Multi Path PERT

Ankit Singh and A. L. Narasimha Reddy
Electrical and Computer Engineering Department,
Texas A&M University; email: reddy@ece.tamu.edu.

***Abstract*—This paper presents a new multipath delay based algorithm, MPPERT (Multipath Probabilistic Early response TCP), which provides high throughput and efficient load balancing. In all-PERT environment, MPPERT suffers no packet loss and maintains much smaller queue sizes compared to existing MPTCP, making it suitable for real time data transfer. MPPERT is suitable for incremental deployment in a heterogeneous environment.**

PERT, being a delay based TCP protocol, has continuous information about the state of the bottleneck queue along its path. This information is valuable in enabling MPPERT to detect subflows sharing a common bottleneck and obtain a smaller set of disjoint subflows. This information can even be used to switch from coupled (a set of subflows having interdependent increase/decrease of congestion windows) to uncoupled (independent increase/decrease of congestion windows) subflows, yielding higher throughput when best single-path TCP constraint is relaxed. The ns-2 simulations support MPPERT as a highly competitive multipath approach, suitable for real time data transfer, which is capable of offering higher throughput and improved reliability.

I. Introduction

TCP faces several challenges to ensure fair and efficient share of the network resources [1]. Today, demand for bandwidth and reliability is much higher for real time applications such as VOIP, IPTV etc. The general approach to provide better reliability is to provide multiple redundant paths through the network. Many devices now come with both Wifi and 3G capabilities. Several networks now employ multihoming to provide multiple paths through the network. These capabilities can be used collectively to improve both reliability as well as total throughput for the end user. These interfaces could be used to establish multiple connections between the end hosts, using multiple paths, to add the desired redundancy. Thus, even if there is a failure in one of the paths, other paths may be used to maintain the connectivity. Multipath protocols could also be used for improving load balance across the network resources (links and routers).

This paper looks at the problem of adopting a delay-based transport protocol, PERT, to a multipath environment. Delay based TCP algorithms can be quite efficient. Algorithms such as PERT(probabilistic Early Response TCP), provide high throughput, minimum loss rates and maintain low queue sizes that makes them suitable for real time applications.

PERT's response to congestion can be broken into two probabilities [2], p and p$'$, which corresponds to early response and observed congestion loss probability respectively. PERT's throughput is controlled by the combined early response and congestion response probabilities and is given by $1-(1-p)(1-p') = p + p' - p\ p'$. If PERT has to roughly get an equal share when competing with TCP, comparing the steady state throughput equations of the two protocols, we can derive that the window increase function of PERT is given by

$$\alpha_{PERT} = p + p' - \frac{p * p'}{p} \approx 1 + \frac{p'}{p}$$

The parameter α may take different values depending on the mode in which PERT operates [3]. Homogeneous environments use only one flavor of transport protocol. However, Internet is heterogeneous as numerous flavors of transport protocols are employed. PERT uses the observed queuing delay as an indicator for depicting the type of competing environment (homogeneous/heterogeneous) and its available link bandwidth. *High speed* : When the observed queuing delay is less than some minimum threshold, PERT infers that the bandwidth is being underutilized and increments α (starting at 1) linearly till it reaches a threshold of 32. This enables PERT to fill up high speed links quickly. *Safe Mode* : When queuing delay is greater than the minimum threshold, but less than half the maximum observed queue length, PERT assumes that all the competing flows are PERT flows and decrements α till it reaches 1. *Compete Mode* : If the observed queuing delay is larger than half the maximum queue length, PERT infers that it is competing in a heterogeneous environment, and increments α till it reaches $\alpha_{PERT} = 1 + \frac{p'}{p}$. Parameter β determines the factor by which a PERT flow will reduce its congestion window in case of early response. The probability of packet loss increases with increase in the queue size. Thus, it is desired to reduce the congestion window by a larger amount as the queue size progresses towards the maximum queue length. Thus β is given by

$$\beta = \frac{q_{curr}}{q_{curr} + q_{max}}$$

where q_{curr} and q_{max} are the smoothed values of the queue sizes.

Multipath transport protocols employ subflows on each path and shift traffic from one path to another based on observed congestion status. When congestion is primarily deciphered through packet losses, the congestion response can be uniform across the multiple paths.

However, as seen above, delay based protocols respond to congestion both based on observed queuing delays and to packet losses. In addition, the need to compete against loss-based protocols such as TCP, may make protocols such as PERT operate in multiple modes. These multiple modes of operation and multiple responses to congestion complicate how PERT could be adopted to multipath environment. This paper proposes a mulipath version of PERT, MPPERT, which ensures

that it receives throughput at least equal to single-path PERT on the best of its available paths.

The remainder of the paper is organized as follows: We will discuss the related work in Section II. We will present the MPPERT algorithm in Section III and its approach to detecting shared bottleneck links in Section IV. We provide an evaluation of MPPERT in V and conclude the paper in section VI.

II. Related Work

Multihoming can improve the performance and resilience by using multiple simultaneous paths [4]. Several solutions for multipath problem have been suggested. mTCP [5] stripes data packets across parallel, independent TCP subflows. It maintains a sequence of fast retransmit intervals for each of the subflows and computes correlation to infer a shared link. mTCP takes upto 15 seconds to detect the shared bottleneck which may not be acceptable. Parallel TCP (pTCP) [6], Concurrent multipath Transfer (CMT) over SCTP [7] send data packets independently using uncoordinated congestion control algorithms. These protocols don't handle/detect common bottlenecks and do not fairly share the available bandwidth. R-MTP [8] targets wireless links.

Fluid-flow modelling [9], [10] was used to show that not only can multipath transport give robustness, but with a right coupled congestion controller, it can balance congestion in a stable manner in the internet. Although the fluid model provided insight and assured stability, However, these algorithms were shown to behave erratically, flipping almost all traffic on one path to another with non-periodicity [11]. Ability to shift traffic off congested path provides resource pooling [12] capability, which promotes fair distribution of network resources among the competing flows. BMC [13] adaptively changes the contributions of subflows to achieve resource pooling. may not be scalable. Recently proposed MPTCP algorithm [16] presents a window based congestion algorithm, which aims at distributing traffic inversely proportional to path loss. It compensates for RTT variability among the multipath subflows and takes as much throughput as it would get with single-path TCP on the best of its paths.

