A Transport Protocol to Exploit MultiPath Diversity in Wireless Networks

V. Sharma, K. Kar, *Member, IEEE*, K. K. Ramakrishnan, *Fellow, IEEE*, and S. Kalyanaraman, *Fellow, IEEE*

***Abstract*—Wireless networks (including wireless mesh networks) provide opportunities for using multiple paths. Multihoming of hosts, possibly using different technologies and providers, also makes it attractive for end-to-end transport connections to exploit multiple paths. In this paper, we propose a multi-path transport protocol, based on a carefully crafted set of enhancements to TCP, that effectively utilizes the available bandwidth and diversity provided by heterogeneous, lossy wireless paths. Our Multi-Path LOss-Tolerant (MPLOT) transport protocol can be used to obtain significant goodput gains in wireless networks, subject to bursty, correlated losses with average loss-rates as high as 50%. MPLOT is built around the principle of separability of reliability and congestion control functions in an end-to-end transport protocol. Congestion control is performed separately on individual paths and the reliability mechanism works over the aggregate set of paths available for an end-to-end session.**

MPLOT distinguishes between congestion and link losses through Explicit Congestion Notification (ECN), and uses Forward Error Correction (FEC) coding to recover from data losses. MPLOT uses a dynamic packet mapping based on the current path characteristics to choose a path for a packet. Use of erasure codes and block level recovery ensures that in MPLOT, the receiving transport entity can recover all data as long as a necessary number of packets in the block are received, irrespective of which packets are lost. We present a theoretical analysis of the different design choices of MPLOT, and show that MPLOT chooses its policies and parameters such that a desirable tradeoff between goodput with data recovery delay is attained. We evaluate MPLOT, through simulations, under a variety of test scenarios and demonstrate that it effectively exploits path diversity in addition to efficiently aggregating path bandwidths, while remaining fair to a conventional TCP flow on each path.

***Index Terms*—Transport protocols, TCP, multi-hop wireless networks, lossy environments, multi-path, diversity gain.**

I. Introduction

THE past decade has witnessed a tremendous interest and growth in wireless networking. Wireless mesh networks are being used to provide communication access to remote areas, and increasingly being deployed in domains traditionally dominated by wired networks. Projects like AT&T Metro Wifi, Google Wifi and municipal deployments seek to replace traditional wired backhaul networks with multi-hop wireless networks. Wireless networks (especially meshed networks) provide increased opportunities to establish multiple paths between source-destination pairs. End-systems that are multi-homed (i.e., can make use of multiple access technologies or service providers for connectivity to the network), or capable of directional transmission, have additional opportunities of using multiple paths in parallel. A transport protocol can potentially use this inherent path-diversity to overcome the limitations imposed by a single path, by transmitting data over multiple paths.

The dominant transport protocol used for reliable end-to-end data transfer – TCP – was designed for wired networks, primarily using a single end-to-end path. Wireless link characteristics can differ from a wired link in fundamental ways. Transmissions over wireless links can be blocked by environmental barriers (e.g., walls), interfere with other transmissions and experience attenuation or shadowing losses. As a result, wireless links exhibit higher bit-error rates than wired links, reflected as high packet loss rates at the transport layer. Recent studies on IEEE 802.11b networks [1] have reported pre-ARQ packet losses as high as 50%, even for static community networks. Link-level ARQ/Hybrid-ARQ (HARQ) schemes can in general reduce these loss rates significantly, usually with targets of 10%. However, due to limits on the latency (or the number of retransmission attempts) at the link layer, the residual (post-ARQ) loss rates can be high as well, particularly in scenarios involving mobility. For example, for communication among fast-moving ground vehicles, it has been observed that post-ARQ loss rates can be high (20% or more) and also have high variance [2], [3]. Such high packet loss rates have serious implications on performance of TCP when used over wireless networks.

In addition to the poor performance under loss, TCP's design intricately couples flow and congestion control with reliability. A small number of out-of-order packets delivered to the destination transport (above a threshold number) results in the triggering of a congestion response from TCP. A desirable characteristic of an end-to-end transport protocol is that it should provide flow-controlled, reliable delivery of data while being able to exploit the availability of multiple paths. Such a protocol should aggregate capacities across multiple, lossy paths and leverage the diversity across different paths to yield stable, high goodput and low latency, while still providing reliable packet delivery. The implicit assumptions that exist in the design of TCP however prevent it from effectively using multiple paths.

In this paper, we present MPLOT, the *Multi-Path LOss tolerant Transport protocol*, to attain the above mentioned goals. MPLOT effectively separates reliability and congestion control functions in an end-to-end transport protocol by organizing

V. Sharma and K. Kar are with the Department of Electrical and Computer Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180, USA. Email:{sharmv, kark}@rpi.edu.

S. Kalyanaraman is with IBM Research - India, Manyata Tech Park, Bangalore 560045, India. Email:shivkuma@gmail.com.

K. K. Ramakrishnan is with AT&T Labs Research, Florham Park, NJ 07932, USA. Email:kkrama@research.att.com.

reliability across paths while performing congestion control on a per-path basis. MPLOT employs a combination of Forward Error Correction (FEC) coding and adaptive HARQ to reduce the number of packet re-transmissions, and exploits Explicit Congestion Notification (ECN) to differentiate between congestion and link losses. MPLOT dynamically maps packets that may be required immediately at the destination onto paths currently experiencing good conditions (low losses, delay and high capacity), while mapping packets required 'less urgently' to paths with poorer conditions. This is a significant departure from previously proposed protocols that perform packet-to-path mapping that is independent of conditions on the path.

In Section II we outline and motivate the overall design of MPLOT. In Section III, we theoretically analyze the associated performance tradeoffs, towards making good choices of key policy and protocol parameters. We present and discuss our simulation study in Section IV. In Section V, we discuss related approaches that address high packet losses or make use of multiple paths. We conclude in Section VI.

II. MPLOT: Overall Design

In this section, we describe the overall design of Multi-Path Loss-Tolerant TCP (MPLOT) protocol, and outline the functionality of its major components. In Section III we theoretically analyze the protocol with the goal of determining good choices for key policies and parameters.

MPLOT enhances the two key functions of a typical transport protocol to exploit multiple end-to-end paths – *reliability* and *congestion control*. The reliability mechanism of MPLOT guarantees delivery of data in the presence of losses and bit errors, through the use of adaptive, erasure coding – FEC[1]. The standard TCP-like congestion control mechanism ensures that that the protocol reacts to queue buildup in the network, and is fair to single-path (TCP) flows, through the use of ECN. However, the reliability and congestion control functions (components) are implemented separately, and differently: reliability is implemented on a transport connection basis whereas congestion control is implemented on a per-path basis. This is illustrated in Figure 1. In addition to these two components, the source at the connection level also implements a *packet mapper*, which maps the erasure coded packets (data and FEC) created by the reliability mechanism to the different paths; the packets mapped to a path are then released on to the individual path according to the congestion control mechanism. Next, we describe how these components work in more detail.

A. Block Erasure Coding & Proactive/Reactive FEC

In MPLOT, reliability is achieved by using an aggregate flow manager, by combining the transmissions and acknowledgements (ACKs) across all the paths used by the transport, on a per connection (also termed flow, in this paper) basis. To counter estimated packet losses, the source adds redundancy (FEC) to a transport level flow through *block erasure coding* of packets. In our approach, the data in the socket buffer is divided into groups, and these groups of data bytes are encoded independent of each other. Consider a group of data bytes organized into F data packets. The encoder takes the F data packets, and adds K FEC packets to create a coded *block* of size $B = (F + K)$ packets. All packets (data or FEC) of a block are made to be of equal size, although packet size can vary across blocks. In the following, we assume Reed-Solomon coding for creating the FEC packets,[2] although our analysis and results extend to other Maximum Distance Separable (MDS) erasure codes. Of the K FEC packets in the block, $k \leq K$ packets, which we refer to as *proactive FEC (PFEC)*, are sent along with the data packets in the block. Intuitively, the purpose of the PFEC packets is to proactively counter potential packet losses in the channel. The set of data and PFEC packets of the block, which constitutes the initial (proactive) transmission, is referred to as the *p-block*. Based on properties of Reed-Solomon block erasure codes, all of the F data packets of the block can be recovered as long as any *unique* F packets (data or FEC) of that block are received. In other words, the destination node can recover all the data packets of the block, if at most k packets of the p-block (data or PFEC) are lost in transit. If $k' > k$ packets of the block are lost in transit, then an additional $(k' - k)$ packets must be received *from the same coded block* to be able to decode all data in the block. This is achieved by having the source send additional FEC packets from the $(K - k)$ packets remaining in the block, which we term *reactive FEC (RFEC)* packets. This set of $(K - k)$ FEC packets kept in "reserve" and sent reactively, is called the *r-block*. The number of RFEC packets to be sent is determined based on how many additional packets are needed for block data recovery, and the estimated loss rate, which are in turn determined from the *Selective ACK (SACK)* feedback sent by the destination. (The design of the SACK feedback is described later in more detail.) RFEC packets can be sent redundantly, i.e., more than the minimum necessary or $(k' - k)$, to account for possible losses in transit. If these RFEC packets are still not enough for the destination to recover all data of the block (due to more losses of RFEC packets than what was accounted for), additional rounds of RFEC transmission may be needed. Note that in this approach, 'retransmissions' use "new" FEC packets from the r-block. Specifically, when data or FEC packets from the p-block are detected to be lost, then new FEC packets from the r-block are used to recover from the lost packets. Similarly, when FEC packets from the r-block previously transmitted are lost, these are also recovered using new FEC packets from the r-block. However, if the entire r-block is exhausted before the block-data is recovered fully, the transmission "wraps around" to the beginning of the block, and continues as before, starting from the first packet of the p-block. This wrap-around may result in duplication in the data received, which is detected by having sequence numbers in the packets indicating their position in the block. This is discussed in detail later. We use $GF(2^8)$

[1]In MPLOT, FEC is generated through erasure coding of *blocks* of packets, to aid recovery in presence of packet losses (erasures). This should be distinguished from FEC computed on individual packets using channel coding, to recover from bit errors.

[2]For ease of discussion, we implicitly assume *systematic* Reed-Solomon encoding, which retains the original data packets and creates the encoded block by adding FECs to it; our protocol, analysis and results however applies to non-systematic encoding as well.

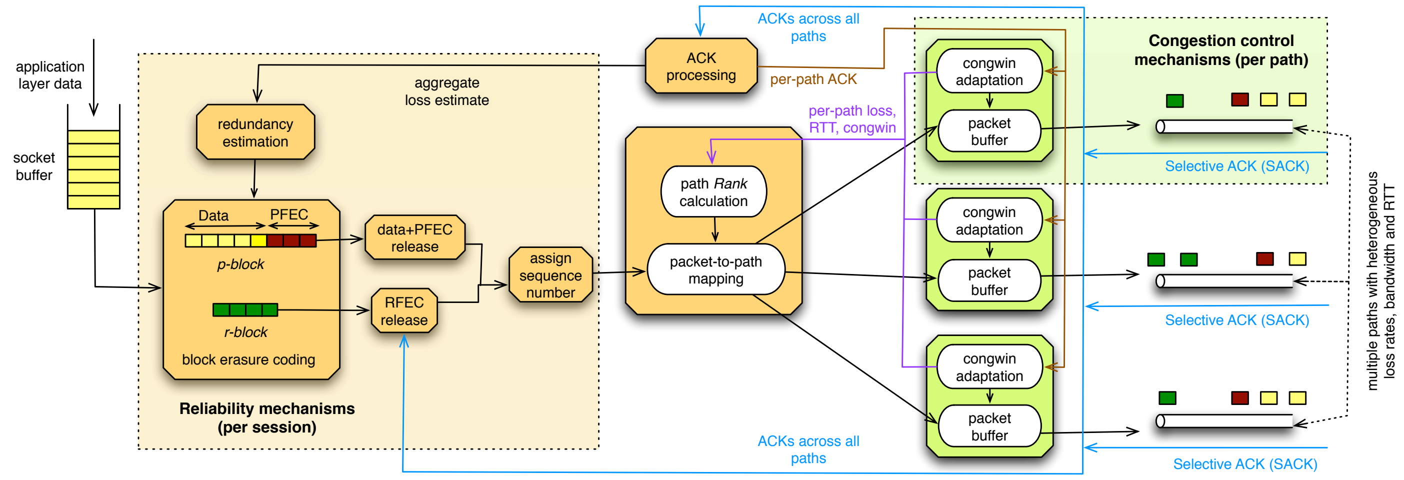


Fig. 1. Overall design of MPLOT (sender side) and key components.

for the block erasure code computation, which allows byte arithmetic, and limits the block size to a maximum of 255 packets. This block size must be apportioned appropriately into the p-block and r-block sizes, taking into account the packet loss rates, so that the block-data is recovered with high probability before the entire r-block is exhausted.

