in this paper, we attempt to determine the correct level of forward error correction (FEC) protection for an application which meets the following characteristics.

1. The application is able to adjust its coding rate.
2. Provided the application meets the coding rate, the application requires reliable (or close to reliable).
3. The application has certain per-packet sequential latency constraints that it would like to meet. As an example, it may want the probability that PPSL is greater than or equal to 150ms to be less than 1%.

In this paper, we make the following contributions for applications with the above characteristics.

1. Describe a forward error correction (FEC) code by applying a simple modification to a Reed-Solomon style Maximum Distance Separable (MDS) code for applications which require lossless, in-order delivery.
2. Derive a mathematically accurate probability distribution of the per-packet sequential latency (as defined above) given network characteristics of bandwidth, delay, and loss and coding parameters for the modified MDS code.
3. Given the probability distribution of the per-packet sequential latency, provide an optimization framework to find FEC coding parameters to meet desired application requirements. For example, given a channel with a loss rate of 1%, RTT of 100ms, bandwidth of 2Mbps, and packet size of 1400 bytes, what is the FEC strategy so that the application sees a 99% delay of $\leq$ 175ms while maximizing the rate available to the application?
4. Provide an efficient (memory and complexity-wise) implementation which fairly accurately approximates the optimization.

In this paper, we do not address the issue of network parameter estimation and control for which there is already a significant body of research as we will show in Sec. 8. Although these are critical pieces of a practical system, they are somewhat orthogonal to the issue of the optimization presented here. The use of congestion control protocols to control the rate of transmission in the network is a well researched issue and determines *how much data to send*. What we address here is not *how much to send*, but rather the issue of *what to send*.

In fact, any good congestion control protocol should be completely independent of *what the application sends* as the network itself is completely agnostic of the actual data being sent, and at least for congestion purposes the only thing that matters is *how much is sent*. For example, even in the basic TCP protocol, one could arbitrarily send any data (even for example when TCP says to retransmit), and there would be no effect on the network's congestion characteristics.

Since we are not controlling *how much to send*, the use of FEC or other proactive techniques to recover from losses will not increase the congestion seen on the network. Rather it will only reduce what is available to the application.

In addition, there is also already a rich body of literature on network parameter estimation, namely estimating the current bandwidth, delay, and loss which we will present in Sec. 8. It is well-known that these parameters may be exhibit a certain degree of memory (i.e. each sample in time may not be independent of the previous) and may even exhibit non-stationary behavior [22]. Also, the congestion control protocol itself will have an effect on these parameters. In this paper, we are not addressing the issue of techniques to improve the estimation, but rather are concerned with what to do once we have these estimates.

In any application which requires lossless, in-order delivery of data, we cannot solely rely on FEC codes as there may still be losses even after the use of FEC. Therefore, the application must use FEC codes in combination with retransmissions. However, the amount of FEC (and thus the amount of loss recovered via retransmissions) can be controlled depending on application requirements and network characteristics. By addressing the issue of optimally finding FEC parameters given the above, we can improve the performance of applications which have rate control, but require lossless (or near-lossless), in-order delivery of data.

2. FORWARD ERROR CORRECTION CODE

We consider using a systematic (N, K) *block FEC code*, where K original source packets – defined as a *coding block* – are sent along with $N - K$ FEC packets giving a *redundancy* of $\frac{N-K}{K}$. We use a Cauchy Reed-Solomon code [2] as the basis for the block code. This code is a Maximum Distance Separable (MDS) code which means that any K packets out of the N are sufficient to recover all K packets in the *coding block*. Let $x_i, i = 1, 2, ..., K$ be the original K packets and let $y_j, j = K + 1, K + 2, ..., N$ be the $N - K$ FEC packets. We use the following to compute the FEC packets,

$$y_j = \sum_{i=1}^{K} \frac{1}{i+j} x_i, \quad j = K+1, K+2, ..., N \tag{4}$$

where all mathematical operations (division, addition, and multiplication) are done over a finite field, e.g. $GF(2^8)$. Using matrix notation,

$$\mathbf{y} = \mathbf{Gx}$$

$$\begin{bmatrix} x_1 \\ \vdots \\ x_K \\ y_{K+1} \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \\ \frac{1}{1+K+1} & \cdots & \frac{1}{K+K+1} \\ \vdots & \ddots & \vdots \\ \frac{1}{1+N} & \cdots & \frac{1}{K+N} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_K \end{bmatrix} \tag{5}$$

.

2.1 Modified Code for Lossless, In-Order Delivery

If we wish to minimize the per-packet sequential latency for lossless, in-order delivery of data, we can modify the code so that FEC packets for a given block also include terms corresponding to packets in previous blocks In this way, previous blocks with missing packets may be able to derive additional FEC packets to allow for recovery. In case there are no packets missing in the previous coding blocks, the previous block's packets can be removed from the FEC packet thus not hurting decodability of the current block.

Given two consecutive blocks, the first block consisting of packets $x_{1,1}, \ldots, x_{1,K}$ and the second block of packets $x_{2,1}, \ldots, x_{2,K}$, we can modify the FEC packets for the second block to include additional protection for the first block using

$$y_{2,j} = \sum_{i=1}^{K} \frac{1}{i+j} x_{2,i} + \frac{1}{i+j+(N-K)} x_{1,i}, \tag{6}$$

where $j = K+1, K+2, ..., N$. This results in the following generator matrix,

$$\mathbf{G} = \begin{bmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 & \cdots & 0 \\ \frac{1}{1+K+1} & \cdots & \frac{1}{K+K+1} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{1+N} & \cdots & \frac{1}{K+N} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 1 \\ \frac{1}{1+N+1} & \cdots & \frac{1}{K+N+1} & \frac{1}{1+K+1} & \cdots & \frac{1}{K+K+1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{1+2N} & \cdots & \frac{1}{K+2N} & \frac{1}{1+N} & \cdots & \frac{1}{K+N} \end{bmatrix}. \tag{7}$$

That is FEC packets in block 2 also include additional terms in the linear sum corresponding to additional FEC packets from block 1 (beyond the $N - K$ FEC packets that have already been sent for block 1).

This modification can help in decoding block 1 without hurting block 2.

1. If block 2 is already decodable without a given FEC packet (say the jth FEC packet in block 2), then we can remove the term $\sum_{i=1}^{K} \frac{1}{i+j} x_{2i}$ from y_{2j} to obtain an additional FEC packet $y_{1,N+j-K}$ for block 1.
2. If block 1 is already decodable, block 2 is still equally decodable since the term $y_{1,N+j-K}$ can be removed from y_{2j} and give an FEC packet which is the same as before.
3. If block 1 is not decodable, then the additional terms corresponding to block 1 packets in block 2's FEC packets can prevent block 2 from being decodable. However, for reliable applications which require lossless, in-order delivery, this cannot hurt performance since block 2 is not useful without block 1 anyways.

