MULTIPATH PROBABILISTIC EARLY RESPONSE TCP

A Thesis

by

ANKIT SINGH

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2012

Major Subject: Computer Engineering

MULTIPATH PROBABILISTIC EARLY RESPONSE TCP

A Thesis

by

ANKIT SINGH

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	A. L. Narasimha Reddy
Committee Members,	Riccardo Bettati
	Srinivas Shakottai
Head of Department,	Costas N. Georghiades

August 2012

Major Subject: Computer Engineering

ABSTRACT

Multipath Probabilistic Early Response TCP. (August 2012)
Ankit Singh, B.Tech., Indian Institute of Technology, Guwahati
Chair of Advisory Committee: Dr. A. L. Narasimha Reddy

Many computers and devices such as smart phones, laptops and tablet devices are now equipped with multiple network interfaces, enabling them to use multiple paths to access content over the network. If the resources could be used concurrently, end user experience can be greatly improved. The recent studies in MPTCP suggest that improved reliability, load balancing and mobility are feasible. The thesis 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. It also presents a parametrized approach to tune the amount of traffic shift off the congested path.

Multipath approach is benefited from having multiple connections between end hosts. However, it is desired to keep the connection set minimal as increasing number of paths may not always provide significant increase in the performance. Moreover, higher number of paths unnecessarily increase computational requirement. Ideally, we should suppress paths with low throughputs and avoid paths with shared bottlenecks. In case of MPTCP, there is no efficient way to detect a common bottleneck between subflows. MPTCP applies a constraint of best single-path TCP throughput,

to ensure fair share at a common bottleneck link. The best path throughput constraint along with traffic shift, from more congested to less congested paths, provide better opportunity for the competing flows to achieve higher throughput. However, the disadvantage is that even if there are no shared links, the same constraint would decrease the overall achievable throughput of a multipath flow.

PERT, being a delay based TCP protocol, has continuous information about the state of the queue. 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.

To my parents.

ACKNOWLEDGMENTS

I thank my research advisor, Dr. A. L. Narasimha Reddy, for his continuous guidance, feedback and support. His immense knowledge and key inputs were pivotal for the smooth and successful completion of my research objective. I heartily appreciate his cooperation, compassion and patience during my acute knee injury. I thank former group members Bin Qian and Kiran kotla for their valuable inputs and help in getting me started with the research. I am grateful to my elder brother Amrit for the encouragement and guidance that streamlined my academic and professional career. Above all, I thank my parents for their unconditional love, encouragement and support, through thick and thin, that helped me achieve my goals. Last but not the least, I thank Almighty God for giving me wisdom and strength to overcome my limitations.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

A. Motivation

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. 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. This facilitates reliable mobility of the end hosts.

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.

B. Related Work

Simultaneous use of multiple end-to-end paths between two end-hosts is becoming an increasingly important problem as growing number of end-hosts are now multihomed. Multihoming can improve the performance and resilience by using multiple simultaneous paths [2].

This thesis follows the style of *IEEE Transactions on Automatic Control*.

Several solutions for multipath problem have been suggested. mTCP [3] stripes data packets across parallel, independent TCP subflows. It integrates a bottleneck detection mechanism to identify and suppress the subflows that traverse the same congested link. 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) [4], Concurrent multipath Transfer (CMT) over SCTP [5] 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 [6] targets wireless links. It probes the bandwidth availability periodically for each of the subflows and adjusts the rates accordingly. It uses packet's inter-arrival times and jitter information to detect congestion. Increase in jitter is used as an indicator of the mounting congestion.

Kelly [7] has shown that congestion control at endsystems can be thought of as a distributed control system for solving a network wide optimization problem. Fluid-flow modelling [8, 9] 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, algorithms behaved erratically, flipping almost all traffic on one path to another with non-periodicity [10]. Fully-coupled multipath algorithms [10] offer high traffic shift capability but they suffer from the traffic flips. Ability to shift traffic off congested path provides resource pooling [11] capability, which promotes fair distribution of network resources among the competing flows. BMC [12] adaptively changes the contributions of subflows to achieve resource pooling. BMC assigns weights to individual subflow so that a bundle of subflows can have the same aggressiveness as