Presently, we are not aware of any delay based transport protocol that supports multiple paths.

III. MPPERT algorithm

Multipath PERT allows a single data stream to be split across multiple paths. A multipath connection consists of a set of subflows R, each of which may take a different route through the Internet. Each subflow r $\in$ R maintains its own congestion window W_r. An MPPERT sender stripes packets across these subflows as space in the subflow windows becomes available. The new MPPERT algorithm can be given as:

- Each ACK on subflow r, increase window Wr by $\min\left(\frac{a\alpha_r}{W_r^k}, \frac{\alpha_r}{W_r}\right)$
- Each early response/packet loss on subflow r, decrease the window Wr by $\beta W r$.

Here α_r is the window increase factor of subflow r, β is the window decrease factor and 'a' is the scaling factor which scales the window increase to meet the total throughput requirement. By treating early responses and responses to packet losses similarly, we minimize the complexity in adopting PERT to multipath scenarios. Since PERT adopts its window decrease function to the observed queue length (and hence to early response and packet losses), the response across multiple paths takes the level of congestion into account. In order to promote resource pooling, similar to MPTCP, it is desired that MPPERT flow should take throughput equal to the single-path TCP throughput available on the best of its paths. Differential equation for PERT can be given as :

$$f(W, W_r, T_q, P) = \frac{\alpha}{R} - \frac{\beta W(t) W_r(t) P(t)}{R} \tag{1}$$

$$g(W, T_q) = \frac{NW(t)}{RC} - 1, \tag{2}$$

where P(t) is the congestion response rate at time t, N is the number of flows, R is the RTT and C is the link capacity.

In steady state, setting equations 1 and 2 to zero gives us

$$W_{tcp} = \sqrt{\frac{\alpha}{\beta P}} = \frac{RC}{N} \tag{3}$$

Equation 3 gives steady state window size for PERT. In steady state, each MPPERT subflow would have equal increase and decrease in its window size. This can be given as follows:

Assuming $1 - P_r \approx 1$

$$\frac{a\alpha_r}{W_r^k} = \beta_r P_r W_r$$

$$\frac{a\alpha_r}{\beta_r P_r} = W_r^{k+1}$$

$$a^{1/(k+1)} W_{tcp_r}^{2/(k+1)} = W_r \tag{4}$$

Now aplying the throughput constraint of single-path TCP on the best available path, we have

$$\sum_r \frac{W_r}{RTT_r} = \max_r \frac{W_{tcpr}}{RTT_r} \tag{5}$$

using equation 4

$$\sum_r \frac{W_r}{RTT_r} = \max_r \frac{W_r^{(k+1)/2}}{RTT_r a^{1/2}}$$

$$a = \frac{\max_r \frac{W_r^{(k+1)}}{RTT_r^2}}{\left(\sum_r \frac{W_r}{RTT_r}\right)^2} \tag{6}$$

Thus, scale parameter 'a' ensures that the total throughput of MPPERT flow is at least equal to PERT throughput on the best of its available paths. Equation 4 provides window size relationship between a MPPERT subflow and a PERT flow

competing on the same path. The resource pooling parameter 'k' controls the sensitivity of MPPERT flows to path loss. As k is varied from +1 to -1, the sensitivity to path loss increases, which promotes traffic shift from more congested to less congested paths.

Subflow Throughput

$$\propto \frac{W_{tcp_r}^{2/(k+1)}}{RTT_r}$$

ANALYSIS : TWO SUBFLOW MPPERT

In the present analysis, MPPERT stripes packets across two subflows which maintain their own congestion windows w_r. MPPERT flow sends packets across these subflows as space in the subflow windows becomes available. For simplicity, we consider equal RTTs for all the subflows. Using equation (4), the ratio of congestion windows, in steady state, of single-path PERT flows can be given as

$$\frac{w_{tcp1}}{w_{tcp2}} = (\frac{w_1}{w_2})^{(k+1)/2} = \ n(say)$$

$$a = \frac{\max\limits_r \frac{W_r^{(k+1)}}{RTT_r^2}}{(\sum\limits_r \frac{W_r}{RTT_r})^2}$$

Assuming RTT1=RTT2, we can rewrite

$$a = \frac{n^2 w_2^{(k+1)}}{(w_2 + n^{2/(k+1)} w_2)^2}$$

$$a = \frac{n^2 w_2^{(k-1)}}{(1 + n^{2/(k+1)})^2} \tag{7}$$

The increment for each subflow is given as :

$$(\frac{a\alpha_1}{w_1^k}, \frac{a\alpha_2}{w_2^k})$$

or

$$(\frac{n^2 \ n^{2(1-k)/(k+1)}\alpha_1}{(n^{2/(k+1)}+1)^2 w_1}, \frac{n^2\alpha_2}{(n^{2/(k+1)}+1)^2 w_2})$$

The common factor in addition to normal tcp increase of $\frac{\alpha}{w}$ is

$$\frac{n^2 \ \alpha}{(n^{2/(k+1)}+1)^2 w}$$

In order to provide higher increase to the subflow with higher available bandwidth, we maximize factor $n^{2(1-k)/(k+1)}$ and keep $n^{2(1-k)/(k+1)} >= 1$. Solving these constraints we have

$$\frac{(1-k)}{(k+1)} >= 0 \implies -1 < k <= 1$$

The constraint suggests that varying k from +1 to -1 will increase the amount of traffic shift by increasing the sensitivity of a subflow to the corresponding path loss. In order to at least provide traffic shift equivalent to single-path TCP throughput distribution, we have

$$(\frac{w_1}{w_2}) = \ n^{2/(k+1)} >= n, \quad where \ n >= 1$$

or

$$\frac{2}{k+1} >= 1 \implies -1 < k <= 1$$

The constraint also holds true for n-MPPERT flow (MPPERT flow with n subflows) for n>=1. Selection of parameter 'k' affects the amount of traffic shift achieved by MPPERT subflows off the congested path. However, performing floating point operations in the kernel is generally avoided. This would suggest choosing k=0 would present a trade off between the computational requirement and amount of traffic shift achieved by the subflows.