An alternative retransmission (RFEC transmission) policy would be to resend the data and PFEC packets that were lost as RFEC packets. With this approach an additional reserve of FEC packets need not be maintained, and therefore the size of the p-block can be the same as the block size. In this approach, the number of data packets in a block can be larger, for the same size of the coded block. This approach requires a more complex ACK mechanism, however, as the sender has to detect *which* packets in the block were lost; in the first approach described above in which only new FEC packets are sent as RFECs, the sender only needs to know *how many* packets of the block have been received so far. As we describe shortly, since MPLOT already uses SACK maps anyway (necessary to obtain good loss estimates, as discussed in Section III-A), the ACK mechanism necessary to implement this retransmission approach is already built into MPLOT: the SACK maps could be used to pin-point which packets (data or PFEC) were lost, and they could be used in retransmission.

In our experimentation and evaluation, we have used the first retransmission approach – that of maintaining an r-block of FEC packets, and sending new packets from this r-block for retransmission until all packets in the r-block gets exhausted – for the following reasons. Firstly, note that this retransmission policy can thus be viewed as an *incremental redundancy* HARQ approach, where each retransmission (RFEC transmission, in our case) uses a different set of coded bits than the previous transmission, so that the receiver can obtain new information with each packet it receives. This approach maximizes the benefits from the retransmission, as it implies that the retransmitted packet would be useful in data recovery even if the packets that were determined as lost (that prompted the retransmission) would appear later at the receiver (i.e., in case packet loss was incorrectly assumed due to delayed packet delivery). Since we can have multiple blocks in transit simultaneously, we observed that the reduced number of data packets in a block that we have in this retransmission approach did not pose a serious performance issue (MPLOT would scale up the number of blocks in transmit simultaneously to fill up the congestion window). Finally, since MPLOT adopts a somewhat conservative PFEC allocation policy (taking into account both mean and standard deviation of the loss rate, as we discuss in Section III-B), we have observed that most of the lost data gets recovered by the PFEC; in other words, the number of RFEC packets that need to be transmitted for block recovery is typically quite small, and even with a modest size of the r-block, we rarely exhaust all the FECs kept in reserve before the block-data is fully recovered.

Note that since packets for a given flow (transport connection) are sent across multiple paths, they may be received at the destination out-of-order. The aggregate flow manager at the destination must recover and reorder them appropriately to deliver them in sequence, in order, to the application layer. Note that all the F data packets in a block must be buffered before encoding the block at the source. Similarly, packets of the block must be buffered at the destination until F packets of the block are received; then the block-data is ready to be decoded and sent to the application layer. The key parameters – the size of the p-block, $B^{\mathrm{p}} = F + k$ $(\leq F + K = B)$, the fraction of PFEC packets in the p-block $(k/B^{\mathrm{p}} = k/(F+k))$, the redundancy in the number of RFECs sent in each round, and to a limited extent the size of each packet in the block – are dynamically chosen; in Section III we analyze and justify how these choices are made.

B. *Protocol Header, Sequence Numbers & Feedback Design*

While the sender is waiting for recovery of the data of one block, if transmission opportunities are available (i.e., there is space in the sender's individual path windows), the packets from the next block are transmitted. Thus there can be multiple blocks in transit at the same time. However, since the p-block size is scaled up to the (weighted) sum of congestion windows over all paths (see Section III-C), subject to block size limits imposed due to the erasure coding field size as discussed in Section II-A, we observe in our simulations that the number of blocks in transmission simultaneously at any time is typically small (i.e., unless the bandwidth-delay product of the paths is very large).

The aggregate reliability mechanism described above can be implemented using the same acknowledgment/feedback structure as provided in standard TCP. The Selective ACK (SACK) feedback that MPLOT uses is similar to that used by TCP, with the SACK option enabled (TCP-SACK). In MPLOT, this SACK feedback, as well as the cumulative ACK (included by default in the TCP header), are provided *on an aggregate (per flow) basis*. This feedback allows the destination to indicate "holes" in the packet stream, i.e., groups of packets that it has not received so far. This is used by the source to identify packet losses and determine how much RFEC should be sent, and when.

MPLOT assumes the default TCP header, and availability of the SACK option. In addition, the following information fields need to be included in each packet header (data or FEC): (i) block id, (ii) block size (in packets), (iii) number of data packets in the block, (iv) sequence number of the packet in the block. These fields are used to convey necessary information in the forward direction (i.e., from the sender to the receiver). Simple calculations reveal that 1-2 bytes per field is sufficient for use in practice. The block size $(F + K)$ along with the number of data packets in the block (F) are parameters that are need for the block-data decoding process. Note that the *packet* sequence numbers mentioned in (iv) indicate the position of the packets in the block (and are therefore reused across blocks), and are different from the *byte* sequence numbers that are part of the TCP header. PFEC packets get assigned sequence numbers that follow those of the data packets in the block; RFEC packets naturally get assigned sequence numbers that are greater than the p-block size.

Byte sequence numbers (as in the default TCP header) are assigned on a *per session (aggregate flow)* basis, and are assigned when the packets are sent out. All packets (data, PFEC or RFEC) sent out are assigned distinct sequence numbers; in other words, byte sequence numbers are not repeated, even in the case some packet is retransmitted (which may happen if the end of the block is reached before the block-data is recovered, requiring the sender's transmission to wrap around to the beginning of the block). These byte sequence numbers are used for determining the cumulative ACK and SACK information included in the ACK packets sent by the receiver to the sender, which are also calculated on a per session (*not* per path) basis, naturally. The cumulative ACK field represents one more than all bytes received in order for all "outstanding" blocks, i.e., blocks that are known to the receiver as being in transit (in other words, blocks for which the receiver has received at least one packet), but whose data have not been fully recovered yet. For each packet (sequence number identified by the packet) sent, the sender keeps track of (i) which path it is sent on, and (ii) which block id it corresponds to. The former piece of information is used to determine the path-specific loss and RTT information from the cumulative ACK and SACK feedback, which in turn is used to provide self-clocking and window adaptation as in per-path TCP (refer to Figure 1). The latter information is used to determine how many unique packets corresponding to each outstanding unrecovered block have been delivered successfully to the receiver so far, and thus determine whether and how many RFEC packets should be sent for the block.

From the above, we observe that MPLOT uses a HARQ/FEC approach to ensure reliable delivery of data – FEC is used proactively along with the data transmission, and reactively in response to feedback of the delivery status. This is in contrast with TCP which uses an ARQ-only approach for data recovery. Furthermore, whereas TCP requires *specific* segments to be retransmitted on loss, with MPLOT delivery of *any* (unique) F data or FEC packets from the block suffices for block-data recovery. This *sequence-agnostic* property of erasure codes helps reduce the number of transmission rounds required for data recovery for a given packet loss (erasure) rate, thereby improving goodput and latency.

Our hybrid approach also allows us to attain a desirable goodput-latency tradeoff. A greater degree of redundancy, in terms of the PFEC to data packet ratio in the p-block, or the amount of over-provisioning in the RFEC transmissions, will in general result in fewer rounds (round trip delays) for block-data recovery, or lower latency. However, greater redundancy also reduces goodput due to higher FEC overhead per packet. Intuitively, however, it is easy to observe that the fraction of PFEC packets in a p-block, and the degree of redundancy in the number of RFEC packets sent in any round, should be related to the overall loss rate experienced by the flow (aggregated across all paths). Our analysis of the PFEC/RFEC allocation policy, as described in Section III, captures this intuition and related tradeoffs.

C. Design of Per-path Congestion Control

We recognize the fact that different paths may have different characteristics or temporarily experience different conditions (e.g., different bandwidths, cross-traffic, end-to-end delay etc.). Responding to congestion at the aggregate level will result in performance levels dominated by the worst (or slowest) path. Consequently, congestion control is performed by each path, independent of the other paths.

Each path i maintains the usual congestion control variables: congestion window size (w_i), round trip time estimate (RTT_i) and a timeout value (RTO_i). Recall that ACKs in MPLOT is provided on an aggregate (per flow basis), irrespective of which reverse path the ACK is sent on. The latest aggregate feedback is sent by the destination upon receipt of every packet (on any path). An interesting feature of MPLOT is that it allows the ACK packets to be sent back (by the destination to the source) on *any reverse path* – not necessarily the path on which the last packet was received by the destination (which may have prompted the destination to send the ACK). In MPLOT, the destination sends the ACK on a reverse path chosen uniformly at random, among all available reverse paths. Choosing reverse paths for the ACKs at random (as opposed to choosing a specific reverse path) increases the robustness of the protocol to large delays and failures on individual paths.

The motivation for this is to reduce the overall round-trip time. For a specific path, the aggregate selective/cumulative ACK feedback is filtered to obtain the delivery status of the packets sent on that path; this is done by the ACK processing component as shown in Figure 1. This path-specific ACK information is then used to slide the congestion window of the path (in accordance with the TCP self-clocking mechanism). MPLOT adapts the per-path congestion window size w_i based on ECN – this is done as specified in the standard for TCP using ECN [4]. With use of ECN, packet losses (which could be due to poor channel quality or link disruptions as well, in addition to congestion) need not be used as the congestion signal. Instead the congestion window could be reduced only in response to ECN feedback, which provides an unambiguous indication of congestion. In MPLOT, the destination maintains the latest ECN information for each (forward) path. While sending an ACK on a reverse path, the destination copies the ECN bit of the corresponding forward path. Note that we modify TCP's congestion window size adaptation by dividing the window increase of each path i by its packet "acceptance rate" ($(1-p_i)$), where p_i is the estimated packet loss rate on path i) to account for the packets lost on that path.

A path congestion controller will timeout if no feedback is received for a period RTO_i. Hence, timeouts are handled separately for each path. The response to a timeout is identical to conventional TCP response to a timeout. Unlike TCP-SACK, MPLOT does not respond to duplicate-acks (dupacks) at the aggregate level, i.e., MPLOT does not perform fast-retransmit after a certain number of dupacks are received. Fast retransmit on dupacks (duplicate cumulative ACKs) would typically lead to a lot of unnecessary transmissions. Fast retransmit can be made more efficient if SACK information is taken into account. However, since MPLOT already adds FEC protection against losses, fast retransmit of lost packets may be redundant anyway, and is therefore avoided. As described in Section II-B however, MPLOT implements a fast-retransmit like mechanism at the block level (for fast retransmit of RFEC packets), taking into consideration block erasure coding of the packets.

