MPRTP: Multipath Considerations for Real-time Media

Varun Singh
Aalto University
Espoo, Finland
varun@comnet.tkk.fi

Saba Ahsan
Aalto University
Espoo, Finland
saba.ahsan@aalto.fi

Jörg Ott
Aalto University
Espoo, Finland
jo@comnet.tkk.fi

ABSTRACT

The Internet infrastructure often supports multiple routes between two communicating hosts and, today, especially mobile hosts usually offer multiple network interfaces, so that disjoint paths between the hosts can be constructed. Having a number of (partly or fully) disjoint paths available may allow applications to distribute their traffic, aggregate capacity of different paths, choose the most suitable subset of paths, and support failover if a path fails. Exploiting multipath characteristics has been explored for TCP, but the requirements for real-time traffic differs notably. In this paper, we devise a multipath communication model for Real-time Transport Protocol (RTP); present minimal set of required protocol extensions; develop algorithms for scheduling RTP traffic across multiple paths at the sender and a corresponding de-jittering algorithm at the receiver side; and evaluate our proposal in varying scenarios using media traffic across different emulated mobile access network setups.

Categories and Subject Descriptors

C.2.2 [**Network Protocols**]: Applications (SMTP, FTP, etc.)

General Terms

Algorithms, Design, Performance, Experimentation

Keywords

Multipath, RTP, Multimedia, Scheduling

1. INTRODUCTION

The Internet backbone has evolved over the past decades to a mesh of service providers with manifold peerings that are generally capable of offering a number of (independent) paths, intra- and inter-domain, between two nodes. This extends to stub networks often using multiple attachment points for resilience purposes, such as data enterprise networks or data centers (and even routers for SOHO networks support multiple access networks [8, 14]). And as many hosts today feature multiple network interfaces (e.g., WLAN and 3G for laptop computers, tablets, or mobile phones), this may yield the possibility for two endpoints to communicate via multiple, at least partly disjoint paths. With at least two disjoint paths, those may be exploited for balancing traffic load, for aggregating capacity, and for failover in case one path becomes unavailable.

Operators already use alternative paths internally, e.g., for traffic engineering, load balancing, or fast re-routing, albeit without interaction with the endpoints. While no explicit path information or control is available to the endpoints, they may use their peer's IP addresses (paired with standard IP routing) to obtain a first approximation of multipath capabilities [21].[1]

Several transport protocols were developed to exploit such capabilities: *SCTP* [45] uses multiple interfaces for failover purposes and *Multipath TCP (MPTCP)* [19] offers parallel usage of multiple paths for resource pooling [48]. Both offer reliable transport services, motivated by balancing bulk data transfer and increasing robustness. However, in the single path case, TCP can only serve real-time communication within tight constraints of network characteristics [9]. In the multipath case, the scheduling algorithms do not consider real-time bounds when spreading data segments across different paths.

The increasing share of high-bit-rate video traffic on the Internet [15] motivates multipath support for real-time media as well. Today's dominant form of over-the-top streaming in the web usually are variants of HTTP streaming [4] that perform rate adaptation by switching rates, but this is fairly constrained and requires large playout buffers. MPTCP could be applied to media streaming but, as noted above, it does not consider real-time data and diverse paths may lead to worst case delay and thus even longer buffering time. Moreover, the compelling use-case for multipath media delivery is to mobile devices/tablets either for aggregating throughput of the WLAN and 3G or for path failover. It should also be noted that the mobile website of YouTube[2] uses RTSP instead of HTTP[3]. Additionally, pre-buffering is not an option for conversational media that gains importance with video communication clients implemented as ubiquitous applications such as *Skype* or as part of web browsers (see, e.g., the WebRTC effort[4]).

Currently, the *Real-time Transport Protocol (RTP)* cannot efficiently operate over multiple paths because RTP operates at the media session level and not at the transport level, i.e., the receiver reports per media (audio or video) flow and not per underlying 5-tuple. In this paper, we explore *Multipath RTP* that extends RTP

MMSys '13 February 26-March 1, 2013, Oslo, Norway.

[1] Further mechanisms are being investigated (e.g., [46]) but those are not available in today's Internet.
[2] http://m.youtube.com
[3] http://code. google.com/apis/youtube/2.0/reference.html
[4] http://www.w3.org/2011/04/webrtc/, http://datatracker.ietf.org/wg/rtcweb/

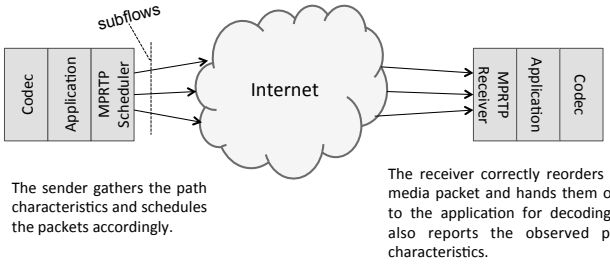


Figure 1: System Overview: A sender uses multiple paths to stream media to a receiver. The receiver uses a dejitter buffer to reorder packets and sends per-path characteristics to the sender that distributes the packets based on the reported values.

towards multipath communication. We focus on spreading *constant bit rate (CBR)* media streams across multiple paths, for which we present algorithms for allocating traffic on each path based on path characteristics and evaluate those in different scenarios. We use a short playout buffer ($\leq$ 500ms) at the receiver so that the algorithms will be applicable even to other scenarios like live video streams and interactive video. Our work is orthogonal to media rate adaptation—which would just change the total media rate to spread—and we defer adaptation as a function of the joint path characteristics to future work.

2. MULTIPATH REAL-TIME VIDEO

The Real-time Transport Protocol (RTP) supports end-to-end (e2e) delivery for data with real-time characteristics [37]. RTP was designed to carry data over UDP/IP but can be used in conjunction with other transport protocols, such as TCP and DCCP. RTP is suitable for applications such as live streaming and broadcast, video-on-demand, and interactive multimedia communication. RTP uses the associated RTP Control Protocol (RTCP) for monitoring the e2e media delivery and for other control operations. The baseline RTCP defines two reports for session monitoring: 1) The Sender Report (SR) carries synchronization information for media playout and reference points for RTT and packet loss calculation. 2) The Receiver Report (RR) provides mostly long-term statistics on the observed session characteristics (loss, jitter) and assists in RTT calculation [37].

In this paper, we present *Multipath RTP (MPRTP)*, an extension to RTP that allows splitting a single RTP stream into multiple subflows, which are transmitted over different paths. This allows pooling the capacity of multiple Internet paths so that higher bit rate media can be delivered and the system becomes more robust against path variations or disruptions. From the application perspective, the available bandwidth between the two endpoints increases or becomes more stable. Figure 1 shows a macroscopic system overview of MPRTP.

For a constant bit rate (CBR) media stream, a codec generates packets of a constant size or packets produced in a short interval, will average to a constant size. Video streaming services overcome congestion by either pre-buffering *enough* content or by rate-switching. The latter requires the streaming server to pre-encode content at different video bit rates, so that a receiver can easily pick a rate that is suitable for its e2e path capacity. Alternatively, interactive media streams try to match the e2e path capacity by change the encoding rate at the media source. In either case, the endpoints should avoid changing the rate too often or by too much because it would impede the user experience [51].

Instead of using rate switching, MPRTP can split the stream across multiple paths and thereby sustain the media bit rate (and thus the quality of experience) and shift more load to the less congested paths—provided that the aggregate capacity of all paths exceeds the average bit rate of the CBR video stream. A side effect of the video traffic distribution is that the video stream *appears* somewhat fair from the perspective of an individual more congested path: even though MPRTP does not reduce the bit rate, load is shifted away to another path. This may reduce the impact of video stream on the other traffic transmitted over the former, potentially at the expense of traffic on the latter path.

RTP is a generic real-time transport protocol and MPRTP is not restricted to video either; e.g., voice calls may benefit from its path failover properties. However, in this paper we focus on streaming video with low buffering requirements and can therefore be easily extended to interactive multimedia sessions. Our contributions are: algorithms deciding which paths to use, how to distribute the RTP traffic across them, and the matching receiver side operation.

3. MPRTP DESIGN GOALS

MPRTP is designed as an extension to RTP and thus can benefit from the following RTP functionality: end-to-end (e2e) transmission of media packets, intra- and inter-stream synchronization, e2e monitoring and session control, and a wealth of payload formats to encapsulate encoded media content. What is novel is the ability to explicitly distribute media packets across multiple paths, which puts forward requirements on scheduling and dejitter algorithms.

From a functional perspective, MPRTP must be able to make use of multiple paths and adapt to their relative capacity changes by redistributing the load, which should be done in a way that avoids oscillation (Section 3.1). As different paths will likely exhibit different RTTs, mechanisms must be put in place to overcome the resulting skew (Section 3.2). The choice of suitable transmission path should reflect the demands of the application (Section 3.3). From a protocol perspective, RTP must be extended to provide the additional information necessary to perform these functions, yet maintain backwards compatibility (Section 4).