IV. MPPERT PATH SELECTION

One of the goals of MPPERT is not to harm or take unnecessary bandwidth advantage over normal single-path flow. MPPERT tries to get as much throughput as it would get with a single-path flow on the best of its paths. Increasing the number of paths improve the chances of getting better throughput. However, having large number of multipath subflows may not produce any significant benefit over a minimal set of efficient subflows. Thus, one may suppress the subflows that do not add significantly to the throughput improvement and select only those subflows that offer reasonable bandwidth advantage.

Another approach could be to promote resource pooling and avoid the shared bottleneck link. This would avoid subflows competing with each other at the shared bottleneck. So, if MPPERT can successfully detect the shared bottleneck, it can be used to suppress the self competing subflows. It would help in maintaining a smaller set of subflows which lower the computational requirements. Thus, there is a need to detect common bottleneck successfully to be able to promote disjoint subflows and obtain a smaller, efficient set of multipath subflows. PERT continuously monitors queue sizes based on RTT values and accumulates important data for detection of the shared bottleneck. As Internet is highly dynamic, probability of different routers having same queue dynamics at a given instant is highly unlikely. The likelihood is further reduced if the set of routers are reduced to the ones used by a multipath flow. Thus, correlation among the measured queue sizes for subflows having a shared bottleneck should be higher than with disjoint queues [18]. Although the idea is quite apparent, we may face issues realizing it. Some of the issues are as follows : (a) Rate at which a subflow receives queue information may not be sufficient, (b) Queue sizes could get subjected to random changes, (c) Determining time stamp for the queue signal and (d) Two different subflows sharing a bottleneck can have different sending rates. A large time difference between the signals of different subflows may lower the correlation.

Smoothed RTT provides reliable control over the dynamics of the queue. It may increase the response time required to determine shared bottlenecks, but helps in countering random variations in queue estimates. In order to compare signals of subflows having different throughput, we take average

over frequently occurring samples between two consecutive timestamps of the slower subflow. It is important not to take average over samples which are far apart in time, to avoid large fluctuations in queue sizes.

We collect a series (q_i) of queue samples for each subflow. We use averaging to reduce the mismatch between the number of available samples. We then calculate cross-correlation between the pair of subflows using perason's correlation coefficient given by

$$r_{q1,q2} = \frac{\sum_i (q_{1i} - \bar{q_1})(q_{2i} - \bar{q_2})}{\sqrt{(\sum_i (q_{1i} - \bar{q_1})^2 \sum_j (q_{2j} - \bar{q_2})^2)}}$$

This gives us instantaneous cross-correlation between the two subflows. We take moving average over these instantaneous values to obtain average cross-correlation between the subflows. Allow (t) sec to collect sufficient samples to measure average correlation and use thresholding to detect shared bottleneck subflows. Larger time (t) would provide better shared link detection. Choosing a higher threshold value would reduce false positives (disjoint subflow as shared). However, it may also increase false negatives (shared subflow as disjoint). Thus, depending on the system requirement, suitable choice of threshold value would enhance its performance. Shared bottleneck detection can be used to suppress the subflows sharing the bottleneck links or can be used to aggregate throughput across disjoint paths as shown in later section V.

V. EVALUATION

We carry out the evaluation of MPPERT in three stages. First, we present an evaluation of MPPERT's performance in different multi path environments, where we compare its performance to single path PERT. We then compare MPPERT with MPTCP. We then evaluate MPPERT's bottleneck detection algorithm and show how that can be employed.

A. Comparison with single path PERT

We use ns-2 simulations to test and evaluate MPPERT algorithm. The experiment setup is shown in Fig. 1. There are two bottleneck links which connect (two subflow) MPPERT sender to its destination MPPERT receiver. In the homogeneous environment, all the background flows are of PERT flavor. Background flows carry FTP traffic with a packet size of 1000 bytes. The start time of the traffic is uniformly distributed on the interval (0, 20) sec with RTTs set to 40ms. In PERT-TCP 50-50 mix scenario, total number of background flows contain 50% of PERT and 50% of TCP Reno flows.

Throughput Distribution: It is desired that MPPERT be capable of efficiently shifting traffic off the congested path. In order to precisely compare the amount of traffic shifts, we compare the performance by varying the number of background flows while keeping the link bandwidths constant. The decrease in the amount of background traffic increases

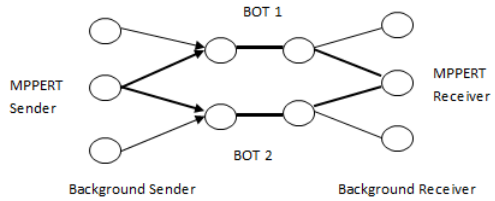

Fig. 1. Experiment setup for two multipath flow

the available BW for MPPERT subflows. We maintain 40 background flows on one path and vary the number of flows on the other path from 40 to 10. The notation used for simplicity is given as path 1 B.W. (number of background flows on path1) + path 2 B.W. (number of background flows on path2).

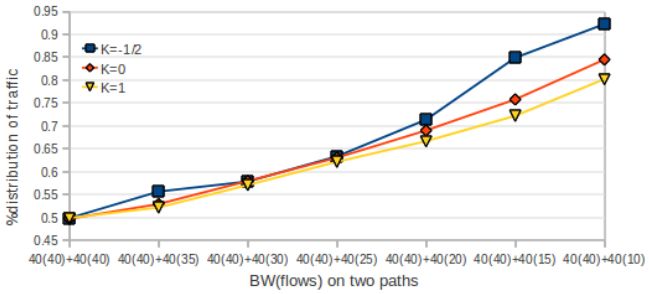

Fig. 2. MPPERT throughput distribution with varying k

For instance, case 40(40)+40(20) suggests two 40Mbps bottleneck links with 40 and 20 background flows on path1 and path2 respectively. Fig. 2 shows traffic distribution with variation in available BW. When both links have the same available B.W, case 40(40)+40(40), traffic is distributed equally for all values of k. The results suggest that as we move from k=1 to -1, the amount of traffic shift increases. For k=-1/2, we observe the traffic shift to go from 50% to 93% with the variation in available BW.