D. Packet Mapping: From Aggregate Block to Path Windows

In general, unless the p-block size is allowed to scale up arbitrarily – which has the undesirable effect of making the coding/decoding delays very large – a single block may not be sufficient to fill up the available capacity over all paths. Therefore, MPLOT allows multiple outstanding blocks to be in transit at any time. The packets of all these blocks must be mapped to the paths that the aggregate connection/flow uses. As different paths may have different RTTs, capacities and loss-rates, this mapping must be done carefully. Intuitively, the mapping should be such that packets of the earliest unrecovered blocks arrive at the destination quickly with a high probability. In other words, packets that are required earlier should be mapped to "better" paths, i.e., paths that have higher bandwidth, lower loss and shorter RTT. We capture the notion of the "goodness" of a path through a *rank* function can be viewed as the *estimated loss-free bandwidth* of a path. More precisely, the rank of a path i, denoted π_i, is defined as

$$\pi_i = \frac{w_i(1-p_i)}{RTT_i}. \tag{1}$$

Symbol(s)	Meaning
w_i, RTT_i, BW_i	cong. window, RTT on path i; $BW_i = w_i/RTT_i$
$p_{\text{agg}}, \hat{p}_{\text{agg}}, \hat{\sigma}^2_{\text{agg}}$	aggr. loss rate, and its EWMA estimates (avg., var.)
F, k, B^{p}	# data and PFEC packets in a block; $B^{\text{p}} = F + k$

TABLE I
IMPORTANT NOTATION.

This rank function assigns a higher rank to paths with larger windows (w_i), lower path loss rates (p_i) or shorter round trip times (RTT_i). In Section III-D, we formally justify this ranking function through analysis, and describe the exact mapping function that MPLOT uses. Broadly speaking, mapping packets to paths in MPLOT is done using a few simple, intuitive rules: (i) paths with higher rank are preferred over paths with lower rank; (ii) if packets of multiple blocks are to be sent simultaneously, packets of earlier unrecovered blocks are preferentially mapped to paths of higher rank; (iii) a packet to be sent is mapped to a path only if the congestion window of the path has a transmission opportunity; otherwise, the next ranked path is considered. Note that due to the temporal equivalence between packets of the same block (again due to the sequence-agnostic property of erasure codes), for rule (ii) above, consideration of the temporal ordering of the blocks suffices (i.e., it is not necessary to consider the temporal order between the packets of the same block). Finally note that our choice of the packet mapping function is designed to minimize block recovery time, as we formally justify in Section III-D. A different packet mapping policy can be plugged in if needed, without affecting the other components of MPLOT, if a different metric is desired to be optimized.

III. Policy and Parameter Choices

The design of the complete MPLOT protocol involves determination of four key policies: (i) Estimation of per-path and aggregate statistics, (ii) Redundancy (PFEC/RFEC) provisioning, (iv) p-block and packet sizing, and (iii) Mapping of packets to paths. The policy and parameter choices in MPLOT can be motivated when we consider the overall objective of MPLOT as being that of maximizing goodput while meeting desirable delay characteristics. In this section we first describe how MPLOT estimates statistics like loss rate and bandwidth (per-path as well as aggregate), which are subsequently used in the redundancy provisioning, p-block sizing and packet mapping policies. The goal of the PFEC/RFEC provisioning policy is to maximize goodput for a given constraint on the per-round data recovery probability, while the p-block sizing policy aims to limit the average block recovery time. The packet mapping policy uses the path parameters for mapping packets to paths so as to minimize the recovery time of a block of a given size.

A. Estimation of Statistics

We consider a multi-path session using M paths, where we characterize each path i with three parameters: loss rate estimate (p_i), bandwidth estimate (BW_i), and round trip time estimate (RTT_i). First we describe how MPLOT estimates loss rates, which are updated every time an ACK is received on any path. Let the number of packets sent by the source for block b on path i be denoted by $S_i(b)$. Also let $R_i(b) \leq S_i(b)$

denote the number of these packets delivered successfully at the destination (determined from the ACKs sent across all paths, by the aggregate ACK processing unit as shown in Figure 1). Then $p_{i,\text{inst}}$, the *instantaneous* loss rate of an individual path i, and p_{inst}, the instantaneous aggregate loss rate across M paths, are calculated as

$$p_{i,\text{inst}} = \left(1 - \frac{R_i(b)}{S_i(b)}\right); \quad p_{\text{inst}} = \sum_{i=1}^{M} \left(\frac{S_i(b)}{\sum_{i=1}^{M} S_i(b)}\right) p_{i,\text{inst}} \tag{2}$$

The running estimates of the *average* path and aggregate loss-rates, $\hat{p}_i$ and $\hat{p}_{\text{agg}}$ respectively, are then obtained by smoothing using an EWMA with a parameter value of 0.5, on a per-packet basis. The instantaneous variance σ^2_{inst} is calculated as

$$\sigma^2_{\text{inst}} = (p_{\text{inst}} - \hat{p}_{\text{agg}})^2. \tag{3}$$

The running estimate of the *variance* of the aggregate loss rate $\hat{\sigma}^2_{\text{agg}}$ is calculated from σ^2_{inst}, again using EWMA with a parameter value of 0.5. The aggregate loss rate estimates (mean and variance) is used for redundancy provisioning, as described in Section III-B.

The per-path RTT (RTT_i) is estimated the same way as performed in TCP. The *bandwidth*[3] BW_i of a path i is then estimated as $BW_i = w_i/RTT_i$, where w_i is the congestion window size for path i. The aggregate bandwidth across all the M paths used by the flow is then estimated as

$$BW \;=\; \sum_{i=1}^{M} BW_i \;=\; \sum_{i=1}^{M} \frac{w_i}{RTT_i}. \tag{4}$$

The following analysis, which focuses on the recovery of a single block of packets, is done under a reasonable "quasi-static" assumption that the w_i, RTT_i, and p_i parameters (and therefore the bandwidth BW_i) do not change significantly during the transmission and recovery period of a single block.

B. PFEC/RFEC Provisioning

For determining the aggregate flow parameters like PFEC/RFEC allocation and p-block size, we model the multiple paths used by a flow as a single composite virtual pipe with aggregate loss rate p_{agg} and aggregate bandwidth BW. We first discuss the PFEC provisioning policy. Consider the transmission of F data packets (each of size S) over this aggregate virtual pipe from the source to the destination. To aid data recovery despite potential packet losses in the aggregate virtual pipe, k FEC packets are sent along with the data packets. We can view F as the number of data packets in a block, and k as the number of PFEC packets that are added to create a p-block of size $B^{\text{p}} = F + k$. Let $x = \frac{k}{F}$ denote the amount of redundancy (i.e., fraction of PFECs in the block); we express the PFEC provisioning policy in terms of x.

The goal of our PFEC provisioning policy is to ensure that the block-data is recovered "with high probability" in the first round itself, i.e., without any additional round of RFEC packet transmission. With this objective, the PFEC policy (as well as the RFEC policy) can be derived, under fairly general assumptions, by simply bounding the loss tail probability (using Chebyshev inequality, for example). In the analysis provided next, however, we first assume an i.i.d. aggregate loss model with a fixed, known loss probability p_{agg}. This allows us to derive exact expressions, and also helps in analyzing the p-block sizing and packet mapping policies that are discussed subsequently in Sections III-C and III-D.

Let $h(i)$ denote the probability that exactly i FEC packets are required to recover the F data packets. Assuming that packet loss process in the aggregate virtual pipe is i.i.d. with probability p_{agg}, then $h(i)$ is given by

$$h(i) \;=\; \binom{F+i-1}{i} (p_{\text{agg}})^i (1-p_{\text{agg}})^F. \tag{5}$$

Let the given block recovery probability constraint, i.e., lower bound on the probability of recovering all data in the block (p-block) in a single round, be $(1-\epsilon)$. Recall that due to the property of erasure codes, all data in the p-block can be recovered if the number of packets in the p-block that is lost is k or less. This recovery probability, denoted by $H(k)$, is given by $H(k) = \sum_{i=0}^{k} h(i)$. Therefore, the block recovery probability constraint translates to $H(k) \geq 1-\epsilon$.

It is easy to observe that $H(k)$ is non-decreasing in k, there exists an $k_{\min}(\epsilon)$ such that $H(k) \geq 1-\epsilon \; \forall k \geq k_{\min}(\epsilon)$. Let $x_{\min}(\epsilon) = \frac{k_{\min}(\epsilon)}{F}$. Therefore, given F, the amount of redundancy must be $x_{\min}(\epsilon)$ or greater, so as to satisfy the block recovery probability constraint. A closed form expression for $x_{\min}(\epsilon)$ may not exist in general; however, $x_{\min}(\epsilon)$ can be computed numerically. Through curve-fitting, we observe that $x_{\min}(\epsilon)$ can be well approximated as follows:

$$x_{\min}(\epsilon) \approx \beta(p_{\text{agg}}, \epsilon) = \alpha_\epsilon \frac{p_{\text{agg}}}{1-p_{\text{agg}}}, \tag{6}$$

where values of α_ϵ for different ϵ are listed in Table II. The amount of PFEC in a block is then $\lfloor F\beta(p_{\text{agg}}, \epsilon)\rfloor + 1$.

Numerical results show that the error in approximating $x_{\min}(\epsilon)$ by $\beta(p_{\text{agg}}, \epsilon)$ is negligible for $F \geq 90$. The difference between $x_{\min}(\epsilon)$ and $\beta(p_{\text{agg}}, \epsilon)$ for small F exists because $x_{\min}F$ must be an integer.

ϵ	0.01	0.10	0.20	0.30	0.50
α_ϵ	1.28	1.20	1.12	1.08	1.00

TABLE II
CALCULATED VALUES OF α_ϵ FOR DIFFERENT ϵ.

Let $\Lambda(k)$ be the average number of redundant packets (FECs) that are required to recover the F data packets. Then

$$\Lambda(k) \;=\; k\sum_{i=0}^{k} h(i) + \sum_{i=k+1}^{\infty} i h(i). \tag{7}$$

Then, $GP(x)$, the flow goodput (aggregated over all paths that the flow uses), expressed as a function of the redundancy

[3]Note that this *bandwidth* is not the raw per-path bandwidth (capacity), but the time-varying share of that bandwidth as perceived by the end-to-end flow. It can also be viewed as the sending rate (Mb/s) on path i that is achievable with the current congestion window (w_i) and round-trip time (RTT_i).

$x = \frac{k}{F}$, is

$$GP(x) = \frac{BW}{1+\frac{\Lambda(k)}{F}} = \frac{BW}{1+x\sum_{i=0}^{xF} h(i) + (\sum_{i=xF+1}^{\infty} ih(i))/F}, \quad (8)$$

where BW is given by (4) and $h(i)$ are given by (5).

Numerical results show that $GP(x)$ is non-increasing in x: it is relatively flat for smaller values of x, but decreases sharply for values of x larger than $\frac{p_{\rm agg}}{1-p_{\rm agg}}$. Thus it follows that the value of x that maximizes $GP(x)$, while satisfying the $1-\epsilon$ bound, is $x_{\min}(\epsilon)$. Thus, MPLOT must transmit at least $k = x_{\min}(\epsilon)F$ PFEC packets in the p-block.