We can also modify this to include additional previous blocks. For example, the FEC packet for block m can include additional FEC protection for blocks $m-1, \ldots, m-M$, where M is the number of blocks which still have unacknowledged packets remaining. Additionally, we can also modify the code in case successive blocks use differing coding structures, for example if block m uses a (N_m, K_m) code.

For analysis purposes the block coding structure is assumed, even though for practical purposes, the modified code can be used. The modified code can only improve the delay seen by the application and thus if the traditional block code is able to meet the application requirements for a given set of network parameters, then the modified one will also meet the requirements.

3. PER-PACKET SEQUENTIAL LATENCY ANALYSIS

We define lossless, in-order delivery of packets as *reliable*. The per-packet sequential latency for reliable packet transmission is defined in Eqn. 3. We use the following notation in the analysis.

1. τ denotes the *per-packet sequential latency*.
2. (N, K) denotes parameters in the block FEC code.
3. P denotes the size of each packet.
4. T is the current transmission rate.
5. δ_{RTT} is the round-trip time between the sender and receiver.
6. δ_{OWD} is the one-way network delay between the sender and receiver.
7. ϵ is the loss rate between sender and receiver. We use $\rho = 1 - \epsilon$ to denote the probability of reception.

In practice, P may be variable although for FEC coding purposes, the best performance is achieved when packets are of similar size. If an FEC packet consists of a linear combination of packets of varying packet sizes, then the FEC packet size is simply the largest of all packet sizes.

The network parameters T, δ, and ϵ will indeed vary over time as the network changes and in response to congestion control. We assume that the values are averages over some reasonable time period. In addition, the exact strategy for parameter estimation may be different. For example, a common parameter estimation strategy for loss may utilize a two-state (or multi-state) model such as a Gilbert-Elliot model [5] where the channel is assumed to be in one of two or more states, where the underlying network characteristics (such as loss rate) are different in each state. The parameter estimation scheme then attempts to classify which state the channel is in and update the model parameters for the state only. However, for our purposes regardless of what estimation scheme is used, we will have an estimate of T, δ, and ϵ.

In addition, in practice the acknowledgments may also be lost (i.e. the loss rate between the receiver and sender may be non-zero). However, if the acknowledgment information is sufficiently small, then we can add sufficient redundancy to make the reverse loss rate arbitrarily small.

The per-packet sequential latency for packet n for reliable data transmission is given by

$$\tau_n = \max(\{r_i, i \in 1, \ldots, n\}) - s_n, \tag{8}$$

where s_n is the time packet n is sent and r_i is the time at which packet i is *received*.

To simplify the discussion when analyzing latency, we compute all time and latency measurements in units of packets. We can easily go from units of packets to seconds (and vice-versa) via $t_{seconds} = \frac{t_{packets} P}{T}$, where P is the packet size in bits/packet, and T is the transmission rate in bits/second, and t is some measure of time.

3.1 Per-Packet Delay

For any given FEC coding block of K packets, we define the *round* to be the number of times a packet in the coding block has already been sent. For example, in the first round (round 0), all K packets of a given block are sent followed by $N - K$ FEC packets. In round 1, packets which have still not been acknowledged by the receiver are sent again followed by potentially additional FEC packets for the block. This continues until all packet of the coding block have been acknowledged as being received.

In order to derive the probability distribution of the *per-packet sequential latency*, we first need to obtain an expression of the *per-packet delay*.

DEFINITION 3. *We define the* **per-packet delay** *of packet n to be $r_n - s_n$, where s_n is the first time packet n is sent (in round 0) and r_n is the time at which packet n is either received or successfully decoded via FEC.*

Since it takes δ_{OWD} time for a packet to be sent from the sender to the receiver, if a packet is received in round 0, it experiences a delay of δ_{OWD}. If it is recovered by means of FEC decoding in round 0, then it encounters a delay of $\delta_{OWD} + \delta_{FEC,n}$, where $\delta_{FEC,n}$ is the additional delay for a sufficient number of FEC packets for the block to be decoded.

If a packet is not received or recovered in round 0, then it takes $\delta_{RTT} - \delta_{OWD}$ for this feedback to come back and an additional δ_{OWD} time for the second transmission. Thus, a packet received or recovered in round 1 experiences a delay of $\delta_{OWD} + \delta_{RTT} + \delta_{FEC,n}$, where $\delta_{FEC,n}$ is the additional delay in case the retransmission is lost and the packet is recovered via FEC in round 1.

In general, if packet n is received or decoded in w_n rounds, it experiences a *per-packet delay* of

$$\begin{aligned} d_n &= r_n - s_n \\ &= \delta_{OWD} + w_n \delta_{RTT} + \delta_{FEC,n}, \end{aligned} \tag{9}$$

where $\delta_{FEC,n}$ is the additional delay in case FEC is used to decode the packet.

In a practical system, it may take slightly longer than δ_{RTT} prior to declaring a packet to be lost. For example, many protocols may declare a packet to be lost when (i) once acknowledgments from future packets have been received, (ii) once the protocol receives information from which it can infer that future packets have been received (e.g. via Dupacks in TCP), or (iii) once a certain timeout has been reached. Regardless of the exact mechanism used, it would simply add a mostly constant delay to the term and thus it is easy to incorporate in the analysis.

For purposes of obtaining a probability distribution, in Eqn. 9, we can see that w_n and $\delta_{FEC,n}$ are random variables whose probability distribution is determined by network characteristics as well as the coding parameters (N, K).

3.2 Per-Packet Sequential Latency

Manipulating the expression for *per-packet latency* from Eqn. 3 using the *per-packet delay* from Eqn. 9, we get

$$\begin{aligned}\tau_n &= \max(r_n, r_{n-1}, \dots, r_0) - s_n \\ &= \max_{i=0}^{n}(r_{n-i} - s_{n-i}) - (s_n - s_{n-i}) \\ &= \max_{i=0}^{n} d_{n-i} - (s_n - s_{n-i}) \\ &= \delta_{OWD} + \max_{i=0}^{n}\left(w_{n-i}\delta_{RTT} + \delta_{FEC,n-i} - (s_n - s_{n-i})\right)\end{aligned} \tag{10}$$

Given Eqn. 10, we make the following claim which we prove in Appendix A.

CLAIM 1. *For probability distribution purposes, with the reasonable assumption that $\delta_{RTT} \geq N$, the following random variable has the same probability distribution as τ_n,*

$$\tau_n' = \delta_{OWD} + \delta_{FEC,n} + \max_{i=0}^{n}\left(w_{n-i}\delta_{RTT} - (s_n - s_{n-i})\right). \tag{11}$$

where $\delta_{FEC,n}$ is the additional delay required to make the coding block to which packet n belongs decodable.

For simplicity going forward, we simply use τ_n to represent τ_n'.

Although the exact value for the term $s_n - s_{n-i}$ in Eqn. 11 depends on the indices of the coding block for each packet and the indices of the packets within the block, a simple approximation is $\frac{iN}{K}$ since there are K source packets for every N packets being sent and thus the separation between each source packet is approximately $\frac{N}{K}$.

