RPT: Re-architecting Loss Protection for Content-Aware Networks

Dongsu Han **Ashok Anand**[†] **Aditya Akella**[†]
Srinivasan Seshan

June 2011
CMU-CS-11-117

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
[†] University of Wisconsin-Madison, Madison, WI, USA

Keywords: interactive video chat, real-time communication, content-aware networks, redundancy elimination

Abstract

Many types of time-critical communication occur in today's networks. However, providing reliability or robustness to such transfers is often challenging. Retransmission-based schemes can violate timing constraints, and redundancy-based FEC schemes impose significant bandwidth overhead and are difficult to tune.

In light of recent advances in content-aware networking, we argue that redundancy-based approaches need to be re-engineered. Content-aware networks allow the overhead of redundancy to be made very small, if the redundancy is introduced in a way that the network can understand. Based on this insight, we propose redundant packet transmission (RPT), a new loss protection scheme for content-aware networks. Using redundant video streaming (RS) as an example, we show that our approach, unlike FEC, provides low latency communications with a high degree of robustness and is insensitive to parameter selection. We tackle practical issues, such as minimizing the impact on other traffic and the network. Furthermore, we show that RPT provides a simple and general mechanism for application-specific timing control and flow prioritization.

1 Introduction

A variety of current and future Internet applications require time critical or low latency communication. Example applications include live/interactive video streams, online games, and video-based calls (e.g., Apple's FaceTime), all of which send real-time data. Certain classes of inter-datacenter transfers, such as mirroring financial data, also require real-time communication [17]. Within a data center, many soft real-time applications that interact with users require low latency communication [13]. Market reports indicate that real-time video, including Internet TV and streams from monitoring cameras, will account for 13% of consumer Internet traffic [4] in the coming few years.

The central challenge in supporting such applications is protecting them from network loss. For example, data loss in video streams can degrade video quality and user experience. Acknowledgment-based retransmission protocols are traditionally used to provide reliability, but they are not appropriate for real-time communication as retransmissions triggered by timeouts can take several RTTs and violate applications' timing constraints [41, 34]. Alternatively, redundancy-based schemes like Forward Error Correction (FEC) can be used to improve robustness against packet loss. However, a tension exists in these schemes between robustness and the bandwidth overhead of redundancy [20, 22], making them either difficult to tune or inefficient in practice.

In this paper, we argue that redundancy-based approaches need to be re-evaluated and redesigned in light of recent advances in content-aware networking. Thus, we challenge the conventional wisdom on the trade-offs of redundancy and show that it is now possible to use redundancy to ensure robustness while imposing little or no impact on the network or on existing other applications, and making it easy to integrate loss protection with other design constraints in applications (e.g., adhering to delay bounds).

In content-aware networks, such as Redundancy-Elimination (RE) enabled networks [14, 1, 10] and data centers [8, 9], or content-centric network architectures [29], the network caches content and effectively removes duplicate transfers of the same content. This provides a tremendous opportunity to use redundancy-based protection schemes. However, redundancy must be introduced in the right way to ensure: *(a)* the network can eliminate it optimally to provide the desired efficiency and *(b)* the impact on other applications can be controlled.

We describe Redundant Packet Transmission (RPT) – a scheme that intelligently sends multiple copies of the same packet. While sending duplicates copies of a packet instead of using FEC may seem counterintuitive and high-overhead, we show that RPT on content-aware networks can effectively support a variety of time-critical applications far better than existing approaches. When packets are not lost, the duplicate transmissions are compressed and add little overhead to the network. In contrast when the network is congested, the loss of a packet prevents the compression of a subsequent transmission. This ensures that the receiver still gets at least one decompressed copy of the packet. In some situations, the fact that packet losses do not directly translate to the less bandwidth use may raise the concern that RPT streams are obtaining an unfair share of the network. However, existing congestion control schemes can be applied to RPT flows with some modifications to address this concern.

In essence, RPT signals the network the relative importance of a packet by transmitting multiple copies. At the same time, since the network compression behavior automatically adapts to current loss patterns, RPT requires far less tuning that FEC-based approaches. Finally, existing

FEC schemes closely tie delay guarantees with redundancy encoding since the receiver cannot reconstruct lost packets until the batch is complete. In contrast, RPT decouples redundancy from delay and can, therefore, more easily accommodate application timing constraints.

To explain the benefits of RPT concretely, we use redundant real-time video streaming (RS) in a redundancy elimination network as an example. Our evaluation of RS, using a combination of real-world experiments, network measurements and simulations, shows that RS achieves far better video streaming than traditional FEC-based schemes. RS decreases the data loss rate by orders of magnitude more than FEC schemes applicable to live video communications. As a result, it achieves better video quality than FEC for a given bandwidth budget, or uses 10 to 20% less bandwidth than FEC schemes to deliver the same video quality. Furthermore, it is simple and easy to use as it is insensitive to parameter selection unlike FEC-based coding schemes. Finally, our results also show that RS enables effective control of network resource consumption and can be made to coexist with existing traffic in a TCP-friendly fashion.

The rest of the paper is organized as follows. Section 2 reviews related work and background on content-aware networks. Sections 3 and 4 respectively explore the idea and design of redundant streaming and TCP-friendly RS on redundancy elimination networks. We discuss how general RPT schemes can be implemented on other content-aware networks (including networks with partial support) in Section 5, evaluate the performance of RPT in Section 6, and conclude in Section 7.

2 Background and Related Work

Loss protection today. Packet losses are often inevitable on the Internet, especially across heavily-loaded links, such as cross-country or trans-continental links. Many prior works use timeout-based retransmission to recover lost data [34, 18, 24, 38, 41]. However, retransmission causes large delays [41] which are often difficult to hide. Also, its performance depends on correct timeout estimation [34] which is non-trivial [34, 18]. Because of the intrinsic limitations, more sophisticated end-to-end behavior such as selective retransmission [24, 38], play-out buffering [38], and modification to codecs [41] are often required to augment retransmission based loss recovery.

FEC is another option. In fact, it is widely used in real-time communication today. However, the use of FEC is constraining in many ways. *(1)* In FEC, the receiver cannot recover lost packets until the batch is complete. This limits the size of the batch. For example, at most 5 packets are typically batched in video chat applications such as Skype [45]. *(2)* Small batch size makes FEC more susceptible to bursty loss. For example, adding a single coded FEC packet for every five original data packets is not to recover from two consecutive lost packets. Therefore, in practice, the amount of redundancy used is high, e.g., 20% to 50% [20, 22, 45],which is much higher than the underlying packet loss rate. *(3)* Furthermore, FEC needs to adapt to changing network conditions [19, 46], which makes parameter tuning even more difficult.

Many previous works have looked at fine tuning FEC parameters within various environments to make it usable by controlling the overhead. LIVE [46] uses complex feedback from routers to perform adaptive FEC coding. Frossard and Verscheure [26] show that FEC parameter selection is non-trivial and explore a joint source rate and FEC rate selection scheme to optimize video quality.

However, these approaches require complex modifications to routers and/or applications and, yet, their benefits in practice are unclear.

Opportunities in content-aware networks. In this paper, we show that content-aware networks provide a tremendous opportunity to design highly effective, yet low-overhead loss protection schemes that impose few or no practical constraints. Content-aware networks commonly suppress or eliminate duplicate transmissions of the same content to enhance network efficiency. Examples include content-centric networks, and redundancy elimination (RE)-enabled ISPs and data center networks. Partial deployments of such networks have already become popular [5, 10, 3]. Our idea is to leverage this key property of content-aware networks and introduce redundancy in a way that the network can understand. We refer to our approach as Redundant Packet Transmission or RPT. We argue that FEC is not the right way to introduce redundancy in content-aware networks, and redundancy should be introduce in a way that the network understands.

For example, in CCN [29], content is transferred only once on a particular link even if the network receives multiple interest packets for the content. While interest and data packets may be lost in an unreliable network, redundant interest packets can be used to signal the importance of the request and provide robustness against such loss. Furthermore, the cost of sending redundant interest packets is minimal compared to transferring a large content.

Redundancy elimination. In this paper, we focus largely on how RPT functions on a network with packet-level redundancy elimination networks [14], although our discussions apply to other content-aware networks as well (see Section 5). Furthermore, to highlight the benefits of RPT to the design and performance of time-critical applications, we use the example of redundant streaming (RS), which is RPT applied to video streams. The rest of this section provides additional details on redundancy elimination.

In Anand et al's [14] approach for IP layer RE in ISP networks, RE is deployed across individual ISP links. An upstream router remembers packets sent over the link in a cache (each cache can hold a few tens of seconds' worth of data) and compares new packets against the packets in the cache. It encodes new packets on the fly by replacing redundant content (if found) with the pointers to the cache. The immediate downstream router maintains an identical packet cache, and decodes the encoded packet by looking up these pointers. RE is applied in a hop-by-hop fashion. Caches across each hop are assume to be in synch (this is easy to enforce in practice [14]).

RE encoding and decoding are deployed on the line cards of the routers as shown in Figure 1. The decoding operation happens on the input interface before a packet enters the virtual output queue, and the encoding operation happens on the output interface before the packet leaves the router. The router's buffers (virtual output queues) contain fully decoded packets. It is shown that encoding and decoding can be done at line speed [14, 8].

Consider losses occurring due to overflow at router buffers in RE networks. In this situation, since packets are decoded before they reach the router buffer, the buffer would see duplicate packets. If the first of the duplicate packets is dropped, the next of the duplicates is transmitted without compression, thereby allowing downstream routers and nodes to reconstruct the transmitted data.

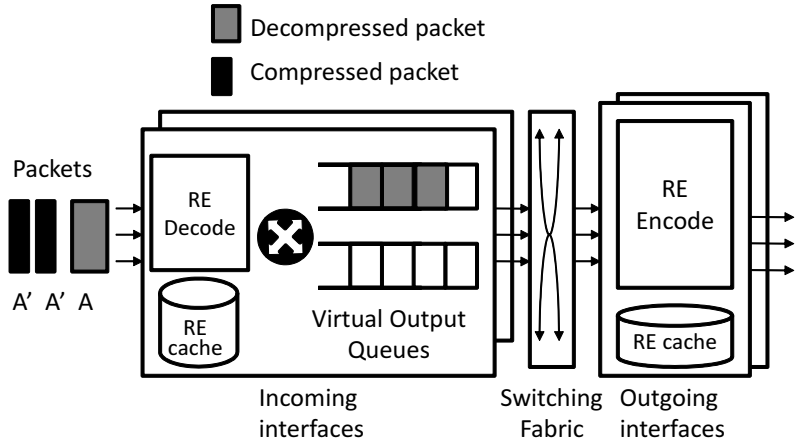


Figure 1: Illustration of Redundant Streaming in a redundancy elimination router.