3.1 Adapting to Bandwidth Changes

An MPRTP endpoint should be able to redistribute the traffic load to other paths when one or more of the current paths become congested or fail. In such situations, [26] suggests moving all the traffic to less congested paths, but for MPRTP we propose sending some small fraction of the traffic on these congested paths to continuously monitor the path characteristics. This is similar to MPTCP [49], which keeps some traffic on a congested path to monitor the path as well. If loss is one of the motivating indicators for reassigning traffic, a scheduling algorithm should not change the per path traffic distribution at short timescales.

MPRTP avoids aggressively reassigning traffic and continues monitoring all available paths. The latter is especially important for mobile devices since those will often have only two interfaces and those interfaces (or paths) may not only see sudden degradation but also quick recovery.

RTP/UDP has no inherent congestion control and is unfair to competing traffic and CBR media streaming limits congestion control to rate-switching. As a result, a CBR media stream will occupy its share of the e2e capacity and may not give way to any other flow. MPRTP supports some minimal fairness by detecting interactions with other flows: it is sensitive to changes in path characteristics such as jitter, RTT, and packet inter-arrival time at the receiver. An MPRTP sender aims at balancing the system by spreading the load across all their (suitable) paths, essentially being less unfair and thereby leaving more room for other traffic.

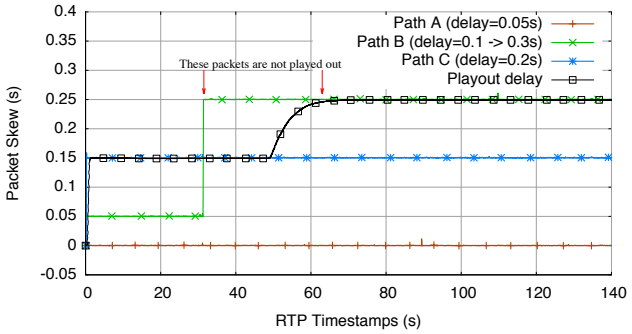


Figure 2: Shows the adaptive playout capabilities of a legacy endpoint receiving media data over multiple paths with different latencies. The latency on path A and C is constant through-out the session while Path B experiences a change in path latency at $t = 30s$. The dark line is the instantaneous skew and the figure shows that the playout takes 15 to $20s$ to converge to the highest value when the fraction traffic distribution per path is equal.

3.2 Overcoming Packet Skew

Packets sent over multiple paths will likely encounter different path delays and may arrive out-of-order at the receiver, i.e., the receiver dejitter buffer would have to compensate for the variation in packet inter-arrival time (*packet skew*). Conceptually, if the dejitter buffer is larger than the differences in the path latencies, the receiver would be able to re-order the packets. However, in reality the RTT is variable and can increase beyond the calculated dejitter buffer size, which may also happen for a single path. Diverse paths thus increase the chances for buffer underflow. Using an adaptive playout buffer can overcome this limitation.

A single path receiver calculates the optimum playout using a low-point or mid-point averaging algorithm [18]. The algorithm averages the *packet skew* over a large window size[5] to reduce the impact of the few packets that get delayed. However, in a multipath scenario, the packets may be scheduled on paths with widely varying latencies and the above method may be insufficient. As an example, consider figure 2 that shows the instantaneous playout point (*black line*) for each packet using the method described in [18] for a multipath scenario. In this scenario three paths are available with different path latencies but the same e2e capacity. The sender sends an equal share of the media stream along each path. The figure also shows the packet skew (*Path A* =$50ms$, *Path B* =100 *to* $300ms$, *Path C* =$200ms$) relative to the low-latency path (*Path A*=$50ms$). Initially, the algorithm is able to compensate for the different path latencies. However, at $t = 30s$ the latency of *Path B* changes from 100 to $300ms$ and the algorithm takes about $20 - 30s$ to play packets from that path. Therefore, the algorithm proposed in [18, 33] is not suitable for the multipath scenario and a multipath receiver should implement an alternate algorithm to calculate optimum playout to adapt to paths with different latencies (See Section 5.3 for our proposed solution).

3.3 Choosing Transmission Paths

A multipath endpoint needs to choose which of its available paths to use for sending media (one, a subset, all). This is an optimization problem with multiple parameters that an endpoint can optimize for. Examples include minimizing losses, minimizing latency or maximizing e2e capacity. While we mentioned above that as many paths as possible should be used, distributing the media stream widely should still not come at the expense of media quality at the receiver. The impact on the media quality is application-dependent so that MPRTP has to provide an interface that allows an application to specify its preferences: for example, some multimedia applications are more tolerant to losses than others.

Video streaming applications can pre-buffer few seconds of media data, while live streaming and interactive multimedia can only pre-buffer *hundreds of ms* of media data. So an application prioritizes between expected e2e *latency* and *capacity* (by choosing a different encoding bit rate). Since capacity is additive for paths with similar latency [48], an MPRTP sender needs to aggregate as much capacity as needed by combining paths with as little latency divergence as possible.

In the real-time media case, quality may suffer quickly from packet losses and therefore in this paper we choose packet loss as another factor an MPRTP endpoint takes into account for path selection: if the losses observed across multiple paths differ widely, those with lower loss rates will be prioritized for media transmission (and only monitoring traffic will be sent on the high loss rate paths to observe changes in path characteristics).

Obviously, MPRTP cannot assume static path characteristics and needs to continuously measure those using media traffic and, for any passive paths, probing traffic. As noted above, the changes in path selection should not be too aggressive to avoid oscillation.

4. PROTOCOL DETAILS

RTP uses as abstraction from the lower layers an interface via which it can send and receive RTP and RTCP packets to one or more peers. The implicit assumption is that all these packets follow a common path at a time so that the path characteristics can be measured and the results can serve as input to application layer adaptation mechanisms (if any). The abstraction offered as an "API" to the application is the transmission/reception of a media packet stream plus information about the path.

With MPRTP, we seek to preserve this abstraction towards the application. But, obviously, multiple paths may have different characteristics and hence the RTCP monitoring functions tailored to a single path are insufficient. MPRTP needs to be able to determine if multipath communication is possible and, if so, mark packets sent over separate paths accordingly and monitor characteristics per path so that packet traffic distribution can take those into account. The receiver side has to automatically take into account the potentially wider variation in latency without any explicit notification from the sender. The entire operation must be backwards compatible with RTP.

In the following sub-sections, we present the protocol extensions required for MPRTP. These extensions assist in implementing packet scheduling and adaptive playout by identifying packets sent on a particular path and reporting per-path characteristics. The complete protocol specification is available as Internet Drafts [39, 44].

4.1 Subflow Information

An MPRTP sender assigns a subflow identifier (*subflow ID*) to each unique path. A path is considered unique if sending or receiving IP address and port number (5-tuple) are different. An MPRTP sender also adds a *subflow-specific sequence numbers* to enable the receiver to determine subflow-related packet jitter, packet loss, and packet discards. Both the subflow ID and the subflow-specific sequence numbers are carried in an RTP header-extension and this preserves backward compatibility (see below).

4.2 Subflow Path Characteristics

The receiver reports characteristics per subflow because the paths

[5]Usually, 256 or 512 packets [18, 33].

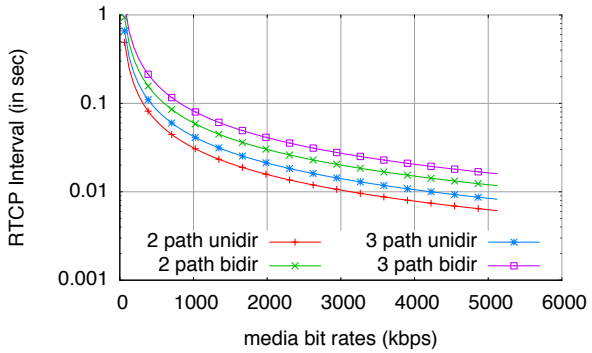


Figure 3: Multipath RTCP Reporting Interval for non-compound RTCP packets in a unidirectional and bidirectional media session.

may exhibit widely differing properties. For this purpose, we introduce RTCP per-subflow reporting using new variants for SR, RR, and extended report (XR) packets. The receiver reports the path jitter, losses and discards for each subflow. It also reports information to calculate per path RTT. The scheduler at the sender compares path characteristics of all the subflows to determine their respective traffic shares. In addition, the application also reports aggregate session level statistics as defined in RFC3550 [37].

4.3 RTCP Interval Calculation

The RTCP transmission rate for each of the two endpoints is calculated as 2.5% of the session *media rate* [32]. Moreover, the reporting frequency for each subflow should be similar to the amount of traffic it sends/receives, i.e., flows which send/receive more media data should report more frequently than those which send/receive less. Figure 3 shows the variation of reporting interval with respect to media bit rate. For low bit rates the feedback depends on the situation, i.e., is it bidirectional (bi)/unidirectional (uni) media flow. We use non-compound RTCP reports [24] because of their relatively lower header overhead. In our experiments, we observe that the RTCP feedback interval should not be shorter than $2 \times RTT$ under normal operation. For urgent feedback (e.g., NACKs) the client still has the option to send immediate feedback.