3.2.1 Computing $\delta_{FEC,n}$

To compute the term $\delta_{FEC,n}$, we define $\iota_n \in \{1, \dots, K-1\}$ to be the *index* of the n-th packet in an FEC block where the earliest packet in the block is $\iota_n = 1$ and $\iota_n = K$ for the last packet in the block. Let $f_n \in \{0, \dots, N-K\}$ to be the number of FEC packets used to decode the block for packet n. Then,

$$\delta_{FEC,n} = \begin{cases} (K + f_n) - \iota_n & \text{if } f_n > 0 \text{ and packet } n \text{ lost} \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

For example, if the first packet in the FEC block is lost, then it encounters a delay of $K + f_n - 1$, whereas if the last packet is lost, it only encounters a delay of f_n. Since the probability distribution of $\delta_{FEC,n}$ is not dependent on w_n, we can assume η_n and $\delta_{FEC,n}$ to be *independent* random variables.

3.2.2 Components of τ_n

Since $\delta_{FEC,n}$ is independent of w_{n-i} for all i, we can write τ_n as the sum of two *independent* random variables, η_n and $\delta_{FEC,n}$,

$$\tau_n = \delta_{OWD} + \delta_{FEC,n} + \eta_n, \tag{13}$$

$$\eta_n = \max_{i=0}^{n}\left(w_{n-i}\delta_{RTT} - \frac{iN}{K}\right). \tag{14}$$

This allows for a much simpler procedure for obtaining the distribution of τ_n. We can now see that the per-packet latency can be seen as the sum of three independent components.

1. δ_{OWD} which is the network delay for a single transmission. This consists of the network propagation delay plus any queuing delay which is caused by congestion (including that due to the congestion control protocol itself). This component cannot be controlled by a coding scheme. The congestion control protocol may have some effect on it.
2. η_n which can be seen to be the delay from retransmission of the head-of-line (HOL) blocking packet. This HOL blocking packet could be packet n itself or some other previous packet which is missing.
3. $\delta_{FEC,n}$ which is the additional delay needed in case FEC is being used to recover the HOL blocking packet.

We note that except for the term δ_{OWD}, τ_n is a function of (i) δ_{RTT} in units of packets which is essentially the bandwidth-delay product in units of packets and (ii) the loss rate over the network. In addition, the per-packet latency increases as both of these increases, so a network with a higher bandwidth-delay product will be more severely affected by head-of-line blocking.

4. PROBABILITY DISTRIBUTION OF PER-PACKET LATENCY

In order to derive a distribution for τ_n, we first derive the probability distribution for the two independent random variables, η_n and $\delta_{FEC,n}$.

4.1 Probability Distribution of η_n

Assume that $\delta_{RTT} = RN$ in units of packets, that is there are R coding blocks in one RTT as shown in Fig. 1. Then we can write $\eta_n = RN \max_{i=0}^{n}\left(w_{n-i} - \frac{i}{KR}\right)$. We see that there is exactly one packet which determines η_n. Let I be the packet which determines η_n, that is,

$$\begin{aligned} I &= \operatorname{argmax}\left(\max_{i=0}^{n}\left(w_{n-i} - \frac{i}{KR}\right)\right), \\ \eta_n &= RN\left(w_{n-I} - \frac{I}{KR}\right). \end{aligned} \tag{15}$$

We see that η_n can take on the following values, $0, \frac{N}{K}, \frac{2N}{K}, \dots, \frac{VN}{K}$ for all integer V. The probability for η_n can be written as

$$P\left(\eta_n = \frac{VN}{K}\right) = \sum_{\{(L,J) s.t. (LRK-J)=V\}} P(w_{n-I} = L, I = J). \tag{16}$$

Now we need to determine the joint probability of w_{n-I} and I. We write this probability by conditioning on

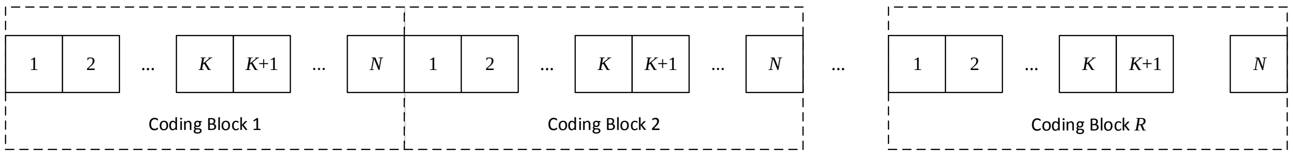

Figure 1: Coding blocks sent on network. There are R coding blocks per RTT.

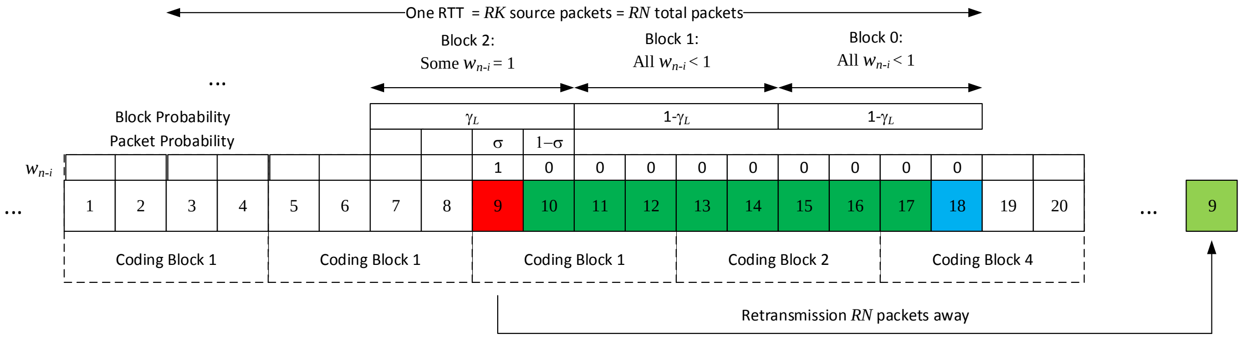

Figure 2: An example showing coding using $K = 4 packets$ over a network with $\delta_{RTT} = 4N$ packets. In the example, we show how a loss by packet 9 results in per-packet sequential latency for packet 18. For example, if $N = 6$, then packet 9 retransmission arrives at time $s_9 + 24$. Since $s_{18} \approx s_9 + \frac{9\times 6}{4}$, it results in packet 18 being delayed by $\approx 24 - 13.5 = 10.5$ (in units of channel packets). The figure also shows how the probability of this event can be computed using block probabilities γ and packet probabilities σ.