3 Redundant Streaming

In this section, we describe the design of Redundant Streaming (RS), a special variant of RPT optimized for delivering real-time video streams. Throughout this section, we assume RS flows travel through a network with unmodified hop-by-hop RE [14] enabled on every router and end host, and packet losses happen only due to congestion. Later in Section 5, we relax this assumption and also explore RPT in content-aware networks of various other forms.

We envision a scenario in which real-time video traffic and other traffic coexist, with no more than 50% of the traffic on a particular link being real-time traffic. We picked this scenario because it is representative of forecasts of future network traffic patterns that predict that real-time video will account for 13% of the consumer Internet traffic by 2014 [4].

3.1 Basic Idea

As explained earlier, the basic idea of redundant streaming (RS) is to send multiple copies of the same packet. If at least one copy of the packet avoids network loss, the data is received by the receiver. In current network designs, transmitting duplicate packets would incur large overhead. For example, if two duplicates of every original packet are sent, the overhead is 200% and a 1Mbps stream of data would only contain 0.33Mbps of original data. However, in network with RE, duplicate copies of packet are encoded to small packets, and this overhead would be significantly reduced.

Figure 1 illustrates how RS works with redundancy elimination. From the input link, three duplicate packets are received. The first packet is the original packet A, and the other two packets A', are encoded packets which have been "compressed" to small packets by the previous hop RE encoder. The compressed packet contains a reference (14 bytes in our implementation) used by the RE decoder of the next hop. At the incoming interface, the packets are fully decoded, generating 3 copies of packet A. They are then queued at the appropriate output queue. The figure illustrates a router that uses virtual output queuing. When congestion occurs, packets are dropped at the virtual

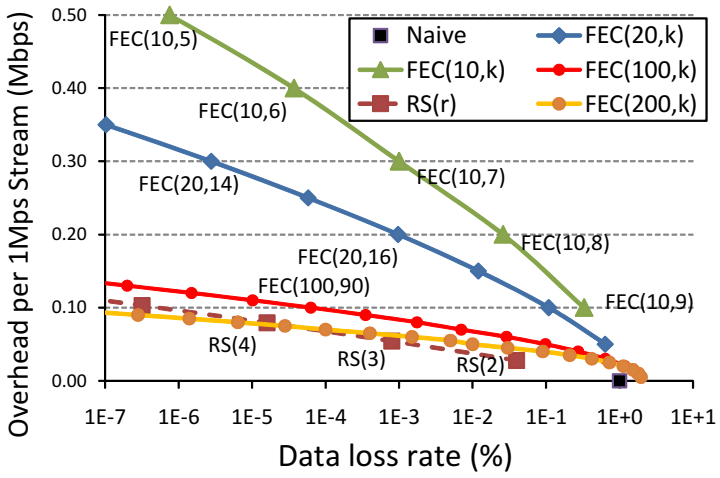


Figure 2: RS and FEC under 2% random loss.

output queue. Only packets that survive the loss will go through the RE encoder on the output interface. If some of the copies survive the network loss, they will again be encoded to small packets by the RE encoder. In this manner, multiple copies of packets provide robustness to loss and RE in the network reduces bandwidth overhead due to any additional copies.

3.2 Key Features

Next, we discuss three practically important properties of RS: *high degree of robustness for applications with low bandwidth overhead*, *ease of use and flexibility for application developers*, and *flow prioritization in the network*.

3.2.1 Low Overhead and High Robustness

As discussed in Section 2, the packet caches in the RE encoder and decoder are typically designed to hold all packets sent within the last tens of seconds. This is much longer than the timescale in which redundant packets are sent ($\sim$60ms). Thus, all redundant packets sent by the application will be encoded in RE with respect to the original packet. The extra bandwidth cost of each redundant packet is only the size of the encoded packet, which is 43 bytes in our implementation. [1] The overhead of an extra redundant packet, therefore, is less than 3% for 1,500 byte packets. This overhead is 7 to 17 times smaller than the typical FEC overhead for a Skype video call [22, 20].

To compare the performance of RS with FEC, we generate a uniform random packet loss pattern and measure the data loss of a 1Mbps RS and FEC streams. Each stream operates on a fixed budget and splits its bandwidth between original data and redundancy. The underlying loss rate is set to 2%. Figure 2 shows the overhead and data loss rate. RS(r) denotes redundant streaming that sends r duplicate packets. FEC flows with various parameters are shown for comparison. FEC(n,k) denotes that k original packets are coded in to n packets. For FEC, we use a systematic coding

[1] Our implementation does not encode IP and transport layer headers.

approach (e.g. Reed-Solomon) that sends k original packets followed by $n-k$ redundant packets. The vertical axis of the figure represents the amount redundancy in the 1Mbps stream.

FEC schemes with a small group size, i.e. small n, incur large overheads, and are much less effective in loss recovery than RS. For example, FEC(10,8), which adds 0.2 Mbps of redundancy, has similar data loss rates as RS(2), which only adds 0.03Mbps of redundancy. FEC with group size 200 performs similar to RS. However, it takes 2.4 seconds to transmit 200 1,500 byte packets at 1Mbps. This violates timing constraints of real-time communications because a packet loss may be recovered only 2.4 seconds later. Thus, in practice, RS provides high robustness against packet loss at low overhead.

3.2.2 Ease of Use and Control

RS is easy to use, i.e., application developers can easily tailor it to fit application needs. Three unique aspects of RS achieve this property:

1) Detailed parameter tuning is not necessary.

2) RS allows per-packet redundancy control.

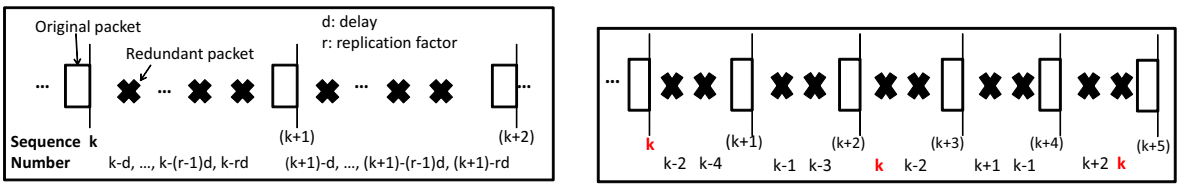
3) Delay and redundancy are decoupled from each other.

Ease of parameter selection: With FEC, the sender has to carefully split its bandwidth between original data and redundant data in order to maximize the quality of video received. If the amount of redundancy is larger than the amount of network loss, the stream tolerates loss. However, this comes at the cost of quality because less bandwidth is used for real content. If the amount of redundancy is too low, the effect of loss shows up in the stream and the quality degrades. This trade-off is clear in FEC(10,k) of Figure 2. Determining the optimal parameters for FEC is difficult and adapting it to changing network conditions is even more so.

A unique aspect of RS is that even though the actual redundancy at the sender is high, the network effectively reduces its cost. Therefore, the sender only has to only ensure that the amount of redundancy (r) is high enough to tolerate the loss and can worry less about its overhead. This makes RS simple to use, as it is simple to choose r.

Packet-by-packet redundancy control: RS introduces redundancy for each packet as opposed to groups of packets in FEC, and therefore the extent of redundancy can be controlled on a packet-by-packet basis. More important packets, e.g., those corresponding to I frames, could simply be sent more repeatedly than others to increase robustness. In essence, RS provides fine-grained unequal error protection (UEP) and enables the use of any of the proposed video encoding schemes that take advantage of UEP [33, 12, 40]. Thus, RS is simple to adapt to application-specific needs and data priorities.

Each encoded packet can be viewed as a signal to the network. Importance of the data is encoded in the number of encoded packets, $r-1$. When an original packet gets lost, routers try to resend the original packet when the signal arrives. As such the network tries up to r times until one original copy of the packet goes through.



(a) Sequence of packets sent by RS(r)

(b) Sequence of packets sent by RS(3) with d=2

Figure 3: Sequence of packets sent by RS

Decoupling of delay and redundancy: Unlike FEC, RS separates the redundancy decision from delay. In live communications, timing is crucial as data is useless after some deadline. FEC schemes closely tie timing with the encoding since the receiver cannot reconstruct lost packets until the batch is complete. In contrast, RS more easily accommodates application timing constraints. For example, sending 3 redundant packets spaced apart by 5 ms is essentially asking every router to retry up to 3 times every 5 ms to deliver the original packet. This signaling mechanism lends itself to application specific control to meet timing constraints. We explore the trade-off in controlling delay in Section 3.3.

3.2.3 Flow Prioritization

A unique property of RS-enabled traffic is that it gets preferential treatment over other traffic under lossy conditions. RS flows do not readily give up bandwidth as quickly as non-RS flows under loss. This is because for RS flows packet losses do not directly translate into less bandwidth use due to the "deflation" of redundant packets. Therefore, *RS flows are effectively prioritized in congested environments.* As a result, RS could get more than a fair-share at the bottleneck link especially in lossy conditions. We believe that this is a desirable property for providing stronger guarantees about the delivery rate of data, and analyze this effect in Section 6. However, this preferential treatment may not be always desirable. Section 4 provides an alternative solution that retains other benefits of RS without flow prioritization.

With this basic design, we now discuss details of RS including the scheduling of redundant packets and its interaction with video codecs.

3.3 Scheduling Redundant Packets

In this section, we discuss the transmission schedule used by RS to transmit data from a video encoder that outputs packets at a specified rate. Each original packet is transmitted as soon as it is generated by the video encoder. We use two parameters to control RS's transmission of redundant packets: redundancy (r) and delay (d).

Figure 3(a) shows the packet sequence of an RS(r) flow. $r - 1$ redundant packets are sent compressed in between two original packets. Thus, compared to a non-RS flow of the same bitrate, r times as many packets are sent by a RS(r) flow. In general, a large redundancy parameter r increases reliability at the cost of bandwidth overhead and packet processing overhead. Assuming

total bandwidth is fixed, if we increase redundancy, the amount of bandwidth left for original data is reduced. Also, the number of packets that a router processes increase.

The delay parameter specifies the number of original packets between two redundant packets that encode the same data. The first redundant packet of sequence number n is sent after the original packet of sequence number $(n+d)$. If the loss is temporally bursty, having a large interval between two redundant packets will help. However, extra delay incurs extra latency in recovering from a loss. So, delay (d) can be adjusted to meet the timing requirements of applications.