4.4 Backwards Compatibility

MPRTP is backwards compatible, i.e., an MPRTP sender does not require the receiver to implement the extensions and therefore can interoperate with an RFC3550 [37] RTP receiver. An MPRTP receiver will naturally process packets from an RTP sender. Legacy RTP entities simply ignore the MPRTP extension headers and extended RTCP reports. However, an RTP receiver may drop packets due to large variance in path delays (limited adaptive playout). Moreover, the RTCP Receiver Reports (RR) do not carry sufficient information to make proper per-path scheduling decisions. Therefore, MPRTP capabilities can only be exploited when both endpoints support MPRTP.

5. ALGORITHMS

The scheduling algorithm uses subflow reports to estimate the bit rate on each of the available paths and distributes the traffic based on the reported values. The receiver buffers the incoming packets, reorders and recombines the subflow into one stream for the application. Figure 4 shows a schematic diagram of the sender side buffer; the scheduling algorithm assigns a priori the bandwidth share per path based on the short term history and long term path heuristics.

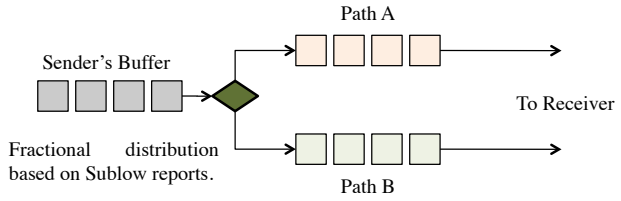


Figure 4: Sender-side Scheduler

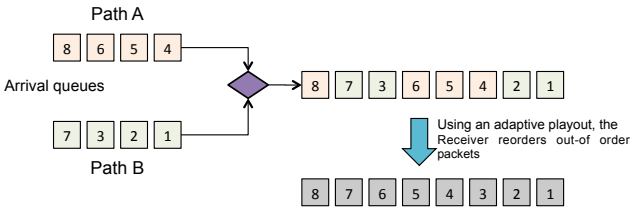


Figure 5: Receiver-side dejitter buffer

The multipath receiver is responsible for recombining the subflows into the original flow. The arrival queues in each subflow re-combine RTP packets that belong to the same video frame and the recombined packets are passed to the dejitter buffer. The dejitter buffer should be sufficiently large to overcome variation in per path RTT (see Fig. 5), also an adaptive playout delay ensures smooth playback. The receiver routinely reports the per path (i.e., subflow) jitter, losses and RTT to the sender and the sender uses the information to make scheduling decisions.

5.1 Scheduling Algorithm at the Sender

At startup, if the number of paths is known, the scheduling algorithm assigns equal fractions of the media rate to all the available paths; so 50% each for two paths. Every time a new packet arrives at the scheduler, it is assigned to a particular path depending on the relative traffic distribution. Since congestion control is an orthogonal problem to packet scheduling and outside the scope of the paper, losses cannot be avoided if the aggregate end-to-end channel capacity across all the paths is below the media encoding rate.

In the following subsection we describe the various steps involved in packet scheduling. The MPRTP sender first calculates the estimated receiver rate on each path based on the Subflow Receiver Reports. Second the sender characterizes the paths based on the observed packet discards and losses. Third the sender chooses a set of *active paths* from the available paths. Fourth the sender follows a set of packet scheduling rules. Fifth the sender calculates the timescales at which it will re-calculate the fractional distribution and lastly, it calculates the per-path fractional distribution.

Calculating the Path Receiver Rate

The sender calculates the receiver rate per-path using information from two consecutive receiver reports (RTCP RRs) and by maintaining a queue of packets since the last RR [41]. This queue can also be used for retransmitting packets in case of receiving NACKs. The bandwidth on *path j* when receiving the i^{th} RR is:

$$RR[j] = \frac{(\sum_{k=HSN_{i-1}}^{HSN_i} sizeof(X_k)) \times (1 - L_i)}{(t_i - t_{i-1})}$$

HSN_i, HSN_{i-1} are the Highest Sequence Received reported by the receiver in two consecutive RRs. t_i, t_{i-1} are the reception times of the two RRs at the sender, L_i is the reported loss rate in the latest RR. $sizeof(X_k)$ is the size of the packet and the sum gives

the bytes sent by the endpoint during the last reporting interval. If the receiver also reports the discarded bytes [31] in the interval, the sender can calculate the goodput by subtracting the discarded from the total bytes.

Path Characterization

Based on the circuit breaker conditions [34], we define some rules for marking paths as congested, lossy or non-congested. A path that reports discards and losses in a single or consecutive intervals can be considered *mildly congested.* If this behavior is observed over 3 successive intervals then it is considered *congested*. Furthermore, if a path reports only losses and no discards in successive intervals then it is considered *lossy*. A path without losses or discards is considered *non-congested*.

Choosing Paths

A multipath endpoint needs to choose all or a subset of its available paths for sending media. To find a suitable group of paths, the multipath sender needs to evaluate the paths that can meet the capacity and latency requirements.

When there are many paths available, an endpoint can group them based on the path latencies—bandwidth is additive for paths with similar latencies [48]. The endpoint then sorts these path groups in decreasing order of $\frac{bandwidth}{delay}$, so that groups with high bandwidth and low delay are preferred. The multipath sender chooses the suitable group(s) of paths from the $\frac{bandwidth}{delay}$ list that meet its e2e capacity requirements. The chosen paths are marked as *'active'* and used for media delivery while the rest of the paths are marked as *'passive'* and used when the chosen paths fail. Depending on the multimedia application, packet loss may affect the quality of experience. Therefore, an MPRTP sender should avoid scheduling packets on paths with losses.

Packet Scheduling Rules

The scheduler continuously assigns part of the media traffic to each path based on the path's receiver rates, some of these paths may be congested while some others may not be. The scheduler observes the following rules:

- If the next scheduled frame is an I-frame then the resulting RTP packets are assigned to the path with the highest $\frac{bandwidth}{delay}$, bandwidth and lowest loss rate.
- On receiving a NACK, transmit the requested packet on the path with the highest $\frac{bandwidth}{delay}$, least RTT and lowest loss rate.
- Reduce the fractional traffic distribution on the *mildly congested* and *congested* paths in an attempt to reduce congestion on those paths.

Scheduling Interval

Calculating the fractional distribution for each path can be computationally intensive; therefore, the redistribution should not be calculated at reception of each RTCP report.

The scheduler calculates the fractional distribution at the expiry of the scheduling interval, which is:

$$\begin{aligned} SchInt = &\quad \gamma \times S_{interval}, \quad 0.5 \leq \gamma \leq 1.5 \\ \gamma = &\quad 0.5 + rand(0.0, 1.0) \end{aligned}$$

The randomization factor (γ) is used to prevent synchronization of multiple senders with common network paths. $S_{interval}$ is a value between the minimum RTCP interval and maximum RTCP interval [32, 39]. At startup, $S_{interval}$ is set to the minimum RTCP interval and is increased in steps of the minimum RTCP interval until it reaches maximum RTCP interval. If congestion is reported on multiple paths then the sender may reduce the scheduling interval to recalculate the fractional traffic distribution of the paths more frequently.

Algorithm 1 Algorithm for calculating fractional traffic distribution for each path

Require: *SchInt* timeout
Require: RR for each *active* path
Require: Sort Paths based on Path Characteristics:
1: path = ['m' congested, 'r' mildly-congested,'w' non-congested]
2: $MR \leftarrow media\_rate$
3: $paths_{total} \leftarrow len(path)$
4:
5: **for** $j = 1 \rightarrow paths_{total}$ **do**
6: **if** path[i] is congested **then**
7: $SB[j] \leftarrow MIN\left(\frac{RR[j]}{\sum_{k=0}^{m} RR[k]} \times MR, \quad \alpha \times RR[j]\right)$
8: **else if** path[j] is mildly-congested **then**
9: $SB[j] \leftarrow MIN\left(\frac{RR[j]}{\sum_{k=0}^{r} RR[k]} \times MR, \quad \beta \times RR[j]\right)$
10: **else if** path[j] is non-congested **then**
11: $SB[j] \leftarrow \left(\frac{RR[j]}{\sum_{k=0}^{w} RR[k]} \times (MR - AP)\right)$
12: **end if**
13: $AP \leftarrow AP + SB[j]$
14: $j \leftarrow j + 1$
15: **end for**
16:
17: $j \leftarrow 0$
18: **while** $AP < MR$ **do**
19: $extra \leftarrow \frac{RR[j]}{MR} \times (MR - AP)$
20: $SB[j] \leftarrow SB[j] + extra$
21: $AP \leftarrow AP + extra$
22: $j \leftarrow j + 1$
23: **end while**

Calculating the Fractional Distribution

Initially the scheduler allocates an equal rate to each path, which may be incorrect. Therefore, as soon as the scheduler has enough information[6] about the path characteristics, it should recalculate the fraction distribution for each path.