one TCP flow. When sending rates of flows are different, BMC increases loss event rates [13] on subflows by being less aggressive. An alternative solution for balancing traffic is to use a centralized scheduler [14]. The scheduler assigns large flows to a lightly loaded path and reassigns existing flows such that the overall throughput is maximized [14]. With the arrival of new flows, the scheduler is required to collect flow level statistics and perform placement computations during the scheduling period which raises serious scalability concerns. Recently proposed MPTCP algorithm [15] takes into account these issues and 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 multipath algorithm that supports multihoming. Congestion oriented protocols such as TCP depend on packet loss as the sole indicator of congestion. The corresponding multipath algorithms use these loss signals to adjust the traffic distribution among the subflows. On the other hand, delay based algorithms use RTTs to estimate queue dynamics. These continuous delay signals provide better opportunity to detect the onset of congestion. The corresponding multipath algorithms can use the increase in queue size as an indicator of congestion, to shift traffic from more congested to less congested paths. Continuous queue dynamics present better opportunity to efficiently detect a shared bottleneck. This can be used to ensure fairness at the bottleneck link. Thus, delay based algorithms have the potential to provide fair, responsive multipath algorithms.

CHAPTER II

BRIEF DESCRIPTION

Today, demand for internet resources are ever increasing but due to protocol constraints, network resources are underutilized. Even when the end devices are connected to the network through multiple paths, the realized throughput can be impacted by the congestion on both the paths, if the transport protocol does not manage the resources well [16]. One approach to increase the utilization of entire system is to make a collection of resources behave like a single pooled resource. This is called Resource Pooling [11, 17]. As per Fig. 1(a), three different multipath flows are connected to their respective destinations using two subflows each. The principle argues that even though all the multipath flows may have different connections, the complete network resource of 36Mbps should get distributed equally (12Mbps each), treating it as a single pool of 36Mbps resource that is getting shared by 3 multipath users. Conceptually, it is similar to max-min fairness, if we treat the entire network resource as a single pool of resources available for the competing flows. The two goals that resource pooling aims to realize are

- Increase the resilience of the connectivity by providing multiple paths, protecting end hosts from the failure of one.
- Increase the efficiency of the resource usage, and thus increase the network capacity available to end hosts.

Multipath TCP allows a single data stream to be split across multiple paths. A connection consists of 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 Wr.

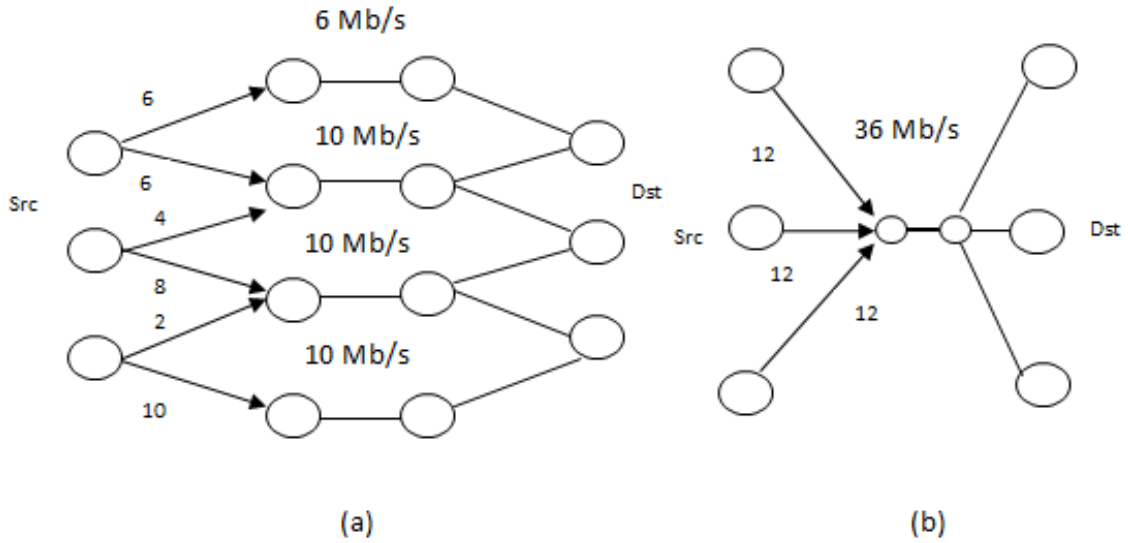

Fig. 1.: Resource pooling

An MPTCP sender stripes packets across these subflows as space in the subflow windows becomes available. There are a number of window based congestion algorithms for multipath TCP. Some of them are as follows:

Equally-Weighted TCP

- For each ACK on path r, increase window Wr by a/Wr.
- For each loss on path r, decrease window Wr by Wr/2.

Here, Wr is the window size on path r and 'a' is a scaling constant. Throughput of a TCP flow can be given as

$$T = \frac{s}{R\sqrt{\frac{2p}{3a}} + t(3\sqrt{\frac{3p}{8a}})p(1+32p^2)}$$

In order to scale subflow throughput by a factor D, one can select $a = D^2$ [12]. By choosing $a = 1/n^2$ (n is the number of subflows used by a multipath flow) and

assuming equal RTTs, multipath flow gets the same throughput as a regular TCP at the bottleneck link. Here, each subflow gets 1/n times the throughput of single-path TCP. However, EWTCP does not shift traffic from more congested to less congested paths.