Figure 3(b) shows an example with $r = 3$ and $d = 2$. Three copies of packet k are sent, each spaced apart by two original packet transmissions. In Section 6, we evaluate the performance of RS and its sensitivity to parameter selection.

3.4 Interaction with Codecs

In this paper, we focus on improving real-time video delivery by improving loss recovery. An alternative approach that has been widely explored is to make video codecs more resilient to packet loss. Examples of designs that take this approach include layered video coding for multicast [36], multiple description source coding [39], ChitChat [45] and SoftCast [30]. Greater loss resilience does not come for free in these designs: These designs typically have lower compression rate than existing schemes because they must be able to reconstruct the video with arbitrary packet loss patterns. This ability to accommodate arbitrary loss also makes these schemes somewhat more computationally complex than existing approaches, which, in turn, makes them difficult to support on all devices. Also, some of these approaches do not accommodate arbitrary loss and focus on particular loss behavior [30]. We note that RS is agnostic to the choice of video codec and makes video communication robust regardless of the underlying codec. Of course, the exact video quality gains may differ based on the loss rates, loss patterns and codec used.

4 RS with Congestion Control

As we have discussed, enforcing a high sending rate is much easier with RS than FEC schemes. In some environments, fair bandwidth sharing may be more desirable than prioritizing RS flows. Ideally the sending rate should adapt to the network conditions to achieve a "fair-share". To achieve this goal, we use TCP friendly rate control (TFRC) [25].

However, applying TFRC to an RS flow raises some surprisingly subtle problems regarding the packet transmission rate and loss event rate estimation. We describe these challenges and our modifications to TFRC below.

Packet transmission: In TFRC for RS, we calculate the byte transmission rate from the equation just as the original TFRC. Note that RS(r) must send *original* packets and r duplicates of each packet. To match the byte sending rate, we adjust the length of the packet so that equal number of bytes are sent by TFRC RS as the original TFRC as shown in Figure 4 . As a result, in a TFRC RS flow, we send the original packets at the same rate as original TFRC, but also send r redundant packets in the time interval between the two original packets sent. Thus, given a computed send

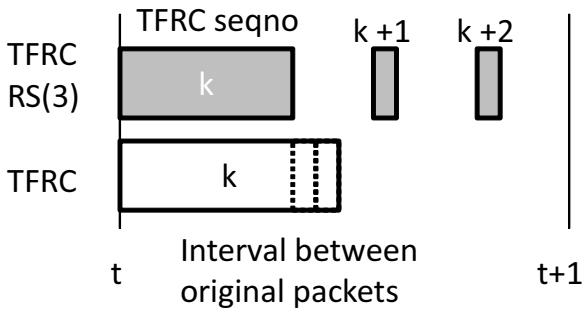


Figure 4: Packets sent by TFRC and TFRC RS(3): TFRC RS(3) sends equally as many bytes, but three times as many packets as TFRC given the same target rate.

rate, TFRC RS(r) sends r times as many packets. Note that each packet, original or duplicate, carries an individual sequence number and a timestamp for TFRC's rate calculation purposes.

Loss event rate estimation: In the original TFRC, the sending rate is calculated for the measured loss event rate p, where loss event rate is defined as an inverse of the average number of packets sent between two loss events. A loss event is a collection of packet drops (or other congestion signal) within a single RTT-length period.

Ideally, we would want TFRC RS(r) to have the same loss event rate as the original TFRC, as that would also make TFRC RS obtain a TCP friendly fair share of bandwidth. However, the observed loss event rate for RS depends on the underlying packet loss pattern. For ease of exposition, we look at the two extremes of loss patterns: one that is purely random and the other that is strictly temporally correlated.

Purely random packet drops may occur in a link of very high degree of multiplexing. In a strictly temporal loss pattern, on the other hand, losses occur during specific intervals. One might see such a loss pattern when cross traffic fills up a router queue at certain intervals. In reality, the two patterns appear in an inter-mixed fashion depending on source traffic sending patterns and other factors.

Next, we discuss the behavior of TFRC RS loss estimation in these extreme settings:

- *Uniform random packet loss:* In a random packet loss pattern setting, TFRC RS(r) behaves in a TCP friendly manner without any adjustment to loss estimation. This is because the inter-arrival between two loss events does not change even though the packet sending rates change.

- *Temporal packet loss:* In a temporal packet loss pattern setting, packets are lost at specific times. During the time interval between loss, TFRC RS(r) sends r times as many packets. Thus, the observed loss event rate p for TFRC RS(r) is only $\frac{1}{r}$ of that of the original TFRC.

Adjusting the loss event rate: As stated earlier, in practice, the two extreme patterns appear in an intermixed fashion. We therefore want to choose an adjustment factor α so that when the loss event rate is adjusted to α times the measured loss event rate p, TFRC RS(r) is TCP friendly.

As seen in the two extreme cases, α has values 1 for uniform random losses and r for temporal losses, respectively. So, in practice, α should be between 1 and r, which means α can be r times off from the correct value in the worst case. Even in this wost case, since the TCP-friendly rate is inversely proportional to $\sqrt{loss}$, a TFRC-RS flow would have performance similar to $\sqrt{r}$ many TCP or TFRC flows. Therefore, even if an incorrect value of α is used, TFRC RS(r) flow would be still friendly to a group of TCP connections. In practice, we find in Section 6 that TFRC-RS with $r = 3$ and $\alpha = 1.5$ closely approximates the bandwidth share of a single TCP flow under wide range of loss rates and realistic loss patterns.

5 RPT in various Networks

So far, we have explored Redundant Streaming on hop-by-hop RE networks as a special case of redundant packet transmission (RPT). In this section, we look at more realistic deployment scenarios and explore RPT in other content-aware networks.

5.1 Incremental Deployment

Corporate Networks: WAN optimizer is the most popular form of RE technology's deployment. Market reports indicate that its market is growing rapidly [5], and many vendors including Riverbed, Cisco and Juniper networks have such products [1, 2, 11]. In a typical deployment, these devices are placed at each branch and main offices to reduce the traffic between them. It has been deployed within corporate networks of 58+ businesses by Riverbed alone [10], and Cisco deployed its product in more than 200 Cisco's remote offices worldwide [3].

These branch offices are commonly connected with limited capacity by dedicated wires or VPN-enabled "virtual wires". ISPs offering VPN services typically state bandwidth, latency and loss requirements in their SLA, and grantee a fixed bandwidth and negligible loss rate [7].

The use of VPN and WAN optimizers effectively creates Redundancy Elimination "tunnels" on which RPT can operate. Important, real-time data can be sent with redundancy, and will compete with less important background traffic for bandwidth when entering this tunnel. Packets will be lost when the total demand exceeds the capacity of the tunnel, but important RPT flows will have protection against loss.

An alternatively scheme is to use traditional QoS schemes such as priority queuing within a corporate's virtual network. However, this typically involves in extra management/facilities including dynamic resource allocation and admission control. For businesses not willing to maintain such infrastructure, using RPT on existing RE-enabled VPN would be an excellent alternative for delivering important and time-critical data stream. .

Partially Content-Aware Networks: Not all routers have to be content-aware to make RPT work. The only requirement is that RE is deployed across links with relatively low bandwidth; this matches the common deployment scenario for RE that prior works propose [44, 14]. One concern may be that non-RE links end up carrying several duplicate packets. However, due to the higher

capacity of these links, this poses no issue. We evaluate the bandwidth use of RPT on non-RE links in Section 6.1.

Detection: In reality, it is hard to ensure that RE is deployed on all bandwidth-constrained links as traffic demand is dynamic in nature. To ensure safe operation of RPT, we can detect the presence of RE on bandwidth-constrained links, and use RPT only when it would not harm other traffic and otherwise fall back to FEC. We outline possible approaches here, and leave details as a future work. One way is to use end-point based measurement. For example, Pathneck [27] allows to detect bottleneck based on available bandwidth. It sends traceroute packets in between load packets and infers (multiple) bottleneck location(s) from the time gap between returned ICMP packets. We can send two separate packet trains: one with no redundancy and the other with redundant factor of r (with r as many packets). If all bandwidth constrained links are RE-enabled and RPT is safe to use on other links, packet gap would not inflate on bottleneck links detected by Pathneck and the redundant packet trains would not report additional bottlenecks. Another way is to use network-level information. I-plane [35] measures and exposes link/path attributes such as available bandwidth in a scalable manner. I-path [37] allows end-hosts to directly query routers for such information. Additional link attributes such as RE-functionality can be augmented to such systems for end-hosts to check RPT safety.

5.2 RPT in other Content-aware Networks

RPT also can be integrated with a broad class of content-aware networks. This section discusses various forms of such networks and how RPT can be implemented in them. We discuss partially content-aware networks, CCN [29], SmartRE [15] and content-aware networks with lossy links.

Content Centric networks: In CCN [29], the data consumers send Interest packets, and the network responds with at most one Data packet for each Interest. Inside the network, each router caches content and eliminates the duplicate transfers of the same content over any link. CCN is designed to operate on top of unreliable packet delivery service, and thus Interest and Data packets may be lost [29]. To provide reliability, consumers retransmit Interests after a timeout.

Let's consider voice-over-CCN as an example [28]. Instead of the delay-prone, timeout-driven retransmission, redundancy-based recovery can be used. In an equivalent FEC scheme in CCN, the content publisher would encode k data packets obtained from the voice codec and add $(n-k)$ coded data packets, where $n > k$. The data consumer would then generate n Interest packets. The receiver will be able to fully decode the data when more than k Interest packets go through. However, up to n Data packets will be delivered to the receiver. In contrast, RPT would not code data packets, but generate redundant Interest packets. This obviously provides robustness against Interest packet loss. Also when a Data packet is lost, subsequent redundant Interest packet will re-initiate the Data transfer. Since Interest packets are small compared to Data packets the bandwidth cost of redundancy is minimal. Also, duplicate Interests do not result in duplicate transfers of the Data packet. As a result, at most k Data packets will be transferred instead of n in the FEC scheme.

SmartRE networks: In SmartRE [15] networks, each router only decodes a subset of packets instead of decoding every packet at every hop to reduce the computational cost. A centralized element in the ISP's network decides which router is responsible for decoding which set of packets. This coordinated RE approach can create a problem for RPT in the following way. Suppose that an encoder E receives an original packet A, caches it and forwards it along. Assume A is dropped before reaching its designated decoder D, which is several hops away from E. A redundant packet A' arrives at E and is encoded with respect to A; however, it cannot be decoded as D does not have bytes for the original A.