We assign the fractional traffic distribution starting from the congested to the the non-congested paths because the congested paths were originally the preferred paths and due to our initial rate allocation or cross-traffic become congested. Consequently, the packet scheduler still wants to continue using these paths (which are 'active') over the others (which were earlier 'passive' or for failover). This yields the following cases:

Paths that are congested— The scheduler wants to send some minimal media data on these paths, so that it is able to observe the changes in the path characteristics instead of totally ignoring these paths. From changes in the path characteristics the sender may infer when to reallocate again more media to the path. The sending bit rate ($SB[j]$) for a *path j* for a total number of congested paths (m) is the minimum of the fair share bit rate of the congested path and α times the path's receiver rate (See line 7 of Algorithm 1). We choose the minimum of the two because we do not want the fair-share to be larger than the path's receiver rate.

Paths that are mildly congested— The scheduler wants to reduce the load on these paths slightly so that the paths can go back to being un-congested. The sending bit rate ($SB[j]$) for a *path j* where the total number of mildly congested paths (r) is similar to the congested case, except here we use a factor β instead of α and

[6]optimally after reception of at least one RTCP report for each subflow.

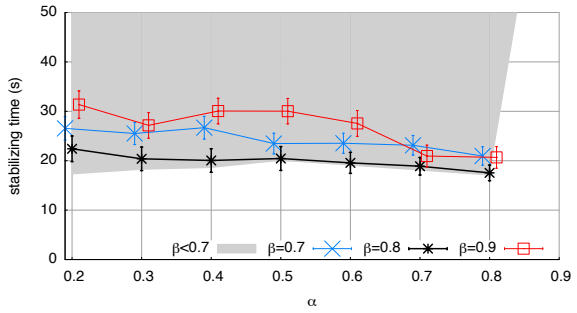


Figure 6: In each experiment we vary the values of alpha (α) and beta (β) and observe the time it takes for the scheduling algorithm to reach optimum fractional distribution. We conduct experiments for a 1Mbps media stream using 2- and 3-paths and the feedback interval per path is 1s. In every scenario the cumulative bandwidth of the paths is 1Mbps, but each path has different e2e capacity. We run each scenario 500 times and the error-bars represent the 95% confidence-level.

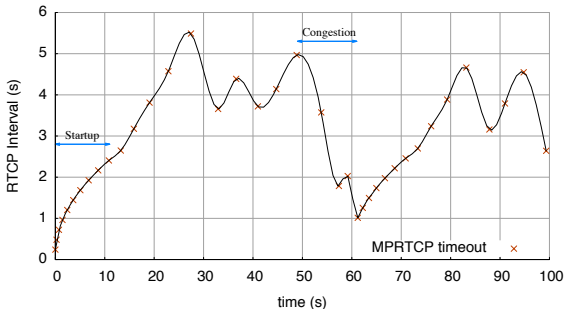


Figure 7: At startup and when congestion occurs the RTCP interval becomes shorter. In steady state the interval is typically $5 \pm 2.5s$.

because the paths are not completely congested, we use $\beta > \alpha$ (See line 9 of Algorithm 1).

Paths that are non-congested— The remaining traffic is divided fairly between the non-congested paths, i.e., as a ratio of the path's receiver rate with the total receiver rate for all the non-congested paths. Therefore, the rest of the media bit rate is assigned to the remaining paths (w) using the formula on line 11 of Algorithm 1. If the allocated bit rate (AP) at the end of the process is still lower than the media bit rate, then we allocate *extra bit rate* to each of the paths (See lines 17:23 of Algorithm 1).

Based on a series of experiments with our proposed scheduling algorithm, we choose $\beta = 0.8$ and $\alpha = 0.5$. Figure 6 shows that $\beta = 0.8$ on average gives quicker stabilizing times[7] compared to other values of β (In the figure the grey region represents average stabilizing times for different values of β). We use $\alpha = 0.5, (\alpha < \beta)$ because we want to reduce the load on the *congested path* by choosing a sending bit rate much lower than the receiver rate for that path. We also observe that $\alpha = 0.5$ gives quicker stabilizing times when $paths > 3$ and it is similar to TCP behavior.

5.2 RTCP Reporting Interval

The RTCP reporting interval for MPRTP is similar to that of RTP, i.e., typically reports every 5 ± 2.5s. However, to ensure reporting fairness across all the *active subflows*, MPRTP uses the ratio of the path's receiver rate and media bit rate to schedule per-path RTCP. If a path requires more frequent reporting, the receiver can allocate a larger fraction of the media bit rate for reporting RTCP, i.e., if the current RTCP reporting rate is less than the maximum RTCP rate (2.5%$\times$ media rate), the receiver will increase the RTCP reporting rate and thus, decrease the reporting interval proportionally per path. Several factors that influence the RTCP reporting interval are:

- If discards and losses are observed in one or more successive reporting intervals (*mildly congested link*), the receiver reduces the RTCP interval to send more frequent feedback. This enables the sender to change the fractional distribution more quickly and thus mitigate congestion on those path(s).
- If only sporadic losses are observed then this may be a transient condition and the receiving endpoint only reports the loss earlier than scheduled [32], so that the sender can retransmit if the lost packet is an I-frame (or part of it).
- When the paths characteristics are stable and enough information is known about the paths then the endpoint uses a relatively constant reporting RTCP interval.

Figure 7 shows an example of variation in MPRTCP timeouts based on the above rules. The timeouts are shorter during startup and when congestion occurs. Rest of the time when the paths are stable the RTCP Interval is also longer.

5.3 Path Skew & Playout Delay Calculation at the Receiver

In section 3.2, we show that when a new high latency path appears the current RTP implementation (legacy) takes nearly 15-20s to include packets from the high delay path for playout. This is because the legacy implementations calculate playout delay at the RTP session level and not at the subflow level. Our proposal is that the receiver calculates: 1) the Path Skew, based on the packet skew on each path, and 2) the Playout Delay, based on the per Path Skew (and not packet skew).

Path Skew

The endpoint calculates the packet skew of each packet received on a path by:

$$Packet\ Skew = (TR_j - TR_i) - (TS_j - TS_i)$$
$$TR = reception\ time$$
$$TS = RTP\ timestamp$$

Where *'i'* and *'j'* are two consecutive packets received on the path. However, they may not necessarily be two consecutive packets in the media stream. For each path the receiver maintains a Drift Window (DW), which is a sliding window of 2 seconds of media packets or 100 packets, whichever is lower. We use a relatively small window size because we want to avoid receiver under-run by changing the playout very late. Every time the endpoint receives a packet on a path it calculates the drift and inserts it in to the window. The receiver then sorts the window and chooses the median ($\widetilde{DW}$) value for calculating the path skew:

$$Path\ Skew = 0.01 \times \widetilde{DW} + 0.99 \times PathSkew_{prev}$$

We do not use the maximum or minimum values of the drift window (DW) in the calculation of the skew because the minimum value will cause buffer underflow while the maximum value is sensitive to temporary congestion. Moreover, if the path latency increases and the fractional distribution on the path remains the same, it will take $1s$ (or 50 packets) for the Path Skew to converge to the new higher measured latency.

[7] time for the scheduling algorithm to reach the optimal solution.

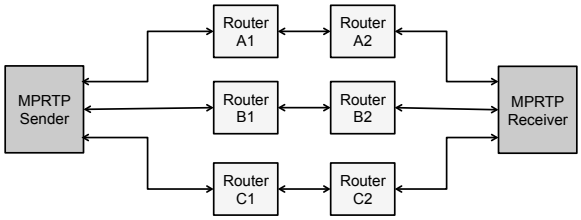


Figure 8: Evaluation Environment.

Playout Delay

The endpoint also maintains a window of the last 256 Path skew values, i.e., Path Skew for each path is inserted into the Skew Window (SW). The Skew Window (SW) is sorted and the maximum value is used to calculated the playout delay ($Playout_{delay}$). We use the MAX value here instead of the median because we want to include the high latency path as soon as possible. Moreover, the Path Skew calculation makes sure that transient congestion events do not show up. To calculate the playout delay, we use the same weighing factor as the legacy implementations [18, 33].

$$Playout_{delay} = \frac{MAX([SW]) + 124 \times Playout_{prev}}{125}$$

Depending on the fractional traffic distribution per path, our experiments show that our proposed method performs better in adapting the playout quickly for a short playout buffer of $500ms$. Our method takes $\approx 3s$ while the one implemented in [18, 33] takes 15-20s. Also our algorithm converges more quickly than the receiver can report the RTT to the sender, the typical RTCP interval is 5 ± 2.5s.

6. PERFORMANCE EVALUATION

To evaluate the performance of our MPRTP algorithms, we have implemented MPRTP using open-source libraries: Gstreamer[8] and x264[9]. The application can encode and decode streams, therefore, it can also be used as a streaming server and client. To evaluate MPRTP, we use the *"Foreman"* video sequence[10]; it is $265s$ long and is encoded at $1Mbps$ average media rate, 30 FPS and a group of pictures (GOP) set to 16. The receiver is configured to use a short dejitter buffer (maximum playout delay is 500ms) to emulate a live video stream.

Even though we have an implementation that can be deployed in the real-world, we use network emulation to evaluate the performance of the scheduling algorithm. The main reason is the need to have a controlled environment where we can vary the path characteristics and observe the response of the scheduling algorithm. We nevertheless did some qualitative real-world experiments as well, to which we will return in section 7.4.

