MPLOT: A Transport Protocol Exploiting Multipath Diversity using Erasure Codes

V. Sharma*, S. Kalyanaraman*, K. Kar*, K.K. Ramakrishnan†, V. Subramanian*
* Rensselaer Polytechnic Institute, Troy, NY 12180.
Emails: {sharmv, kalyas, kark, subrav}@rpi.edu.
† AT&T Research, Florham Park, NJ 07932.
Email: kkrama@research.att.com.

***Abstract*—In this paper, we propose a novel transport protocol that effectively utilizes available bandwidth and diversity gains provided by heterogeneous, highly lossy paths. Our Multi-Path LOss-Tolerant (MPLOT) protocol can be used to provide significant gains in the goodput of wireless mesh networks, subject to bursty, correlated losses with average loss-rates as high as 50%, and random outage events. MPLOT makes intelligent use of erasure codes to guard against packets losses, and a Hybrid-ARQ/FEC scheme to reduce packet recovery latency, where the redundancy is adaptively provisioned into both proactive and reactive FECs. MPLOT uses dynamic packet mapping based on current path characteristics, and does not require packets to be delivered in sequence to ensure reliability. We present a theoretical analysis of the different design choices of MPLOT and show that MPLOT makes an optimal trade-off between goodput and delay constraints. We test MPLOT, through simulations, under a variety of test scenarios and show that it effectively exploits path diversity in addition to aggregating path bandwidths. We also show that MPLOT is fair to single-path protocols like TCP-SACK.**

I. Introduction

Wireless mesh networks are being increasingly deployed in domains traditionally dominated by wired networks. Projects and standards like AT&T Metro Wifi, Google Wifi and municipal deployments seek to replace traditional wired backhaul networks with multi-hop wireless networks. As a result, it is becoming increasingly important for transport protocols to offer applications stable, high goodput (data rate) and low latency in spite of the inherent volatility of the underlying multi-hop wireless paths. One way to accomplish this objective is to use the diversity/parallelism offered by wireless networks. A main source of diversity is the existence of multiple paths in the wireless network. It is highly desirable for a transport protocol to leverage this diversity by using these paths as a higher capacity stable network "tunnel", with good loss and delay behavior to deliver application data.

In wireless networks, accumulated high bit-error rates (possibly over multiple wireless hops) and random delays translate to significant and dynamic packet loss/erasure rates ($\geq 10\%$), high jitter and volatile delay & capacity for a transport protocol operating over such networks. Recent studies of IEEE 802.11b based wireless mesh networks [1], [2], have reported packet loss-rates as high as 50%. This typically translates to low application level goodput and high delay when using current transport protocols, which only use a single path.

Wireless networks provide increased opportunities to form multiple paths from the source to destination. This is due to the fact that two nodes do not require an explicit physical link to exchange information. The advent of multi-homing and directional transmission has also made possible the use of multiple paths in parallel with negligible inter-path interference. Transport protocols can potentially use this inherent path diversity to counter the adverse effects of a single unpredictable path by transmitting data across multiple paths. Such a transport protocol should ideally aggregate capacities across multiple, lossy paths and leverage diversity among the paths to yield stable, high goodput and low latency.

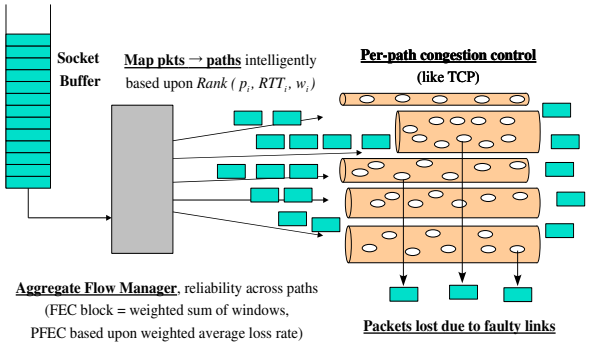

Fig. 1. Key concepts in MPLOT: aggregate flow management, intelligent packet mapping and per-path congestion control. Diferrences in Paths' loss-rates,RTTs and bandwidths are aggregated to form a stable network "tunnel".

In this paper, we present *MPLOT*, the *Multi-path LOss-tolerant Transport* protocol, to attain the above mentioned goals. To counter high losses, MPLOT uses block Forward Error Correction (FEC) coding at the transport layer. MPLOT effectively separates reliability and congestion control by organizing reliability across the available paths while performing congestion control on a per-path basis. To counter the effects of instantaneous high loss rates or delays on certain paths, MPLOT estimates path parameters (loss-rate, capacity and RTT) continuously, and provisions FECs and maps packets to paths adaptively. In particular, MPLOT maps packets that are not required immediately to paths with longer delays, while mapping the more immediately useful packets to paths with shorter RTTs. Previously proposed multiple path protocols fail to perform such latency-aware packet mapping, thereby wasting the capacity of available, heterogeneous paths. MPLOT also overcomes the traditional problems of out-of-order delivery with protocols such as TCP-SACK when using multiple paths, by leveraging the sequence-agnostic properties of FEC, and intelligent packet mapping. MPLOT essentially extends TCP-SACK to aggregate capacities of multiple paths while effectively utilizing diversity gains among the paths and

providing high loss resilience. Our performance evaluation results, described later in this paper, demonstrate that MPLOT is able to effectively utilize the benefits provided by multiple paths even in the presence of heterogeneous path delays, bursty, correlated packet losses and random outage events.

II. Related Work

Lee *et al.* [3] propose simple TCP modifications (increasing the fast retransmission threshold ,delayed ACKs), and the use of flow-aware routers, to address reordering in multi-path transport. However, they do not consider lossy channels. Lim *et al.* [4] propose a multi-path TCP framework for lossy networks, where they transmit multiple copies of a packet on different paths The performance of the scheme degrades sharply as loss-rates increase beyond 20%.