COUPLED

- For each ACK on path r, increase window Wr by 1/Wtotal .
- For each loss on path r, decrease window Wr by Wtotal/2.

COUPLED [8, 9] provides the traffic shift and coupling required for a multipath algorithm. In steady state, the total window size is proportional to the path with the minimum loss.

$$\frac{1}{W_{total}} = \frac{pW_{total}}{2} \quad \Longrightarrow \quad W_{total} = \sqrt{\frac{2}{p_{min}}}$$

Since COUPLED tries to send traffic only to path with the least path loss, it results in rejection of alternate paths with higher path loss. Moreover, if congestion in alternate paths improve over time and the main path degrades, there is no way to determine the change due to lack of traffic signals from the alternate paths. Thus, it is desirable to always keep sufficient traffic on all the paths, as a probe, to efficiently respond to the change in congestion.

MPTCP

MPTCP [10] aims to keep moderate amount of traffic on each path while having bias towards less congested paths. It thus strikes a balance between the aggressive traffic shift of COUPLED and no traffic shift of EWTCP. Some of the design goals for MPTCP are as follows:

- A multipath flow should give a connection at least as much throughput as it would get with single-path TCP on the best of its paths. This ensures there is an incentive for deploying multipath algorithm.

- A multipath flow should take no more capacity on any path or collection of paths than if it was a single-path TCP flow using the best of those paths. This guarantees it will not unduly harm other flows at a bottleneck link, no matter what combination of paths passes through that link.

MPTCP Algorithm can be given as :

- Each ACK on subflow r, increase the window Wr by $\min\left(\frac{a}{W_{total}}, \frac{1}{W_r}\right)$

- Each loss on subflow r, decrease the window Wr by Wr/2.

The steady state distribution for Wr can be given as

$$\frac{2a}{W_{total}\ p_r} = W_r$$

MPTCP [10] offers coupling of paths and distributes traffic inversely proportional to path loss. This helps in maintaining probe traffic in high loss paths. The issue of RTT mismatch [18] may lead to decrease in overall throughput. The factor 'a' provides a scale to the window increase to achieve total throughput equal to at least the best single path available throughput.

The basic TCP/IP stack architecture [15] is given in Fig. 2. This requires a wrapper layer, MPTCP, over the normal TCP layer which stripes packets across these subflows for transmission. It uses two levels of sequence space. One is the connection level sequence number (MPTCP level) and the other for each subflow at TCP level. This

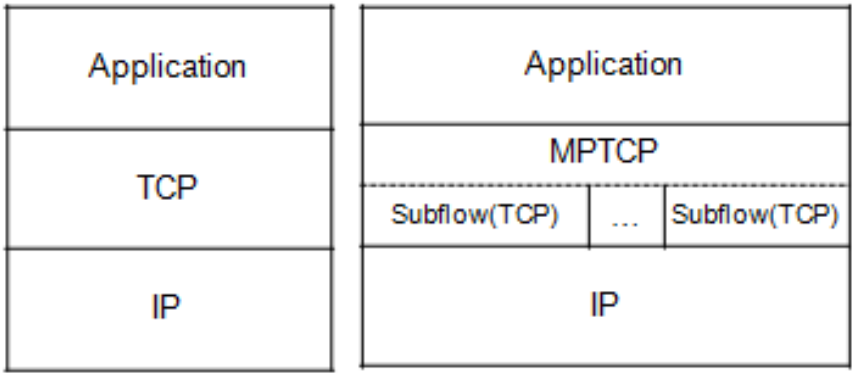

Fig. 2.: MPTCP TCP/IP stack

permits connection-level segmentation, reassembly and retransmission of the same part of connection-level sequence space on different subflow-level sequence space.

CHAPTER III

NEW MP-PERT ALGORITHM

A. Algorithm

Multipath PERT adopts PERT to employ multiple paths in the network. PERT measures delays in a path and responds early to perceived congestion before a packet is dropped. In the homogeneous environment, PERT is shown to offer zero packet losses. A natural question is how should PERT respond to different levels of congestion when it utilizes multiple paths through the network.

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 $(\frac{a\alpha_r}{W_r^k}, \frac{\alpha_r}{W_r})$
- Each early response/packet loss on subflow r, decrease the window Wr by βWr.