To emulate a multipath network, we use a set of physical and virtual machines (VMs). The endpoints are physical machines with 3 network interfaces and the intermediate routers are virtual machines with two interfaces each. The setup is shown in Figure 8. Network Emulator (NetEm)[11] runs at each interface and emulates various link properties such as link capacity, delay and loss rate.

In the following sub-sections, we show representative plots for individual runs as well as tables with results averaged over 10 independent runs, including standard deviation (σ). We use Peak Signal-to-Noise Ratio (PSNR), Packet Loss Rate (PLR), and Average Bandwidth Utilization (ABU) as metrics [40] to compare the performance of the scheduling algorithm.

[8]http://gstreamer.freedesktop.org/

[9]http://www.videolan.org/developers/x264.html

[10]http://xiph.org

[11]http://swik.net/netem

Path Characteristic		Avg. PSNR	σ_{PSNR}	PLR
No Losses on any path	1-Path	48.427	0.00	0.00
	2-Path	48.427	0.00	0.00
	3-Path	48.427	0.00	0.00
0.5% Loss on every paths	1-Path	40.887	0.506	0.49
	2-Path	40.314	0.576	0.505
	3-Path	40.406	0.849	0.494
1% Loss on every paths	1-Path	36.172	0.705	1.01
	2-Path	36.564	1.006	0.94
	3-Path	36.212	0.572	0.99
Dissimilar RTTs	2-Path	48.303	0.278	0.004
Different BW per path	2-Path	45.4	4.6	0.0767

Table 1: Single Path vs Multiple Paths

6.1 Comparison to a Single Path

We first compare the performance of a multipath to single path setup in a static scenario, i.e., an environment with fixed end-to-end delay, losses and channel capacity.

Single Path: An endpoint receiving a streaming CBR video uses a dejitter buffer to compensate for variable one-way delay. Losses are detrimental in video streaming but depending on the trade-off (delay vs bandwidth), packet loss can be fixed using ARQ or FEC. On the other hand, bandwidth fluctuations will cause momentary pausing or freezing due to buffer underflow. This can be observed by comparing the average PSNR for single-path in each scenario listed in Table 1. We use the results of single-path as the benchmark to compare the performance of MPRTP.

Paths with similar losses: Each path has the same loss-rate (0%, 0.5%, 1.0%), one-way delay and e2e capacity. The scheduling algorithm performs at par with single path in this scenario. The per-path bit rate across all the paths is also similar, around 50% in the case of 2-paths and 33% in the case of 3-paths.

Paths with dissimilar RTT: The two paths have different one-way delays (Path A=50ms, Path B=200ms) but the e2e capacities and loss rates are similar. By using an adaptive playout the receiver overcomes the difference in per-path delay (skew) and is able to maintain an average PSNR similar to the lossless scenario. Since no losses occur, the scheduling is able to maintain an equal fractional distribution per path. This shows that the receiver can adapt the playout delay independently of the sender side algorithm.

Aggregating capacity: The two paths have fixed but different e2e channel capacity (Path A is 1 Mbps, Path B is 512 Kbps). We observe that the average bit rate sent on Path B is lower than the capacity of Path B, therefore, the scheduling algorithm is able to adapt to the capacity constraints of the individual links. At startup, per algorithm the sender allocates equal fractional distribution on each path. Since Path B's capacity is equal to the allocated fractional distribution the initial subflow RTCP RRs report higher jitter and the sender recalculates the fractional distribution and sends more media on Path A than on Path B.

MPRTP header overhead: The header overhead for transmitting MPRTP media packets is independent of the number of paths they are transmitted on and in our experiments consumed an additional 1.275 Kbps over the standard RTP media packets. However, RTCP is sent over each path and accounted for 72% (0.24 kbps) and 79% (0.39 kbps) of the total RTCP bandwidth.

In summary, MPRTP can spread a CBR media stream across multiple paths with diverging properties and provide comparable quality (PSNR) to a single sufficiently dimensioned path, i.e., path aggregation does not impact optimal performance.

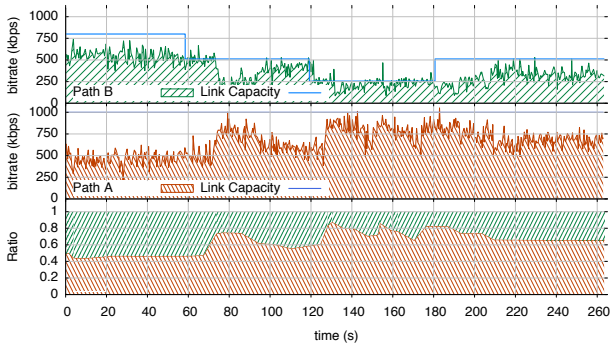


Figure 9: Path A has constant capacity while on Path B capacity changes at $60s$ intervals (possibly due to cross-traffic). We observe that the media is offloaded to the other link when a path is constrained but is slowly shifted back as the constrained link improves.

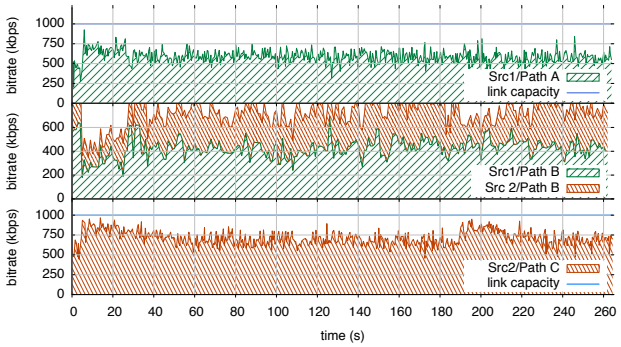


Figure 10: Two flows sharing a common bottleneck. Path A and Path C have an e2e channel capacity of $1Mbps$ and Path B has an e2e channel capacity of $800kbps$.

	Avg. PSNR	σ_{PSNR}	PLR
Variable losses per path			
2-Path (0-0.5%)	43.4	1.9	0.24
3-Path (0-1.0%)	40.5	0.49	0.48
Variable RTT per path			
Multi-Path	48.164	0.32	0.0121
Variable channel capacity per path			
Multi-Path	42.93	2.23	0.772

Table 2: Varying Link Properties

	Avg. PSNR	σ_{PSNR}	PLR
Source 1	44.41	0.03	0.132
Source 2	44.50	0.23	0.11

Table 3: MPRTP sharing a common bottleneck.

6.2 Varying Link Properties

We now discuss the performance of the scheduling algorithm using time-varying link properties.

Paths with varying losses: Each path has a varying loss-rate but the same one-way delay and e2e capacity. The loss-rate for a path is randomly chosen for every one second interval. On detecting losses the scheduling algorithm attempts to move the load to the other available links. A scheduling algorithm that favors one path over the others will perform poorly because of the variable loss rate. This can be observed by the significantly higher average PSNR values for the varying losses scenario (in Table 2) than the average PSNR for the cases with static loss rate (in Table 1). We also observe an equal fractional distribution per path.

Paths with varying delay: The setup comprises of 3 paths with different e2e latencies but same loss-rate (0.5%). We emulate varying path delay by randomly changing the link delay by $\pm 25\%$ at 1 second intervals. These delays reflect routing updates or queuing delay caused by *cross-traffic*. Additionally, improper rate allocation to a path may cause congestion resulting in additional queuing delay. This is the only scenario where we enable retransmissions to show that the scheduling algorithm assigns lower traffic share to the low latency path and favors it for re-transmission, i.e., Path A=27% (50ms), Path B=33% (100ms) and Path C=40% (200ms). By allocating more of the media rate to the high latency path ($200ms$), the scheduler frees up bandwidth on the low latency path ($50ms$) for retransmissions. We also observe that the receiver is able to adapt its playout and provides a comparable PSNR to the lossless scenario despite the varying path latencies and losses.

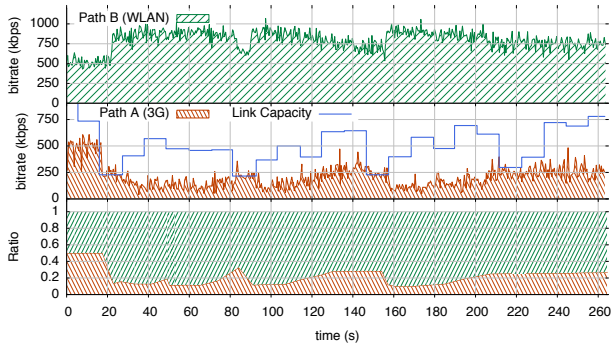
Paths with varying capacity: In this scenario, the e2e capacity on one path is variable, it demonstrates the sensitivity of the scheduling algorithm to the changes in network capacity, which may be caused by *cross-traffic*. Path B in Figure 9 shows the link with variable e2e capacity and the per-instant bandwidth utilization by an MPRTP subflow. Note that the scheduling algorithm uses cues on one path to reallocate the media on to the other paths (observe the points where the link rate drops). The scheduling algorithm also tries to probe the network, so that an equilibrium state of fair sharing can be achieved. However, this is done at long timescales (order of seconds) so that the per-path load does not oscillate.