$P(w_{n-i} \leq L)$.

$$P(w_{n-I} = L, I = j) = P(w_{n-i} \leq L)P(w_{n-I} = L, I = J | w_{n-i} \leq L). \quad (17)$$

We then start at some L where we know $P(w_{n-i} \leq L) \approx 1$. We can recursively compute

$$P(w_{n-i} \leq L-1) = P(w_{n-i} \leq L) - \sum_{I=0}^{LRK-1} P(w_{n-I} = L, I = J | w_{n-i} \leq L). \quad (18)$$

In order to compute $P(w_{n-I} = L, I = J | w_{n-i} \leq L)$, we realize that by definition

$$P(w_{n-I} = L, I = J | w_{n-i} \leq L) = P(w_{n-J} = L) \prod_{i=0}^{J-1} P(w_{n-i} \leq L). \quad (19)$$

If $P(w_{n-i} \geq L)$ for any $i = 0, \ldots, J-1$, then that packet would have a higher value for $w_{n-i} - \frac{i}{KR}$ violating the definition of I. Also since $P(w_{n-i} \leq L)$, all packets $n - i$ for $i = J+1, \ldots$ will have a lower value for $w_{n-i} - \frac{i}{KR}$ with probability 1. That is, Eqn. 19 is the probability that packet $n - J$ is the closest packet to packet n which is not received (either received or decoded) in rounds 0 to $L - 1$ which is then subsequently received in round L given the condition that at least one such packet exists within LRK packets.

To obtain the probability that packet $n - I$ is the first such packet, we divide the packets prior to packet n into "virtual coding blocks" as shown in Figure. 2. That is, they need not align with the actual coding structure of the FEC. This is because for probability purposes the probability that a group of packets has a certain pattern of loss is independent of the packets in the group provided the same coding parameters are used for all packets in the group. The need for analyzing probabilities over coding blocks arises from the fact that the marginal probability of loss for any given packet is not completely independent of the other packets in the block. However, it is independent outside the block.

We can concretely write this as a product of three terms

$$P(w_{n-I} = L, I = J | w_{n-i} \leq L) = P(w_{n-J} = L) \times \prod_{i=0}^{\lfloor \frac{J}{K} \rfloor - 1} P(w_{n-Ki} \leq L, \ldots w_{n-(Ki+K-1)} \leq L) \times \prod_{i=0}^{\bmod (J,K)-1} P(w_{n-(K\lfloor \frac{J}{K} \rfloor + i)} \leq L), \quad (20)$$

where $\bmod (J,K) = J - K\lfloor \frac{J}{K} \rfloor$ is the "modulo" operator. Suppose $1 - \gamma_L$ is the probability that a group of K packets is completely recovered in L rounds $(0, \ldots, L-1)$ and γ_L

is the probability that at least one packet in the block is still missing after L rounds. Since $P(w_{n-J} = L)$, it implies that it comes from a coding block which has not been fully recovered in L rounds. Let $1 - \sigma$ be the probability a packet is received given that the coding block in which it is contained has not yet been received. Then, we can write Eqn. 20 as

$$P(w_{n-I} = L, I = J | w_{n-i} \leq L) = \frac{1}{A}\gamma_L(1-\gamma_L)^{\lfloor \frac{J}{K} \rfloor}\sigma(1-\sigma)^{\mod (J,K)} \quad (21)$$

$$A = \sum_{I=0}^{LRK} P(w_{n-I} = L, I = J | w_{n-i} \leq L), \quad (22)$$

Since $w_{n-i} \leq L$, if the packet J is still missing after rounds $0, \ldots, L-1$, then with probability 1, it will be recovered in round L. We show this probability computation pictorially in Fig. 2.

The reasoning behind this is that there are $\lfloor \frac{J}{K} \rfloor$ coding blocks prior to the block containing packet J. Since the probability of recovering each of them within L rounds is $1 - \gamma_L$, the probability of recovering all of them in L rounds is $(1-\gamma_L)^{\lfloor \frac{J}{K} \rfloor}$. For the coding block containing packet J, it has not been recovered which has a probability of γ_L. Within this coding block, there are $\mod (J,K)$ packets which have been recovered (each with probability $1-\sigma$ followed by the missing packet which has probability σ.

For a specific and practical case, consider the case when we start with $P(w_{n-I} \leq 1) = 1$. That is we assume that $w_{n-i} \in \{0, 1\}$. This itself is a reasonable assumption since ϵ is small and thus the probability of two losses even in the absences of FEC is much less than the chance of one since $\epsilon^2 << \epsilon$. With FEC, $P(w_n = 2) << P(w_n = 1)$ since $P(w_n = 1)$ will likely be significantly smaller than ϵ and very low.

In this case, γ_1 is the probability that at least one packet is missing in a block of K packets after round 0. We can write this as the probability of receiving less than K packets in a coding block,

$$\gamma_1 = \sum_{i=0}^{K-1} \binom{N}{i} (1-\epsilon)^i \epsilon^{N-i}. \quad (23)$$

The probability of recovering in round 1 given failure in round 0 is more difficult to compute as some of the packets have already been received in round 0. However, we can approximate it as $\gamma \approx \gamma_1$ and typically we can simply choose the number of FEC packets in the successive rounds so that they all have approximately the same γ.

To compute σ, we need to find the probability that any one particular packet is missing in a block of K packets given that at least one packet is missing. We compute this by conditioning on the number of received packets and realizing that if r packets are received the probability that any one particular packet is missing is $\frac{N-r}{N}$.

$$\sigma = \frac{\sum_{r=0}^{K-1} \binom{N}{r} (1-\epsilon)^r \epsilon^{N-r} \frac{N-r}{N}}{\gamma}. \quad (24)$$

Even without the assumption that $P(w_{n-I} \leq 1) \approx 1$, we can start with a higher value of L and perform the recursion fairly easily using Eqn. 18.

4.2 Probability Distribution of $\delta_{FEC,n}$

The additional delay due to FEC decoding can take on the values $0, 1, \ldots, N-1$ since the most the first packet in a coding block has to wait is $N-1$. It is clear that $\delta_{FEC,n} = 0$ if and only if all source packets in the block are received thus giving $P(\delta_{FEC,n} = 0) = (1-\epsilon)^K$.

For other $F > 0$, we can condition on the index of the packet ι using

$$P(\delta_{FEC,n} = F) = \sum_{i=1}^{K} P(\iota = i) P(\delta_{FEC,n} = F | \iota = i) \quad (25)$$

$$= \sum_{i=1}^{K} \frac{P(\delta_{FEC,n} = F | \iota = i)}{K}, \quad (26)$$

since the packet index is uniformly distributed, $P(\iota_n = i) = \frac{1}{K}$ for $i = 1, \ldots, , K$.

Since $\delta_{FEC,n} = (K + l) - i$ if l FEC packets are used to decode the block, $P(\delta_{FEC,n} = F | \iota = i) = P(l = F + i - K)$, where l is a random variable to represent the number of FEC packets used in the decoding. In order to use $l = F + i - K$ FEC packets to decode packet n, we must receive exactly $K - l$ out of K source packets, receive $f_n \geq l$ FEC packets, and packet $\iota_n = i$ must be lost.