Total Throughput Constraint: To have an incentive for MPPERT deployment, it should at least provide throughput equal to (single-path) PERT throughput on the best of its available paths. Fig. 3 shows throughput comparison of the MPPERT flow with average throughput of the background flows. For the MPPERT flow, we observe an increase in throughput share of subflow 2 (path2) as we change the number of background flows on path2 from 40 to 10, keeping the number of background flows on path1 constant. The results suggest that MPPERT flow maintains total throughput close to the best single-path throughput of the background PERT flows.

Incremental Deployment of MPPERT: We further test the feasibility of incremental deployment of MPPERT in 50-50 PERT-TCP mix environment. In Fig. 4, we observe that MPPERT maintains its total throughput more than the single-path TCP throughput on the best of its paths. Here, PERT tries to pump in packets early in the queue and decreases the amount of traffic when the queue becomes large, to minimize packet loss. TCP on the other hand, continuously sends packets till a packet gets dropped.

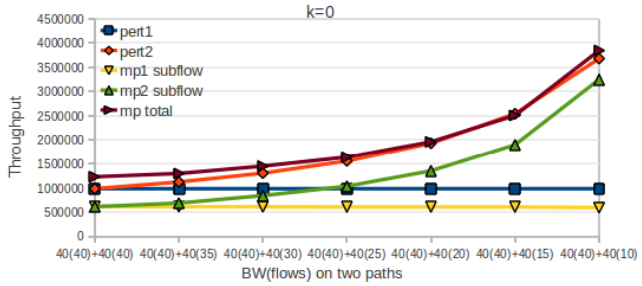

Fig. 3. MPPERT throughput (bytes/s) with varying number of background flows

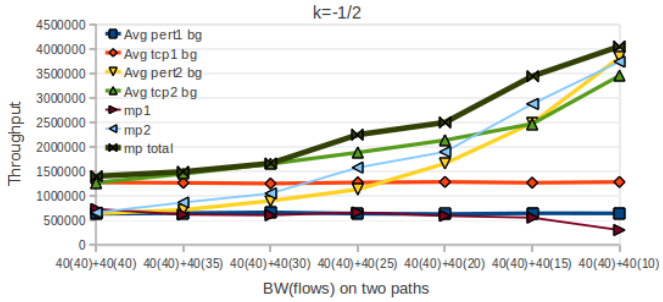

Fig. 4. MPPERT (k=-1/2) throughput in heterogeneous 50-50mix environment with varying number of background flows

Resource Pooling: We compare MPPERT with independent multipath PERT (uncoupled flows), to analyze the performance of resource pooling.

%endfigure

We consider scenario as shown in Fig. 5. Here, S2 is a multipath capable flow. Bottleneck links 1,2 are 12Mbps and 18Mbps respectively. RTT is set to 40ms. Ideally, the resource pooling should distribute the total capacity of 30Mbps equally among the 3 flows (10Mbps each). We perform experiments with independent multipath PERT and MPPERT flows. We observe from Table I that MPPERT achieves distribution closer to the ideal share of 10Mbps each.

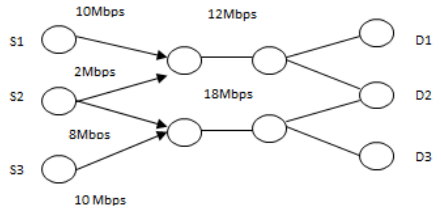

Fig. 5. Network Setup for comparing multipath flows with single path flows

We carried out several other experiments to test the responsiveness of MPPERT to traffic dynamics such as sudden arrival and departure of flows or traffic. We also carried out experiments to test the flappiness of MPPERT i.e., how frequently does it switch traffic among the multiple paths

TABLE I. RESOURCE POOLING COMPARISON OF MPPERT AND INDEPENDENT SUBFLOW MULTIPATH WITH COMPETING PERT FLOWS (MBPS)

	Independent Multipath	**MPPERT (K=-1/2)**	**MPPERT (K=0)**
Throughput S1	6.80	9.42	8.90
Throughput S2 path 1	5.08	2.56	3.05
Throughput S2 path 2	8.31	7.40	6.84
Throughput S2 Total	12.97	9.96	9.89
Throughput S3	13.39	9.65	10.32
Total System	28.89	29.02	29.12

TABLE II. PERFORMANCE COMPARISON MPPERT, MPTCP FOR TAIL DROP QUEUE MANAGEMENT

10(10)+10(10)	**MPTCP**	**MPPERT**
Avg Background Throughput path1 (KBps)	982.9	949.2
Avg Background Throughput path2 (KBps)	981.9	947.7
Multipath subflow 1 (KBps)	158.8	523.1
Multipath subflow 2 (KBps)	167.9	538.3
Multipath total (KBps)	326.7	1061.4
Avg Queue Size path 1 (Packets)	32.39	13.21
Avg Queue Size path 2 (Packets)	32.50	12.77
System Throughput (MBps)	19.98	20.03

available between the source and destination. MPPERT was found to exhibit good properties on these tests, for more details, please see [19].

B. Performance Comparison MPPERT and MPTCP

This section provides performance comparison and analysis of our new multipath algorithm, MPPERT, with MPTCP [17]. We employ (a) Queue Management, (b) Packet Loss, (c) Throughput and subflow traffic Distribution, and (d) Resource Pooling in our evaluation.