This can be remedied using one of two approaches: (1) At flow initialization time, an RPT flow can negotiate with the centralized element to do hop-by-hop RE for the flow. This approach is suitable in an environment where sending RPT flows is a privileged operation and the network operator wants to do admission control on RPT flows. (2) When an uncompressed packet arrives at some router R and the output queue for the packet is full, the packet is inserted into the local RE cache at R, where R is a router between the encoder E and decoder D for the packet. When a compressed packet arrives, each router checks if it is referencing a dropped packet in its local RE cache. If it is, then the router decompresses the packet and removes the original packet from the local RE cache. The extra cache space required at each router for this solution is minimal compared to the already existing RE cache because only a fraction of packets get dropped, and the dropped packet needs to be cached for maximum delay bound of a real-time communication (less than 200ms).

Wireless links RE networks with lossy links (i.e. from link-layer corruption) are the most tricky scenario to operate RPT in. Caches at the two ends of a lossy link will get de-synchronized and, as a result, RPT cannot recover from losses. However, in most such situations, link loss can be detected by link-layer acknowledgments. In this case, RE can use this packet loss notification to resynchronize the cache contents at the two ends to the link, which, in turn, makes RPT work normally.

Although, this is a viable alternative, there are many systems, such as SoftCast [30] and Medusa [43], that are specialized for wireless streaming. These systems take advantage of the fact that wireless links appear only on first and/or last hop link in most cases, and tightly integrate the system with wireless end-points and exert more control to deliver high performance and robustness against loss. We believe such systems are more suitable in wireless environments.

6 Evaluation

In this section, we evaluate various aspects of redundant packet transmission (RPT) using the specific instantiation of RS. We examine the end-to-end performance benefits of RPT through the video quality achieved by RS flows. We also carefully examine the side-effects to the network that RPT may cause and how RPT affects the performance of other competing flows. To better understand RPT, we answer the following questions:
(*i*) Does RS deliver better the video quality? How well does it work in practice? (Section 6.1)

Quality	Excellent	Good	Fair	Poor	Bad
PSNR (dB)	> 37	$31 \sim 37$	$25 \sim 31$	$20 \sim 25$	< 20

Table 1: Conversion between PSNR and user perception

(*ii*) Is RS sensitive to its parameter setting? Does RS or RPT require fine tuning of parameters to get it work? (Section 6.2)
(*iii*) How do RPT flows affect other flows and the overall network behavior? (Section 6.3)
(*iv*) Can we make RPT flows adapt to network conditions and be TCP-friendly? (Section 6.4)

First, we discuss our framework and the metric used for performance evaluation.

Evaluation framework: We use a combination of real-world experiments, network measurements and simulations. To this end, we implemented a RE encoder and decoder using the Click modular router [32], and we created a router similar to that of Figure 1. Using this implementation, we create a testbed of a hop-by-hop RE network in our lab, which serves as our evaluation framework.

We use implementation-based evaluation to show the overall end-to-end performance of RS, and simulations to unravel the details and observe how it interacts with other cross traffic. To obtain realistic packet traces and loss patterns from highly multiplexed networks, we also performed active measurements to collect real-world Internet packet traces. We also created background traffic and simulated RS and FEC flows in a hop-by-hop RE network using the ns-2 simulator. These video flow packet traces are then fed into *evalid* video performance evaluation tool [31] to obtain the received video sequence with loss. For the video, *football* video sequence in CIF format was used from a well known video library [6]. We use H.264 encoding using x264. I-frames were inserted every second and only I-frame and P-frames were used to model live streams.

RE implementation: Our RE implementation implements the max-match algorithm described in [14]. It is further modified to only store non-redundant packets in the packet cache, and not store the fully redundant packets. Therefore, sending multiple redundant packets do not interfere with other cached content. The implementation of the encoder encodes a 1500 byte fully redundant UDP packet to a 43 byte packet. It does not compress network and transport layer headers. Note that the redundant packets may be compressed even further, if we had used header compression. Apart from the Click-based implementation, we also implemented encoder and decoder modules that emulate RE in ns-2.

Evaluation metric: We use the standard Peak-to-Signal-to-Noise Ratio (PSNR) [42] as the metric for the video quality. PSNR is defined using a logarithmic unit of dB, and therefore small difference in PSNR results in visually noticeable difference in the quality of the video. The MPEG committee reportedly uses a threshold of PSNR = 0.5dB to decide whether new coding optimizations should be incorporated [42]. Typical values for PSNR for encoded video are between 30 and 50 dB. Table 1 maps the PSNR value to a user perceived video quality [31].

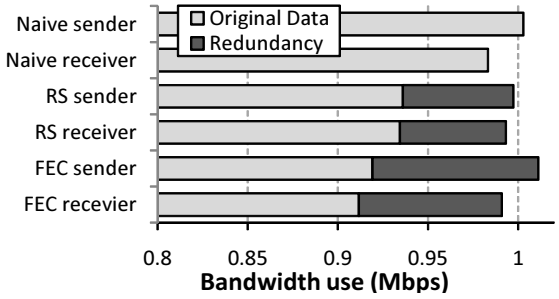
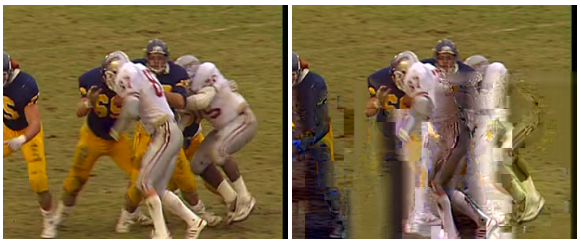


Figure 5: Bottleneck bandwidth use



(a) RS flow (b) Naive flow

Figure 6: Snapshot of the video

6.1 End-to-end Video Performance

In this section, we evaluate the end-to-end performance of RS and examine key characteristics.

Experimental testbed: First, we use our testbed based on our hop-by-hop RE router implementation, and create a video streaming application that uses redundant streaming. We created a dumbbell topology where an RE router in the middle connects two networks, one at 100Mbps and the other at 10Mbps. To create loss, we generate traffic from the well-connected network to the 10Mbps bottleneck link. When the network is congested, packets are dropped at the output queue of the router.

We generated a 1Mbps UDP video stream and long-running TCP flows as background traffic. We adjust the background traffic to create a 2% loss on the video flow. We then compare the video quality achieved by RS, Naive, and FEC senders that use the same amount of bandwidth. For the RS sender, we use a redundancy factor (r) of 3 with a delay parameter of 2, which has 6% of overhead. For FEC, we used FEC(10,9) coding which has 10% of overhead. FEC(10,9) was chosen to meet the latency constraint as well as to closely match the overhead of RS. The naive sender sends UDP packets without any protection.

Figure 5 shows the sending rate and the received data rate after the loss. The RS and FEC senders respectively use about 6% and 10% of their bandwidth to send redundant packets, while the Naive sender uses the full 1Mbps to send original data. The sending rates of the RS, Naive, and FEC senders are virtually the same, within a small margin of error (1%). The Naive receiver loses 2% of the data and receives 0.98Mbps because of the loss. The FEC receiver also receives 0.98Mbps, but recovers about 66% of the loss because of the bursty loss. On the other hand, the RS receiver receives virtually the same amount of original data. Note that only the amount of redundancy has slightly decreased. This is because when an original packet is lost, a subsequent redundant packet is naturally expanded inside the network.

As a result, the RS flow gives much higher video quality. Figure 6 shows an example snapshot of the video for RS and Naive flows. Table 2 shows the video quality of an encoded video and the received video. Encoded video column shows the quality of video generated at the sender before packet loss. When RS and FEC are used, the encoded video quality is slightly lower because of the bandwidth used towards redundancy. **However, because the RS flow is highly robust against**

	Encoded	Received
RS	37.3 dB	**37.1 dB**
FEC	36.9 dB	**35.3 dB**
Naive	37.5 dB	**31.4 dB**

Table 2: Average PSNR of videos

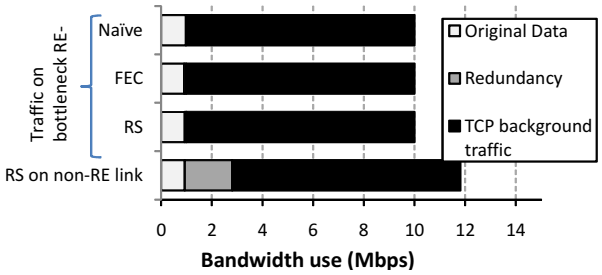


Figure 7: Bandwidth use on non-RE links.

loss, it provides the best video streaming quality (1.8 dB better than FEC and almost 6dB better than the unprotected Naive).

Partially content-aware networks: In Section 5.1, we mentioned RPT poses no harm when all bandwidth constrained links are RE-enabled, and other links are well provisioned, RPT should be only used in such case. To demonstrate such a case and to quantify RPT's overhead on a non-RE link, we disabled the RE engine on the non-bottleneck 100Mbps links of our testbed. Figure 7 shows the bandwidth use on both 100Mbps and 10Mbps (bottleneck) links. An RS flow occupying 1Mbps on RE-enabled bottleneck link almost has 2Mbps of overhead on the non-RE 100Mbps link. However, this cause no harm since the 100Mbps link is not bandwidth constrained.

RE-enabled Corporate VPN of Section 5.1 is the most common deployment scenario of RPT in today's network. To demonstrate the feasibility of this, we emulated a RE-enabled VPN tunnel with bandwidth and loss grantees in Emulab [23] using our implementation and Linux kernel's priority queue. We set up a network of four routers and isolate the traffic coming from a branch office. We generated cross traffic over links that carries the VPN traffic so that they experience congestion. The tunnel capacity was set to be 5Mbps, and we introduced 1Mbps real-time traffic and 5 TCP connections over the link. We use RPT and FEC(10,9) whose overhead is similar. The real-time traffic was experiencing around 2.5% loss rate and tunnel's link utilization was nearly 100% in both cases. PSNR of the RPT flow and FEC was 37.3dB and 34.1dB respectively. We can see that RPT works better in this environment compared to traditional loss protection.

Real traces: To study the performance of RS in a wide-area setting under various loss conditions, we collected a real-world packet trace on the Internet. We generated UDP packets from a university in Korea to a wired host connected to a residential Internet service in Pittsburgh, PA. The sending rate was varied from 1Mbps to 10Mbps, each run lasting at least 30 seconds.