6.3 Paths Share a Common Bottleneck

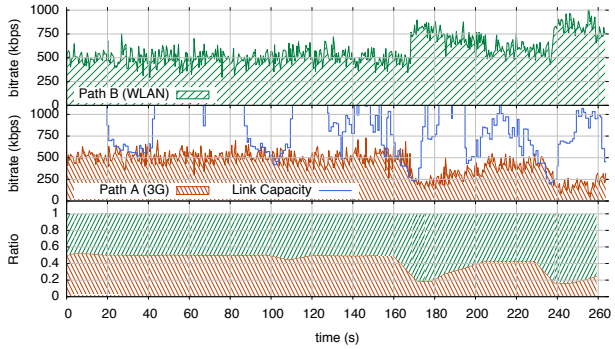
Not all e2e paths in the Internet will be disjoint, some may share an intermediate bottleneck path. We conduct two experiments where: 1) all paths between two MPRTP endpoint share a common bottleneck, 2) one path in each MPRTP endpoint shares a common bottleneck path.

Subflows share a common bottleneck: All the paths have different e2e capacity but the bottleneck path has sufficient capacity to carry the $1Mbps$ stream. In this case, we observe no variation in the relative traffic distribution for the paths once the scheduling algorithm converges to the optimum traffic distribution.

MPRTP flows share a common bottleneck: The previous subsection has shown that the scheduling algorithm tries to be fair to other streams by moving part of its media data to another link. We now evaluate the performance of the scheduling algorithm when it competes with another MPRTP flow on a common bottleneck. Path A and Path C are $1Mbps$ each and Path B is $800kbps$. Source 1 uses paths A and B, while Source 2 uses Path B and C. So Path B is shared between the two multipath sources. Furthermore, each individual path cannot carry the entire stream. At startup, the MPRTP senders allocate 50% traffic distribution on each of their paths: both endpoints cause congestion by sending about $500kbps$ each on Path B (capacity $800kbps$). Figure 10 shows the per-path bandwidth utilization including the common bottleneck link (Path B). Initially, Source 2 detects *congestion* and reduces its fractional traffic allocation for Path B. Meanwhile, Source 1 detects *mild congestion* because Source 2 has already reduced its load on the bottleneck link and therefore Source 1 gradually decreases its traffic load on Path B. Since Path B is not completely loaded after the initial period, it shows that the scheduling algorithm is capable of being *'fair'* to another MPRTP flow on the common bottleneck. Table 3 shows that the average PSNR and PLR for each media stream is comparable to the other and no stream benefits more than the other.

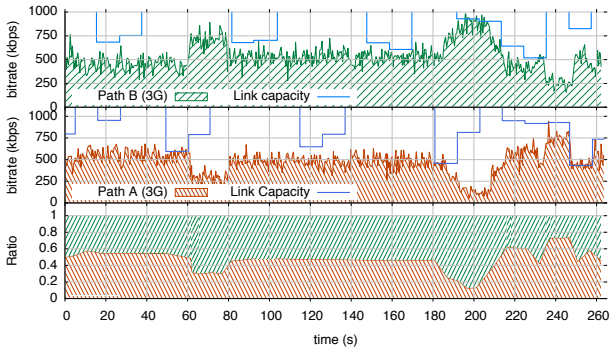


(a) Path A (3G link) capacity changes every 10s

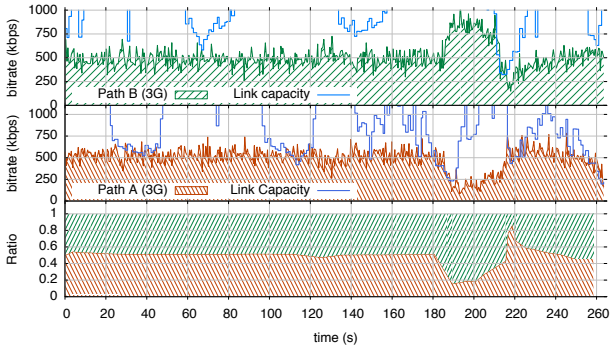


(b) Path A (3G link) capacity changes every 1s

Figure 11: MPRTP using WLAN and 3G paths



(a) Path capacity changes every 10s



(b) Path capacity changes every 1s

Figure 12: Using MPRTP to stream media over multiple 3G paths

Path Characteristic	Avg. PSNR	σ_{PSNR}	PLR
3G link with varying e2e capacity every 10s	42.483	0.551	0.85
3G link with varying e2e capacity every 1s	46.7173	0.21	0.33

Table 4: Performance of scheduling with WLAN and 3G paths

Path Characteristic	Avg. PSNR	σ_{PSNR}	PLR
Two 3G links with varying capacity every 10s	39.2680	1.9	1.41
Two 3G links with varying capacity every 1s	46.1704	0.18	0.95

Table 5: Performance of scheduling with multiple 3G links

6.4 MPRTP using WLAN and 3G paths

Many mobile devices have WLAN and 3G interfaces that can be exploited by MPRTP to deliver higher quality video streams. In addition, *wireless offloading*, i.e., shifting the load from the cellular network to a WLAN when a mobile device gets in range of a suitable WLAN access point would benefit from using MPRTP for changing the traffic allocation. YouTube on mobile/tablet uses RTSP and it alone accounts for 22% of the mobile data bandwidth [5]. Therefore, these devices would benefit from receiving media over multiple interfaces.

We consider the case of a multihomed server is delivering a live CBR video stream to a multihomed client over a WLAN and 3G path. The WLAN path provides $1Mbps$ e2e capacity and 0.25% loss rate. We emulate the 3G link using the throughput trace provided in [1, 42] and compare the performance in the case of quick (link rate changes at $1s$ intervals) and slow bandwidth changes (link rate changes at $10s$ intervals). The 3G link also emulates 1.0% bit-error losses based on traces [2] and the RLC is set to *unacknowledged mode* (RLC-UNACK) to keep link layer delays to a minimum [3]. For simplicity, the link latency on the WLAN and 3G path is kept constant.

Figure 11(a) shows the bandwidth utilization of each path and the overall bandwidth distribution between the paths. The 3G link is constrained and the scheduling algorithm does not quickly reallocate the bandwidth it took away from a link to avoid bandwidth oscillations. Moreover, the 3G link encounters packet losses more often than on the WLAN link, which makes the scheduling algorithm prefer sending more media over the WLAN than the 3G link. Alternatively, in Figure 11(b) the 3G link has higher capacity (on average) but varies more quickly. In this case, the bandwidth is more evenly shared except when the capacity on one path is limited and the load is moved to the other path.

In both the cases, the scheduling algorithm does not quickly reallocate the bandwidth it moved to another link but does this in a more controlled manner so that the load distribution does not oscillate. This is a useful feature for the scheduling algorithm because it can then use the passive or idle paths for fallback. Despite using two lossy paths the PSNR of the media stream (see Table 4) in this scenario is better than the single path with 0.5% and 1.0% loss rate (see Table 1).

6.5 MPRTP using Multiple 3G paths

Many live streaming events take place outdoors, oftentimes in places where broadband Internet is inaccessible. However, these places may have wireless 3G coverage but one 3G link may be insufficient to cater to the capacity needs of the video stream. Moreover, Internet connectivity to trains is become widespread and such systems often use multiple 3G access links (possibly complemented by WLAN or WiMax) to increase capacity, which could be used by onboard infotainment systems that are aware of the diverse interfaces. In either case, combining many 3G links to form one virtual (composite) 3G link, may be an option for real-time video (interactive or live) streams.

In this scenario, a video stream is sent over two 3G links from a server to a client where it is reconstructed and played back. The two links are different combinations of the throughput traces provided in [1,42] and the RLC is set to *unacknowledged mode* (RLC-UNACK) to keep link layer delays to a minimum [3]. The losses on the path vary (1.0% based on [2]) and variation in link latency between the two path is low. Like in the previous scenario, we choose slow ($10s$ intervals) and quick ($1s$ intervals) bandwidth changes to evaluate performance of the scheduling algorithm. One added constraint is that the sum of the instantaneous capacities of the two 3G links should be sufficient to carry the multimedia stream ($\approx 1Mbps$). We do not implement rate-switching, which may be used when the required capacity exceeds available capacity.

Figure 12(a) and (b) show the bandwidth utilization of each flow per path for $10s$ and $1s$ link rate changes. Initially, the scheduling algorithm keeps the per-path traffic distribution equal as long as each path is capable of carrying the individual subflows. However, as soon as one link becomes constrained, the scheduling algorithm offloads the rest of the media to the other link (around 180-200s in Fig. 12). The PSNR of the media stream (see Table 5) in the quick bandwidth scenario is better than the single path with 0.5% and 1.0% loss rate. And in the slow bandwidth change scenario it comparable to the single path with 1.0% loss rate.