Suppose we receive j packets from the $i-1$ packets prior to packet i and $l-j$ packets from the $K-i$ packets following packet i. Then, conditioning on j gives

$$P(l = F + i - K) = \sum_{j=0}^{\min(i-1,l)} P_J(j) P(l = F + i - K | j)$$

$$= \sum_{j=0}^{\min(i-1,l)} \binom{i-1}{j} \rho^j \epsilon^{i-1-j} \epsilon \binom{K-i}{l-j} \rho^{l-j} \epsilon^{(K-i)-(l-j)} P(f_n \geq l), \quad (27)$$

where $P_j(j)$ is the probability of receiving j packets prior to packet i. The probability that $P(fn_n \geq l)$ is given by

$$P(f_n \geq l) = \sum_{m=l}^{N-K} \binom{N-K}{m} \rho^m \epsilon^{N-K-m}. \quad (28)$$

4.3 Combined Distribution

Going forward, we drop the subscript n from the expressions as we assume the distributions to be stationary given

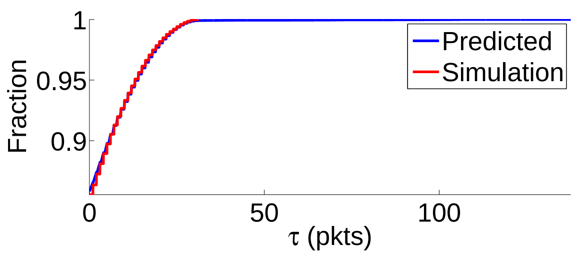

Figure 3: Cumulative probability distribution of $\tau_n - \delta_{OWD}$ using probability model and simulation.

the network estimates. Since η and δ_{FEC} are independent, we can write the probability that the per-packet sequential latency, τ, is some D larger than the base network delay of δ_{OWD} as

$$\begin{aligned} P(\tau = \delta_{OWD} + D) &= P(\eta + \delta_{FEC} = D) \\ &= \sum P(\delta_{FEC} = A) P(\eta = D - A). \end{aligned} \tag{29}$$

We can validate the accuracy of the probability distribution model by comparing the distribution from the probability model with that from simulation. We show the cumulative distribution of $\eta + \delta_{FEC}$ (the additional sequential latency beyond the minimum δ_{OWD}) in Fig. 3 for $\delta_{RTT} = 106$ packets, $\epsilon = 0.01$, $K = 30$ packets, $N = 33$ packets (3 FEC packets). We see that the cumulative distributions match showing that the derived probability distribution is correct.

5. FEC OPTIMIZATION

Suppose we wish to find the optimal (N, K) for a given per-packet sequential latency of D which is defined to be the maximal desired θ percentile value for $\tau - \delta_{OWD}$. That is we want to find (N, K) to minimize $\frac{N}{K}$ such that

$$P(\tau \leq \delta_{OWD} + D) \geq \theta. \tag{30}$$

In practice, the sequential latency target would be in units of seconds. We can convert this to units of packets via $D_{packets} = \lfloor \frac{D_{seconds} T}{P} \rfloor$. By taking the floor, the actual target in seconds can be met. We can write

$$P(\tau \leq \delta_{OWD} + D) = P(\eta + \delta_{FEC} \leq D) \tag{31}$$

$$== \sum_{A=0}^{N-1} P(\delta_{FEC} = A) P(\eta \leq D - A). \tag{32}$$

since η and δ_{FEC} are independent. To compute $P(\eta \leq D - A)$, we need to find the largest i such that $\frac{iN}{K} \leq D - A$, which is $i = \lfloor \frac{(D-A)K}{N} \rfloor$. Thus,

$$P(\eta + \delta_{FEC} \leq D) = \sum_{A=0}^{N-1} P(\delta_{FEC} = A) \sum_{i=0}^{\lfloor \frac{(D-A)K}{N} \rfloor} P\left(\eta = \frac{iN}{K}\right), \tag{33}$$

where the probabilities for δ_{FEC} and η are obtained as in Sec. 4.

Since it is not possible to find a closed solution for all (N, K) such that $P(\eta + \delta_{FEC} \leq D) \geq \theta$, we search over the space of (N, K) to find the optimal (N, K) which meets the delay target D and minimizes $\frac{N}{K}$. For a given K, we can stop searching over increasing N once we meet the target (otherwise we are over-protecting). Also, for a given K, no minimal N may be found to meet the criteria because the FEC decoder delay itself may exceed the delay threshold. In this case, we can stop increasing K once this condition is reached. We perform optimization using the following algorithm.

```
Set $K = 1$
Set $IncreaseCnt = 0$
Set current minimal redundancy $\lambda_{\min} = \infty$
Find minimal $N$ such that $P(\eta + \delta_{FEC} \leq D) \geq \theta$.
**Step 2:**
**if** $N$ found and $\frac{N}{K} < \lambda_{\min}$
    $\lambda_{\min} \leftarrow \frac{N}{K}$
**if** no $N$ found or $\frac{N}{K} > \lambda_{\min}$
    $IncreaseCnt \leftarrow IncreaseCnt + 1$
**if** $IncreaseCnt < \frac{K}{2}$
    $K \leftarrow K + 1$
    goto step 2
**else**
    exit
```

We note that although we present work to optimize the code to meet percentile delay constraints, it can easily be modified to allow the application to meet arbitrary *average* delay constraints or any other constraint which we can derive from a probability distribution. Meeting average delay constraints can be done by finding codes so that $E[\tau] \leq \delta_{OWD} + D$.

6. IMPLEMENTATION

Since it is difficult to compute the optimal (N, K) in real-time as it involves fairly complicated probability computations and searching over the space of (N, K), we can pre-compute the optimal values of (N, K) for various values of δ_{RTT} (in packets), ϵ, θ, and D (in packets) and store them in a table.

Since the number of possible values for each of the above parameters can be large, simple pre-computation for all possible values is not possible and thus the parameters must be quantized. We use log quantization for δ_{RTT} and D since the dynamic range of δ_{RTT} can be very large. δ_{RTT} in units of packets is the bandwidth-delay product (BDP) which can be on the order from a few packets up to several thousands. If we use linear quantization, then the percentage error in δ_{RTT} for small δ_{RTT} would be large. We use the following parameter quantization in our implementation.

- δ_{RTT} is quantized using log quantization in the range of $[5, 2000]$ using 50 bins. That is the quantized value is given by $Q(\delta_{RTT}) = 2^{I(\delta_{RTT})\Delta}$, where $\Delta = \frac{\log_2(2000)-\log_2(5)}{50}$ and $I(\delta_{RTT}) = \lceil \frac{\log_2(\delta_{RTT})-\log_2(5)}{50} \rceil$. Here ceiling is used since larger RTT requires more protection. Log quantization ensures that the *percentage error* after quantization is not large for small values.

- ϵ is quantized linearly in increments of 0.005 (0.5%) from 0 to 0.1. giving 21 possible values.
- θ is quantized to one of the following values $\{0.95, 0.98, 0.99, 0.995, 0.999\}$. Quantization gives a value larger than the targeted θ. For example, to target the 98.5% sequential latency, we would quantize θ to 0.99.
- D is quantized using the same way as δ_{RTT} except using a floor instead of ceiling since by meeting a smaller sequential latency target, the actual target will be met.