Assuming that the packet loss patterns do not change with RS[2], we apply the loss pattern obtained from the measurement to an RS flow, an FEC flow and a Naive UDP flow. For RS, we use a redundancy parameter of $r = 3$ and delay parameter $d = 2$. For FEC, we choose the parameter so that the overhead matches closest to that of RS, while the additional latency incurred by FEC at the receiver do not exceed 150ms. Figure 8 shows the video quality for each scheme. The solid line shows the packet loss rate. The encoded video bar represents the ideal PSNR without any

[2]We later verify this and see how RS affects the loss rate in Section 6.3.

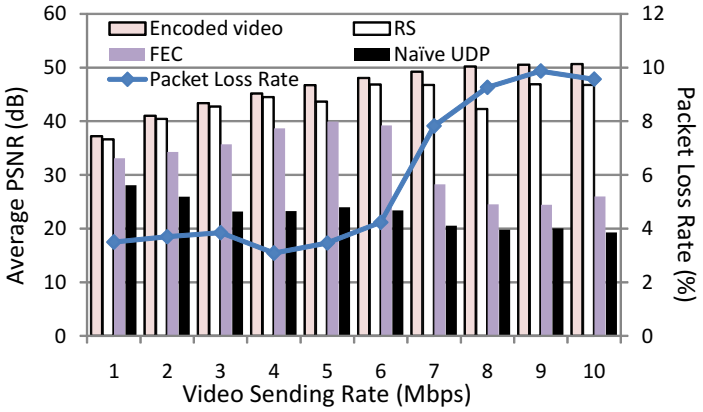


Figure 8: Video quality and loss rate for real traces

loss when all the bandwidth is used towards sending original data, and is presented as a reference. As the sending rate increases, the quality of the encoded video increases. However, the loss rate from Korea to U.S. was 3.5% at 1Mbps but increased to 9.8% at 10 Mbps as the sending rate was increased. Data loss with RS was between 0 and 0.2%, and the data loss with FEC was 1.4% to 5.1% [3]. However, because of the increased loss rate, the naive UDP sender performs poorly (PSNR well under 30dB). FEC achieves better performance, but its performance is only acceptable under low packet loss, and it gives marginal benefit as loss rate increases. In contrast, **RS gives the best performance, closely following the quality of the encoded video when the loss rate is below 8%. Even at the higher loss rates, the impact on quality is much less than the FEC scheme.** This is due to the fact that RS is highly robust to a wide range of packet losses, and gives much better protection against loss than FEC at similar overhead.

6.2 Parameter Selection and Sensitivity

In this section, we provide an in-depth performance evaluation of RS. In particular, we compare RS and FEC's parameter sensitivity using simulations that produce packet loss patterns of highly multiplexed networks with realistic cross traffic. We vary the amount of loss and the fraction of RS flows.

Simulated RE Network: We use the ns-2 simulator to create a realistic loss pattern by generating a mix of HTTP and long-running TCP cross traffic. We use a dumbbell topology with the bottleneck link capacity set to 100Mbps, and simulate a hop-by-hop RE network and RS flows. RS traffic flows from senders through routers to receivers each of which has a RE decoder and encoder. We generate 100 long-running TCP flows and 100 HTTP requests per second. We used the packmime [21] module to generate HTTP traffic. Packmime generates representative HTTP traffic by modeling the size of HTTP requests, the size of response messages and the inter-arrival time between requests. We also generate 10 video flows each using 1Mbps of bandwidth. Video

[3]Data loss is defined as the fraction of data not recovered.

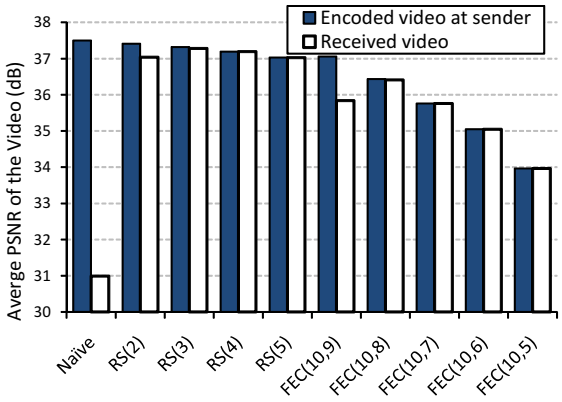


Figure 9: Sensitivity to parameter: RS' performance is not sensitive, but FEC's performance is quite sensitive to its parameter setting.

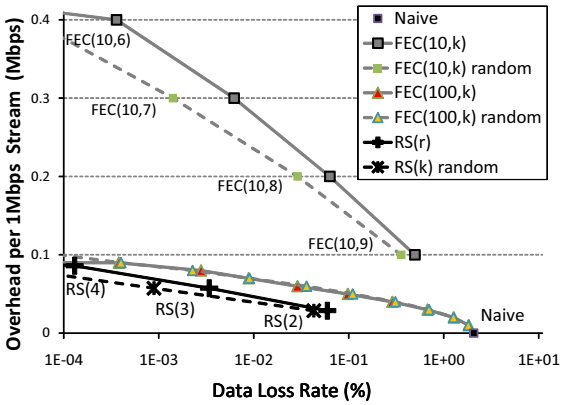


Figure 10: Data loss and overhead: RS flows perform much better than FEC flows with small group size. FEC flows with small group are susceptible to bursty loss.

flows are sent with RTP header encapsulated in a UDP packet. The results presented are averages of 10 simulation runs with each simulating 5 minutes of the network traffic. We first look at the final video quality seen by the end receiver under different parameter settings, and then analyze the underlying loss rate and overhead.

How do RS flows and FEC flows perform with different redundancy parameters? For RS, we vary the redundancy parameter r from 2 to 4, while fixing the delay parameter d to 2. For FEC, we use a group size $n = 10$ and vary the number of original data packets k from 5 to 9. We use a group size of 10 to meet the latency constraints. Figure 9 shows the quality of the video seen by the receiver compared to the encoded quality at the sender.

The quality of encoded video decreases as the overhead of redundancy is increased. For example, the Naive sender fully uses the 1Mbps of bandwidth to encode the video, while the FEC(10,5) sender only uses half the bandwidth to encoded the video and uses the other half to encode redundant information. The gap between the encoded video and the received video within the same scheme represents the effect of loss. Since the Naive sender does not have any protection against loss, the resulting video quality is severely degraded from the original video.

The result shows that **RS flows perform better than any FEC flows regardless of the parameter, and their performance is stable across different parameter settings. In contrast, FEC's performance is sensitive to the parameter selection.** With FEC, the sender has to carefully tune the parameter to balance the amount of redundancy and encoding rate to achieve higher quality.

Figure 10 shows the underlying data loss rate and overhead of the video flows in Figure 9. The horizontal axis shows the data loss rate in log-scale, and the vertical axis shows the amount of overhead in the 1Mbps stream. For comparison, the performance of RS and two FEC families (n=10 and n=100) under 2% uniform random loss (dotted lines) from Figure 2 is also shown. The Naive sender does not incur any overhead, but loss rate is high: 2%. RS(4)'s data loss rate is several orders of magnitude lower than the loss rate of FEC(10,9) whose overhead is similar. RS(4) even

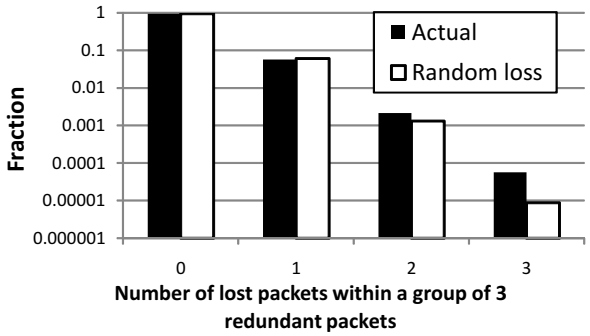


Figure 11: Pattern of loss: Small bursts of lost packets occur more commonly.

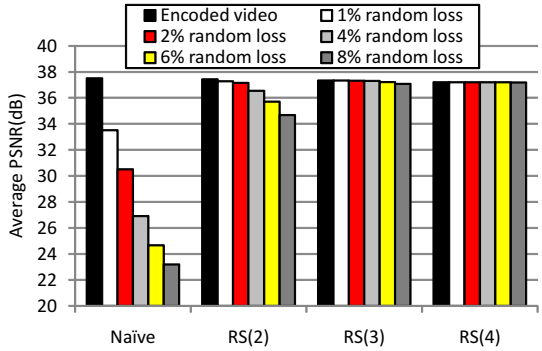


Figure 12: Video quality under 1 to 8% loss. RS(3) steadily delivers high quality video even under high loss.

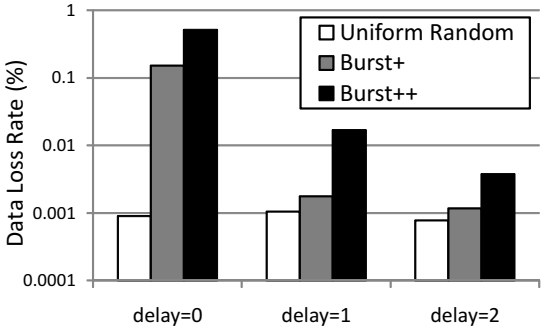


Figure 13: Bursty loss causes increase the data loss at the receiver especially when the delay parameter is small.

performs better than FECs with large group size, such as FEC(100,91), whose latency exceeds the real-time constraint. RS(3) and FEC(10,7) achieves similar data loss rate but FEC(10,7) has 6 times higher overhead. Therefore, the average PSNR (video quality) of RS(3) was 2 dB greater than that of FEC(10,7) (Figure 9).

The gap between the uniform loss and actual loss lines in Figure 10 represents the effect of bursty loss performance. FEC(10,k) and RS shows a relatively large gap between the two lines. Analyzing the underlying loss pattern, we observe that within a group of 10 packets, losses of 2 to 4 packets appear more frequently in the actual loss pattern. This shows that the underlying traffic is bursty. On the other hand, loss bursts of more than 5 occur less frequently in the actual pattern. This is because TCP congestion control kicks in when packets are dropped. Figure 11 shows the frequency of multiple redundant packet losses in RS(3). When all three packets are lost, the data is lost. It shows that three packet losses appear more frequently in the actual loss pattern, which explains the gap in Figure 10.

We now show how the parameter should be set in RS under various loss conditions.