Several recent works have proposed TCP based multi-path transport protocols for use over lossy links [5], [6], [7]. 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.* [5], 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.* [6] a packet is transmitted redundantly (over two paths) only if it immediately follows a timeout. RCP, described in [7], 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 and/or loss dynamics/path heterogeneity. Tsirigos and Haas [8], [9] derive 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.* [10], [11] 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 [12], although only a small number of paths (that differed only in their loss behaviors) were considered. Miu *et al.* [13] 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 [14] 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.

Loss-Tolerant TCP (LT-TCP), proposed in [15], 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, LT-TCP is designed to operate over a single path and cannot leverage additional capacity and diversity benefits available through use of multi-paths.

Jurca *et al.* [16] and Rao *et al.* [17] propose algorithms to schedule packets on multiple paths for bandwidth aggregation while minimizing delay but ignore losses due to faulty links. Nguyen *et al.* in [18] 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.

The authors in [19] 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.

To summarize, some key limitations of the existing work on this topic are: (i) proposed schemes may not scale well to a highly lossy environment, (ii) 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) specific 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 (HARQ)/FEC strategies using erasure coding to extract diversity gains from multiple paths with heterogeneous characteristics. Our proposed scheme adapts to the changing channel conditions quickly (within a window) to achieve a stable, aggregate goodput despite higher volatility and poor mean values on individual paths.

III. Scheme Design

In this section, we present the Multi-Path Loss-Tolerant TCP (MPLOT) scheme, and describe its major components in detail. We present a theoretical justification for the choices made in this section in Appendix A. An overview of key MPLOT functions is shown in Figure1. MPLOT is a realization of a few simple, but effective ideas:

- *Construct a block* of data and FEC packets that will be transmitted across the available paths. The size of the block is determined according to the delay characteristics of the application, e.g., if the application can tolerate large delays, MPLOT can form large block sizes to reduce the impact of bursty losses. MPLOT constructs a block of size B given by the following expression:

$$B = \sum_{i=1}^{M} w_i \frac{RTT_{\text{med}}}{RTT_i}. \tag{1}$$

  Here, M is the number of paths , RTT_{med} is the median Round Trip Time (RTT) of all paths, w_i and RTT_i respectively represent the window size and RTT of path i. The block size B adapts to the path-windows and allows the transmission of the block across M paths in an average time of RTT_{med} provided the packets are mapped on each path accordingly.
- Organize reliability at the *aggregate flow manager*, across paths (Figure 1). In a recent work [15] that only considers a single lossy path, the authors have established that highly lossy and bursty conditions require extension of the traditional TCP window/ARQ mechanisms to a hybrid FEC/ARQ based reliability framework. In such a

framework, FEC is *proactively* provisioned within each block, and if there are bursty losses, FEC can be used *reactively* in response to status feedback as well. In the multi-path scenario that we consider in this paper, we propose to perform such hybrid FEC/ARQ functions at the aggregate level across individual paths as shown in Figure 1. Provisioning FEC packets based on aggregate parameters help in averaging out the volatility of individual paths leading to a smoother, more stable performance.

- *Congestion control* is done on a per-path basis (Figure 1). The per-path congestion window (w_i) determines when a path i can accept packets from the aggregate flow manager. Explicit congestion notification (ECN) on a path is used to distinguish congestion losses from those due to faulty/lossy links. Latest aggregate reliability status is fed back on all paths. Thus the information about a packet received on a long path can reach the source through a shorter reverse path, shortening the effective round trip times for feedback. Moreover, if any single reverse path is subject to heavy loss or disruption, the reliability and self-clocking feedback for that path can arrive at the source through other paths. The source can thus advance the window for any path based upon self-clocking feedback received on a shorter or error-free reverse path. Per-path disruptions in the forward direction will lead only to per-path timeouts like in TCP, but will not affect the congestion window dynamics of other paths.
- Use *intelligent packet mapping* strategies to map packets from the buffers at the aggregate flow manager to individual paths. When a path's congestion window advances and offers a transmission opportunity, an appropriate data or FEC packet (possibly out-of-order from a future block) is mapped to that path. As shown in Figure 1, per-path parameters (loss rate, RTT, window) are combined into a *rank* function that is used to decide which packet is picked for a given transmit opportunity. In particular, higher ranked paths have shorter RTTs, lower loss rates and higher window sizes, and data and FEC packets from the earliest un-recovered block are mapped to these paths. We show that such a *heterogeneity aware* mapping is far more effective in aggregating path capacities than a *heterogeneity blind* baseline approach.

We demonstrate that this simple and modular division of functions is sufficient to extract the synergies from multiple paths and efficiently aggregate them.

A. Aggregate Flow Manager: Hybrid ARQ/FEC Framework

The aggregate flow manager is responsible for reliability functions. As discussed above, the reliability framework is decoupled from per-path congestion control. Providing reliability across paths instead of on a per-path basis allows us to average across the volatility of individual path performance. However, the reliability scheme has to deal with the fact that aggregate loss rates could still be high.

As paths become highly lossy, we have to go beyond TCP's ARQ/window/SACK framework and use a HARQ/FEC framework. First, in presence of high loss-rates, TCP-SACK's selective retransmission policy would result in multiple rounds of retransmissions and timeouts to recover from losses, reducing goodput. Second, if we want to reduce end-to-end latency, we would like to recover missing data without many rounds of retransmission. The use of FEC pro-actively in the original block transmission, and reactively in response to status feedback allows a block to be recovered when a given threshold of packets (F) is received. Further, unlike SACK which requires specific sequence-number feedback and specific segments to be retransmitted, FEC is *sequence-agnostic*, i.e., *any* F data or FEC packets suffice for block recovery. However, it is important not to over-provision FEC in order to maximize goodput. The goal therefore, is to construct a reliability scheme that offers an optimal goodput-latency trade-off even under highly lossy and bursty path conditions.