We note that the range of D can only be up to δ_{RTT} since we assume that we can always receive a packet in two rounds.

By using the above quantization, we only need to store $\frac{50*21*5*50}{2} = 131250$ values for the optimal (N, K). This makes the table size needed to about 260KB which is very manageable. In addition, computing the optimal values for all combinations does not take long ($<$ 10 minutes).

In order to implement usage in an application, the application needs an estimate of the packet size P, transmission rate T, loss rate ϵ, and RTT δ_{RTT}. The application specifies a target percentile for τ using D and θ. Using P and T, δ_{RTT} and D are converted into units of packets. Then, using δ_{RTT} (in units of packets), ϵ, θ, and D (in units of packets), we look up the optimal (N, K) in the precomputed table. Since the application may not be CBR, it waits for a minimum of K packets or for $\frac{KP}{T}$ seconds to declare a block boundary. Thus, the actual number of packets in a block may be less than K (say K'). It then inserts $\lceil \frac{N-K'}{K'} \rceil$ packets of FEC and moves to the next block.

7. EVALUATION

We first compare the optimal redundancy as a function of each of the four parameters, δ_{RTT}, ϵ, D, and θ. In order to do this, we vary each of the four parameters and fix the other parameters using $\delta_{RTT} = 106$ packets, $\epsilon = 0.01$, $D = 31$ packets, and $\theta = 0.99$ (99% delay of less than or equal to 31 packets). The results are shown in Fig. 4. We find the following four trends which is as expected. The required redundancy increases as δ_{RTT} increases, as ϵ increases, as the sequential latency target D decreases, and as the percentile for the latency target $theta$ increases. We note that even with a 1% loss rate, if we wish the 99% delay to be no more than a quarter of the RTT, then the redundancy required for high-BDP networks (2000 packets) is actually over 13%. Although this seems high, it is actually correct since a single packet loss causes a large number of packets to be delayed due to head-of-line blocking. In addition, we note that as the sequential latency target approaches the RTT, the required redundancy drops to zero. In such cases, the application can rely on retransmissions alone to meet the target.

We also show the optimal value of the FEC coding block size (K) using the same methodology in Fig. 5. Here we see the following trend. The optimal value for K is independent of δ_{RTT}, ϵ, and θ. It is mostly a function of the latency target D. This is most likely due to the fact that larger K is actually more optimal from an FEC standpoint. For example, $(N, K) = (24, 20)$ code is more optimal than a $(N, K) = (12, 10)$ from a decodability perspective even though both have the same amount of redundancy. The only reason K cannot go beyond a certain point is that a larger K results in not being able to meet the delay target, D. The only reason the optimal K does not grow beyond 30 for increasing values of D is because of the constraint placed when performing the search for the optimal (N, K).

In Fig. 6, we show the achieved delay target (the θ percentile value of $tau - \delta_{OWD}$) as a function of each of the four parameters along with the desired latency target, D, by performing simulation. For the cases when δ_{RTT}, ϵ, and θ are varied, the latency target, $D = 31$ is used. For the case when D is varied, the other parameters are fixed as before. This shows that the proposed optimization is indeed able to achieve arbitrary per-packet sequential latency targets as expected. In fact, the targets are met for all cases of δ_{RTT}, ϵ, D, and θ. The results in Fig. 6 simply show a subset of the results.

7.1 Sensitivity Analysis

We also attempt to perform sensitivity analysis to see what the effects are in case the network parameters are misestimated. The results are shown in Fig. 7. We first compute the optimal (N, K) with estimated network parameters of $\delta_{RTT} = 106$ packets and $\epsilon = 0.01$ for application requirements of $D = 31$ packets and $\theta = .01$, that is $P(\tau \geq 137) \leq 0.01)$. However, then we suppose that our estimates of the bandwidth-delay product are incorrect and we simulate what the effects of that would be.

From Fig. 7 (a) and (b), we see that even if the true bandwidth-delay product is between 50% to 150% of the estimated, there is no effect on the performance. From Fig. 7 (c) and (d), we find that if the true loss rate is double the estimated, then it has little effect on the performance. However, if the loss rate is 3% and is estimated at 1%, then the 99% delay increases to 3x of what is desired and the target delay becomes the 95% delay instead of the 99% delay as desired.

Overall, we see that performance is dependent on getting accurate network estimates, but some amount of misestimation is tolerated fairly well.

8. RELATED WORK

There is a significant amount of work related to network protocols. In particular, there is a large body of literature on congestion control protocols using TCP, TCP variants, and other bandwidth estimation techniques [4,7–9,11,16,17,19]. Much of this work can be used to provide the rate control

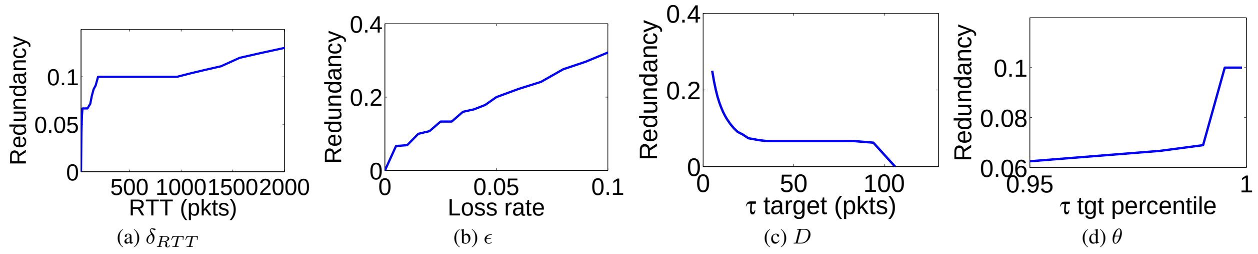

Figure 4: Optimal redundancy as function of each of the four parameters, δ_{RTT} (network RTT), ϵ (loss rate), D (sequential latency target), and θ (sequential latency target percentile). The specified parameter is varied while other three out of four are fixed at $\delta_{RTT} = 106$ packets, $\epsilon = 0.01$, $D = 31$ packets, and $\theta = .01$

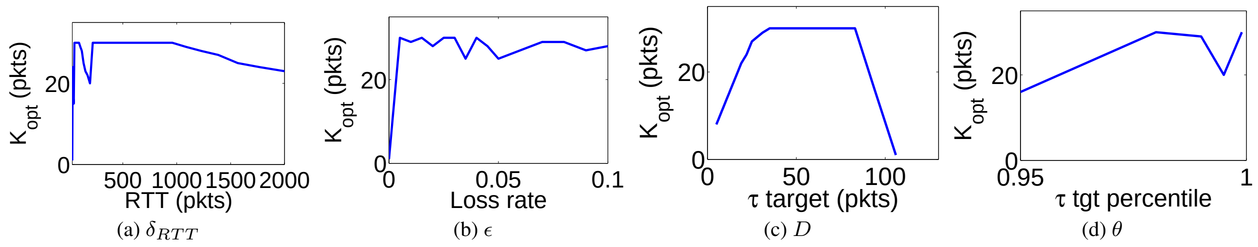

Figure 5: Optimal FEC block size (K) as function of each of the four parameters, δ_{RTT}, ϵ, D, and θ. The specified parameter is varied while the other three are held constant.

framework needed by any application. This can tell our algorithm *how much to send*. With minor modifications in signaling mechanisms and acknowledgments, much of this work can be modified so that it does not control *what to send*. Note that *what to send* should be application dependent as differing applications have differing requirements.