6.6 Summary

Our evaluation shows that 1) for paths with static properties, multipath's performance is similar and comparable to that of the single path, 2) the scheduling algorithm tries to maintain proportional fairness, 3) when one path is congested the scheduling algorithm is able to offload the remaining media traffic to the other links, 4) by using an adaptive playout delay the receiver is able to compensate for paths with different latencies without any explicit notification from the sender, 5) MPRTP subflows can share a common bottleneck and be fair to each other, 6) on lossy links the multipath is more robust and produces fewer losses when compared to a single path with the same average loss rate.

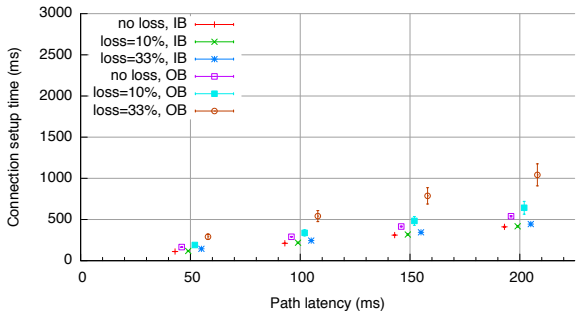
7. SYSTEM CONSIDERATIONS

MPRTP will not be used stand-alone but in the context of a complex system that comprises of signaling for setting up and tearing down media streams (such as SIP or RTSP) in today's Internet, which includes dealing with NATs and firewalls. Also, MPRTP will encounter non-MPRTP endpoints and have to interact with them, and multipath operation requires extending RTCP functionality for RTT calculation to be revisited. We will discuss these aspects in the following subsection.

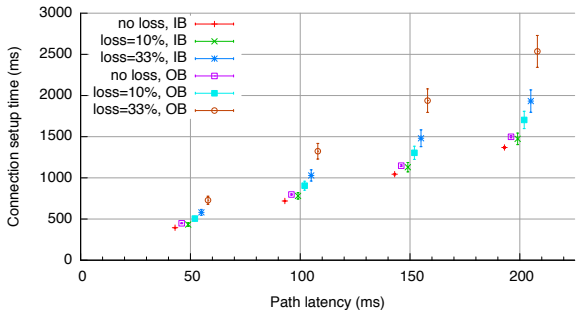
7.1 Call Establishment and NATs

At session startup and when new interfaces appear they can be advertised in-band (in RTP and/or RTCP) or out-of-band (in SDP).

In-band Interface Advertisement (IB): The RTP packets carry interface advertisements, i.e., RTP header extensions contain a set of additionally available interfaces. On receiving interface advertisements, a receiver responds with its own available interfaces in the RR, yielding a set of receive interfaces on each side [39]. For NAT traversal, both endpoints probe which of the remote interfaces they can reach and when enough of these *connectivity checks* succeed, the sender starts scheduling packets on multiple interfaces. We define new connectivity check messages in RTCP and send them at 20ms intervals (similar to STUN [36]). While sessions with bidirectional media stream provide interface information already in the first RTP packets, with unidirectional media, the receiver uses an RTCP extension for signaling, thus completing the exchange in one RTCP interval. However, the interface advertisement in RTCP is sent as a non-compound packet, i.e., if there is sufficient RTCP bandwidth then the endpoint can send the non-compound packet earlier than the next scheduled RTCP Report. Packet loss does not overly affect the connection setup time because the endpoint sends more frequent updates when it observes higher packet loss (see Fig. 7 in Section 5.2).



(a) Without connectivity-checks and NAT traversal



(b) With connectivity-checks for NAT traversal

Figure 13: Comparison of connection setup times for in-band (IB) and out-of-band (OB) call setup a) without NATs (i.e., no STUN connectivity checks [36]), b) with NAT traversal for different path latencies and link error rates. We run each call setup 100 times and the error-bars represents the 95% confidence-level.

Out-of-band Interface Advertisement (OB): An MPRTP attribute [44] in SDP [35] carries the MPRTP interface advertisement. If NAT traversal is not required then the endpoint sends the MPRTP interfaces in the first SDP Offer/Answer. However for NAT traversal, the endpoint first sends the ICE candidates in SDP [35] and the ICE agent performs the connectivity checks. After enough connectivity checks have succeeded, the endpoint advertises the MPRTP interfaces in SDP. Similarly, to advertise out-of-band in RTSP [20, 38] requires extensions to RTSP [44].

In both in-band and out-of-band call establishment, if the set of available interfaces changes, updated interface advertisements are sent (and the corresponding procedure repeats), i.e., the most recently received addresses override earlier ones.

In-band vs Out-of-band Setup: In-band call setup does not have to wait for a response from the other endpoint before sending media on the path. Typically, out-of-band call setup would require at least one additional RTT for SDP Offer/Answer procedures to succeed. This is especially useful when an interface (e.g., WLAN) at an endpoint routinely appears and disappears.

Figure 13 shows the comparison of connection setup times using in-band and out-of-band signaling for paths with different latencies and error-rates. If signaling packets are lost, the sender times out and re-transmits (typically, retransmission timeout is 500ms if an RTT estimate is unavailable) leading to longer call setup times.

Whereas in in-band, the endpoint creates redundancy by sending the interface advertisement in every RTP packet in the RTCP interval and this leads to a shorter call setup time.

In Figure 13(a), the endpoints do not run connectivity-checks[12] and the call setup times are much shorter than Figure 13(b), where endpoints run connectivity checks. Moreover, the call setup times for in-band is shorter than out-of-band for the same error-rates and increasing path latency. For instance in Figure 13(b), the connection setup at $100ms$ link latency for in-band setup with 10% loss rate is shorter than out-of-band with 0% loss rate.

7.2 Backward Compatibility

We did a series of simple tests using Gstreamer (legacy endpoint) and our MPRTP application. The following cases were successful: 1) Gstreamer to MPRTP over a single path, 2) MPRTP to Gstreamer over a single path, and 3) MPRTP to Gstreamer over multiple paths with similar path characteristics. However in 3), when the path characteristics are different, Gstreamer is able to playback packets from all the paths only if the path skew is smaller than the size of the dejitter buffer (which is typically 512 packets [18]). If the dejitter buffer is smaller than the path skew, Gstreamer just discards the packets from the slow path and does not play them out because they arrived late. Since a standard RTCP report [37] does not report packets discarded after arrival at the receiver, the MPRTP scheduler will not be able to adjust the fractional distribution.

7.3 RTT measurements

The media sender sends an RTCP SR on each active path. For each SR the receiver gets, it echoes one back to the same IP address-port pair that sent the SR. Therefore, the receiver tries to choose the symmetric path (based on the same 5-tuple) and if the routing uses the same return path then the per-path RTT calculations will work out correctly. However, if the paths are not symmetric, the sender would at maximum, under-estimate the RTT of the path by a factor of half of the actual path RTT. The scheduling algorithm should therefore not depend solely on RTT as an indicator for scheduling.

7.4 MPRTP Scheduling in the Real World

We measure the performance of the scheduling algorithm by initiating a live video stream between a multi-homed server (WLAN, Ethernet$\times$2, 3G) hosted at the university and a laptop computer (WLAN, 3G). We observe varying results between each successive result because of varying amounts of cross traffic on the public Internet. Even though we observe diverse results, it shows that MPRTP endpoints can operate on the public Internet.

8. RELATED WORK

A lot of work has gone into multihoming and multipath at the transport layer: SCTP [45], Multipath TCP (MPTCP) [49], and bandwidth aggregation for mobile hosts [23, 30], but these do not consider real-time properties. On the theoretical aspects, [16, 22, 27, 40, 50] provide the foundation and reference simulation setups for multipath transmission.

Specifically for multimedia, Liang *et al.* [28] show that transmitting redundant voice traffic over multiple paths perform better than a FEC protected single stream. While Chesterfield *et al.* [12] show that by sending media over one 3G interface and Unequal Protection (UEP) packets over a separate 3G interface can compensate for losses on the first path. To minimize bursty losses, [7] recommends sending odd and even frames over multiple paths. Westwood SCTP-PR [17] balances real-time media traffic using a bandwidth-aware scheduler but uses reliable transport.

Chebrolu *et al.* [11] propose bandwidth aggregation for multimedia applications by computing the earliest delivery time for each packet. They further propose to drop less important frames (e.g., B-frames) if the available capacity is smaller than the current encoding rate [10]. In our media stream, we do not use B-frames and do not discard any packets at the sender. Furthermore, we try to maintain optimal playout by choosing paths that meet the latency constraints and try to maintain a very short de-jitter buffer ($\leq 500ms$), so that the scheduling algorithm can be extended to include interactive applications. Jurca *et al.* [25] propose a frame-aware scheduling algorithm that sends key-frames and other important media packets over less lossy paths and this approach is similar to the one proposed in this paper. However, they also propose sending future packets over high latency paths by reading ahead in the media stream. While this is an interesting concept, it would require larger buffers and more state at the sender (typically, RTSP servers) to read ahead the stored media stream and would not work for interactive and live video streams where it cannot read ahead. While the proposed solution in this paper does not do congestion control, it borrows a lot of ideas from [13, 25, 29, 34, 47], for sensitivity towards reported network cues to perform load balancing.