For a 50% block recovery rate, i.e., $\epsilon = 0.50$, we have $\alpha_\epsilon = 1$ (Table II), and the PFEC allocation policy reduces to $x = \frac{k}{F} = \frac{p_{\rm agg}}{1-p_{\rm agg}}$. In practice, however, $p_{\rm agg}$ is not exactly known, and varies with time. To account for these, we take into account the estimates (mean as well as variance) of the loss rate, computed as described in Section III-A. For this purpose, we replace $p_{\rm agg}$ with $(\hat{p}_{\rm agg}+\kappa_1\hat{\sigma}_{\rm agg})$, where κ_1 is a parameter that must be chosen depending on the desired tradeoff between amount of redundancy and block recovery probability. Thus, MPLOT's PFEC provisioning policy allocates the amount PFEC redundancy according to:

$$x = \frac{k}{F} = \frac{\hat{p}_{\rm agg}+\kappa_1\hat{\sigma}_{\rm agg}}{1-(\hat{p}_{\rm agg}+\kappa_1\hat{\sigma}_{\rm agg})}, \quad (9)$$

where the average ($\hat{p}_{\rm agg}$) and variance ($\hat{\sigma}^2_{\rm agg}$) of the loss rate is estimated using measurement and EWMA as described in Section III-A. Note that if X (random variable) denotes the fraction of packets lost in the block, then $\hat{p}_{\rm agg}$ and $\hat{\sigma}^2_{\rm agg}$ represent the estimates of the mean and variance of X. The probability that the block-data is not recovered in a single round is then given by $\mathrm{P}\left(X > \frac{k}{F+k}\right) = \mathrm{P}\left(X > \frac{x}{1+x}\right)$. If the PFEC redundancy x is determined according to (9), then from Chebyshev inequality, we have $\mathrm{P}\left(X > \frac{x}{1+x}\right) = \mathrm{P}\left(X > \hat{p}_{\rm agg}+\kappa_1\hat{\sigma}_{\rm agg}\right) \leq \frac{1}{1+\kappa_1^2}$. Choosing $\kappa_1 = 1$, we observe that the PFEC provisioning policy of (9) yields a block recovery probability 0.5 that we desire. We therefore use $\kappa_1 = 1$ in the simulations shown in Section IV.

The process of deriving the RFEC provisioning policy is similar to that for PFEC. Recall that the aggregate flow manager needs to transmit RFEC packets when the PFEC packets in the block are insufficient to recover the block-data. In order to balance rounds of RFEC transmissions and goodput, a small amount of excess, $(1+y)r$, RFEC packets are sent in response to a request for r RFEC packets needed to recover the block-data. Note that receiving any r packets out of the $(1+y)r$ packets sent would be sufficient to recover the block-data. MPLOT's RFEC provisioning policy chooses y according to

$$y = \frac{\hat{p}_{\rm agg}+\kappa_2\hat{\sigma}_{\rm agg}}{1-(\hat{p}_{\rm agg}+\kappa_2\hat{\sigma}_{\rm agg})}, \quad (10)$$

where κ_2 is a parameter that needs to be chosen. As argued previously, $\kappa_2 = 1$, would ensure that that r packets out of the $(1+y)r$ sent are recovered in the same round. However, since there are multiple paths with diverse characteristics, and number of RFEC packets sent in one round is typically small, and RFECs tend to be sent on "better" paths (which have lower loss rates than $\hat{p}_{\rm agg}$, and lower RTTs). Thus, as our experiments show, a smaller value of κ_2 (even $\kappa_2 = 0$, which is used in the simulations in Section IV), works quite well.

Note that the PFEC allocation x is also updated when a new block is created, while the RFEC allocation y is updated at the beginning of each RFEC transmission round.

C. p-block and Packet Sizing

Note that the PFEC provisioning policy specifies $x = \frac{k}{F}$, but does not specify the individual values of F and k. The p-block sizing policy we describe next specifies $B^{\rm p} = (F+k)$, from which F and $k = xF$ can be obtained using the PFEC allocation x. The optimal p-block size is determined by bounding the block recovery time. In general, a larger p-block size implies larger block recovery time, as we observe shortly. We again model the packet loss process in the aggregate virtual pipe as i.i.d., with probability $p_{\rm agg}$. Recall from the analysis in Section III-B, $h(i)$, as given by (5), is probability that exactly i FEC packets are required to recover the F data packets. Since $h(i)$ follows a negative binomial distribution, the average number of FEC packets required to recover the F data packets is $\frac{p_{\rm agg}F}{1-p_{\rm agg}}$. Therefore, the average number of all packets (data and PFEC) that are required to recover the block-data is $F + \frac{p_{\rm agg}F}{1-p_{\rm agg}} = \frac{F}{1-p_{\rm agg}}$. Note that with the optimum PFEC policy as given by (6), the average number of packets required for block-data recovery reduces to $\frac{B^{\rm p}}{(1+\beta(p_{\rm agg},\epsilon))(1-p_{\rm agg})}$. For a given maximum block recovery delay constraint Δ, the p-block size $B^{\rm p}$ is then bounded as

$$B^{\rm p} \leq BW\Delta(1-p_{\rm agg})(1+\beta(p_{\rm agg},\epsilon)),$$

where BW is the aggregate virtual pipe bandwidth, given by (4). Using (6) and (4), we then obtain

$$B^{\rm p} \leq \sum_{i=1}^{M} w_i \frac{\Delta}{RTT_i}(1+(\alpha_\epsilon - 1)p_{\rm agg}) \triangleq B^{\rm p}_{\max}. \quad (11)$$

A larger p-block size will result in a better goodput, as it allows a selection of PFEC policy x closer to the optimal value $\beta(p_{\rm agg},\epsilon)$ (because of reduced rounding error). Therefore, the optimal p-block size is $B^{\rm p}_{\max}$, as defined in (11).

Note that for $\epsilon = 0.50$ and $\Delta = RTT_{\rm med}$, $B_{\max}$ reduces to the following expression, which MPLOT uses as the p-block sizing policy:

$$B^{\rm p} = \sum_{i=1}^{M} w_i \frac{RTT_{\rm med}}{RTT_i}. \quad (12)$$

Note that the p-block size is updated when a new block is created, based on the latest values of w_i, RTT_i.

In practice, there may be additional constraints that limit the size of the p-block. Note that $B^{\rm p}$ cannot exceed $B(= B^{\rm p} + (K-k))$, the total size of the coded block, which may be limited by the field size in the block erasure code calculation, as discussed in Section II-A. Thus one may want to choose the size of the p-block such that it allows keeping an r-block of "reasonable size", so that the block can be recovered with high probability before all FEC packets in the block are exhausted. The analysis of Section III-B and this section can be extended to compute the appropriate p-block and r-block sizes taking such limits into account. Further, note that if the amount of data remaining in the buffer is not enough (after adding the necessary PFECs) to fill up the p-block size (determined according to (12)), then the MPLOT sender does not wait any longer to fill up the desired p-block size: it just takes up the data that is available, adds PFECs to it according to (9), and sends it out.

Finally, a note on the packet sizing policy is in order. In general, larger packet sizes cause less header overhead, and therefore seem more desirable. A larger packet size implies fewer number of packets per block, which makes the losses more bursty, and the loss rate estimate per block less reliable (due to fewer samples that can be used in the loss rate estimation). Increased burstiness in the losses, or less reliability in the loss estimates, imply increased PFEC provisioning (due to higher loss variance, as in (9)), or non-optimal PFEC/RFEC allocation, thereby reducing goodput. In MPLOT, therefore, we adapt the packet size S (within some pre-determined upper and lower bounds) with each block: we use the largest packet size that still enables us to maintain a given minimum number of packets per block, if possible. Thus a larger p-block size in general implies a larger packet size. In the MPLOT version used in the simulations in Section IV, we vary the packet size across three values: 750 bytes, 550 bytes and 250 bytes (with a 50 byte header), choosing a maximum size that maintains a minimum of 5 packets in a block. MPLOT also ensures that the packet size does not exceed the minimum MTU across all paths.

D. Mapping Packets to Paths

The choice of a mapping policy is important as it significantly affects the amount of goodput we can gain from the available path diversity. We first formalize the question of mapping packets to paths, with the objective of minimizing the block recovery delay, and motivate the packet mapping policy that MPLOT uses, from the analysis. Recall that each candidate path i is characterized by its congestion window w_i, its round-trip time RTT_i and its loss probability p_i. Additionally, at the time of mapping, let $\nu_i \leq w_i$ be the number of packet transmission opportunities on path i. i.e., ν_i is the number "vacant positions" in path i's congestion window. Therefore, ν_i represents a limit on the number of packets of the p-block that can be mapped onto path i. When mapping a p-block of B^{p} packets onto the paths, let z_i be the number of packets mapped onto path i; therefore $\sum_{i=0}^{M} z_i \leq B^{\mathrm{p}}$, and $z_i \leq \nu_i \; \forall i$. In accordance with our goal of keeping the congestion control policy (and hence the packet mapping onto paths) separable from the reliability mechanism, in the optimal packet mapping formulation we ignore the fact that the packets are erasure coded and recovered at an aggregate level. Therefore, we derive the optimal mapping policy assuming that all the packets mapped onto a specific path must be recovered individually. Assuming that the packet loss process on path i is i.i.d., with probability p_i, the number of packets (original packets plus retransmissions) that must be sent to recover the z_i packets mapped to path i equals $z_i/(1-p_i)$ on an average. Since w_i packets are sent over a time interval of RTT_i, the average time taken to recover all packets sent on path i is $z_i/\left(w_i(1-p_i)/RTT_i\right)$. We call the term $\frac{w_i(1-p_i)}{RTT_i}$ the *estimated loss-free bandwidth* of path i, which is the same as the rank function defined in (1).

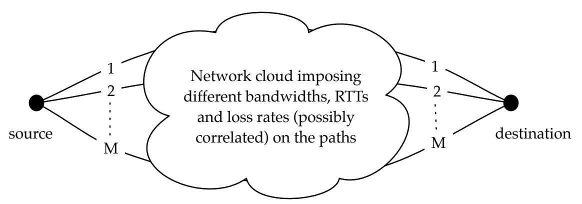
Note that in absence of the upper limit constraints ν_i, the mapping policy that minimizes block recovery time would be to map all packets on the path with the largest estimated loss-free bandwidth (rank) π_i. Due to the upper limit constraints, however, the optimal solution would pick the path $i = i_{\max}$ such that it maximizes π_i among all i, and fill it up to the limit of $\nu_{i_{\max}}$. It then picks the path that attains the second maximum rank, fills it up to the limit, and so on. In other words, this mapping policy picks paths in decreasing order of the rank values, and fills them up to the allowed limits.

MPLOT's packet mapping policy derives the basic idea and the use of the rank function from the solution just described, but differs from the above policy in two aspects. Firstly, when there are multiple outstanding blocks whose packets must be mapped to paths, it also considers the temporal relationship between different blocks during mapping. Refer to Section II-D for the rules followed by the MPLOT packet mapping algorithm: note that MPLOT preferentially maps blocks generated (required) earlier in time to paths with higher rank. Secondly, MPLOT reduces complexity by grouping all paths into two categories: (i) GOOD and (ii) BAD, depending on whether their rank is greater than the median rank (across all paths) or not. GOOD paths are then preferentially used for packets of earlier blocks. More precisely, if there are U unrecovered blocks at any time, then packets of the earliest (latest) $\frac{U}{2}$ blocks are mapped to the GOOD (BAD, resp.) paths, while ensuring that no transmission opportunities are wasted.

IV. Simulation Results

In this section, we present performance results obtained through simulations of the MPLOT protocol on ns-2. We consider a topology comprising M parallel paths between each source and destination node (Figure 2). The different parallel paths in the topology correspond to different physical routes between a source the destination, which may have different capacities, round-trip times and loss rates, and possibly overlap (in terms of physical links) in the underlying network. Since ACK packets are sent back on reverse paths chosen at random, all these paths are utilized for ACK delivery. The overlap between the paths is modeled by considering correlations across losses of different paths in our topology this is studied in Section IV-E. In all our simulations, we have used 1 byte for each of the four additional header fields that MPLOT uses (as mentioned in Section II-B), and the goodput results in this section do take this overhead into consideration, along with that for the rest of the header.