In this paper, we use HARQ/FEC techniques (similar to [15]) for multiple paths and perform the reliability functions at the aggregate level across paths. FEC provisioning is more efficient with larger blocks, and adapts to measured aggregate end-end loss statistics as described below. Specifically, the FEC encoder takes F data packets, and adds k FEC packets using a $(F+k, F)$ block coding scheme. The encoder needs to dynamically choose a block size ($B = (F+k)$) and decide what fraction of the block (k/B) to allocate for *proactive FEC (PFEC)* packets. We assume that the encoder also computes a large inventory of *reactive FEC (RFEC)* packets to be used if the PFEC packets are unable to recover the block.

Observe that, for a $(F+k, F)$ block code, k represents the number of packets that we can afford to lose while still being able to recover F data packets. The aggregate flow manager reads the first F application data packets in the send buffer to generate $F+k$ encoded packets and stores them in the send buffer. The number k must be carefully selected to maintain an optimal trade-off between goodput and delay. A larger number of PFEC packets k reduces the number of retransmissions required to recover the data (thereby reducing delay), but may also waste bandwidth (thereby reducing goodput) as some of the PFEC packets may not be required for data recovery.

The aggregate flow manager dynamically allocates the fraction of PFEC packets (k/B) in a block to adjust to the prevailing aggregate loss-rate $\bar{p}_{\text{agg}}$, aggregate loss-rate variance σ^2_{agg}, and the number of data packets F. The PFEC allocation is given by

$$\frac{k}{F} = \frac{\bar{p}_{\text{agg}} + \sigma_{\text{agg}}}{1 - \bar{p}_{\text{agg}} - \sigma_{\text{agg}}}. \tag{2}$$

The inclusion of σ_{agg} in PFEC allocation allows MPLOT to account for the time variance in loss-rate while still ensuring that at-least 50% of blocks will be recovered by the receiver without any retransmissions.

The aggregate flow manager needs to tranmsit RFEC packets when the PFEC packets in the block are not enough to recover the block-data. In order to balance rounds of RFEC transmissions and goodput, $(1+\kappa)r$ RFEC packets are sent in response to a request for r RFEC packets needed to recover the data. The redundancy κ is expressed as

$$\kappa = \frac{\bar{p}_{\text{agg}}}{1 - \bar{p}_{\text{agg}}}. \tag{3}$$

The transmission of $(1+\kappa)r$ RFEC packets ensures that less

than half the blocks would require another round of RFEC transmissions to recover the data.

1) Aggregate Flow Manager: Measurement of Statistics: In order to estimate the loss rate on a path, the header of a packet transmitted on path i is appended with three pieces of information: (i) The block number b the packet belongs to, (ii) The number of packets sent by the source for block b on path i ($S_i(b)$), (iii) The number of packets received by the destination for block b on path i ($R_i(b)$).The block number b and $S_i(b)$ are updated by the aggregate flow manager when scheduling the packet and are simply echoed back by the receiver. The receiver updates the number of received packets $R_i(b)$ on path i. The loss rate of an individual path p_i is then estimated as $\left(1 - \frac{R_i(b)}{S_i(b)}\right)$. The aggregate loss rate p_{agg} across M paths is estimated as

$$p_{\text{agg}} = \sum_{i=1}^{M} \left(\frac{S_i(b)}{\sum_{i=1}^{M} S_i(b)} \right) p_i. \tag{4}$$

The *mean* aggregate loss rate $\bar{p}_{\text{agg}}$ and mean path loss-rate $\bar{p}_i$ are updated using the EWMA averaging method with a EWMA parameter value of 0.5. The instantaneous variance σ^2_{inst}in the aggregate loss rate σ^2_{agg} is calculated as

$$\sigma^2_{\text{inst}} = (p_{\text{agg}} - \bar{p}_{\text{agg}})^2. \tag{5}$$

The aggregate flow manager maintains a running estimate of the *variance* of the aggregate loss rate σ^2_{agg} which is calculated from σ^2_{inst} using EWMA with a parameter value of 0.5.

B. Per-path Congestion Control and Feedback Design

We recognize the fact that each path may differ in parameters or experience different conditions (e.g. different bandwidths, cross-traffic 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 other paths.

Each path i maintains the usual congestion control variables: congestion window (w_i), round trip time (RTT_i) and a timeout value (RTO_i). The congestion window (w_i) determines when a path i can accept packets from the aggregate flow manager. Explicit congestion notification (ECN) on a flow is used to distinguish losses due to congestion from those due to faulty/lossy links, i.e., the congestion window is reduced only in response to ECN feedback.

At the receiver, recovery status for the latest block (SACK map) is fed back across all paths, and is used at the sender to update per-path congestion controllers. Each path congestion controller receives the "ack-report" from the aggregate flow manager after a SACK map is received on *any* of the paths. This report is used to slide the congestion window (i.e., TCP self-clocking), and increase the window based upon the standard TCP congestion control scheme. We modify the congestion window-sliding by scaling the window increase of each path i by it's acceptance rate ($(1-p_i)^{-1}$) to account for the packets lost on it. A path congestion controller will timeout if no feedback is received for a period RTO_i. Hence, timeouts are independent for each path. The response to a timeout is identical to conventional TCP response to a timeout.