There is also a large body of literature in network parameter estimation [5, 16, 19, 22] which can be used both by congestion control protocols to enhance their performance and also be used by our protocol. This in conjunction with congestion control literature can provide the network control and estimation.

There has also been work which relates to improving the performance of real-time interactive applications by utilizing proactive forward error correction, both with and without retransmissions [12,13,23]. However, some propose adhoc FEC [13] which may result in under-protection or overprotection. In [23], the authors optimize for a different metric which is geared towards message based streaming where only each message must remain intact and some messages may be lost. In [12], the authors attempt to minimize average delay for reliable streaming. However, the optimization there cannot meet arbitrary statistical constraints which may be required by the application as there is no probability distribution. In addition, the average delay minimization does not take compute a *coding rate* for FEC, but rather is only able to generate instantaneous FEC packets based on the current buffer fullness.

As far as we know, there is no work in the literature which proposes to compute the exact probability distribution of sequential latency and then use it to derive optimal coding parameters to meet arbitrary statistical constraints which are needed by the application.

We also argue that it is actually extremely desirable to allow applications to decide exactly *what to send* as each of them has different requirements, some may want to be as close to channel capacity as possible, while others would trade capacity for reducing per-packet sequential latency. Allowing each application to independently decide what to send is entirely possible as it does not have any effect on the congestion characteristics of the network, so long as they obey *how much to send* which should be governed by the congestion control protocol. This separation between *how much to send* and *what to send* should be the future of all transport protocols going forward. Only *how much to send* requires cooperation amongst all the flows which are sharing the link.

9. CONCLUSION

In this paper, we have provided a probability model for the per-packet sequential latency which, in addition to throughput, is the measure which is important for user-perceived performance in interactive applications requiring reliable data transmission. Since both throughput and latency are important for such applications, we have used this probability model to optimize for throughput while still meeting arbitrary latency constraints as required by the application. By using pre-computation and intelligent quantization of network parameters, we have provided a way to efficiently implement this in an application without any significant overhead, both in terms of memory and computation.

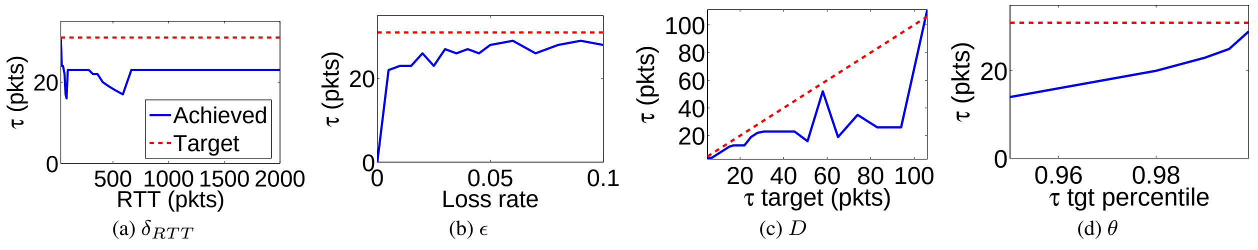

Figure 6: Achieved $\tau - \delta_{OWD}$ vs. the four parameters, δ_{RTT}, ϵ, D, and θ. sequential latency target is $D = 31$ packets except for (c) where D is varied.

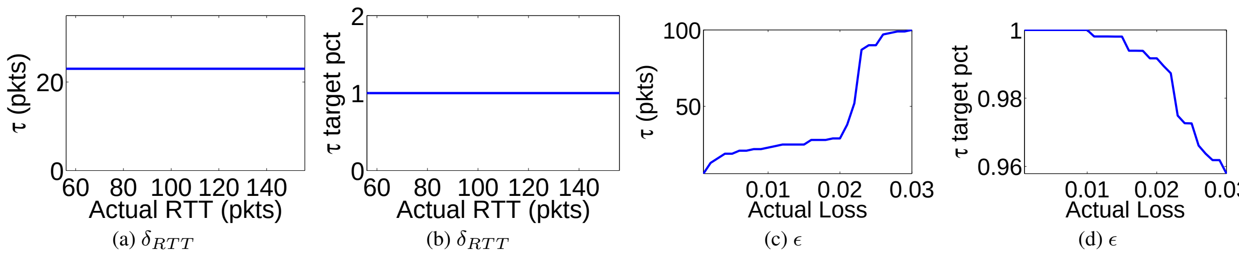

Figure 7: Sensitivity Analysis. Using optimal (N, K) with estimates of $\delta_{RTT} = 106$ packets, $\epsilon = 0.01$, $D = 31$ packets, and $\theta = .01$, (a) τ as function of δ_{RTT} which is different than estimate, (b) the target percentile achieved given D, (c) τ as function of ϵ which is different than estimate, (d) the target percentile achieved for the given D.

APPENDIX

A. PROOF OF CLAIM 1

Here we show that Eqn. 10 and Eqn. 11 have the same probability distribution with the reasonable assumption that $\delta_{RTT} \geq N$. Consider packet I such that

$$I = \operatorname{argmax} \max_{i=0}^{n} w_{n-i}\delta_{RTT} + \delta_{FEC,n-i} - (s_n - s_{n-i}). \tag{34}$$

We first prove the following two Lemmas.

LEMMA 1. *All packets within the coding block containing packet I and future coding blocks must have $w_{n-i} \leq w_{n-I}$.*

PROOF. Suppose $w_{n-i} > w_{n-I}$ for some i. If packet i has index $\iota \in 1, \ldots, K$ in the coding block, then $\delta_{FEC,n-i} = 0$ or $\delta_{FEC,n-i} = (K+1) - \iota, \ldots, N - \iota$. Then, we can write

$$\begin{aligned}
&w_{n-i}\delta_{RTT} + \delta_{FEC,n-i} - (s_n - s_{n-i}) - \\
&w_{n-I}\delta_{RTT} + \delta_{FEC,n-I} - (s_n - s_{n-I}) = \\
&(w_{n-i} - w_{n-I})\delta_{RTT} + \\
&(\delta_{FEC,n-i} + s_{n-i}) - (\delta_{FEC,n-I} + s_{n-I}) \geq \\
&(w_{n-i} - w_{n-I})\delta_{RTT} + (0 + \iota_{n-i}) - (N - \iota_{n-j} + \iota_{n-j}) = \\
&(w_{n-i} - w_{n-I})\delta_{RTT} - N + \iota_{n-i} \geq 1
\end{aligned} \tag{35}$$

since $\delta_{RTT} \geq N$. Thus, if $w_{n-i} > w_{n-I}$ for any i in the coding block, the definition of I is violated. For coding blocks sent after that containing packet i, s_{n-i} would even be larger and thus the definition of I would still be violated. □

Since all packets within the coding block have $w_{n-i} \leq w_{n-I}$, we can now show that $\tau_n = w_{n-I}\delta_{RTT} + \delta_{FEC,n-I} - (s_n - s_{n-I})$ can alternatively be written using

$$\tau_n = w_{n-J}\delta_{RTT} + \delta_{FEC,n-J} - (s_n - s_{n-J}), \tag{36}$$

$$J = \operatorname{argmax} \max_{i=0}^{n} w_{n-i}\delta_{RTT} - (s_n - s_{n-i}) \tag{37}$$

where $\delta_{FEC,n-J}$ is the additional delay required to decode the coding block to which packet J belongs. To show this, we first prove another Lemma.