Here '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. 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 :

$$f(W, W_r, T_q, P) = \frac{\alpha}{R} - \frac{\beta W(t) W_r(t) P(t)}{R} \tag{3.1}$$

$$g(W, T_q) = \frac{NW(t)}{RC} - 1 \tag{3.2}$$

In steady state, setting equations 3.1 and 3.2 to zero gives us

$$W_{tcp} = \sqrt{\frac{\alpha}{\beta P}} = \frac{RC}{N} \tag{3.3}$$

Equation 3.3 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 $1 - P_r \approx 1$

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

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

$$a^{1/(k+1)} W_{tcp_r}^{2/(k+1)} = W_r \tag{3.4}$$

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

$$\sum_r \frac{W_r}{RTT_r} = \max_r \frac{W_{tcpr}}{RTT_r} \tag{3.5}$$

using equation 3.4

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

$$a = \frac{\max_r \frac{W_r^{(k+1)}}{RTT_r^2}}{(\sum_r \frac{W_r}{RTT_r})^2} \tag{3.6}$$

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 3.4 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.

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

Parameters Adjustment: Homogeneous/Heterogeneous Environment

Homogeneous environments use only one flavor of TCP. Thus, PERT homogeneous environment would require all the competing background flows to be PERT flows. Internet is highly diverse and numerous flavors of TCP are available. The environment where multiple flavors of TCP compete is termed as heterogeneous environment.

MPPERT ensures that it atleast receives throughput equal to single-path PERT on the best of its available paths. In order for MPPERT to compete with TCP, it is desired that its PERT subflows be able to compete with TCP flows. Thus, enabling PERT subflows in MPPERT to individually compete with TCP will enable MPPERT to compete in heterogeneous environment. Steady state throughput of a PERT flow is given by

$$\frac{1}{RTT}\sqrt{\frac{\alpha}{\beta p}}$$

PERT's response to congestion can be broken into two probabilities [19], 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 eventually arrive at

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

In summary, parameter α may take different values depending on the mode in which PERT operates [20]. It uses amount of queue build-up/observed queuing delay as an indicator for depicting the type of competing environment (homogeneous/heterogeneous) and its available link B.W.(high speed links).

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 in-

crease 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.

B. 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 (3.4), 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

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

$$a = \frac{n^2 w_2^{(k-1)}}{(1 + n^{2/(k+1)})^2} \tag{3.7}$$

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 atleast 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.

CHAPTER IV

EVALUATION OF MPPERT

A. Experiment Setup

We use ns-2 simulations to test and evaluate MPPERT algorithm. The experiment setup is shown in Fig. 3. 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.

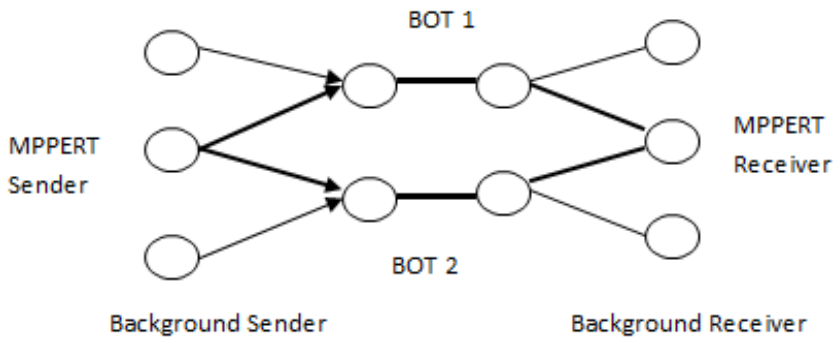

Fig. 3.: Experiment setup for two multipath flow

B. Flappiness

Flappiness [10] occurs when one of the two paths having the same drop probability, due to fluctuations, suffers from time to time, a few extra drops and looks momentarily more congested. Thus, depending on random occurrences of loss, either subflow

windows can reach zero. Another issue could be that if one flow experiences a couple of drops, the other subflow needs to experience many more drops to get the traffic rates back into balance.

We monitor the congestion window distribution between two subflows to analyze the flappiness induced by various algorithms. Fig. 4 shows that when both subflows experience same path loss, there is no flappiness observed in the congestion windows of the subflows. When path 1 experiences lower loss than path 2, congestion window 1 (path 1) dominates the traffic distribution. This shows that with change in path loss one can change the distribution of total throughput among the available paths. As we vary k from 1 to -1, we observe that the amount of traffic shift increases. However, this may also accentuate the variations in congestion window based on momentary fluctuations in path loss.