Queue Management: MPPERT performs Active Queue Management (AQM) at the end hosts. This relaxes the need for AQM capability at the routers and provides greater control to the end hosts. MPPERT performs well with both RED and Tail Drop queue management at the network routers. We employ experiment setup similar to Fig. 1 with RTT 40ms, buffer size 1BDP and each bottleneck link of 10Mbps capacity carrying 10 background traffic each (case 10(10)+10(10)).

Table II shows a performance comparison of MPTCP and MPPERT flows in a homogeneous environment using Tail Drop Queue Management. We observe that the total throughput of MPTCP flow is considerably less than the average throughput of background TCP Reno. MPTCP also maintains higher average queue length of 65% of the total buffer size in comparison with 26% for MPPERT. MPPERT maintains throughput close to 1Mbps, which is the best single-path background PERT throughput.

MPTCP performs much better with RED queue management as per Table III. MPPERT performs similarly with both droptail and RED management at the routers.

Packet Loss: MPPERT is a delay based protocol which proactively responds to congestion to avoid packet losses. On the other hand, TCP is a loss based protocol which requires packet loss to perform congestion control. In an all-PERT environment, after initial congestion, MPPERT quickly achieves stable state and avoids any further packet loss. MPTCP on the other

TABLE III. PERFORMANCE COMPARISON MPTCP, MPPERT FOR RED ACTIVE QUEUE MANAGEMENT

10(10)+10(10)	MPTCP	MPPERT
Avg Background Throughput path1 (KBps)	931.5	859.5
Avg Background Throughput path2 (KBps)	928.4	864.8
Multipath subflow 1 (KBps)	498.7	625.1
Multipath subflow 2 (KBps)	528.4	638.0
Multipath total (KBps)	1027.1	1263.2
Avg Queue Size path 1 (Packets)	5.08	4.98
Avg Queue Size path 2 (Packets)	4.96	4.85
System Throughput (MBps)	19.63	18.51

hand, continuously introduces a large number of packet losses and retransmissions which decrease the total throughput of the system. Fig. 6 shows packet loss at the two bottleneck links for MPTCP with TCP background and MPPERT with PERT background flows for 10(10)+10(10) configuration. The result suggests that after initial stabilization, PERT environment suffers no packet loss whereas TCP environment continuously keeps on losing packets at the bottleneck link.

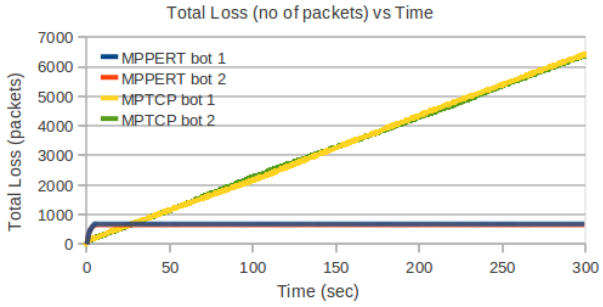

Fig. 6. MPPERT and MPTCP pathloss comparison at bottleneck 1 and 2

Throughput and Subflow Traffic Distribution: We compare total throughput achieved by MPPERT and MPTCP for the cases 10(10)+10(10) and 10(10)+10(2) as shown in Table IV. Here, both the flows operate in their homogeneous environments with MPTCP using RED and MPPERT using Tail Drop queue management schemes (both using favorable queue management schemes). The results in Table IV suggest that in a homogeneous environment, in comparison with MPTCP, MPPERT flow maintains throughput much closer to the throughput of the best single- path background flow. Moreover, in all the experiments, the MPPERT network consistently maintains higher total system throughput than the MPTCP network. The traffic distribution is quite similar for both MPTCP and MPPERT, which ranges from equal distribution of 0.5 to 0.9 for 10(10)+10(2) configuration. Here, traffic distribution is given by the ratio of higher throughput subflow and the total multipath throughput.

Fig. 7 shows performance of 2-MPPERT (two subflow MPPERT) and 2-MPTCP (two subflow MPTCP) competing together in a 50-50 mix (50% PERT and 50% Reno) environment. We set RTT to 160ms and buffer size to 1BDP. We observe that the MPPERT flow acts more aggressively and aquires higher total throughput compared to the MPTCP flow. MPPERT is much closer to the throughput of the best-single path background flow. The simulation results suggest that a

TABLE IV. TOTAL THROUGHPUT AND SUBFLOW DISTRIBUTION COMPARISON

10(10)+10(10)	MPTCP KBps	MPPERT KBps
Avg Background Throughput path1	931.5	949.2
Avg Background Throughput path2	928.4	947.7
Multipath subflow 1	498.7	523.1
Multipath subflow 2	528.4	538.3
Multipath total	1027.1	1061.4
Multipath throughput distribution	0.485	0.492
System Throughput	19,626.1	20,030.0
10(10)+10(2)	**MPTCP**	**MPPERT**
Avg Background Throughput path1	950.4	967.9
Avg Background Throughput path2	3,482.6	3,317.4
Multipath subflow 1	310.0	336.1
Multipath subflow 2	2,500.0	3,242.4
Multipath total	2,810.0	3,578.5
Multipath throughput distribution	0.889	0.906
System Throughput	19,279.5	19,892.2

MPPERT flow, though slightly more aggressive, is able to compete and successfully shift traffic off the congested path in a heterogeneous environment.