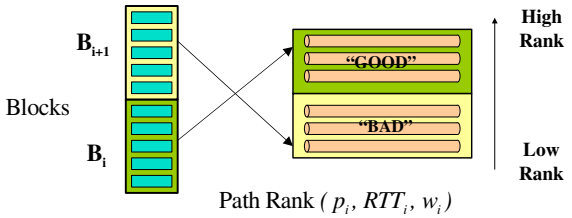

Fig. 2. Mapping blocks to GOOD and BAD paths. Earlier packets are mapped to GOOD paths, and vice versa.

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 is received. This is done because MPLOT may use multiple reverse paths for feedback and a fast-retransmit feature does not allow the use of multiple paths effectively as it would respond to dupacks from different paths often, reducing goodput.

C. Packet Mapping: From Aggregate Block to Path Windows

Different paths may have different RTTs, capacities and loss-rates. Hence, it is important to transmit packets on paths such that packets required for decoding the earliest unrecovered block at the receiver arrive quickly with a high probability. Mapping earlier block packets to longer paths is counter-productive because by the time they arrive at the destination, the block may have been fully recovered from packets received on shorter or low-loss paths, thus wasting capacity on longer, less preferred paths. The packets that will be required in the future (later blocks) can be transmitted on paths with higher RTT and loss rates as transmission opportunities arise on those less preferred paths. Observe that we will have multiple outstanding blocks at any time in transit. This is also naturally expected because a single block may not be sufficient to fill up all paths.

To achieve the desired effect, the aggregate flow manager assigns a "rank"(R_i) to each path i as follows:

$$R_i = w_i(1 - p_i)\frac{RTT_{\max}}{RTT_i}. \tag{6}$$

This ranking 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), normalized by the largest RTT, i.e., $\frac{RTT_{\max}}{RTT_i}$.

Once paths have been ranked, for further operational simplicity, they are grouped into just two classes: GOOD or BAD. The median rank is used as a dividing threshold, i.e., paths with rank greater than the median rank are grouped into the GOOD class and the other paths are classified as BAD. Packets from the earliest unrecovered block are then mapped to any transmission opportunities offered by GOOD paths. The packet mapping procedure is illustrated in Figure 2.

In general, the aggregate flow manager may have multiple outstanding blocks at any time. Let the number of unacknowledged blocks be U. The aggregate flow manager schedules packets from the earliest formed $\frac{U}{2}$ blocks to GOOD paths while packets from most recent $\frac{U}{2}$ blocks are scheduled on BAD paths. The scheduling is also work conserving to ensure that no transmission opportunities are wasted. As earlier blocks are acknowledged and the earliest unacknowledged

pointer moves ahead, the transmission of the recent block's packets shifts from BAD paths to GOOD paths. Our choice of ranking function and classes is motivated by conceptual and implentation simplicity. In Appendix A we show that our choices for block construction, PFEC/RFEC allocation and packet-mapping offer a near-optimal tradeoff between goodput and delay.

IV. Simulation Results

In this section, we present the results obtained through simulations of the MPLOT scheme. The simulation platform used is ns-2. We consider a topology consisting of M parallel paths between a source and a destination node. This topology provides an abstraction of the physical routes (paths), where the different parallel paths in the topology correspond to different, possibly overlapping routes in the underlying network. Scenarios like underlying routes sharing bottleneck links are modeled by considering correlations across losses of different paths in our topology. We further assume that the MAC transmits packets after fixed intervals of time. This allows us to focus only on the loss aspects, keeping out the effects of other MAC details on the overall performance.

In particular, we study several key characteristics of MPLOT. Firstly, we investigate the capability of MPLOT to aggregate bandwidth across paths and utilize diversity across paths to gain additional goodput over the possible aggregated tunnel. Secondly, we stress on the importance of heterogeneity awareness by showing that *heterogeneity aware* MPLOT's performance consistently outperforms a *heterogeneity blind* multi-path protocol and the difference between them grows as a function of number of paths. Thirdly, we show that MPLOT is able to share bandwidth fairly (proportionally) with traditional TCP protocols. Finally, we study the effect of correlated loss rates across paths on the protocol performance, and show that the diversity gains attained degrade gracefully with an increase in the degree of correlation.

A. Bandwidth Aggregation and Diversity Gain

In this section, we study the capability of MPLOT to aggregate bandwidths from 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; therefore, the average loss rate of each path is 50%. The capacity of each path is set to 10 Mb/s. 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.

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. For the case in Figure 3, we observed that the standard deviation for the aggregate loss reduces from 0.132 for 1 path to 0.67 for 4 paths, while the mean aggregate loss rate is constant at 50%.

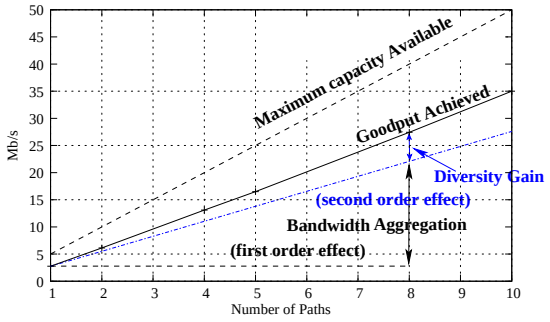

Fig. 3. Bandwidth aggregation and diversity gains obtained by MPLOT. Bandwidth aggregation is linear, whereas MPLOT performs better than linear aggregation due to diversity gain. (2.75 Mb/s out of 5 Mb/s for 1 path to 35 Mb/s out of 50 Mb/s for 10 paths).