Flappiness is usually observed when there is no per subflow steady state distribution of traffic and the coupling constraint only involves the total per flow congestion window. For instance, COUPLED multipath TCP algorithm requires the total window size (per flow) to be inversely proportional to the least path loss. This does not provide a unique throughput distribution for the subflows and may even lead to complete rejection of the higher loss paths. Simulations suggest that MPPERT does not suffer from the issue of flappiness. Moreover, it settles down to a steady state distribution much faster with low variations. Fig. 4 shows flappiness comparison of MPPERT algorithm with varying k=1,0,-1/2. As expected, it shows that traffic shift increases as we progress from k=1 to k=-1/2.

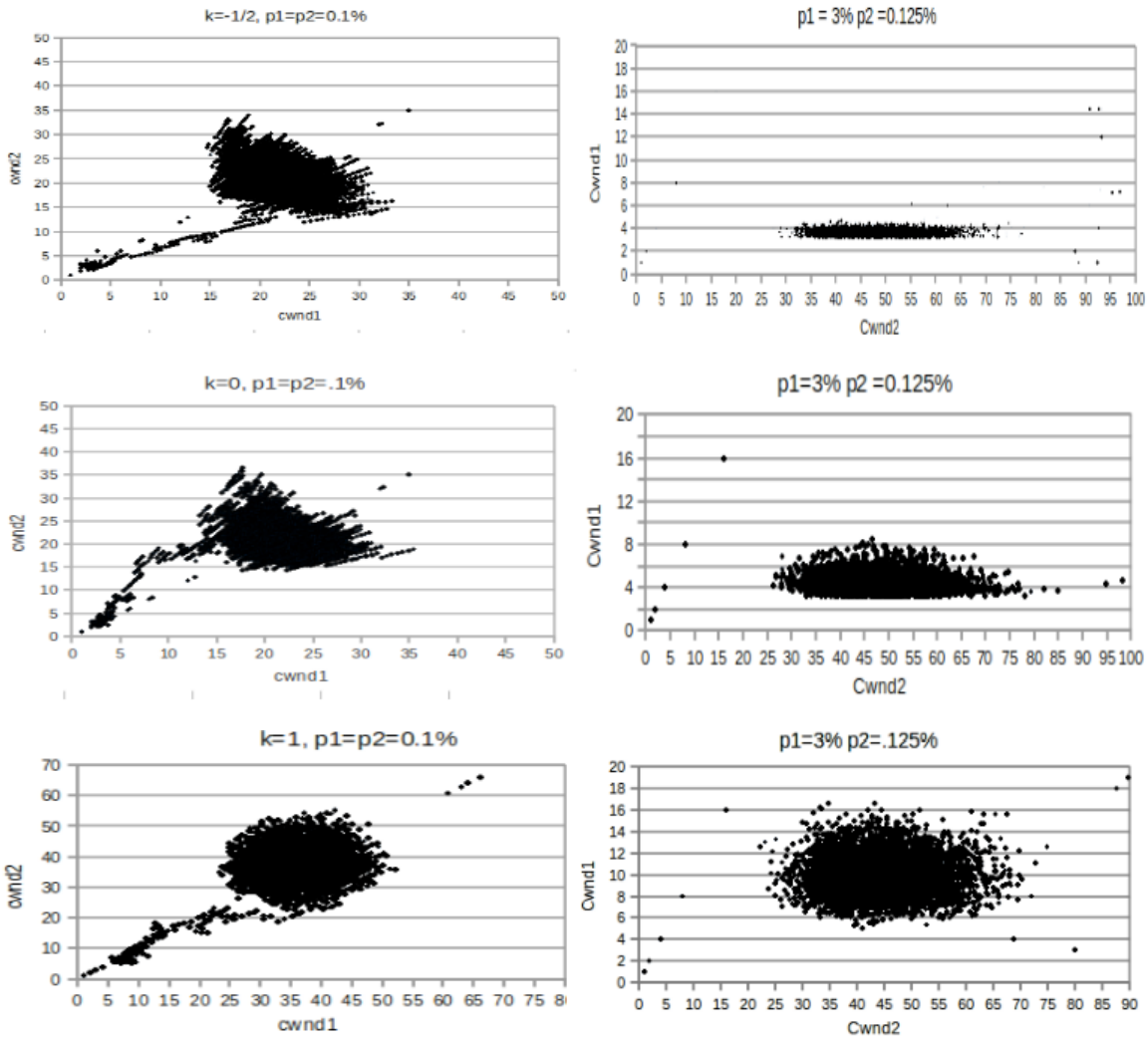

Fig. 4.: Flappiness of MPPERT for equal/unequal path loss with varying k

C. 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 traffic while keeping the link bandwidths constant. The decrease in the number of background traffic increases the available BW for MPPERT subflows. The configuration is same as in Fig. 3 with both bottleneck links having capacity of 40Mbps. We maintain 40 background flows on one path and vary the other 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).