9. CONCLUSION

We have presented MPRTP, a backwards-compatible extension to RTP that spreads packets of a media stream across a number of different paths, that can be discovered and set up dynamically within an RTP session. We have explored the criteria for assigning traffic shares as a function of the diverse path properties and presented considerations for scheduling algorithms. Our evaluation using our complete protocol implementation shows that our design 1) allows exploiting multiple paths without performance degradation compared to suitable single-path cases—so that it is safe to deploy—and 2) enables load distribution and capacity aggregation in diverse scenarios. Mobile users (and operators) may benefit from aggregating or dynamically shifting load between different wireless interfaces and MPRTP may assist well in bundling multiple wireless access networks for vehicular Internet access.

While our initial findings are promising, further work is needed to cover a broader range of media encoding rates, more sophisticated network setups with diverse cross traffic and on the protocol side, we seek to integrate MPRTP with congestion control (e.g., [6, 34, 43]).

10. REFERENCES

[1] 3GPP R1-081955. LTE Link Level Throughput Data for SA4 Evaluation Framework., May 2008.

[2] 3GPP S4-050560. Software Simulator for MBMS Streaming over UTRAN and GERAN, Sept. 2005.

[3] 3GPP TR 26.902. Video Codec Performance, Jan. 2008.

[4] S. Akhshabi, A. C. Begen, and C. Dovrolis. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP. In *Proc. of MMSys*, 2011.

[5] Allot MobileTrends. Global Mobile Broadband Traffic Report, 07 2011.

[6] H. Alvestrand, S. Holmer, and H. Lundin. A Google Congestion Control Algorithm for RTCWEB, 2011. IETF Draft, draft-alvestrand-rtcweb-congestion.

[7] J. Apostolopoulos. Reliable Video Communication over Lossy Packet Networks using Multiple State Encoding and

[12] setup assumes that at least one entity has a globally addressable IP address (in use-cases where the endpoint connects to an IPTV distribution, live streaming server, etc.)

Path Diversity. In *Proc. of IEEE VCIP*, 2001.

[8] F. Baker. Exploring the multi-router SOHO network, 2011. IETF Draft, draft-baker-fun-multi-router.

[9] E. Brosh, S. A. Baset, D. Rubenstein, and H. Schulzrinne. The Delay-Friendliness of TCP. In *Proc. of ACM SIGMETRICS*, 2008.

[10] K. Chebrolu and R. Rao. Selective frame discard for interactive video. In *Proc. of IEEE ICC*, volume 7, june 2004.

[11] K. Chebrolu and R. Rao. Bandwidth aggregation for real-time applications in heterogeneous wireless networks. *IEEE Transactions on Mobile Computing*, 5(4), april 2006.

[12] J. Chesterfield, R. Chakravorty, I. Pratt, S. Banerjee, and P. Rodriguez. Exploiting diversity to enhance multimedia streaming over cellular links. In *Proc. of IEEE INFOCOM*, 2005.

[13] S. Choi and M. Handley. Designing TCP-Friendly Window-based Congestion Control for Real-time Multimedia Applications. In *Proc. of PFLDNeT*, 2009.

[14] T. E. Chown. Home Networking Architecture for IPv6, 2012. IETF Draft, draft-ietf-homenet-arch.

[15] Cisco. Entering the Zettabyte Era. Whitepaper, June 2011.

[16] Y. Dong, D. Wang, N. Pissinou, and J. Wang. Multipath load balancing in transport layer. In *Proc. of IEEE EuroNGI*, 07.

[17] M. Fiore and C. Casetti. An adaptive transport protocol for balanced multihoming of real-time traffic. In *Proc. of IEEE GLOBECOM*, 2005.

[18] D. Fober, Y. Orlarey, and S. Letz. Real time clock skew estimation over network delays, 2005.

[19] Ford, A., Raiciu, C., and M. Handley. TCP Extensions for Multipath Operation with Multiple Addresses, 2011. IETF Draft, draft-ietf-mptcp-multiaddressed.

[20] J. Goldberg, M. Westerlund, and T. Zeng. A NAT Traversal mechanism for media controlled by RTSP, 2012. IETF Draft, draft-ietf-mmusic-rtsp-nat.

[21] M. Handley, C. Raiciu, and M. Bagnulo. Outgoing Packet Routing with MPTCP, 2009. IETF Draft, draft-handley-mptcp-routing.

[22] Y. Hasegawa, I. Yamaguchi, T. Hama, H. Shimonishi, and T. Murase. Improved data distribution for multipath tcp communication. In *Proc. of IEEE GLOBECOM*, Dec 2005.

[23] H. Hsieh and R. Sivakumar. A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts. *Wireless Networks*, 11(1), 2005.

[24] I. Johansson and M. Westerlund. Support for Reduced-Size RTCP: Opportunities and Consequences. RFC 5506, 2009.

[25] D. Jurca and P. Frossard. Video packet selection and scheduling for multipath streaming. *IEEE Transactions on Multimedia*, april 2007.

[26] F. Kelly and T. Voice. Stability of end-to-end algorithms for joint routing and rate control. *ACM SIGCOMM CCR*, 2005.

[27] P. Key, L. Massoulié, and P. Towsley. Path selection and multipath congestion. In *Proc. of IEEE INFOCOM*, 07.

[28] Y. J. Liang, E. G. Steinbach, and B. Girod. Multi-Stream Voice over IP Using Packet Path Diversity. In *Proc. of MMSP*, 2001.

[29] F. Licandro and G. Schembra. A rate/quality controlled MPEG video transmission system in a TCP-friendly Internet scenario. In *Proc. of IEEE Packet Video*, 2002.

[30] L. Magalhaes and R. Kravets. Transport level mechanisms for bandwidth aggregation on mobile hosts. In *Proc. of IEEE ICNP*, 2001.

[31] J. Ott, I. Curcio, and V. Singh. RTP Control Protocol (RTCP) Extended Reports (XR) for Run Length Encoding (RLE) of Discarded Packets, 2011. IETF Draft, draft-ietf-xrblock-rtcp-xr-discard-rle-metrics.

[32] J. Ott, S. Wenger, N. Sato, C. Burmeister, and J. Rey. Extended RTP Profile for RTCP-Based Feedback (RTP/AVPF). RFC 4585, 2006.

[33] C. Perkins. *RTP: audio and video for the internet*. Addison-Wesley Professional, first edition, 2003.

[34] C. Perkins and V. Singh. RTP Congestion Control: Circuit Breakers for Unicast Sessions, 2012. IETF Draft, draft-ietf-avtcore-rtp-circuit-breakers.

[35] J. Rosenberg. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. RFC 5245, 2010.

[36] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing. Session Traversal Utilities for NAT (STUN). RFC 5389, 2008.

[37] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550, 2003.

[38] Schulzrinne, H., Rao, A., Lanphier, R., Westerlund, M., and M. Stiemerling. Real time streaming protocol 2.0 (rtsp), 2011. IETF Draft, draft-ietf-mmusic-rfc2326bis.

[39] V. Singh, T. Karkkainen, J. Ott, and S. Ahsan. Multipath RTP (MPRTP), 2012. IETF Draft, draft-singh-avtcore-mprtp.

[40] V. Singh and J. Ott. Evaluating congestion control for interactive real-time media., 2012. IETF Draft, draft-singh-rmcat-cc-eval.

[41] V. Singh, J. Ott, and I. Curcio. Rate adaptation for conversational 3G video. In *Proc. of INFOCOM Workshop on MoViD*, Brazil, 2009.

[42] V. Singh, J. Ott, and I. Curcio. Predictive Buffering for Streaming Video in 3G Networks. In *Proc. of IEEE WoWMoM*, Jun 2012.

[43] V. Singh, J. Ott, and I. Curcio. Rate-control for Conversational Video Communication in Heterogeneous Networks. In *Proc. of IEEE WoWMoM*, Jun 2012.

[44] V. Singh, J. Ott, T. Karkkainen, R. Globisch, and T. Schierl. Multipath RTP (MPRTP) attribute in Session Description Protocol, 2012. IETF Draft, draft-singh-mmusic-mprtp-sdp-extension.

[45] R. Stewart. Stream Control Transmission Protocol. RFC 4960, 2007.

[46] F. Valera, I. van Beijnum, A. Garcıa-Martınez, and M. Bagnulo. Multi-path bgp: motivations and solutions. *NGI: Architectures and Protocols*, 2011.

[47] J. Viéron and C. Guillemot. Real-time constrained TCP-compatible rate control for video over the Internet. *IEEE Transactions on Multimedia*, 6(4), August 2004.

[48] D. Wischik, M. Handley, and M. B. Braun. The resource pooling principle. *SIGCOMM CCR*, 38, September 2008.

[49] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, Implementation and Evaluation of Congestion Control for Multipath TCP. In *Proc. USENIX NSDI*, 2011.

[50] M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson, and R. Wang. A transport layer approach for improving end-to-end performance and robustness using redundant paths. In *Proc. of USENIX ATC*, 2004.

[51] M. Zink, O. Kuenzel, J. Schmitt, and R. Steinmetz. Subjective impression of variations in layer encoded videos. In *Proc. of IWQoS*, 2003.