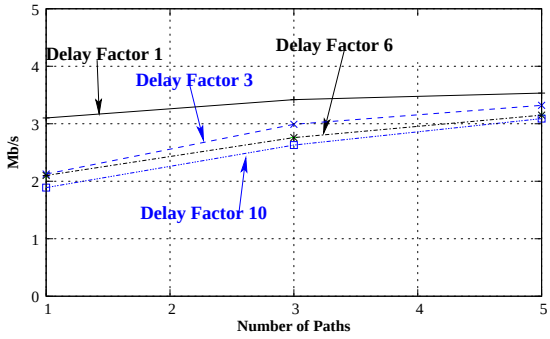

Fig. 4. *Per-path* goodput vs. Number of paths, with different delay factors D. Each path has a capacity of 10 Mb/s, and 50% loss rate. As number of paths increases, the reduction in goodput due to long paths reduces.

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 a single path by a *delay factor* D to $40D$ ms. The RTTs of rest of the paths are kept at 40 ms. We vary D as $1, 2, 6$ and 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 factor D increases. However, we observe that the per-path goodput does not suffer significantly with increasing delay factor, as

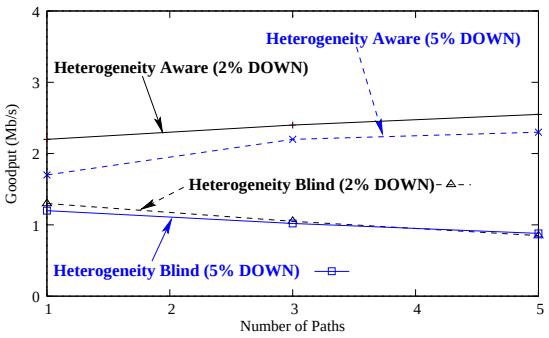

Fig. 5. MPLOT (heterogeneity aware) compared with a heterogeneity blind scheme for different paths. MPLOT exploits diversity gain from paths and improves while heterogeneity blind scheme suffers over multiple paths

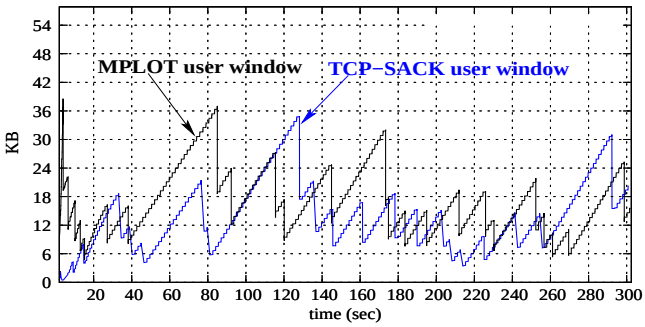

Fig. 6. Path Congestion Window of a MPLOT user and a TCP-SACK user on a path when 5 paths are used. The per-path windows of the 2 sources behave similarly with similar values

the number of paths increases beyond 3. In fact, with 5 paths, the per-path goodput obtained even with a delay factor of 10 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 larger number of paths helps offset the effect of a large delay on a single (or a few) paths. MPLOT is able to use the shorter paths to route the feedback for the longer paths, thus effectively reducing their RTTs.

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 III.

To understand the impact of *heterogeneity awareness*, we compare MPLOT with a *heterogeneity blind* scheme that has the same PFEC, RFEC and block sizing policy as MPLOT, but considers all packets and paths as equivalent. In such a case, the first packet in the queue is mapped to any available path. Consequently, packets from next block are transmitted only after the present block has been recovered and 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 2% and 5%, for two different studies. The bandwidth of each path is fixed at $\frac{10.0}{M}$ 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 from 1.2 Mb/s to 0.9 Mb/s as number of paths increases from 1 to 5. On the other hand, the goodput of the heterogeneity aware scheme (MPLOT) improves significantly (from 1.7 Mb/s to 2.5 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. Fairness with Traditional TCP

The use of multiple paths raises an important question regarding MPLOT's fairness with respect to single-path traditional 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 some other forward path. In this section, we compare MPLOT with TCP-SACK in terms of per-path fairness using the M parallel path topology. For the fairness comparison, The number of TCP-SACK flows and MPLOT flows on each path is kept the same; hence the total number of TCP-SACK flows ($10M$ flows) is M times the number of MPLOT flows (10 flows) . The total aggregate bandwidth is kept fixed at 10 Mb/s and RTT of each path is 40 ms with no losses. Thus if MPLOT and TCP-SACK flows share bandwidth *proportionally* (i.e., have similar per-path bandwidth shares), then the total throughput (goodput) of all TCP-SACK users and that of all MPLOT users must be approximately equal.

When only one path is used, our simulations show that TCP-SACK users share about 4.5 Mb/s of the 10 Mb/s capacity, while MPLOT users share about 4.8 Mb/s, in terms of the overall throughput. The congestion windows of an MPLOT user and a TCP-SACK user are nearly undistinguishable as well. When $M = 5$ paths are used by MPLOT, we observed that the relative bandwidth sharing between MPLOT and TCP-SACK flows was similar to the single-path case. More specifically, in this case, TCP-SACK users share about 4.5 Mb/s of the 10 Mb/s capacity, while MPLOT users share about 4.9 Mb/s. Again, the congestion window evolution on a path for an MPLOT user and a TCP-SACK user are nearly identical, as in Figure 6.

D. 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 of 10 Mb/s capacity each, RTT of 40 ms, as loss rates varying 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. In particular, the correlation is

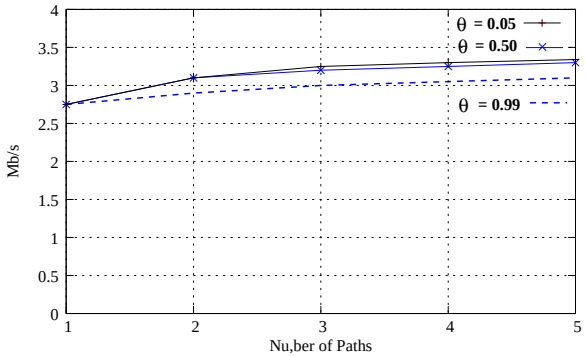

Fig. 7. Effect of interference interference among paths on goodput. As number of paths increases, the reduction in goodput reduces as MPLOT has more paths to select from.