MPLOT as a transport layer protocol could be used over a different link layer technologies and multi-access protocols, and therefore our goal is to evaluate its performance in a manner that is relatively independent of the underlying link layer mechanisms. We start by assuming link characteristics and interactions across flows on individual links are captured in the time-varying loss rates, bandwidths and RTTs of the paths, modeled and measured end-to- end. However, in Section IV-F, we show representative simulation results that capture link layer contention effects on flows for a 802.11-like MAC protocol, and study how an increasing number of paths sharing the same wireless channel impacts MPLOT's performance.



Network cloud imposing different bandwidths, RTTs and loss rates (possibly correlated) on the paths
source
destination
1
2
M
1
2
M

Fig. 2. Generic topology used for most of the simulations: 10 MPLOT flows (sessions), each using (and sharing) the M paths between the source and the destination node. The paths have different (but possibly correlated) loss rates, RTTs and bandwidths, that provide the end-to-end abstraction of the different physical, link and network layer characteristics of the physical hops (links) constituting each path. Each end-to-end path is composed of 4-6 links.

In addition, we investigate the capability of MPLOT to aggregate bandwidth across paths, and utilize diversity across paths to gain additional goodput over the aggregated virtual pipe (Section IV-A). We also stress on the importance of path heterogeneity awareness (Section IV-B), and demonstrate the importance of implementing the reliability mechanism on the aggregate, rather than having per-path reliability (Section IV-C). We show that MPLOT is able to share bandwidth fairly (proportionally) with traditional TCP protocols (Section IV-D). Next, we study the effect of overlapping paths, and correlated loss rates across paths, on the protocol performance (Section IV-E). Finally, we perform a detailed performance comparison on MPLOT with another well-known multipath transport protocol, PTCP (Sections IV-G and IV-G).

Before we describe our experiments and results, a few words on the simulation setup and choice of parameters are in order. In Section IV-A, we keep the bandwidth per path fixed as we vary the number of paths, as we study bandwidth aggregation gains and differentiate it from diversity gains, and study the effect of differential RTT across paths. In the rest of the simulation study, however, we set the total capacity across the M paths to a constant (typically 10 Mb/s or 50 Mb/s) when we vary the number of paths; this allows us to easily compare the results across multiple paths and focus on diversity gains obtained with increasing number of paths. The RTT on any path is kept at 40 ms, except in our study of differentiated path RTTs in Section IV-A. In almost all of our simulations, we use a bimodal loss rate process, where loss rates vary between "low" and "high" levels (25% and 75%, or 5% and 15%, i.e., averaging to 50% or 10% respectively, over time). In Section IV-B, which studies heterogeneity awareness, the loss process is trimodal, as the channel can also be in a state of 100% loss (disruption). In our study involving comparison with PTCP (Sections IV-G and IV-H), the loss process is unimodal (single rate), which allows us to easily vary the loss rate over a wide range for doing this comparison. In our study of wireless channel contention effects in Section IV-F, we have kept a fixed loss rate of 10% to model external interference effects. For any particular loss rate, the loss processes are assumed i.i.d. across packets and paths, except in Section IV-E where we study the effect of loss correlation across paths. The capacity and loss rate parameters chosen for the experiments discussed here are only representative, and simulations across a wide range of these parameters have shown that the nature of the results is generally consistent across different parameter values. In the following, we have presented results for "high" values of the parameters (50 Mb/s for link capacity and 50% for average loss rate) in some cases, and "low" values in the others (10 Mb/s for link capacity and 10% for average loss rate).

A. Bandwidth Aggregation and Diversity Gain

In this section, we study the capability of MPLOT to aggregate bandwidths from multiple, possibly hetergoneous paths. We also study the extent to which MPLOT is able to to suppress volatility in the loss rates and available bandwidths, resulting in further "diversity" gains in goodput. Bandwidth aggregation will typically result in large gains in bandwidth because total bottleneck capacity will increase with the number of paths; this gain can be considered as a "first order effect". The diversity gain of MPLOT is obtained by comparing the goodput achieved by MPLOT using multiple paths with that obtained on a single path *of the same aggregate capacity*. This gain can thus be considered a "second order effect".

In the simulation results presented next, the packet loss rate on each path is implemented as a 2-state time-varying process. The loss rate on each path alternates between 25% and 75%, and the time duration between state transitions is exponentially distributed with the same mean of 250 msec; therefore, the overall time-average loss rate of each path is 50%. The capacity of each path is set to 10 Mb/s, and the RTT of each path is 40 ms. In all our simulations, the buffer capacity at any link is set to its bandwidth-delay product. In Figure 3, we vary the number of paths (M) and plot the goodput (or effective data rate) achieved by MPLOT and the maximum effective capacity available ($M \times 5$ Mb/s), after the 50% loss rate is taken into account. In the simulations, each packet is 550 bytes long with 500 bytes of data. Our packet buffering and ECN marking policy at the intermediate nodes (links) follows the well-known Random Early Detection (RED) scheme.

The bandwidth aggregation curve shown in Figure 3 corresponds to M times the goodput achieved for a single path (2.75 Mb/s). The goodput gained over this value is the diversity gain achieved by MPLOT. We can observe that the diversity gain increases with M. For 10 paths, the goodput achieved by MPLOT is 35 Mb/s as compared to the 27.5 Mb/s estimate from bandwidth aggregation. The diversity gain in this case is as high as 21.4% (7.5 Mb/s) of the total goodput achieved by MPLOT. The diversity gain is attributed to the suppression in path volatility. Our detailed studies reveal that diversity gain increases as $O(1-\frac{1}{\sqrt{M}})$ which matches the order of reduction of aggregate path loss volatility/variance with increasing M. For the case in Figure 3, we observed that the standard deviation for the aggregate loss reduces from 0.132 for 1 path to 0.067 for 4 paths, while the mean aggregate loss rate is constant at 50%.

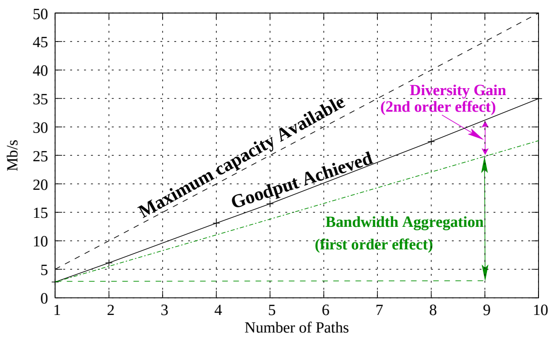


Fig. 3. Bandwidth aggregation and diversity gains obtained by MPLOT. Bandwidth aggregation leads to a linear increase in goodput with increasing number of paths, whereas MPLOT performs even better due to diversity gain (2.75 Mb/s out of 5 Mb/s for 1 path, to 35 Mb/s out of 50 Mb/s for 10 paths).

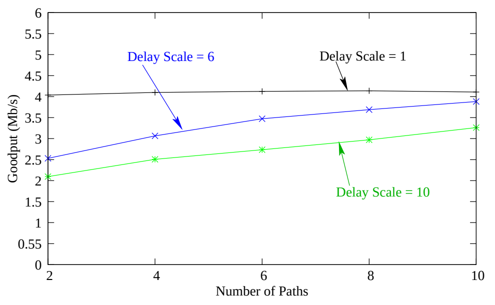


Fig. 4. *Per-path* goodput vs. Number of paths, when the RTT of 50% of the paths is scaled up by the Delay Scale. Each path has a capacity of 10 Mb/s, and 50% loss rate. As number of paths increases, the reduction in goodput due to the longer paths becomes less significant.

Next we consider diversity gain due from paths with different RTTs. Towards this end, we consider similar topology and loss process as before but scale RTT of 50% of the M paths by a *delay scale* D, from 40ms to $40D$ ms (M is chosen even and ≥ 2). The RTTs of the other 50% of the paths are kept at 40 ms. We vary D as $1, 3, 10$. Figure 4 plots the resultant "average per-path" goodput for different D and M. (The total goodput is thus M times the values shown in the figure.) The results show that goodput decreases as delay scale D increases, as expected. However, we observe that the reduction in per-path goodput due to the extra delay scale becomes less significant as the number of paths increases. In fact, with 8 paths or more, the per-path goodput obtained with a delay scale of 6 is quite close to that obtained with D=1. Note also that the per-path goodput increases with an increasing number of paths; this is due to the fact that the shorter delay paths helps offset the effect of a large delay on the rest of the paths. We conclude that MPLOT can take advantage of the diversity offered by multiple paths, even if they have heterogeneous characteristics.

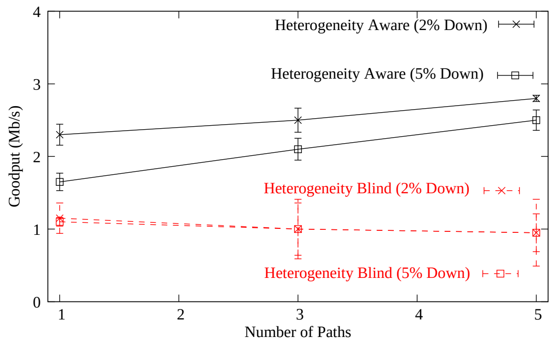


Fig. 5. Average goodput with 95% confidence intervals for MPLOT (heterogeneity aware) compared with a heterogeneity blind scheme for different paths. MPLOT exploits diversity gain from paths and improves while heterogeneity blind scheme suffers over multiple paths.

B. *Importance of Heterogeneity Awareness*

MPLOT is a *heterogeneity aware* scheme, i.e., MPLOT is aware of the differences in packets (block it belongs to) and paths (bandwidth, RTT, loss-rate) which are used to map packets on paths, as discussed in Section II.

To understand the impact of *heterogeneity awareness*, we compare MPLOT with a *heterogeneity blind* scheme that has the same PFEC, RFEC and p-block sizing policy as MPLOT, but considers all paths as equivalent. In such a case, the first packet in the queue is mapped to any available path. Consequently, packets are transmitted in order as opposed to the out-of-order mapping of MPLOT.

We consider the scenario where paths suffer from random disruptions, in addition to bursty packet losses. Loss-rate on a path varies between 25% (OFF), 75% (ON) and 100% (DOWN) states for (exponentially distributed) random time-periods. The average time periods for the ON, OFF and DOWN (disruption period) states are 250 ms, 250 ms and 1 sec respectively, significantly larger than RTT of the paths (40 ms). The likelihood of a transition from the ON or OFF state to DOWN state is kept at 0%, 2% and 5%, for three different studies. The bandwidth of each path is fixed at $\frac{10}{M}$ Mb/s for M paths, keeping the total aggregate bandwidth constant at 10 Mb/s.

We observe in Figure 5 that the goodput of the heterogeneity blind approach actually worsens as number of paths increases from 1 to 5, in each of the three cases (from 1.2 Mb/s to 0.9 Mb/s in case with 5% down state On the other hand, the goodput of the heterogeneity aware scheme (MPLOT) improves significantly Mb/s) with increasing number of paths (from 1 to 5). This demonstrates that consideration of heterogeneity in the path characteristics is crucial to obtaining diversity gains in the goodput.