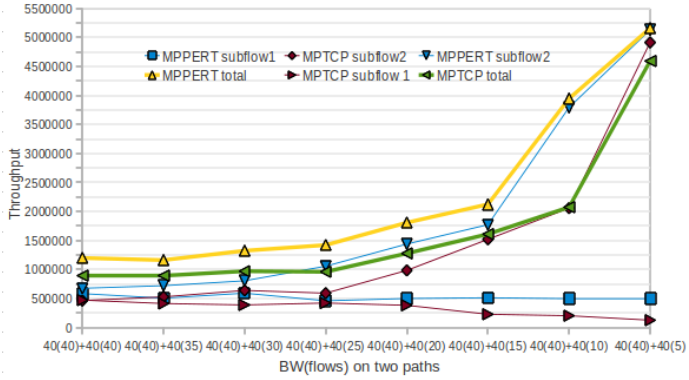

Fig. 7. Throughput (bytes/s) comparison for 2-MPPERT and 2-MPTCP competing in heterogeneous environment

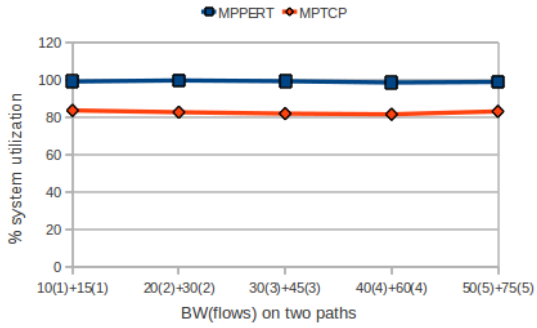

Fig. 8. System resource utilization comparison for 2-MPPERT and 2-MPTCP

It is observed with MPTCP that when the available BW is high, the total achieved system throughput is low. To analyze the performance, we perform experiments with 2-MPPERT and 2-MPTCP systems having available BW of 5 Mbps and above. We then scale the link bandwidth and the background traffic

TABLE V. RESOURCE POOLING COMPARISON OF MPPERT AND MPTCP

	MPTCP MBps	**MPPERT** MBps
Throughput S1	6.74	8.45
Throughput S2 path 1	3.04	3.48
Throughput S2 path 2	5.83	6.86
Throughput S2 Total	8.87	10.30
Throughput S3	8.50	10.24
System Throughput	24.20	29.00

proportionally, maintaining the same available BW, to monitor change in system resource utilization. Fig. 8 shows that the MPPERT system maintains much higher system throughput, close to 100% utilization, compared to MPTCP system which obtains 80% system utilization. MPPERT system, on account of low packet loss, attains higher total system resource utilization.

Resource Pooling: To analyze the resource pooling performance of MPPERT and MPTCP, we compare both algorithms for scenario shown in Fig. 5. Ideal resource pooling should distribute the total 30Mbps capacity equally (10Mbps each) among the 3 competing flows. Table V shows that in homogeneous environment, the traffic distribution of flows S1,S2,S3, for MPPERT is 84.5%, 103% and 102% of the ideal 10Mbps distribution in comparison to 67.4%, 88.75% and 85.04% for MPTCP. Total throughput of the system is about 20% higher for MPPERT than with MPTCP.

C. Shared Bottleneck Detection

The setup uses 3-MPPERT (MPPERT with 3 subflows) with PERT as the background traffic for the homogeneous case as shown in Fig. 9. In the experiment, Subflow1 and 3 share a common bottleneck whereas subflow 2 is disjoint. Background flows carry FTP traffic with their start times uniformly distributed on the interval [0-20] sec. Each bottleneck link has a capacity of 40Mbps. The RTT is set to 80ms. Each link carries 20-20 PERT-Reno background flows giving available bandwidth of approximately 1Mbps. The experiment

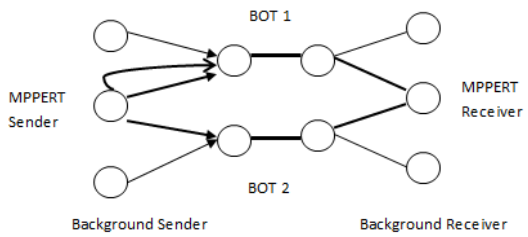

Fig. 9. Experiment setup for three multipath flows

maintains similar background conditions for both links. We perform shared bottleneck link detection test and distinguish shared/disjoint subflows. We decouple(make independent) the disjoint subflows while keeping the shared ones as coupled.

Homogeneous Environment: We analyze the correlation for both shared and disjoint subflows. The data in Fig. 10 shows that subflows with a shared bottleneck have much higher correlation compared to the ones that are disjoint. Usually, for all-PERT flows, the correlation is higher compared to 50-50 mix case. In the present case, we have chosen threshold = 0.5. This successfully separates the shared bottleneck from a disjoint one. This information can be used to decide which subflows share a link and can be suppressed to achieve a smaller, efficient set of subflows.

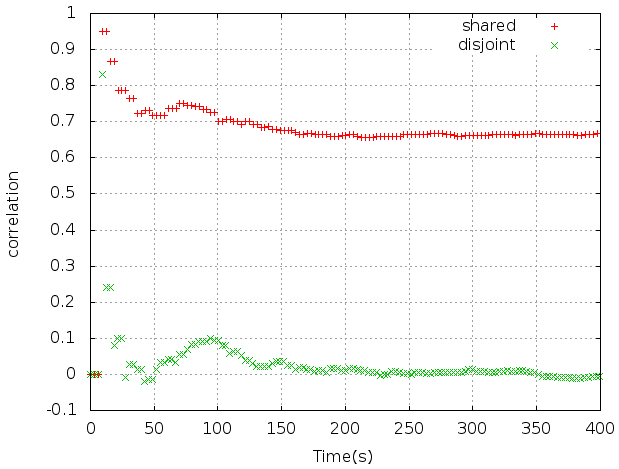

Fig. 10. Correlation of MPPERT subflows sharing a common or disjoint bottlenecks in a homogeneous environment

The result of congestion window variation for the three subflows as per experiment are shown in Fig. 11. Initially all the subflows start as a coupled system which later results in subflow2 becoming independent and subflow 1 and 3 experiencing the coupling. We also observe that congestion windows for both subflows 1 and 3 sharing bottleneck 1 are equally distributed because of coupling and achieve a total window size close to subflow 2, which experiences similar background traffic. This information can be used to suppress shared link flows and is advisable to keep the threshold low to boost the true positives (ones indicating shared bottleneck link).

Heterogeneous Environment: In TCP-PERT 50-50 mix case, keeping aggressiveness factor increase of 0.1 and α_{max} constraint of 16, yield decent performance for PERT. The correlation for shared and disjoint links are shown in Fig. 12. Suitable choice of thresholding is important for the correct detection of shared and disjoint links. The data in Fig. 10 and Fig. 12 show that MPPERT is able to distinguish shared and disjoint paths in both homogeneous and heterogeneous environments.