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 7, demonstrate that although the diversity gains attained gets reduced with increasing correlation, the degradation in goodput is 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. As number of paths increases, MPLOT can use the greater path-diversity to transmit useful more packets on paths that have low degree of loss-correlation.

V. 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 and random outage events. Traditional TCP (e.g., TCP-SACK) is vulnerable to residual loss rates of over 5% (especially with longer round trip times) and is unable to take advantage of multiple paths. In contrast, our proposed solution achieves effective bandwidth aggregation and diversity gains from multiple paths, in the presence of delay heterogeneity across paths, bursty, high, correlated loss rates (which can be as high as 75%, with a mean of 50%, as in our example study), and share bandwidth fairly with single-path TCP-SACK users.

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. Although the separation of per-path congestion control and aggregate reliability functions has been suggested before, we are the first to design a complete scheme, especially in the context of highly lossy paths, involving a hybrid ARQ/FEC reliability strategy. FEC's sequence agnostic property allows us to overcome out-of-order delivery issues naturally. However, as our comparisons with a heterogeneity-blind approach show, the use of an intelligent packet mapping design like the one MPLOT uses is required to maximize aggregate goodput over heterogeneous and dynamic component paths. In MPLOT, these effects are realized by sending the latest feedback on all paths, and mapping packets to paths based upon a rank function that values shorter RTT, lower loss and higher capacity paths.

In future work, we plan to extend MPLOT to include flow management (e.g., dynamically dropping flows based upon measured correlations, dynamically adding flows to explore/discover new sources of diversity), cross-layer issues and a Linux/BSD implementation.

Appendix A
Analysis and Parameter Choices

We now analyze MPLOT with the goal of deriving the optimal policy decisions and parameter values that would maximize goodput while meeting desirable delay characteristics. Design of MPLOT involves determination of four key policies: (i) Block construction, (ii) PFEC allocation in a block, (iii) RFEC allocation in response to a RFEC request, and (iv) Mapping of packets to paths. Note that each of the above policies works in a different domain and towards a different goal. The PFEC/RFEC allocation and block construction policies only use aggregate parameters (aggregate loss-rate, bandwidth etc.). The goal of PFEC/RFEC allocation policies is to maximize goodput, while block construction aims towards limiting the average block recovery time. In contrast, the packet mapping policy uses the path parameters to map packets to paths so as to minimize the aggregate loss-rate. We exploit the differences in domains and goals of the above mentioned policies to incrementally arrive at an optimal choice for each policy. We first determine optimal PFEC and RFEC allocation policy by bounding the fraction of blocks that may require additional transmissions. We compute the block size by limiting the average time for block recovery and finally derive an optimal mapping policy.

We first model multiple paths as a single network tunnel with a total bandwidth BW (sum of bandwidths of individual paths) and an aggregate loss rate p_{agg}. For tractability of analysis, we assume that the source transmits at a rate that equals the bandwidth of the tunnel, and ignore the propagation and queueing delays. Consider F data packets (each of size S) being transmitted over the tunnel followed by the FEC packets necessary for its reconstruction. In such a case, receiver would receive a packet after every $T = \frac{S}{BW}$ time. The probability that exactly k FEC packets are needed for recovery (which results in a recovery delay $D = (F + k)T$) is given by

$$P(D = (F + k)T) = \binom{F + k - 1}{k} (p_{\text{agg}})^k (1 - p_{\text{agg}})^F. \tag{7}$$

Let λ be the ratio of FEC packets (needed for recovery of the data packets) to the data packets; then the probability for $\lambda = i$ is given by $h(i) = P(D = F(1 + i)T)$.

Note that the expected goodput $GP = BW/E[1 + \lambda] = BW(1 - p_{\text{agg}})$, which is achievable if there are no constraints on the block recovery time; in presence of such constraints however, the attainable goodput can be lower.

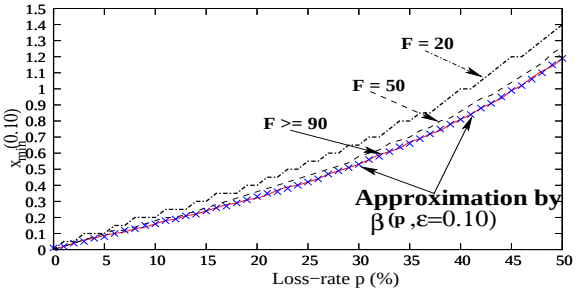

Fig. 8. Comparison of x_{min} vs $\beta(p_{agg}, \epsilon)$ for $F = 20, 50, 90$ and $\epsilon = 0.10$. The approximation $\beta(p_{agg}, \epsilon)$ gets better as F increases for loss-rates $\leq 50\%$.

A. Optimal PFEC/RFEC Allocation Policy

We now consider delay constraints to find an optimal PFEC allocation policy that allows MPLOT to achieve the maximum possible goodput while attaining desired delay characteristics. Consider a block consisting of F data packets and xF PFEC packets. We express the PFEC policy as the ratio x of the PFEC packets and the data packets F in a block.

The probability of recovering the F data packets in $F(1+x)T$ time is $H(x,F) = \sum_{i=0}^{x} h(i)$. We assume that the delay requirement is translated to a minimum probability of block recovery in a single round. This translates to to $H(x,F) \geq 1-\epsilon$ for some given ϵ, i.e., the block-data can be recovered in one round (or within $(1+x)FT$ time) with a probability $\geq 1-\epsilon$.