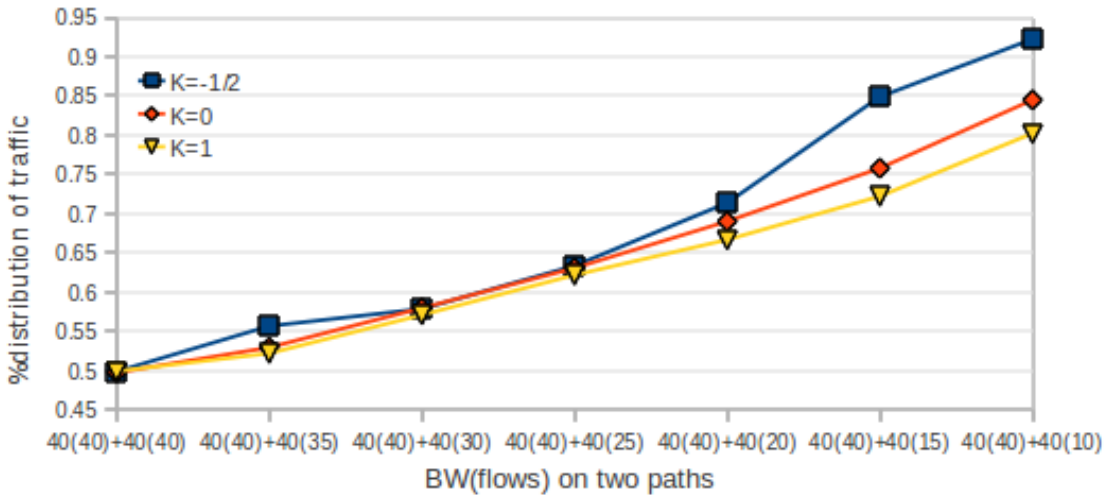

Fig. 5.: MPPERT throughput distribution with varying k

For instance, case 40(40)+40(20) suggests two 40Mbps bottleneck links with 40 and 20 background flows on path1 and path2 respectively. Fig. 5 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.

D. 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. 6 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 also maintains total throughput close to the best single-path throughput of the background PERT flows.