C. *Aggregate vs Per-path Reliability*

Recall that one of the key design decisions that we made for MPLOT was to implement the reliability mechanism at flow level, aggregated over all paths that the flow uses. We next demonstrate the impact of this, by comparing its performance with that of implementing reliability at the per-path level. In this experiment, we consider the topology as shown in Figure 2, with $M = 3$ paths, where each path has an RTT of 40 ms. The aggregate capacity across all paths (equally divided among the paths) is 50 Mb/s, and loss rates vary uniformly between 5% and 15%, with an average of 10%. The throughput across the three paths, along with the aggregate throughput and goodput attained across all flows, are shown in Table III. The "aggregate reliability" case corresponds to MPLOT; in the "per-path reliability" case, the reliability mechanism remains the same as MPLOT (use of PFEC and RFEC etc.), but they are implemented on a per-path basis; note that in both cases, the congestion control is implemented on a per-path basis. From the results, we observe that MPLOT with aggregate reliability attains about 4.5% more throughput, and 7.8% more goodput, as compared to MPLOT with per-path reliability. MPLOT's reliability across the aggregate results in larger p-block sizes, which in turn results in lower variability in the loss rate perceived by the overall transport connection, compared to that observed by the individual transport connections. This implies that with reliability at the aggregate, we have better efficiency in redundancy allocation (PFEC and RFEC), and/or fewer number of RFEC rounds for block recovery. Recall that in MPLOT, acknowledgements can come back across any of the paths, rather than being restricted to the individual path; this can reduce the number of timeouts (including spurious timeouts). All this argues in favor of implementing reliability at the aggregate (flow) level, as done in MPLOT.

	Per-path Reliability	Aggregate Reliability
Path 1 throughput	15.67	16.83
Path 2 throughput	14.89	16.05
Path 3 throughput	15.97	15.75
Total throughput	46.53	48.63
Total goodput	27.46	29.61

TABLE III
COMPARISON OF PER-PATH VS AGGREGATE RELIABILITY.

D. Fairness with Traditional TCP

The use of multiple paths raises an important question about MPLOT's fairness to conventional TCP. The question becomes even more relevant when we consider the fact that unlike single-path transport protocols, MPLOT can use a different reverse path to acknowledge packets sent on a given forward path.

In this sub-section we use TCP-SACK for comparison in our simulations. Since TCP-SACK is a single path protocol, we focus on the sharing of bandwidth on a per-path basis, even though MPLOT uses multiple paths simultaneously. For fair comparison, the TCP-SACK protocol that we use is ECN-capable as well.

We first consider in detail the case where equal number of MPLOT and TCP-SACK users transfer packets on a single lossless path. We compare the throughput behaviors of an MPLOT flow and a TCP-SACK flow in Figure 6. We note that the throughputs of the MPLOT and TCP-SACK flow behave similarly with time. This shows that the MPLOT operating on a single lossless path behaves fairly with TCP-SACK flows not only on an average, but at a finer time-scale as well.

From Figure 6, note that the initial ramp-up for MPLOT is faster than TCP-SACK. This can be explained as follows. Recall from the discussion in Section III-C that packet sizes

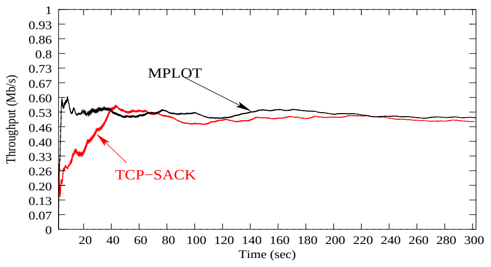


Fig. 6. Throughput comparison between a single-path MPLOT flow and a TCP-SACK flow. MPLOT flow only uses a single path, sharing it with other MPLOT and TCP-SACK flows. The throughput of both flows vary very similarly with time.

are varied based on the p-block size. While the congestion windows for both protocols (TCP-SACK and single-path MPLOT) start out at with the size (1 MSS), more packets are sent out for MPLOT (as the packets are smaller in size, initially). This implies that for MPLOT, a larger number of ACKs come back, and the congestion window grows faster initially. As the congestion window grows, packet sizes become larger, and this effect goes away, and the throughput values for the two protocols become similar over time.

We now consider MPLOT transferring data over multiple paths, and compare the behavior of an individual MPLOT flow with a TCP-SACK flow on one of the paths. We simulate a case where the number of MPLOT and TCP-SACK sources are the same on each path. In particular, we simulate 10 MPLOT sources, each source using M identical paths (M is varied from 1 to 5). Each of the M paths is also used by 10 TCP-SACK users. In this way, the number of MPLOT and TCP-SACK flows on each path are equal. The paths are lossless, so the increased loss-tolerance of MPLOT (due to FEC coding) does not play a significant role in increasing throughput and goodput. Therefore, we expect the throughput for MPLOT and TCP-SACK flows to be equal. We conducted 5 simulations (each of duration 300 secs) with different seed values for each case and report the average across these experiments.

The dynamic behavior of the per-path throughput of MPLOT and that of TCP-SACK were observed to be similar in nature to that shown in Figure 6 – they converge over time to the same values. The congestion windows of the MPLOT and TCP-SACK flows on the path , also behave quite similarly with time. We conclude that even when MPLOT uses multiple paths under lossless conditions, it remains fair to TCP-SACK flows on the paths it is using. These results show that MPLOT treats existing TCP flows fairly, sharing bandwidth equally on the paths it uses.

Figure 7 shows the total throughput of all the MPLOT and TCP-SACK users, as the number of paths is varied (and each MPLOT source uses all the paths). The share of throughput for MPLOT remains constant independent of the number of paths used by each MPLOT source. MPLOT users use a total 5.1 Mb/s while TCP-SACK users use up 4.7 Mb/s out of a total available 10 Mb/s aggregate capacity. This shows that MPLOT and TCP-SACK share bandwidth equally at an aggregate level as well. Hence, we conclude that MPLOT is fair to existing TCP flows on the individual paths as well as at an aggregate

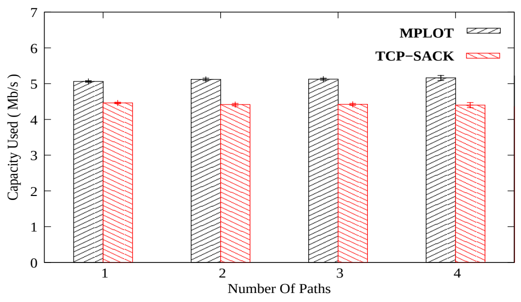


Fig. 7. The average values and 95% confidence intervals of the capacity used by all MPLOT and TCP-SACK users from an aggregate total capacity of 10 Mb/s. We observe that MPLOT and TCP-SACK sources use almost equal amounts of capacity for different number of paths.

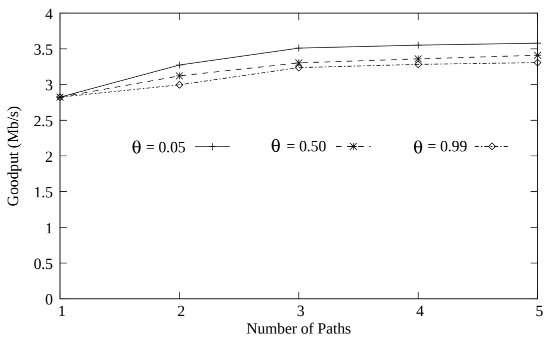


Fig. 8. Effect of loss correlations among paths on goodput attained. As number of paths increase, loss correlations have lesser effect on goodput as MPLOT has more paths to choose from.

level.

The fact that MPLOT attains per-path fairness with TCP-SACK flows, results from MPLOT's use of TCP-like single-path congestion control algorithm separately for each path. However, since MPLOT separates reliability from congestion control, it can work with other congestion control mechanisms as well, and thereby attain different notions of fair bandwidth sharing with single-path (TCP) flows, depending on the specific multi-path congestion control policy used.

E. Effect of Loss Correlations

It is possible that two or more paths may share a lossy link, or their MAC transmissions may interfere, resulting in correlated loss rates across the paths. Intuitively, we would expect the goodput to reduce with correlation due to less diversity among the available paths; next we present some simulation results that quantify this reduction in goodput.

We consider a topology with M paths with an aggregate capacity of 10 Mb/s capacity and RTT of 40 ms, as loss rates vary randomly between 25% and 75%, as described earlier. In this case, however, the packet loss events are correlated, and the degree of correlation is measured by a parameter θ; the correlated loss processes are generated according to the method described in [5]. In particular, the correlation is such that, for two paths numbered i and j, the probability that a packet loss in one path results in a packet loss in the other is equal to $\theta^{|i-j|}$, thereby simulating the scenario where paths that are further apart interfere to a lesser degree. We carry out simulations for $M = 2$ to 5 paths, with $\theta = 0.05, 0.50$ and 0.99. The results, shown in Figure 8, demonstrate that although the diversity gains attained get reduced with increasing correlation, the degradation in goodput with correlation appears gradual and graceful. Moreover, comparing the curves for $\theta = 0.05, 0.50$, we observe that the goodput reduction is small even for correlation factors as high as 50%, and significant reduction in diversity gains occurs only when the degree of correlation is even higher. A larger number of paths provides greater diversity, thereby mitigating the effect of loss correlations among paths.

F. Effect of Wireless Channel Contention

Beyond correlated loss patterns among the paths, MAC protocols also cause correlations in per-path bandwidth variation. To study this, we simulate the topology shown in Figure 9. The 10 MPLOT sessions running between the source and destination node use M paths each. Each of these paths have 3 hops and share the wireless channel on the last hop; in other words, the APi-destination links ($i = 1, \cdots, M$) contend with each other for wireless access. We assume a 802.11b wireless channel (raw data rate of 11 Mb/s) operating in the basic mode (CSMA). On this channel, we assume i.i.d. losses of 10% due to external interference. We the use the well-known model by Bianchi [6] (validated by several simulation studies in the literature) to capture channel bandwidth variations across the last hop of each path. In this case, the aggregate path (last hop link) bandwidth decreases as the number of contending links increase due to packet losses due to collisions, as well as additional protocol overhead. Note that these collision losses are in addition to the 10% loss due to interference.

The capacity of each wired link is 10 Mb/s. Thus, the flows are bottlenecked by the wireless channel. The total throughput and goodput attained is shown in Figure 10. Firstly, the result confirms that MPLOT is able to utilize wireless channel effectively, and even obtain diversity gains, as the number of paths increases up to a point (up to 5 paths). Note that when $M = 5$, then the throughput and goodput attained is observed to be 82% and 70%, respectively, of the wireless link capacity (i.e., the raw bandwidth, discounted by the 10% loss rate). It is worth noting that diversity gain occurs *after* compensating for the bandwidth loss with increasing number of paths due to channel contention. Beyond $M = 5$, however, the diversity gain obtained is not enough to counter the effect of the decreasing bandwidth due to channel contention/collisions. This implies that, when a common wireless channel is shared, we need to strike a balance between using a larger number of paths and the resulting contention. Note that the tradeoffs and the optimality point is highly dependent on the link layer technology and the multi-access protocol used. In the CSMA/CA (with RTS/CTS) mode, the wireless channel bandwidth decreases much more slowly with increasing number of contending links (as compared to CSMA), and therefore the optimal number of paths is likely to be larger. If a contention-free MAC like TDMA is used, the throughput and goodput attained should degrade even more gracefully with increasing number of paths. Note that with paths (links) sharing the

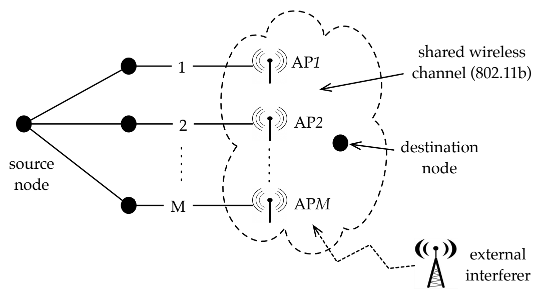


Fig. 9. Topology used for study of the effect of wireless channel contention.

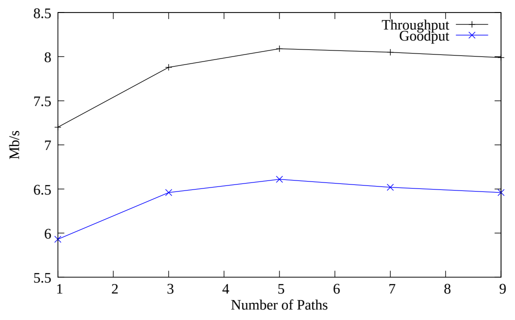


Fig. 10. Variation of throughput and goodput with number of paths (APs) in the topology of Figure 9.

same wireless channel, the loss rates tend to become more correlated, thereby limiting the diversity gains that can be obtained from using of multiple paths (as observed in Section IV-E).