When losses dominate in a heterogenous environment, PERT observes lower correlation of subflows sharing a bottleneck link than when the packet loss rates are lower. This may sometimes delude in concluding shared bottleneck subflows as being disjoint. This may lead to false -ve detection of the shared bottleneck. In order to reduce such cases, one can change the threshold from a fixed value to a set of values proportional to the available BW. As the available BW increases, correlation increases and detection of shared

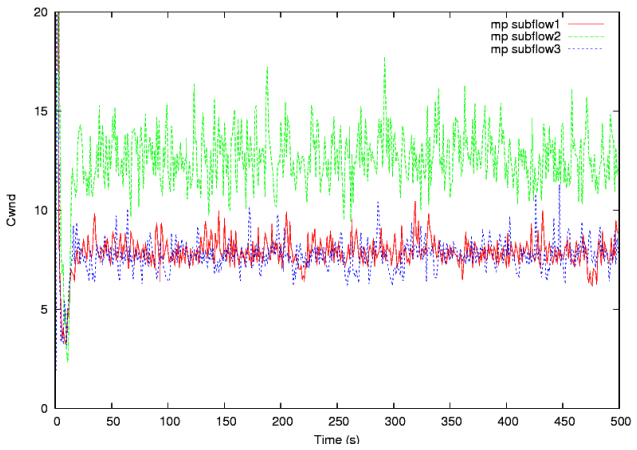

Fig. 11. MPPERT disjoint/shared path subflow congestion window variation

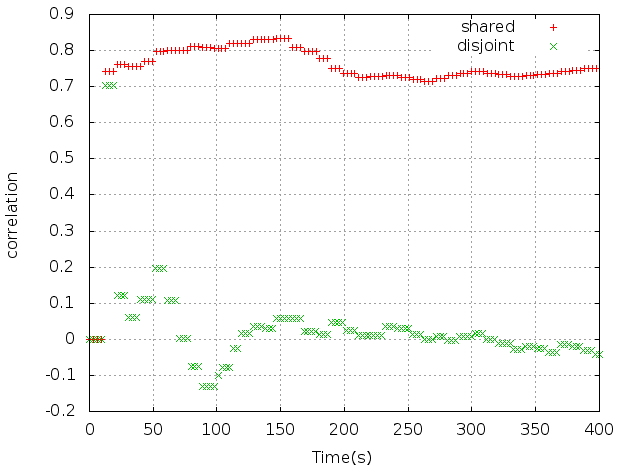

Fig. 12. Correlation for MPPERT flows with shared and disjoint bottlenecks in heterogeneous 50-50 mix environment

bottlenecks becomes easier.

Suppressing Shared Subflows: Multipath approach improves reliability and throughput share by providing multiple concurrent connections between the end hosts. One of the key concerns is fairness. The idea is that a multipath flow should not take higher BW than a single-path TCP at the bottleneck. This can be ensured by coupling only subflows sharing the bottleneck link. Resource pooling strives to turn Internet into a single pool of resources and multipath flows facilitate single path flows by shifting traffic off the congested paths. This idea forces all multipath subflows, having shared/disjoint bottleneck links, to be coupled under the best single-path TCP throughput constraint. Raising the resource pooling restriction would allow multipath to couple only the shared subflows and make the disjoint subflows independent. This would allow multipath flows to achieve total throughput equal to the aggregate of single path TCP throughputs on the disjoint paths. Suppressing subflows with shared bottleneck links help in reducing the total number of paths.

TABLE VI. COUPLED WITH CORRELATION BASED SUPPRESSION ON 20(10) PATHS

Starting 6-MPPERT	**6 disjoint** KBps	**4 disjoint** KBps	**2 disjoint** KBps
Avg. Background Throughput	1,965.5	1,939.5	1,905.8
Multipath subflow 1	503.7	120.8	834.6
Multipath subflow 2	265.1	120.8	120.2
Multipath subflow 3	497.3	555.6	120.3
Multipath subflow 4	626.5	754.4	120.3
Multipath subflow 5	383.7	844.3	120.3
Multipath subflow 6	733.1	631.9	882.0
Multipath total	3,009.3	3,027.6	2,197.6

We analyze two approaches to utilize shared subflow suppression technique. For completely coupled case (like MPTCP), it can be used to supress the shared subflows and maintain complete coupling over the set of disjoint subflows, to promote resource pooling. For coupled/decoupled case, it can be used to supress shared subflows and allow disjoint subflows to become independent (like m-TCP, p-TCP which provide throughput aggregation). For this kind of system, one can start with all-coupled multipath flow and based on shared/disjoint subflow information, create subsets of coupled/independent subflows. It then suppresses shared subflows with smaller throughput to obtain a set of disjoint subflows. In the experiment, we start with 6-MPPERT subflows and depending on the number of available disjoint paths, suppress the subflows sharing a bottleneck link. Each disjoint path uses 20Mbps link with 10 background flow configuration (20(10)). The network has one shared bottleneck link. So, for the 4 disjoint path configuration, it puts subflows 1,2,3 on path1 and the remaining subflows 4,5,6 on the disjoint paths 2,3 and 4 respectively. Table VI shows the performance of coupling with correlation based supression. We observe that the bottleneck detection technique successfully classifies the shared subflows from disjoint subflows. For 6 disjoint path configuration, we observe that none of the subflows get suppressed whereas 4 and 2 disjoint path configurations suppress 1 and 3 of its shared subflows (subflows with throughput $\approx$ 0.12Mbps) respectively. Table VII shows performance of coupling/decoupling with correlation based suppression approach. MPPERT again detects the shared links efficiently and sets the disjoint subflows independent. For the 6 disjoint path configuration, it sets all the subflows independent whereas 4 and 2 disjoint path configurations suppress 1 and 3 of its shared subflows (subflows with throughput $\approx$ 0.12Mbps) respectively. This provides throughput aggregation over the available disjoint paths. When flows are allowed to be decoupled, MPPERT achieves aggregate bandwidth that is approximately equal to the number of disjoint paths * the single path bandwidth.