LEMMA 2. *Packet I and J must be from the same coding block and $w_{n-I} = w_{n-J}$.*

PROOF. If I is contained in a future coding block from block J, then from 1, $w_{n-I} \leq w_{n-J}$. However, $w_{n-I} < w_{n-J}$ would violate the definition of I and thus $w_{n-I} = w_{n-J}$. From this, we would get

$$\begin{aligned}
w_{n-I}\delta_{RTT} - (s_n - s_{n-I}) - \\
w_{n-J}\delta_{RTT} - (s_n - s_{n-J}) = \\
s_{n-I} - s_{n-J} \geq \\
(\iota_{n-I} + N) - (\iota_{n-J}) \geq \\
>= N - K + 1
\end{aligned} \tag{38}$$

which would violate the definition of I. We can similarly show that if J is from future coding block, we would violate the definition of I. Thus, we can conclude that I and J have to be from the same coding block and have $w_{n-I} = w_{n-J}$. □

Now since I and J are from the same coding block, we see that if packet I has an index ι_I and if f_{n-I} FEC packets are used to decode the block, then

$$\tau_n = w_{n-I}\delta_{RTT} + (K + f_{n-I} - \iota_{n-i}) - (s_n - (s'_{n-i} + \iota_{n-i})) = w_{n-I}\delta_{RTT} + (K + f_{n-I}) - (s_n - s'_{n-i}), \quad (39)$$

where s'_{n-i} is the send time of the first packet within the coding block. Therefore, any packet in the coding block containing packet I which is recovered with $w_{n-i} = w_{n-I}$ will achieve the $\max$ and thus not only must packet I and J be from the same coding block, but they can be identical, that is $I = J$.

So for probability distribution purposes, using J will provide the same results as using packet I. Since $\delta_{FEC,n-i}$ is stationary, we can simply use $\delta_{FEC,n}$ for probability distribution purposes instead of using $\delta_{FEC,n-i}$. Thus we have proved the claim.

B. REFERENCES

[1] M. Belshe and R. Peon. SPDY protocolâĂŤdraft 3.1. *URL http://www. chromium. org/spdy/spdy-protocol/spdy-protocol-draft3-1*.

[2] J. Blömer, M. Kalfane, R. Karp, M. Karpinski, M. Luby, and D. Zuckerman. An xor-based erasure-resilient coding scheme. *Technical Report TR-95-048, International Computer Science Institute*, Aug. 1995.

[3] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley and Sons, New York, NY, 1991.

[4] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM Trans. on Networking*, 7(4):458–472, Aug 1999.

[5] E. N. Gilbert. Capacity of a burst-noise channel. In *Bell System Technical Journal*, volume 39, pages 1253–1265, 1960.

[6] K.-J. Grinnemo, T. Andersson, and A. Brunstrom. Performance benefits of avoiding head-of-line blocking in SCTP. In *Autonomic and Autonomous Systems and International Conference on Networking and Services, 2005. ICAS-ICNS 2005. Joint International Conference on*, pages 44–44. IEEE, 2005.

[7] S. Ha, I. Rhee, and L. Xu. CUBIC: A new TCP-friendly high-speed TCP variant. *ACM SIGOPS Operating System Review*, 42(5):64–74, Jul 2008.

[8] M. Jain and C. Dovrolis. End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput. *IEEE/ACM Trans. Networking*, 11:537–549, Aug. 2003.

[9] S. H. Low, L. L. Peterson, and L. Wang. Understanding TCP Vegas: a duality model. *Journal of the ACM*, 49(2):207 – 235, Mar. 2002.

[10] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP selective acknowledgment options. Technical report, RFc 2018, October, 1996.

[11] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. In *SIGCOMM Comput. Commun. Rev.*, volume 27, pages 67–82, New York, NY, USA, July 1997. ACM.

[12] S. Mehrotra, J. Li, and Y. zong Huang. Optimizing FEC transmission strategy for minimizing delay in lossless sequential streaming. *IEEE Trans. Multimedia*, May 2011.

[13] J. Roskind. Experimenting with QUIC. *URL http://blog.chromium.org/2013/06/experimenting-with-quic.html*.

[14] M. Scharf and S. Kiesel. Head-of-line blocking in TCP and SCTP: analysis and measurements. In *Global Telecommunications Conference, 2006. GLOBECOM'06. IEEE*, pages 1–5. IEEE, 2006.

[15] R. Stewart and C. Metz. SCTP: New transport protocol for TCP/IP. *Internet Computing, IEEE*, 5(6):64–69, 2001.

[16] J. Strauss, D. Katabi, and F. Kaashoek. A measurement study of available bandwidth estimation tools. In *IMC*, Oct 2003.

[17] K. Tan, J. Song, Q. Zhang, and M. Sridharan. A compound TCP approach for high-speed and long distance networks. In *INFOCOM*, pages 1 – 12. IEEE, Apr. 2006.

[18] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h. 264/avc video coding standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):560–576, 2003.

[19] K. Winstein, A. Sivaraman, and H. Balakrishnan. Stochastic forecasts achieve high throughput and low delay over cellular networks. In *NSDI*. USENIX, 2013.

[20] D. Wu, Y. T. Hou, and Y.-Q. Zhang. Transporting real-time video over the internet: Challenges and approaches. *Proceedings of the IEEE*, 88(12):1855–1877, 2000.

[21] D. Wu, Y. T. Hou, W. Zhu, Y.-Q. Zhang, and J. M. Peha. Streaming video over the internet: approaches and directions. *Circuits and Systems for Video Technology, IEEE Transactions on*, 11(3):282–300, 2001.

[22] M. Yajnik, S. Moon, J. Kurose, and D. Towsley. Measurement and modelling of the temporal dependence in packet loss. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 345–352. IEEE, 1999.

[23] C. Zhang, C. Huang, P. A. Chou, J. Li, S. Mehrotra, K. W. Ross, H. Chen, F. Livni, and J. Thaler. Pangolin: speeding up concurrent messaging for cloud-based social gaming. In *CONEXT*, page 23. ACM, 2011.