G. MPLOT and PTCP: Goodput Comparison

In this section, we compare the goodput delivered by MPLOT for long-lived flows under different conditions and compare it with other multipath protocols, such as PTCP [7], under similar conditions. For fairness, the PTCP protocol we use for comparison uses ECN for congestion feedback.

PTCP aims for bandwidth allocation across multiple paths, but does not include specific mechanisms for dealing with losses. PTCP maintains a single FIFO queue of packets at the aggregate flow level, performs congestion control on a per-path basis, and maps packets to paths as transmission opportunities in the paths become available. ACKs are sent back on the same path (reverse direction) on which the corresponding packet was sent. MPLOT differs from PTCP in its use of block erasure coded FEC to deal with losses, use of other or all reverse paths for sending ACK feedback, and intelligent mapping that takes into account path loss rates and RTTs and does possibly out-of-order mapping of packets onto paths.

We look at the goodput for long-lived flows, where we separately simulated 10 PTCP and MPLOT sources with i.i.d. losses, over a varying number of paths and packet loss rates.

We first compared the goodput of MPLOT and PTCP under lossless conditions, and observed that in this case, MPLOT and PTCP achieve same goodput irrespective of the number of paths used. We next compare their goodput performance in presence of losses. Figure 11 shows this difference in goodput achieved by MPLOT and PTCP for varying numbers of paths each of which experience 30% packet loss-rate. In this case, we observe MPLOT achieves a goodput that is a significant fraction of the aggregate capacity, but the PTCP goodput is considerably lower.

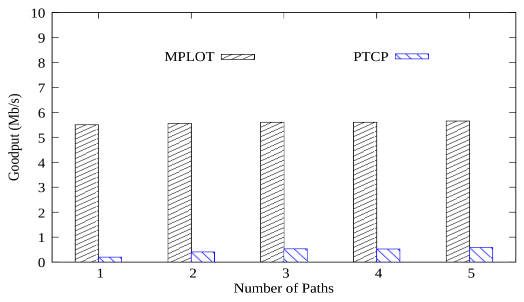


Fig. 11. Goodput achieved by MPLOT and PTCP over different paths when average packet loss rate is 30%. PTCP is unable to achieve more than 1 Mb/s goodput while MPLOT achieves a goodput of more than 6 Mb/s.

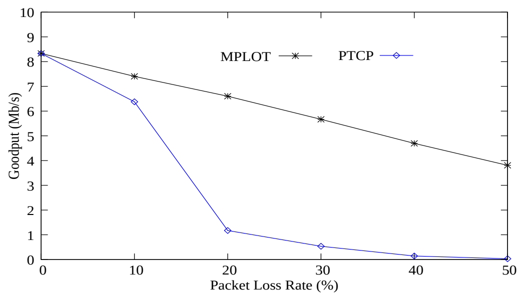


Fig. 12. Goodput achieved by MPLOT and PTCP over 3 paths for different packet loss rates. The goodput of MPLOT is significantly higher than PTCP for loss rates greater than 10%.

As shown in Figure 12, MPLOT only shows a gradual, almost linear decline with increasing loss-rate. PTCP seems to be able to exploit some diversity gain over TCP-SACK at low loss rates (up to about 10%), but shows a sharp decay in goodput, similar to TCP-SACK [8], as the loss-rate increases further. MPLOT is able to maintain high goodput in such lossy conditions because it maps packets to paths in an intelligent manner, considering their bandwidths, round-trip delays and loss-rates while PTCP uses window size as the only parameter in mapping packets to paths. The second reason why MPLOT performs better than PTCP relates to MPLOT's use of FEC, which allows MPLOT to recover from losses without excessive re-transmissions.

H. MPLOT and PTCP: Delay Comparison

We now look at the characteristics of packet delay achieved by MPLOT for long-lived flows. A low average delay and low jitter is desirable for streaming applications, which often use HTTP/TCP, and typically have long-lived flows. Consequently, we measure the average packet delay as well as the variance in packet delay (as a measure of jitter) for MPLOT and PTCP flows. The packet delay is measured from the time a data packet is first sent, to the time it is received/decoded and sent to the application at the destination. This allows us to measure the actual delay experienced by the application. We compare the delay average and variance with PTCP in similar conditions.

We look at the packet delay behavior when individual paths suffer an average 30% packet loss-rate. Now we observe a huge difference in average packet delay between the two protocols. We note in Figure 13, when using 2 paths, MPLOT's average packet delay is less than 100 ms while PTCP's average packet delay exceeds 300 ms. The difference reduces as number of paths increases, but even when using 5 paths, the delay for MPLOT is less than the delay for PTCP by almost 50 ms, which is more than the 40 ms median RTT of the paths. We observe from Figure 14 that for 2 paths the standard deviation of MPLOT's delay was 55 ms, while PTCP's delay showed a deviation in excess of 1600 ms. When using 5 paths, the deviation for MPLOT increases to 500 ms while the delay deviation for PTCP is about 1800 ms.

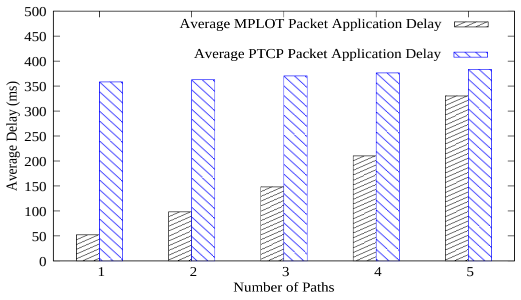


Fig. 13. The average packet delay of MPLOT and PTCP operating over different number of paths when the average packet loss rate is 30%. The average packet delay for MPLOT increases with the number of paths but is still considerably less than the packet delay exhibited by PTCP.

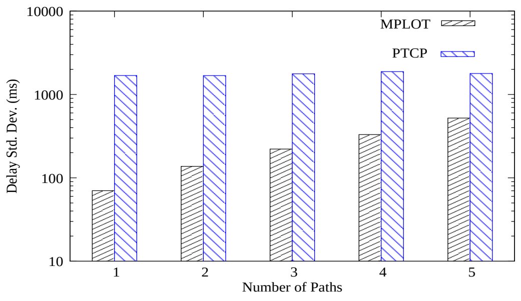


Fig. 14. The standard deviation of packet delay for MPLOT and PTCP operating over various paths when the average packet loss rate is 30%. The std. deviation of packet delay of MPLOT is significantly lower than PTCP for all paths.

Figure 15 shows how PTCP's average delay increases dramatically with increasing loss-rate while MPLOT maintains an average delay that is nearly independent of the loss rate. Coupled with the goodput results from section IV-G, we conclude that MPLOT maintains a good balance between goodput and latency by using multiple diverse paths effectively.

V. Related Work

Lee *et al.* [9] propose simple TCP modifications like increasing the fast retransmission threshold, delayed ACKs and the use of flow-aware routers to address reordering issues in multi-path transport. However, they assume that the underlying link (paths) are reliable and do not consider lossy channels. Lim *et al.* [10] propose a multi-path TCP framework for lossy networks, where they transmit multiple copies of a packet on different paths to counter high loss-rates. The performance of the scheme degrades sharply for loss-rates beyond 20%.

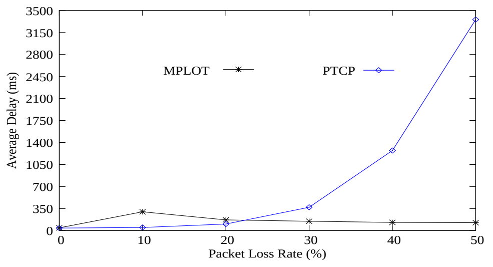


Fig. 15. The average packet delay for MPLOT and PTCP operating on 3 paths for different packet loss rates. MPLOT's packet delay does not show a significant increase as packet losses increase while the packet delay of PTCP increases exponentially.

Rizzo [11] demonstrate the feasibility of high-speed FEC computation at the transport layer. Although [11] mentions the idea of using FEC in TCP, no specific scheme has been proposed or studied. In [12], [13], the authors study the use of FEC in transport control, but in the context of reliable multicast. Anker *et al.* [14] show that using FEC with TCP is a viable option to recover from sporadic packet losses, but the performance depends on how the FEC parameters are chosen. Baldatoni *et al.* [15] propose a version of TCP with FEC (but without any adaptivity in the coding rate) that works for small error rates. TCP Westwood [16] uses an estimate of output rate to guide congestion control, and has been effective for low erasure rates (under 5%). Loss-Tolerant TCP (LT-TCP) [8], developed in our prior work, is a transport protocol designed to be robust in environments with high loss rates and bursty losses. It uses adaptive segmentation, loss estimation and FEC to improve goodput by avoiding expensive timeouts. LT-TCP uses an estimate of the end-to-end loss rate to provision FEC adaptively through both proactive and reactive mechanisms. However, the above mentioned solutions are designed to operate over a single path and cannot leverage additional capacity and diversity benefits available through use of multiple paths.

Several recent works have proposed TCP based multi-path transport protocols for use over lossy links [17], [7], [18]. However, these existing schemes allow a very limited degree of redundancy at the transport layer, due to which they cannot handle multiple highly lossy paths effectively. In mTCP, proposed by Zhang *et al.* [17], no redundancy is introduced at the TCP layer, and all lost packets must be retransmitted resulting in excessive retransmissions and low goodput. Similarly, in pTCP, proposed by Hseih *et al.* [7], a packet is transmitted redundantly (over two paths) only if it immediately follows a timeout. RCP, described in [18], relies solely on retransmission of lost packets to recover from losses, which can seriously limit its performance advantages in a highly lossy environment.

The problem of diversity coding for multi-paths has been modeled theoretically, though the models incorporate only limited protocol adaptivity, loss dynamics and path heterogeneity. Tsirigos and Haas [19], [20] derive analytical expressions for calculating the delivery probability when packets are sent over multiple paths (without much delay heterogeneity), and provide an algorithm for computing the optimal mapping of packets to paths. Vergetis *et al.* [21], [22] address a similar modeling question, but with a more realistic path loss model where loss rates can vary at a faster timescale. The benefits of using multiple paths in improving the packet delivery probability is experimentally evaluated on a 802.11 testbed in [23], although only a small number of paths (that differed only in their loss behaviors) are considered. Miu *et al.* [24] consider how better loss resilience can be provided by exploiting multi-radio diversity, by combining multiple, possibly erroneous, copies of a given frame, and focus on the case where the path introduces bit errors rather than packet erasures. Jain and Das [25] propose a link-layer mechanism to determine the next hop locally based on prevalent channel conditions, but do not provide a mechanism to recover from channel errors.

For real-time streaming applications, Jurca *et al.* [26] and Rao *et al.* [27] propose algorithms to schedule packets on multiple paths for bandwidth aggregation while minimizing delay but ignore losses due to faulty links. In the same context, Nguyen *et al.* in [28] consider using FEC to counter packet losses; however, the scheduling scheme proposed is not adaptive, and requires an exhaustive search that does not scale well with the number of paths. Li *et al.* [29] also propose FEC to counter packet-losses. They propose an algorithm to schedule packets on paths such that the average number of lost packets is minimized while the FEC encoding remains fixed irrespective of the network conditions. However, due to additional path-bandwidth constraints, the algorithm schedules packets sub-optimally.