VI. CONCLUSIONS

In this paper, we have proposed a new multipath transport protocol, MPPERT, adopting PERT to multipath environment. The algorithm offers granular control by providing parameteric adjustments over the amount of traffic shift desired from the multipath subflows. Through ns-2 simulations, we have analyzed the performance of MPPERT based on properties like resource pooling, system throughput and packet loss etc.

TABLE VII. COUPLED/DECOUPLED SWITCH WITH CORRELATION BASED SUPPRESSION ON 20(10) PATHS

Starting 6-MPPERT	**6 disjoint** KBps	**4 disjoint** KBps	**2 disjoint** KBps
Avg Background Throughput	1,835.9	1,833.7	1,838.6
Multipath subflow 1	1,806.2	1,579.6	120.1
Multipath subflow 2	1,834.0	120.1	120.1
Multipath subflow 3	1,764.3	120.1	120.1
Multipath subflow 4	1,801.1	1,884.8	1,249.4
Multipath subflow 5	1,770.1	1,774.4	120.1
Multipath subflow 6	1,797.6	1,801.0	1,811.7
Multipath total	10,779.2	7,279.9	3,541.5

Simulations suggest that MPPERT provides performance boost from single-path PERT.

A comparative study of MPPERT and MPTCP is provided. The results reflect that MPPERT in homogeneous environment outperforms MPTCP in terms of throughput and resource pooling. It suffers no packet loss after initial stabilization and results in higher total system throughput. It also maintains smaller queue lengths offering smaller delays to real time applications. MPPERT can also be incrementally deployed in heterogeneous environment where it helps achieve higher system throughput. This makes MPPERT deployment quite attractive.

The paper suggests methods to detect shared bottleneck link subflows of a MPPERT flow. This then can be used to supress shared link subflows and promote lower number of disjoint paths. This reduces computational requirements while providing similar througput gains.

REFERENCES

[1] M. Handley, "Why the Internet only just works," *BT Technology Journal*, vol. 24, no. 3, pp. 119-129, 2006.

[2] K. Kotla and A. L. Narasimha Reddy, "Making a Delay-based Protocol Adaptive to Heterogeneous Environments," in *Proc. 16th International Workshop on Quality of Service*, 2008, pp. 100-109.

[3] S. Bhandarkar, A. L. Narasimha Reddy, Y. Zhang, and D. Loguinov, "Emulating AQM from End Hosts," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 349-360, 2007.

[4] C. Raiciu, D. Niculescu, M. Bagnulo, and M. Handley, "Opportunistic mobility with multipath TCP," in *Proc. 6th ACM International Workshop on Mobility in the Evolving Internet Architecture*, 2011, pp. 7-12.

[5] M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson and R. Wang, "A transport layer approach for improving end-to-end performance and robustness using redundant paths," in *Proc. USENIX'04 Annual Technical Conference*, 2004, pp. 99-112.

[6] H.Y. Hsieh, and R. Sivakumar, "A transport layer approach for achieving aggregate bandwidths on multihomed mobile hosts," in *Proc. 8th Annual International Conference on Mobile Computing and Networking*, 2002, pp. 83-94.

[7] J. R. Iyengar, K. C. Shah, P. D. Amer, and R. Stewart, "Concurrent multi-path transfer using SCTP multihoming over independent end-to-end paths," *ACM/IEEE Transactions on Networking*, vol. 29, no. 10, pp. 951-964, 2006.

[8] L. Magalhaes and R. Kravets, "Transport level mechanisms for bandwidth aggregation on mobile hosts," in *Proc. IEEE International Conference on Network Protocols*, 2001, pp. 165-171.

[9] F.P. Kelly, T. Voice, "Stability of end-to-end algorithms for joint routing and rate control," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 2, pp. 5-12, 2005.

[10] H. Han, S. Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley, "Multi-path TCP: a joint congestion control and routing scheme to exploit path diversity in the Internet," *IEEE/ACM Transactions on Networking*, Vol. 14, no. 6, pp. 1260-1271, 2006.

[11] D. Wischik, M. Handley and C. Raiciu, "Control of multipath TCP and optimization of multipath routing in the Internet," in *Proc. NetCOOP*, 2009, pp. 204-218.

[12] D. Wischik, M. Handley and M. B. Braun, "The Resource Pooling Principle," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 5, pp. 47-52, 2008.

[13] M. Honda, Y. Nishida, L. Eggert, P. Sarolahti, and H. Tokuda, "Multipath Congestion Control for Shared Bottleneck," in *Proc. 8th International Workshop on Protocols for Future, Large-Scale & Diverse Network Transports*, 2009, pp. 19-24.

[14] I. Rhee and L. Xu, "Limitations of Equation-based Congestion Control," *IEEE/ACM Transactions on Networking*, vol. 15, no. 4, pp. 852-865, 2007.

[15] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang and A. Vahdat. Hedera, "Dynamic Flow Scheduling for Data Center Networks," in *Proc. 7th USENIX Symposium on Networked System Design and Implementation*, 2010, pp. 281-296.

[16] A. Ford, C. Raiciu, and M. Handley, "TCP extensions for multipath operation with multiple addresses," Internet Engineering Task Force, Internet Draft, draft-ietf-mptcp-multiaddressed-06, October 2010.

[17] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in *Proc. of 8th USENIX Symposium on Networked Systems Design and Implementation*, 2011, pp. 8-8.

[18] D. Rubenstein, J. Kurose, and D. Towsley, "Detecting shared congestion of flows via end-to-end measurement", in *Proc. of ACM Sigmetrics*, 2000, pp.145-155.

[19] A. Singh, "Multipath Probabilistic Early Response TCP", Master's thesis, Texas A&M University, http://cesg.tamu.edu/wp-content//uploads/2012/01/ankit_singh_thesis_report.pdf, Aug. 2012.