How should we choose parameters in RS? To answer this question, we study how loss rate and burstiness of loss affect the performance of RS. First, we use random loss pattern and vary the packet loss rate from 1% to 8%. For RS, we vary the redundancy parameter from 2 to 4, but fix the

	FEC(10,6)	FEC(100,92)	RS(3)
Original method	180ms	1.3sec	60ms
Alternative method	240ms	2.4sec	-

Table 3: Maximum one-way delay caused by buffering

delay parameter at 2. For each loss rate, the average PSNR of a naive sender and an RS sender is shown in Figure 12. As more redundancy is added, average PSNR of the encoded video decreases but the decrease is small due to the small cost of extra redundancy in RS (3%). The result shows that video quality for RS flows with redundancy greater than 3 is virtually immune to wide range of packet losses; the average PSNR for RS(3) under 8% loss only decreased by 0.25dB compared to the zero-loss case. The results show that using $r = 3$ is good enough for most cases. We use $r = 3$ for the rest of our evaluation, unless otherwise specified.

Second, we study the affect of bursty loss on the performance of RS. Specifically, we look at the role of the delay parameter. For reference, we generate a 2% random loss pattern, which on average has 1 lost packet every 50 packets. We then create bursty loss patterns by reducing the number of packets between losses by up to 15 and 35, while keeping the average loss rate the same. The three cases are named as `Uniform random`, `Bursty loss+`, and `Bursty loss++` respectively. Figure 13 shows the data loss rate of RS with different delay parameters. The result shows that an increase in burstiness negatively impacts the data loss rate of RS, but as delay is increased the effect of burstiness is decreased. In general, a large delay parameter gives more protection against bursty loss, but incurs additional latency. **We use $d = 2$ and $r = 3$ for RS because of its superior performance in wide range of loss conditions** in the rest of our evaluations. We now look at the latency aspect of RS.

How much latency is caused by RS and FEC flows? Because a loss might be recovered by subsequent redundant packets, the receiver must buffer packets to pass a sequential stream to the decoder. Here, we quantify the delay that this buffering causes.

In RS(r) with delay d, the receiver buffer must hold $d \cdot r$ packets. So the delay in RS is $d \cdot r \cdot intv$, where $intv$ is the interval between packets. RS sender needs no additional buffering, as it transmits the packet as soon as a packet is generated from the encoder.

In FEC(n,k), the receiver buffer must hold n packets, and incurs a delay of $n \cdot intv$. The FEC sender may also need to buffer unlike the RS sender. In FEC(n,k), k original packets are followed by $(n-k)$ coded packets. Assume that each original packet is a separate video frame, for simplicity. In a given interval, the sender wants to send out n packets spaced out evenly, but the CBR encoder only generates k frames (packets). The sender can only generate the coded packets at the end of the interval, but sending packets in a burst make them susceptible to bursty loss. One way is to delay the transmission of the original packets at the sender. The delay d ensures that the original packet is always available before the transmission time of the packet. This can be represented as $\forall i \leq k, \frac{i}{k} \cdot intv \cdot n \leq d + \frac{i-1}{n} \cdot intv \cdot n$. The minimum value of d is $(n - k + 1) \cdot intv$. The total one-way delay is therefore $(2n - k + 1) \cdot intv$. Alternatively, the FEC sender can send original packets right away, but send the coded redundant packets with the next group of original packets.

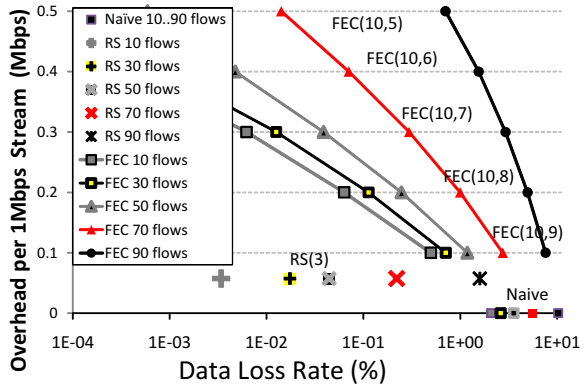


Figure 14: Data loss and overhead: RS flows exhibit much less data loss rate with small overhead even under extreme load and large amount of RS flows.

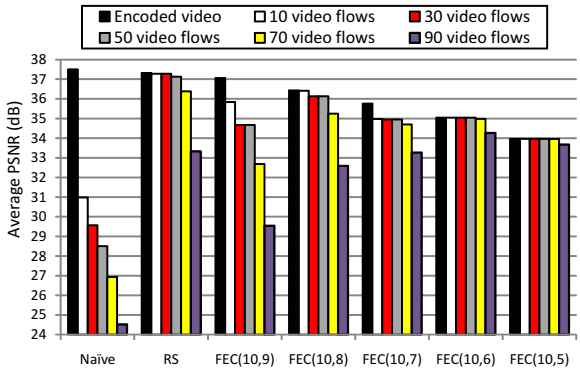


Figure 15: Comparison of Video Quality: RS delivers best performance in most cases.

However, this requires a buffer size of $2n$ packets at the receiver.

Table 3 shows the additional delay caused by the buffer for 1Mbps RS(3) with $d = 2$ and FEC streams that exhibit similar data loss rate from Figure 10. We see that **RS gives a significantly lower delay than FEC schemes with similar strength in loss protection**, and FEC(100,k) is not suitable for real-time communication.

How do RS and FEC flows perform under high load? One might think that in a highly congested link with a high fraction of RS traffic, RS flows would constantly overflow the buffer and the performance would drop. To create such a scenario, we vary the fraction of video traffic in the link from 10% to 90%, while keeping amount of cross traffic and bottleneck bandwidth the same. To vary the fraction of video traffic, we increased the number of 1Mbps video flows from 10 (10%) to 90 (90%). As the number of video flows increases, the network load also increases.

Figure 14 shows the data loss rate and the overhead of RS and FEC. Even when the fraction of RS traffic is high, we see that RS achieves low data loss rate with low overhead compared to FEC. Figure 15 shows the resulting video quality. **RS flows achieve close-to-ideal video quality in most of the cases, and better quality compared to the best FEC scheme except for the extreme case of 90%.** Note that FEC's performance depends heavily on its parameter setting. The performance of RS only degrades in the very extreme condition. Two reasons combined produce high data loss in the 90 video flows case: 1) The underlying packet loss rate is more than 10%. 2) RS flows occupy 90% of the bottleneck link. For an RS flow to recover all data loss under congestion, packets from other flows have to be dropped to give room for subsequent transmission of the original packet. However, in this case, since most packets are from RS flows, packets from other RS flows gets dropped to mask loss from an RS flow, and they often end up competing with each other. However, the extreme case we portray in our experiment is unlikely to occur in practice for two reasons: 1) The loss rates in practice are likely to be much lower. 2) Moreover, even the aggressive estimate suggests that no more than 15% of traffic in future will be real-time in nature [4].

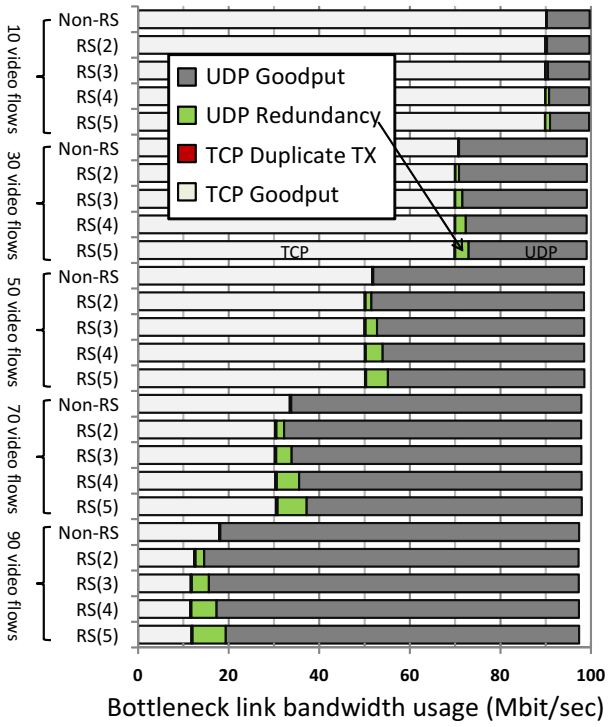


Figure 16: Breakdown on bottleneck link utilization: Link utilization do not change with RS flows and RS flows are prioritized over competing TCP flows.

So far, our results show RS incurs lower overhead, gives far better protection against packet loss, and is far less sensitive to parameters than FEC. Next, we discuss the impact of RS on the other traffic and the network.

6.3 Impact of RS on the network

We examine the effect of RS flows on other cross traffic and the network. For this evaluation, we use the same simulation setup and topology described in Section 6.2.

How do other TCP flows perform? We look at the impact on two different types of TCP flows: long-running TCP flows and HTTP-like short TCP flows.

Impact on long-running TCP flows: To evaluate the effect on long-running TCP flows, we send 100 long-running TCP flows and a varying number of video flows. We also vary the redundancy parameter from 0 (Non-RS) to 5. We used a small router buffer size of $\frac{2 \cdot B \cdot RTT}{\sqrt{100}}$ [16] to maximize the negative impact.

We decomposed the bottleneck link bandwidth use into four categories: UDP goodput, UDP redundancy, TCP duplicate, TCP goodput. UDP goodput is the bandwidth occupied by the original decompressed packet and the UDP redundancy represents the bandwidth occupied by the compressed packets. TCP goodput represents packets contributing to application throughput, and

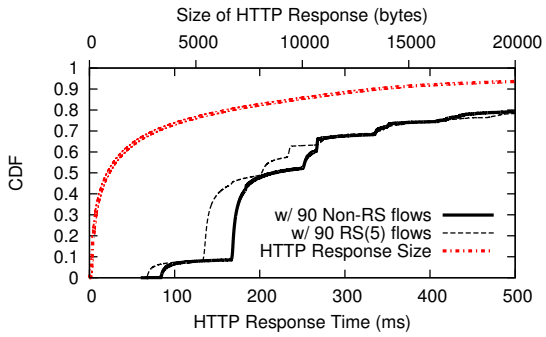


Figure 17: HTTP response size and the response time.

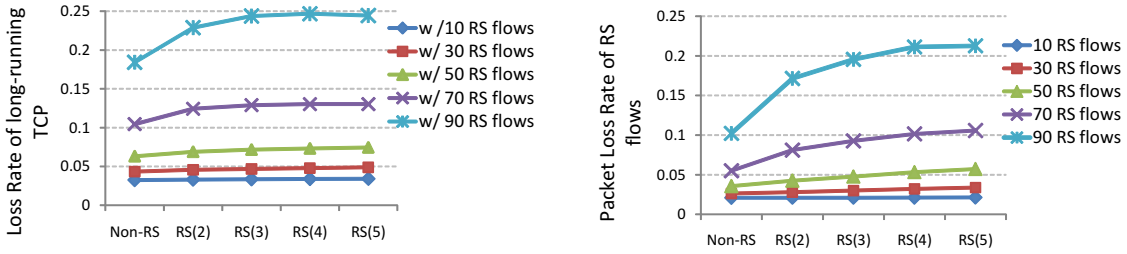


Figure 18: Impact on network loss rate due to RS flows: (a) TCP packet loss rate for long running TCP flows and (b) UDP packet loss rate for video flows. HTTP cross traffic is present for both cases.