Since $H(x,F)$ is increasing in x, there exists an $x_{\min}(\epsilon)$ such that $H(x,F) \geq 1-\epsilon \; \forall x \geq x_{\min}(\epsilon)$. Therefore, MPLOT needs to transmit at least $x_{\min}(\epsilon)F$ PFEC packets to satisfy the block recovery probability or delay requirement. 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:

$$x_{\min}(\epsilon) \approx \beta(p_{\text{agg}}, \epsilon) = \alpha_\epsilon \frac{p_{\text{agg}}}{1-p_{\text{agg}}}, \tag{8}$$

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

Figure 8 compares the exact $x_{\min}(\epsilon)$ and $\beta(p_{\text{agg}}, \epsilon)$ for $\epsilon = 0.10$ for different F and p_{agg}. We observe that the error in approximation is negligible for $F \geq 90$. The difference in $x_{\min}(\epsilon)$ and $\beta(p_{\text{agg}}, \epsilon)$ for small F exists because xF 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 I
CALCULATED VALUES OF α_ϵ FOR DIFFERENT ϵ.

Thus, MPLOT must transmit at least $x_{\min}(\epsilon)F$ PFEC packets in a block. The maximum expected goodput $GP(x)$, when MPLOT tranmsits xF PFEC packets, is given by

$$GP(x) = BW \left(\sum_{i=0}^{x} (1+x)h(i) + \sum_{i=x+1/F}^{\infty} (1+i)h(i) \right)^{-1}. \tag{9}$$

Figure 9 shows values of $GP(x)$ as x varies. Since $GP(x)$ is decreasing in x, it follows that the value of x that maximizes $GP(x)$ while satisfying the $1-\epsilon$ bound is $x_{min}(\epsilon)$.

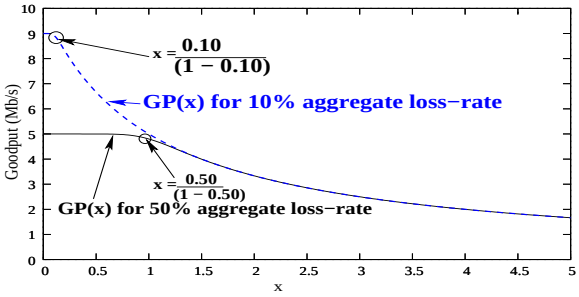

Fig. 9. The variation of goodput $GP(x)$ with PFEC policy x given by (9) for a loss-rate (p) of 10% & 50% with a tunnel bandwidth of $10Mb/s$. Observe that $GP(x)$ decreases as x increases beyond $p/(1-p)$.

Since $\alpha_\epsilon = 1$ for $\epsilon = 0.50$ (Table I), the optimal PFEC policy is very close to the the PFEC policy implemented by MPLOT (as given by (2) in Section III). MPLOT uses $p_{\text{agg}} + \sigma_{\text{agg}}$ instead of p_{agg} to account for any variation in the loss-rates with time. When loss-rates do not change with time, σ_{agg} is negligible making the policy in (2) optimal.

The process of finding the optimal RFEC policy is similar to the PFEC case. MPLOT transmits r RFEC packets followed by κr more RFEC packets in response to a request for r RFEC packets, and receiving any r packets would be enough to recover the block-data. We get the optimal RFEC policy by replacing F, x and ϵ by r, κ and δ respectively. The attainable goodput can be obtained by modifying (9), and is decreasing in κ. The optimal RFEC allocation is then expressed as κ_{opt}, which is approximated by $\beta(p_{\text{agg}}, \delta)$. Note that the MPLOT's RFEC policy (in (3), Section III) is optimum for $\delta = 0.50$.

B. Optimal Block Size

The block size B is the sum of data packets F and xF PFEC packets. With optimal PFEC allocation, $B = F + \lfloor F\beta(p_{\text{agg}}, \epsilon) \rfloor + 1$. The block size B should be chosen so as to limit the average time taken to recover the block-data. Let D_A denote the desired upper bound on the average block recovery time. Then the block size B must be upper bounded as

$$B \leq BW.D_A(1-p_{\text{agg}})(1+\beta(p_{\text{agg}}, \epsilon)).$$

Note that BW is the aggregate sum of path bandwidths BW_i. Moreover, BW_i is estimated as the ratio of the path window w_i and RTT RTT_i. This translates upper bound on B to

$$B \leq \sum_{i=1}^{M} w_i \frac{D_A}{RTT_i}(1 + (\alpha_\epsilon - 1)p_{\text{agg}}) = B_{\max}. \tag{10}$$

A larger block size will result in a better goodput; therefore, the optimal block size is $B_{\max}$. Note that for $\epsilon = 0.50$ and $D_A = RTT_{\text{med}}$, $B_{\max}$ reduces to the expression for the block size used by MPLOT (see (1)).

C. Optimal Mapping Policy

The choice of a mapping policy is important as it significantly affects the amount of goodput we can gain from the path diversity available. The mapping policy of MPLOT is

p	GP_s	GP_{op}	$P_R^s(1)$	$P_R^{op}(1)$	$P_R^s(2)$	$P_R^{op}(2)$
0.10	7.40	7.99	0.74	0.65	0.99	0.98
0.20	6.50	7.05	0.70	0.64	0.97	0.95
0.30	5.65	6.11	0.67	0.59	0.96	0.91
0.40	4.75	5.20	0.66	0.59	0.92	0.89
0.50	3.80	4.32	0.64	0.58	0.86	0.85

TABLE II
MPLOT'S COMPARISON WITH THEORETICAL OPTIMUM. MPLOT ACHIEVES GOODPUT VALUES CLOSE TO THE OPTIMAL LEVELS.

represented by the vector $\vec{Q}_M = \{q_1, .., q_M\}$ where q_i is the probability that a packet is mapped to path i. The aggregate loss-rate is then expressed as

$$p_{\text{agg}} = \sum_{i=1}^{M} q_i p_i. \tag{11}$$

where p_i is the loss-rate on path i. An optimal mapping policy minimizes $\beta(p_{\text{agg}}, \epsilon)$, which implies minimizing p_{agg}.