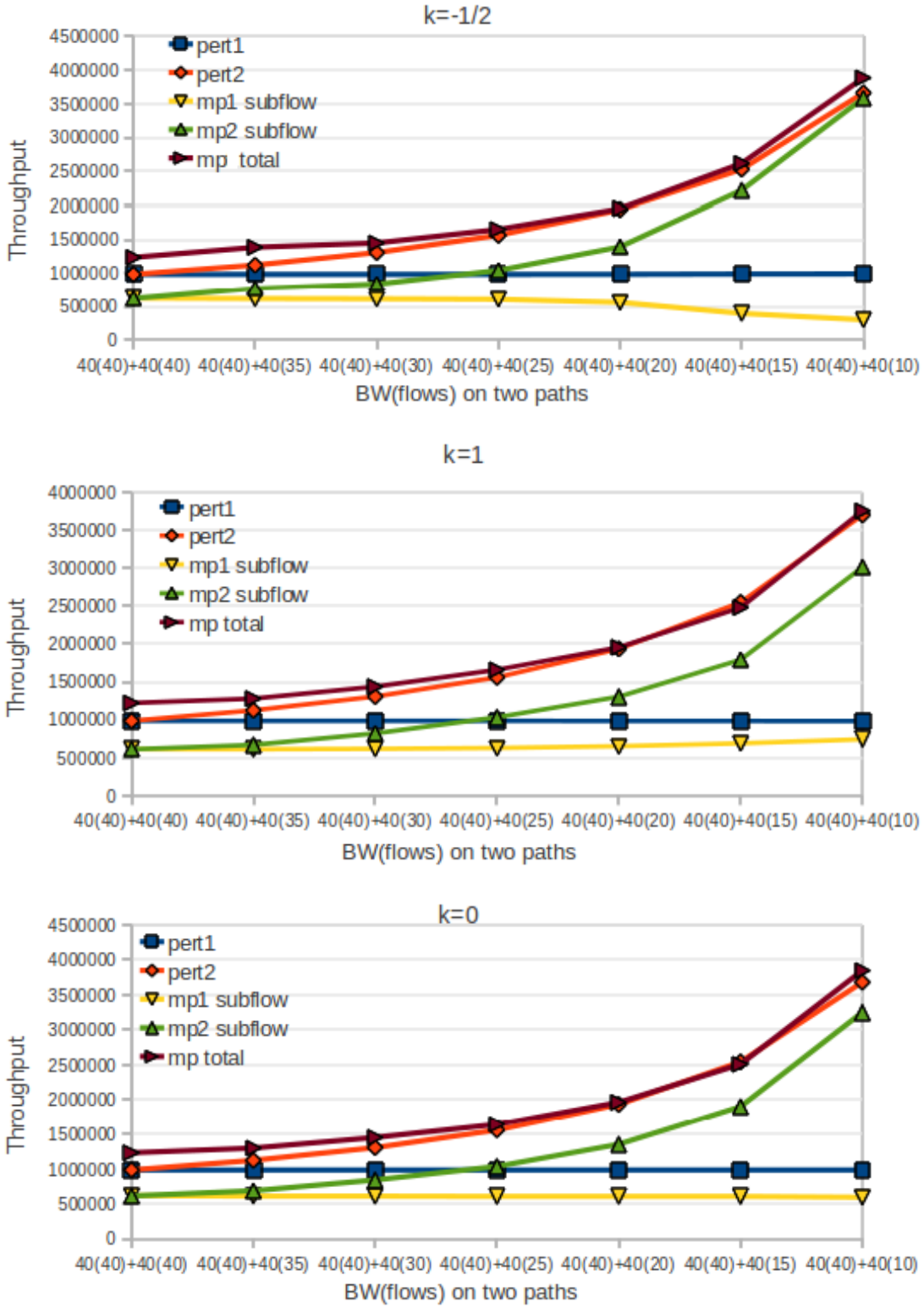

Fig. 6.: MPPERT throughput with varying number of background flows

E. Incremental Deployment of MPPERT

We further test the feasibility of incremental deployment of MPPERT in 50-50 PERT-TCP mix environment. Experiment setup is same as in the homogeneous case. In Fig. 7, 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.

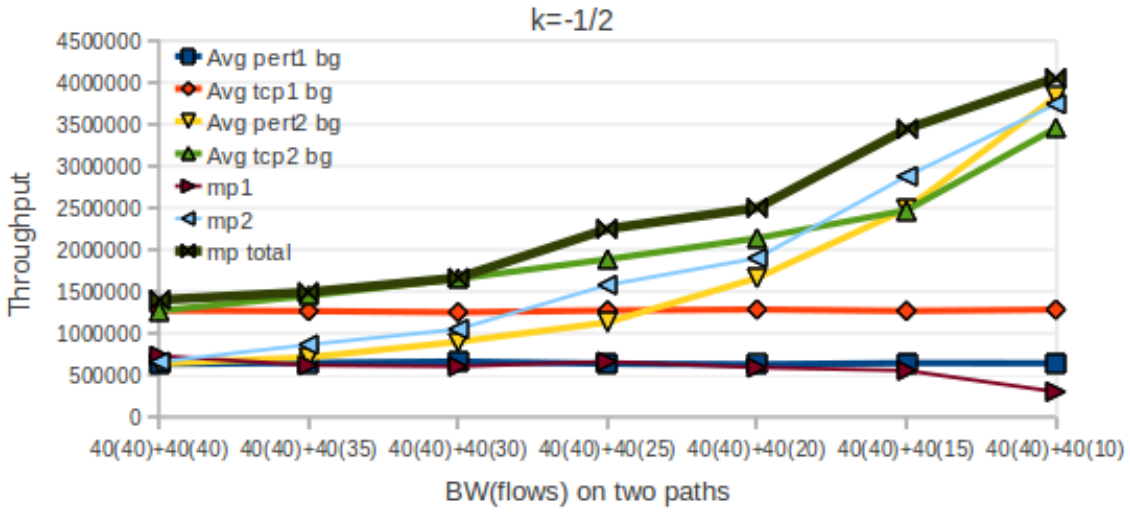

Fig. 7.: MPPERT (k=-1/2) throughput in heterogeneous 50-50mix environment with varying number of background flows

F. Resource Pooling

We compare MPPERT with independent multipath PERT (uncoupled flows), to analyze the performance of resource pooling. The configuration is shown in Fig. 8. The B.W. of links are 5Mbps each with bottleneck delay of 20ms. The RTT is set to 80ms and the bottleneck buffer to 1BDP. The throughput distribution for flows A,B,C are given in Table I for both independent multipath PERT and MPPERT(k=0). Independent multipath PERT competes fairly on all the paths. Thus, flow A, B and C get throughput close to 7.5Mbps, 5Mbps and 7.5Mbps. We observe that the overall throughput of MPPERT (k=0) is higher than the independent multipath PERT. Moreover, for MPPERT, the distribution is quite fair and close to the ideal Resource Pooling distribution of Throughput A = Throughput B = Throughput C = (20/3) Mbps.