TCP duplicate Tx shows the amount retransmission observed at the receiver. Figure 16 shows the decomposition of bottleneck link with varying number of video flows and redundancy.

From the figure, we observe that **1) bandwidth utilization efficiency is not affected by RS, and 2) RS flows are effectively prioritized over non-RS TCP flows.**

In all cases the bottleneck bandwidth utilization was over 97.5%; TCP is able to fill up the bottleneck bandwidth even if the router queue is occupied by many decompressed redundant packets.

TCP throughput, on the other hand is impacted by the RS flow especially when the underlying loss rate is high. For example, with 90 video flows, a non-RS UDP flow experiences 10% loss and the loss directly translates into additional TCP throughput. RS flows also experience high packet loss, however, packet loss does not directly translate into throughput loss. This is because the network recovers the loss through subsequent uncompressed redundant packets being sent out when an original packet is lost, effectively prioritizing RS flows. In other words, RS flows give up less bandwidth than non-RS flows for the same loss.

Impact on HTTP-type short flows: To see the impact of RS on HTTP request/response type short TCP flows, we generate HTTP traffic with long-running TCP and video flows. HTTP traffic is modeled by the packmime [21] HTTP traffic generation module. Figure 17 shows the CDFs of the size of the response messages, and the response times. The distribution of HTTP response size is heavy tailed. To highlight the difference, we only show response times with 90 Non-RS flows and RS(5) flows, but the trend is visible across all RS flows. We only look at the short responses to see the effect on short TCP flows.

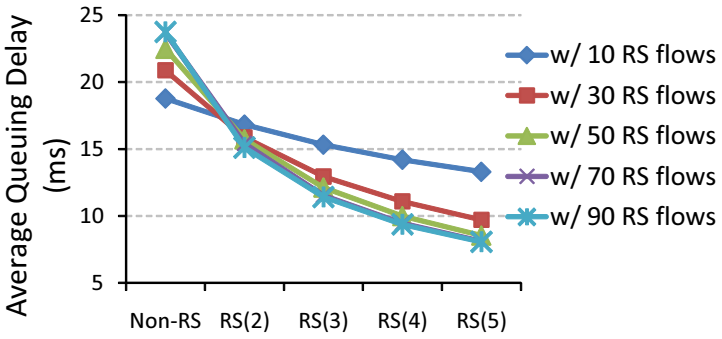


Figure 19: Queuing delay is reduced with RS flows.

The result shows that **the response time for short HTTP messages decreases in the presence of RS flows**. The service rate of the queue increases with RS flows, because redundant packets in the queue are encoded to small packets when they are sent out. Therefore the queuing delay is reduced resulting in a decrease in the response time. However, for large HTTP responses, the response time is increased. As the response messages grow in size, they behave more like long-running TCP flows, which obtain less throughput in the presence of RS flows under congestion.

How does network behavior change with RS flows? Previously, we saw that even with highly redundant traffic, the network does not break down and is able to fully utilize the bottleneck bandwidth. However, there are subtle changes in the behavior of the network. We look at the changes in two different aspects: loss rate and the queuing delay.

Loss rate: Figures 18(a) and (b) respectively show the TCP and UDP packet loss rate at the bottleneck router in the presence of different amount of RS flows. UDP packets represent packets sent by either RS or non-RS video flows.

First, regardless of RS, we observe that TCP loss rate is generally higher than UDP loss rate. This is due to TCP's saw-tooth behavior. Loss usually happens when TCP's congestion window size is large. During this time, the ratio of TCP packets to UDP packets is higher than average as UDP flows are constant bitrate. This is confirmed by the high correlation score (0.78) between the number of incoming TCP packets and the number of lost packets within small intervals.

Second, TCP loss rate increases as redundancy is increased. RS flows get more bandwidth share as redundancy is increased, and the throughput of TCP flows is decreased (See Figure 16). This makes the loss rate go up, which becomes more evident in high loss environments. For example, with 90 RS flows, the difference in TCP throughput from Non-RS to RS is higher in Figure 16. The larger gap in throughput shows up as a larger gap in TCP loss rate.

Third, **when the amount of video traffic is moderate, adding more redundancy has little impact on the underlying packet loss of the video flow.** This is because while redundant packets increase the load, they also increase the service rate of the link. However, we observe that when RS flows dominate the bottleneck link, the underlying loss rate for video flows goes up as redundancy increases. The underlying reason for increased loss is that under congestion RS flows compete with each other for bandwidth when most of the traffic is from RS flows as explained earlier.

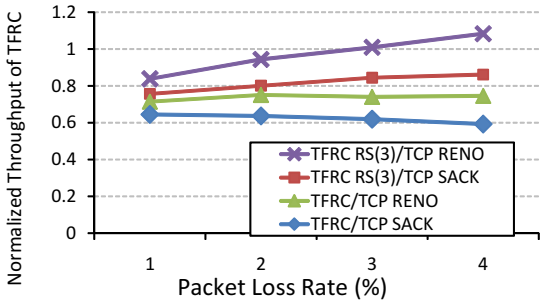


Figure 20: TFRC RS under random loss shows TCP friendliness.

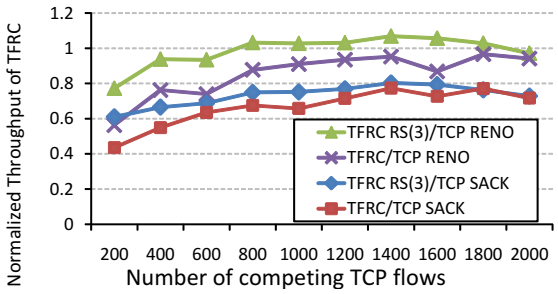


Figure 21: TFRC RS flows exhibit TCP friendliness.

Queuing Delay: We look at how the queuing delay is changed with RS. Figure 19 shows the average queuing delay with varying redundancy parameters and varying number of RS flows. **In the presence of redundant packets, queuing delay is decreased.** This is due to the increased service rate of the queue. Redundant packets appear as decompressed at the router queue but are sent out compressed at the bottleneck link. Therefore, as the number of redundant packets increase, service rate becomes faster.

6.4 TCP Friendly RS

RS flows do not give up bandwidth as easily as as other flows under congestion. In Section 4, we discussed an alternative that makes RS flows achieve fair bandwidth sharing using TCP-friendly rate control. In particular, we showed that incorporating TFRC requires careful adjustment of loss event rate, and explained how it should be done in two distinct loss patterns: *Uniform random* and *Temporal* packet loss.

Is TFRC RS TCP-friendly? We first evaluate our scheme under the two extreme loss patterns created artificially, and then evaluate it under a more realistic loss pattern.

Uniform random loss: In a random loss pattern, TFRC RS(3) behaves in a TCP friendly manner without any adjustment in the loss estimation. Figure 20 shows the normalized throughput of TFRC and TFRC RS(3) with respect to TCP Sack and TCP Reno under 1% to 4% random loss. TFRC RS(3) performs slightly better than TFRC because multiple packet losses within an RTT are counted as one loss event, and therefore loss event rate for RS(3) is slightly lower than that of normal TFRC.

Temporal packet loss: In temporal loss conditions, we adjust the loss event rate of TFRC RS(r) to be r times the observed loss event rate. To validate TCP-friendliness of TFRC RS, we evaluated the performance of TFRC RS(3) and TFRC under the same cross traffic with the same temporal loss pattern. To create the same temporal loss pattern on TFRC RS(3) and TFRC, we artificially modified the router's queue so that redundant packets does not increase the queue length. Indeed, the performance difference of the two was less than 3% with the adjusted loss event rate for RS(3).

Realistic environment: The two cases appear in an inter-mixed way in practice. As discussed in Section 3, an adjustment factor between 1 and r is sufficient for TCP friendliness. To create a

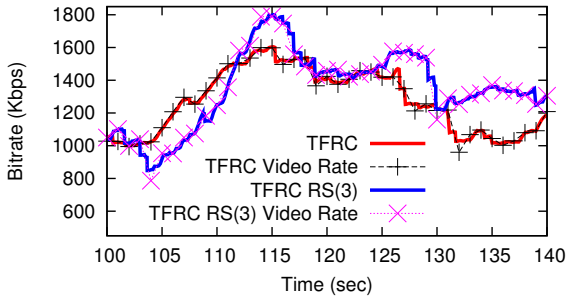


Figure 22: We adapt the encoding rate to TFRC's rate.

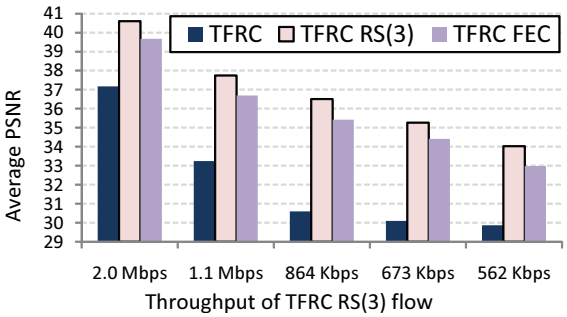


Figure 23: Video quality comparison.

realistic loss, we ran TFRC with competing TCP flows. The same dumbbell topology with 1 Gbps bottleneck link capacity is used. We vary the number of competing TCP flows from 200 to 2000. Each flow's RTT is randomly selected between 40ms and 200ms. Among the TCP flows, five of them are set to have the same RTT as the TFRC flows. We compare the relative throughput of TFRC flows to the average throughput of TCP flows. Our evaluation shows that TFRC RS(3)'s performance reasonably matches that of TCP when $\alpha = 1.5$ across various loss rates. Figure 21 shows the normalized TFRC RS(3)'s performance with respect to TCP Reno and TCP Sack. **We see that TFRC RS(3)'s is TCP friendly**; TFRC RS(3)'s throughput is equivalent to that of normal TFRC.