The delay incurred on path i is a random variable that depends on parameters like q_i, p_i and F. We aim to restrict the probability of delay on path i being greater than D_{avg} (mean aggregate delay) + σ_D (standard deviation of aggregate delay) to be less than half. This gives us

$$q_i \leq \left(1 + \frac{\sigma_D}{D_{\text{avg}}}\right) \frac{BW_i}{BW}. \tag{12}$$

The problem now becomes to find a positive $\vec{Q}_M$ that minimizes (11) while respecting the bounds in (12) and the fact that the q_i must sum up to unity.

We observe that the greedy mapping policy for MPLOT operates within 10% of the optimal values. For 2 paths with bandwidths 4 Mb/s (loss-rate 10%) and 6 Mb/s (loss rate 50%), the greedy algorithm results in an aggregate loss-rate of 32.37% as compared to the optimal value of 30.8%. When the loss-rates for the two paths are interchanged, the greedy algorithm provides an aggregate loss-rate of 24.57% as compared to the optimum of 22.4%. Thus the greedy algorithm achieves near-minimal loss rate with lower complexity.

D. Example and Comparison with Simulation

As argued above, the PFEC & RFEC allocation and the block size determination policies implemented by MPLOT (as described in Section III) optimize goodput under the constraint that each round of packet transmission (data & PFEC or RFEC) recovers at-least half of the blocks. In this section, we present an example to demonstrate this.

We simulate the case of MPLOT transmitting over 2 paths of equal bandwidths 5 Mb/s, equal RTTs 40 ms and equal loss-rates p using the ns2 network simulator. The packets size is 550 bytes (including 50 bytes of header).

Table II compares the simulation and optimal values for the above case. GP_s represents the goodput value (in Mb/s) obtained from simulation and GP_{op} is the optimal value. $P_R^s(1)$ represents the fraction of blocks recovered after 1 round (data & PFEC) of packet transmissions from simulation, and $P_R^{op}(1)$ is the corresponding optimal value. $P_R^s(2)$ and $P_R^{op}(2)$ respectively denote the simulation and optimal values of the fraction of blocks recovered after 2 rounds (1 data & PFEC + 1 RFEC) of transmissions.

We observe that MPLOT achieves aggregate goodput are atleast 92% of the optimal values. We also note that the values for round 1 recovery probabilities from simulation are higher than the optimal values; this is because MPLOT uses a slightly higher PFEC allocation than the optimal to account for time-variance in loss-rate. However, the fraction of blocks recovered after round 2 is nearly the same for both schemes. Therefore, MPLOT attains a near-optimal trade-off between goodput and delay constraints in this case.

ACKNOWLEDGEMENT

We thank the sponsors AT&T Labs Research and MIT Lincoln Labs for their support (grant letter No. 14-S-06-0206) and our collaborators from MIT Lincoln Labs - Biswaroop Ganguly, Stephen McGarry and Linda Zeger.

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, p. 121132.
[2] C. Steger, P. Radosavljevic, and J. P. Frantz, "Performance of ieee 802.11b wireless lan in an emulated mobile channel. 2003. vtc 2003-spring," in *The 57'th IEEE Semiannual Vehicular Technology Conference*, vol. 2, april 2003, p. 14791483.
[3] Y. Lee, I. Park, and Y. Choi, "Improving tcp performance in multipath packet forwarding schemes," vol. 4, no. 2, pp. 148–157, june 2002.
[4] 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 '04)*, 2004.
[5] M. Zhang, "Understanding internet routing anomalies and building robust transport layer protocols," Ph.D. dissertation, Department of Computer Science, Princeton University, sep 2005.
[6] H.-Y. Hsieh and R. Sivakumar, "A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts," in *Proc. ACM Mobicom*, 2002.
[7] 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.
[8] A. Tsirigos and Z. Haas, "Analysis if multipath routing - part i: The effect on the packet delievery ratio," in *IEEE Transactions on Wireless Communications*, vol. 3, no. 1, jan 2004.
[9] ——, "Analysis if multipath routing - part ii: Mitigation of the effectsof frequently changing network topologies," in *IEEE Transactions on Wireless Communications*, vol. 3, no. 1, jan 2004.
[10] 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)*, Beijing,China, Aug-Sep 2005.
[11] ——, "Realizing the benefits of user-level channel diversity," vol. 35, no. 5, oct 2005.
[12] 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.
[13] 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.
[14] 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*, Taormina, Italy, june 2005.
[15] V. Subramanian, S. Kalyanaraman, and K. Ramakrishnan, "An end-to-end transport protocol for extreme wireless environments," in *Proc. IEEE Military Communications Conference (MILCOM 06)*, Washington D.C, USA, october 2006.
[16] D. Jurca and P. Frossard, "Video packet selection and scheduling for multipath streaming," in *IEEE Transactions on Multimedia*, vol. 9, no. 3, april 2007.
[17] K. Chebrolu and R. Rao, "Bandwidth aggregation for real-time applications in heterogeneous wireless networks," in *IEEE Trans. Mobile Computing*, vol. 5, no. 4, april 2006, pp. 388–403.
[18] T. Nguyen and A. Zakhor, "Distributed video streaming with forward error correction," in *Proc. Packet Video Workshop*, Pittsburgh,PA, april 2002.
[19] Y. Li, Y. Zhang, L. Qiu, and S. Lam, "Smarttunnel: Achieving reliability in the internet," in *Proc. of Infocom'07*, 2007.