The current multipath TCP work in the IETF (TCP-M) [30] is another effort at using multiple paths at the transport layer. It is based on adjusting the increase parameter adaptively to balance the load on the different paths so as to improve the aggregate use of all the paths. We go beyond their approach by also being able to tolerate loss on the paths. Moreover, we are able to show that our approach is able to take advantage of the diversity in loss and delay across the different paths.

To summarize, some key limitations of the existing work on this topic are: (i) the proposed schemes may not scale well to a highly lossy environment, (ii) the heterogeneity in path characteristics is not exploited effectively, (iii) in many cases, specific protocols to attain the desired goals have not been proposed, and (iv) algorithms and specific design choices depend heavily on the application. Our contribution lies in the fact that we provide a concrete protocol to address these limitations, by developing hybrid ARQ/FEC strategies using erasure coding to extract diversity gains from multiple paths with heterogeneous characteristics. The scheme we propose, MPLOT, adapts to the changing channel conditions quickly to achieve a stable, aggregate goodput despite higher volatility and poor loss rates on individual paths.

VI. Conclusion

In this paper, we proposed MPLOT, a transport protocol that can realize significant bandwidth gains through the effective use of multiple heterogeneous end-to-end paths subject to very high and bursty loss rates. Traditional TCP (e.g., TCP-SACK) is unable to take advantage of multiple paths, as well as the diversity in loss and delay across these paths. In contrast, MPLOT delivers higher goodput than simple bandwidth aggregation by exploiting diversity from multiple paths in the presence of delay heterogeneity across paths, and even with bursty, high, correlated loss rates (which can be as high as 75%, with a mean of 50%, as in our example study).

MPLOT is based on the fundamental principle of separating reliability and congestion control. It performs congestion control on each individual path, while ensuring reliable, in-sequence delivery by working on the aggregate set of paths. The resources across the set of paths are pooled effectively to present an aggregate capacity end-to-end such that the end-systems view the set of paths as a single larger capacity pipe. MPLOT makes effective use of erasure codes to provide reliability, coupled with loss rate estimation at the aggregate level across paths. It performs per-path congestion control like TCP-SACK using ECN support in the network to distinguish congestion from packet erasure. FEC's sequence agnostic property allows MPLOT to overcome out-of-order delivery issues naturally. However, as our comparisons with a heterogeneity-blind approach show, an intelligent packet mapping design like the adaptive rank based approach used by MPLOT, is required to maximize the aggregate goodput across the heterogeneous and dynamic component paths. In MPLOT, these effects are realized by sending the latest feedback on the best (or all) path(s), and mapping packets to paths based upon a rank function that values shorter RTT, lower loss and higher capacity paths. Through appropriate choice of the amount of redundancy (PFEC/RFEC) and the p-block size, MPLOT allows us to attain a desirable tradeoff point between goodput and delay, and limit the number of retransmissions required for block-data recovery.

In addition to delivering significantly higher goodput compared to other multipath transport protocols proposed, MPLOT also achieves a lower delay. The delay is better behaved (both average delay and variance), thus making MPLOT more suitable for a large variety of applications that seek reliable delivery under lossy conditions. We demonstrated that MPLOT can co-exist with conventional TCP flows, and under lossless conditions MPLOT is fair in that it shares bandwidth equally with connections using traditional TCP. In summary, MPLOT can be effectively used for a wide range of applications to deliver better performance (higher goodput, lower delay) than existing protocols under a wide range of network conditions.

Acknowledgment

The authors would like to thank the sponsors MIT Lincoln Labs and AT&T Research for their generous support of this research, and Biswaroop Ganguly and Stephen McGarry of MIT Lincoln Labs for their valuable advice and feedback throughout this work.

References

[1] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris, "Link-level Measurements From an 802.11b Mesh Network," in *SIGCOMM Computer Communications Review*, vol. 34, no. 4, 2004, pp. 121–132.

[2] C. Steger, P. Radosavljevic, and J. P. Frantz, "Performance of IEEE 802.11b wireless LAN in an emulated mobile channel," in *The 57'th IEEE Semiannual Vehicular Technology Conference*, vol. 2, 2003, pp. 1479–1483.

[3] D. N. Cottingham, I. J. Wassell, and R. K. Harle, ""Performance of IEEE 802.11a in vehicular contexts," in *Proc. of IEEE Vehicular Technology Conference*, Spring 2007.

[4] K. Ramakrishnan *et al.*, "The Addition of Explicit Congestion Notification (ECN) to IP," RFC 3168, September 2001. [Online]. Available: http://www.ietf.org/rfc/

[5] C. G. Park, T. Park, and D. W. Shin, "A Simple Method for Generating Correlated Binary Variates," *American Statistician*, vol. 50, pp. 306–310, 1996.

[6] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, pp. 535–547, 2000.

[7] H.-Y. Hsieh and R. Sivakumar, "A Transport Layer Approach for Achieving Aggregate Bandwidths on Multi-homed Mobile Hosts," in *Proc. ACM Mobicom*, 2002.

[8] V. Subramanian, S. Kalyanaraman, and K. K. Ramakrishnan, "An End-to-End Transport Protocol for Extreme Wireless Environments," in *Proc. IEEE Military Communications Conference (MILCOM)*, 2006.

[9] Y. Lee, I. Park, and Y. Choi, "Improving TCP Performance in Multipath Packet Forwarding Schemes," *Journal of Communications and Networks*, vol. 4, no. 2, pp. 148–157, 2002.

[10] J. Chen, K. Xu, and M. Gerla, "Multipath TCP in Lossy Wireless Environment," in *Proc. IFIP Third Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, 2004.

[11] L. Rizzo, "On the feasibility of software fec," dEIT Technical Report LR-970131. [Online]. Available: citeseer.ist.psu.edu/rizzo97feasibility.html

[12] M. M. J. W. Byers, M. Luby and A. Rege, "A Digital Fountain Approach to Reliable Distribution of Bulk Data," in *Proc. SIGCOMM*, 1998.

[13] J. Nonnenmacher and E. Biersack, "Reliable Multicast: Where to Use FEC," in *Protocols for High-Speed Networks*, 1996.

[14] T. Anker, R. Cohen, and D. Dolev, "Transport Layer End-to-End Error Correcting," The School of Computer Science and Engineering , Hebrew University, Tech. Rep., 2004.

[15] L. Baldantoni, H. Lundqvist and G. Karlsson, "Adaptive End-to-End FEC for Improving TCP Performance over Wireless Links." in *Proc. ICC*, 2004.

[16] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links," in *Mobile Computing and Networking*, 2001, pp. 287–297. [Online]. Available: citeseer.ist.psu.edu/463862.html

[17] M. Zhang, "Understanding Internet Routing Anomalies and Building Robust Transport Layer Protocols," Ph.D. dissertation, Department of Computer Science, Princeton University, Sep 2005.

[18] H.-Y. Hsieh, Y. Z. K.-H. Kim, and R. Sivakumar, "A Receiver-Centric Transport Protocol for Mobile Hosts with Heterogeneous Wireless Interfaces," in *Proc. ACM Mobicom*, 2003.

[19] A. Tsirigos and Z. Haas, "Analysis of Multipath Routing - Part I: The Effect on the Packet Delievery Ratio," in *IEEE Transactions on Wireless Communications*, vol. 3, no. 1, 2004.

[20] ——, "Analysis of Multipath Routing - Part II: Mitigation of the Effectsof Frequently Changing Network Topologies," in *IEEE Transactions on Wireless Communications*, vol. 3, no. 1, 2004.

[21] E. Vergetis, R. Guerin, and S. Sarkar, "Improving Performance Through Channel Diversity in the Presence of Bursty Losses," in *Proc. 19th International Teletraffic Congress (ITC)*, 2005.

[22] ——, "Realizing the Benefits of User-Level Channel Diversity," *Computer Communications Review*, vol. 35, no. 5, 2005.

[23] E. Vergetis, E. Pierce, M. Blanco, and R. Guerin, "Packet-level Diversity - From Theory to Practice: An 802.11-based Experimental Investigation," in *Proc. ACM Mobicom*, 2006.

[24] A. K. L. Miu, H. Balakrishnan, and C. E. Koksal, "Improving Loss Resilience with Multi-radio Diversity in Wireless Networks," in *Proc. ACM Mobicom*, 2005.

[25] S. Jain and S. R. Das, "Exploiting Path Diversity in the Link Layer in Wireless Ad Hoc Networks," in *Proc. of the 6th IEEE WoWMoM Symposium*, 2005.

[26] D. Jurca and P. Frossard, "Video Packet Selection and Scheduling for Multipath Streaming," in *IEEE Transactions on Multimedia*, vol. 9, no. 3, 2007.

[27] K. Chebrolu and R. Rao, "Bandwidth Aggregation for real-time applications in heterogeneous wireless networks," in *IEEE Trans. Mobile Computing*, vol. 5, no. 4, 2006, pp. 388–403.

[28] T. Nguyen and A. Zakhor, "Distributed Video Streaming with Forward Error Correction," in *Proc. Packet Video Workshop*, 2002.

[29] Y. Li, Y. Zhang, L. Qiu, and S. Lam, "SmartTunnel: Achieving Reliability in the Internet," in *Proc. Infocom*, 2007.

[30] C. Raiciu, M. Handley, and D. Wischik, "Coupled congestion control for multipath transport protocols," draft-ietf-mptcp-congestion-01 (work in progress), January 2011. [Online]. Available: http://datatracker.ietf.org/doc/draft-ietf-mptcp-congestion/

Vicky Sharma received the M.S. and PhD. degrees in Electrical Engineering from Rensselaer Polytechnic Institute, Troy, NY in 2006 and 2010, respectively. He received the B.Tech. in Electrical Engineering from the Indian Institute of Technology, Kanpur, India in 2002. His research interests include include traffic engineering, wireless transport layer design and disruption/loss tolerant networking.

Koushik Kar has been a faculty member with the Electrical, Computer & Systems Engineering department at Rensselaer Polytechnic Institute, Troy, NY, since 2002. He received the Ph.D. and M.S. degrees in Electrical & Computer Engineering from the University of Maryland, College Park in 2002 and 1999, respectively, and the B.Tech. degree in Electrical Engineering from the Indian Institute of Technology, Kanpur, in 1997. His research interests include the study of access control, traffic engineering, congestion control and energy management issues in the Internet as well as wireless networks. Dr. Kar received the CAREER Award from the National Science Foundation in 2005.

K. K. Ramakrishnan joined AT&T Bell Labs in 1994 and has been with AT&T Labs-Research since its inception in 1996. Prior to 1994, he was a Technical Director and Consulting Engineer in Networking at Digital Equipment Corporation. Between 2000 and 2002, he was at TeraOptic Networks, Inc., as Founder and Vice President.

A DMTS at AT&T Labs-Research, K. K. is involved in several technical and strategic activities in networking and information distribution. He has published nearly 200 papers and has over 100 patents issued. K.K. is an IEEE Fellow and an AT&T Fellow, recognized for his contributions to communications networks, including congestion control, traffic management and VPN services. K.K. is an editor of the Networking Science Journal and has been on the editorial board of the IEEE/ACM Transactions on Networking and IEEE Network Magazine.

K. K. received his MS from the Indian Institute of Science (1978), an MS (1981) and Ph.D. (1983) in Computer Science from the University of Maryland, College Park, Maryland, USA.

Shivkumar Kalyanaraman is a Senior Manager at IBM Research - India, Bangalore. Previously, he was a Professor at the Department of Electrical, Computer and Systems Engineering at Rensselaer Polytechnic Institute in Troy, NY. He holds degrees from Indian Institute of Technology, Madras, India (B.Tech), Ohio State University (M.S., Ph.D.) and RPI (Executive MBA). His research in IBM is at the intersection of emerging wireless technologies, smarter energy systems and IBM middleware and systems technologies. He is a Fellow of the IEEE, and ACM Distinguished Scientist.