Adapting video encoding rate: From the TFRC's sending rate, we generate the video stream in the following way. At every RTT interval, TFRC yields a sending rate based on the average loss event rate over a much longer period of time. We adapt the video encoding rate at every second to the current sending rate of TFRC. Since TFRC's sending rate can change during a one second interval, we account for the difference in the next interval. The difference is small as TFRC's sending rate is already a smooth value. Figure 22 shows the sending rate of TFRC streams and the video rate. The video rate sampled every second, closely matches with the original TFRC's sending rate.

Video quality: Using the above scheme, we stream video with TFRC under various amounts of TCP cross traffic and create TFRC flows that give a throughput ranging from 562Kbps to 2.0Mbps. We compare the video quality of normal TFRC, normal TFRC with FEC, and TFRC RS under the same amount of cross traffic. For TFRC with FEC, we choose the parameter which gives the best PSNR among the ones that give buffer delay under 150ms. Figure 23 shows the video quality (average PSNR) achieved by TFRC flows of different throughput. We see that TFRC RS gives the best video quality in all cases.

6.5 Evaluation summary

In summary, our evaluation highlights three key properties of RS that we claim in Section 3.2: *high degree of robustness for applications with low overhead*, *ease of use and flexibility for developers*,

and *flow prioritization in the network*. In Sections 6.1, we showed that RS provides high robustness and low bandwidth overhead, which translated into higher quality for video applications. In Section 6.2, we showed that, unlike FEC, RS is easy to use since careful parameter tuning is not necessary, and delay can be independently controlled with the delay parameter. Section 6.3 showed that RPT flows are effectively prioritized over non-RPT flows in congestive links and may occupy more bandwidth than their fair-share. In cases where fair-sharing is desired, we show RPT can share bandwidth in a TCP friendly way while still retaining other benefits of RS in Section 6.4.

7 Conclusion

This paper explore issues arising from the confluence of two trends – growing importance and volume of real-time traffic, and the growing adoption of content-aware networks. We examine a key problem at this intersection, namely that of protecting real-time traffic from data losses. We show that adding redundancy in a way that network understands reduces the cost and increases the benefits of loss protection quite significantly. We refer to our loss protection approach as RPT. Using Redundant Streaming (RS) as an example, we highlight various features of RPT and establish that is a promising candidate to use in several practical scenarios. We show that RPT decreases the data loss rate by orders-of-magnitude more than typical FEC schemes applicable in live video communications, and delivers higher quality video than FEC using the same bandwidth budget. RS provides fine-grained control to signal the importance of data and satisfies tight delay constraints. Yet, it is easy to use as its performance is much less sensitive to parameter selection. We show that constant bitrate RPT flows are prioritized over non-RPT flows, but can share bandwidth fairly by incorporating TCP friendly rate control into RS. We also show that RPT provides an efficient and cost-effective loss protection mechanism in other general content-aware networks.

References

[1] Cisco Wide Area Application Services (WAAS) Software. http://www.cisco.com/en/US/prod/collateral/contnetw/ps5680/ps6870/prod_white_paper0900aecd8051d5b2.html, 2009.

[2] Juniper Networks Datasheet. http://www.juniper.net/us/en/local/pdf/datasheets/1000113-en.pdf, 2009.

[3] Cisco Internal WAAS Implementation. http://blogs.cisco.com/ciscoit/cisco_internal_waas_implementation/, 2010.

[4] Cisco visual networking index: Forecast and methodology, 20092014. http://www.cisco.com/, 2010.

[5] Magic Quadrant for WAN Optimization Controllers. http://www.gartner.com/technology/media-products/reprints/riverbed/article1/article1.html, 2010.

[6] YUV CIF reference videos. http://www.tkn.tu-berlin.de/research/evalvid/cif.html, 2010.

[7] ATT VPN Service. http://new.serviceguide.att.com/portals/sgportal.portal?_nfpb=true&_pageLabel=avpn_page, 2011.

[8] Infineta systems. http://www.infineta.com/technology/network_dedupe, 2011.

[9] Riverbed Could Products. http://www.riverbed.com/us/products/cloud_products/cloud_steelhead.php, 2011.

[10] Riverbed Customer Stories. http://www.riverbed.com/us/customers/index.php?filter=bandwidth, 2011.

[11] Riverbed Steelhead Product Family. http://www.riverbed.com/us/assets/media/documents/data_sheets/DataSheet-Riverbed-FamilyProduct.pdf, 2011.

[12] Andres Albanese, Johannes Blmer, Jeff Edmonds, Michael Luby, and Madhu Sudan. Priority encoding transmission. *IEEE Transactions on Information Theory*, 42:1737–1744, 1994.

[13] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. Data center TCP (DCTCP). In *Proc. ACM SIGCOMM*, 2010.

[14] Ashok Anand, Archit Gupta, Aditya Akella, Srinivasan Seshan, and Scott Shenker. Packet caches on routers: the implications of universal redundant traffic elimination. In *Proc. ACM SIGCOMM*, 2008.

[15] Ashok Anand, Vyas Sekar, and Aditya Akella. SmartRE: an architecture for coordinated network-wide redundancy elimination. In *Proc. ACM SIGCOMM*, pages 87–98, 2009.

[16] Guido Appenzeller, Isaac Keslassy, and Nick McKeown. Sizing router buffers. In *Proc. ACM SIGCOMM*, 2004.

[17] Mahesh Balakrishnan, Tudor Marian, Ken Birman, Hakim Weatherspoon, and Einar Vollset. Maelstrom: transparent error correction for lambda networks. In *Proc. USENIX NSDI*, 2008.

[18] Ali C. Begen and Yucel Altunbasak. Redundancy-controllable adaptive retransmission timeout estimation for packet video. In *Proc. ACM NOSSDAV*, 2006.

[19] Jean-Chrysostome Bolot, Sacha Fosse-Parisis, and Don Towsley. Adaptive FEC-based error control for internet telephony. In *Proc. IEEE INFOCOM*, 1999.

[20] O. Boyaci, A.G. Forte, and H. Schulzrinne. Performance of video-chat applications under congestion. In *Proc. IEEE International Symposium on Multimedia*, December 2009.

[21] Jin Cao, W.S. Cleveland, Yuan Gao, K. Jeffay, F.D. Smith, and M. Weigle. Stochastic models for generating synthetic HTTP source traffic. In *Proc. IEEE INFOCOM*, volume 3, pages 1546 –1557 vol.3, March 2004.

[22] Luca De Cicco, Saverio Mascolo, and Vittorio Palmisano. Skype video responsiveness to bandwidth variations. In *Proc. ACM NOSSDAV*, 2008.

[23] Emulab. http://www.emulab.net/.

[24] Nick Feamster and Hari Balakrishnan. Packet loss recovery for streaming video. In *Proc 12th International Packet Video Workshop*, 2002.

[25] Sally Floyd, Mark Handley, Jitendra Padhye, and Jörg Widmer. Equation-based congestion control for unicast applications. In *Proc. ACM SIGCOMM*, 2000.

[26] P. Frossard and O. Verscheure. Joint source/FEC rate selection for quality-optimal MPEG-2 video delivery. *IEEE Transactions on Image Processing*, 10(12):1815 –1825, December 2001.

[27] Ningning Hu, Li (Erran) Li, and Zhuoqing Morley Mao. Locating Internet bottlenecks: Algorithms, measurements, and implications. In *Proc. ACM SIGCOMM*, 2004.

[28] Van Jacobson, Diana K. Smetters, Nicholas H. Briggs, Michael F. Plass, Paul Stewart, James D. Thornton, and Rebecca L. Braynard. VoCCN: Voice over content-centric networks. In *Proc. ReARCH*, 2009.

[29] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. Networking named content. In *Proc. ACM CoNEXT*, 2009.

[30] Szymon Jakubczak, Dina Katabi, and Hariharan Rahul. SoftCast: One video to serve all wireless receivers. Technical Report MIT-CSAIL-TR-2009-005, Massachusetts Institute of Technology, February 2009.

[31] Jirka Klaue, Berthold Rathke, and Adam Wolisz. EvalVid - a framework for video transmission and quality evaluation. In *Proc. Performance TOOLS*, 2003.

[32] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, August 2000.

[33] Christian Leicher. Hierarchical encoding of mpeg sequences using priority encoding transmission (PET). Technical Report TR-94-058, ICSI, 1994.

[34] Dmitri Loguinov and Hayder Radha. On retransmission schemes for real-time streaming in the internet. In *Proc. IEEE INFOCOM*, pages 1310–1319, 2001.

[35] Harsha V. Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas E. Anderson, Arvind Krishnamurthy, and Arun Venkataramani. iPlane: An information plane for distributed services. In *Proc. 7th USENIX OSDI*, Seattle, WA, November 2006.

[36] Steven McCanne, Martin Vetterli, and Van Jacobson. Low-complexity video coding for receiver-driven layered multicast. *IEEE Journal on Selected Areas in Communications*, 15(6):982–1001, 1997.

[37] Kiyohide Nakauchi and Katsushi Kobayashi. An explicit router feedback framework for high bandwidth-delay product networks. *Comput. Netw.*, 51, May 2007.

[38] Christos Papadopoulos. Retransmission-based error control for continuous media applications. In *Proc. NOSS-DAV*, 1996.

[39] R. Puri and K. Ramchandran. Multiple description source coding using forward error correction codes. In *Proc. Asilomar conference on signals, systems, and computers*, pages 342 –346, 1999.

[40] Finally, A Truly Scalable SVC Solution For The Masses. http://blog.radvision.com/videooverenterprise/2009/04/21/finally-a-truly-scalable-svc-solution-for-the-masses/.

[41] Injong Rhee. Error control techniques for interactive low-bit rate video transmission over the internet. In *Proc. ACM SIGCOMM*, 1998.

[42] D. Salomon. *Data Compression; the Complete Reference*. Springer, 2007.

[43] Sayandeep Sen, Neel Kamal Madabhushi, and Suman Banerjee. Scalable wifi media delivery through adaptive broadcasts. In *Proc. USENIX NSDI*, 2010.

[44] Neil T. Spring and David Wetherall. A protocol-independent technique for eliminating redundant network traffic. In *Proc. ACM SIGCOMM*, 2000.

[45] Jue Wang and Dina Katabi. ChitChat: Making video chat robust to packet loss. Technical Report MIT-CSAIL-TR-2010-031, MIT, July 2010.

[46] Xiaoqing Zhu, Rong Pan, Nandita Dukkipati, Vijay Subramanian, and Flavio Bonomi. Layered internet video engineering (LIVE): Network-assisted bandwidth sharing and transient loss protection for scalable video streaming. In *Proc. IEEE INFOCOM*, pages 226–230, 2010.