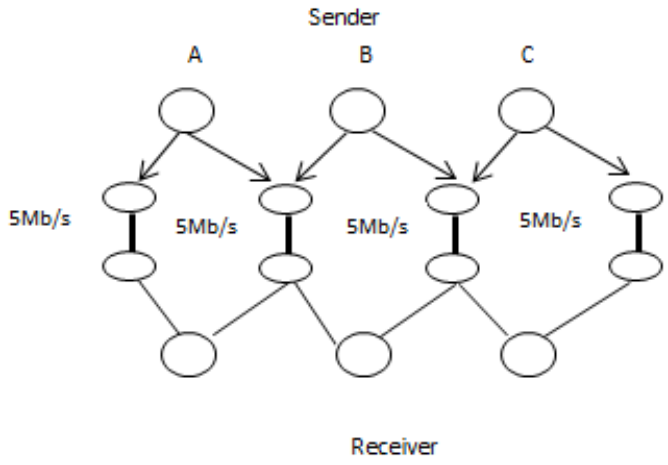

Fig. 8.: Resource pooling for cascaded configuration

We take another similar scenario as shown in Fig. 9. 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

Table I.: Resource pooling comparison of cascaded links for MPPERT k=0 and independent subflow multipath

	MPPERT Throughput	**Independent Multipath Throughput**
Flow A1	4723231	4699947
Flow A2	1239600	2206109
Flow A Total	5962832	6906057
Flow B1	3664999	2648762
Flow B2	2846471	1916552
Flow B Total	6511470	4565314
Flow C1	2015261	2699072
Flow C2	4302408	4462613
Flow C Total	6317669	7161686
Total	18791971	18633057

among the 3 flows (10Mbps each). We perform experiments with independent multipath PERT and MPPERT flows. We observe from Table II that MPPERT achieves higher system throughput with distribution closer to the ideal share of 10Mbps each.

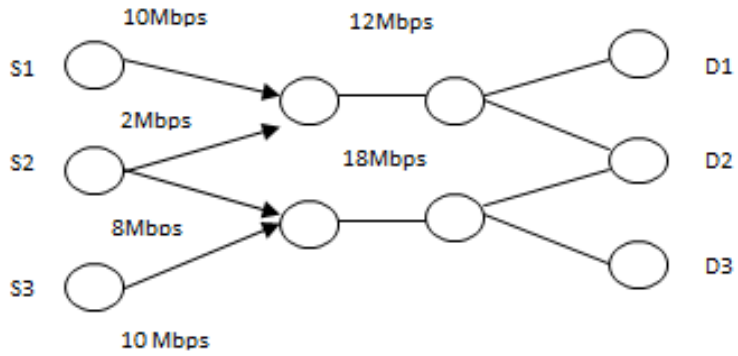

Fig. 9.: Resource pooling MPPERT (k=-1/2) with competing PERT flows

Table II.: Resource pooling comparison of MPPERT and independent subflow multipath with competing PERT flows

	Independent Multipath Throughput	**MPPERT (K=-1/2)**	**MPPERT (K=0)**
Throughput S1	6800789	9416288	8902788
Throughput S2 path 1	5082264	2555420	3051184
Throughput S2 path 2	8306036	7400033	6840525
Throughput S2 Total	12967781	9955453	9891709
Throughput S3	13388301	9650701	10324615
Total System	28889432	29022442	29119112

CHAPTER V

MPPERT PATH SELECTION

Multipath goal is not to harm or take unnecessary bandwidth advantage over normal single-path TCP. It tries to get as much throughput as it would get with single-path TCP 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. When MPPERT flow has more paths, the probability of getting a higher BW path increases. This would increase MPPERT's chances of getting better throughput without compromising the BW share of the competing flows. 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.

A. Detection of Bottleneck Link

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. Although the idea is quite apparent, we may face issues realizing it. Some of the issues are as follows :

- Rate at which a subflow receives queue information may not be sufficient.
- Queue sizes could get subjected to some random changes.
- Determining time stamp for the queue signal.
- 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.

B. Experiment Setup

The setup uses 3-MPPERT (MPPERT with 3 subflows) with PERT as the background traffic for the homogeneous case as shown in Fig. 10. 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. In the 50-50 PERT-TCP mix case, we use 50% of total background flows as PERT and the other 50% as TCP Reno. Each link carries 20-20 PERT-Reno background flows giving available bandwidth of approximately 1Mbps. The experiment 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. A few highlights of the experiment are as follows :

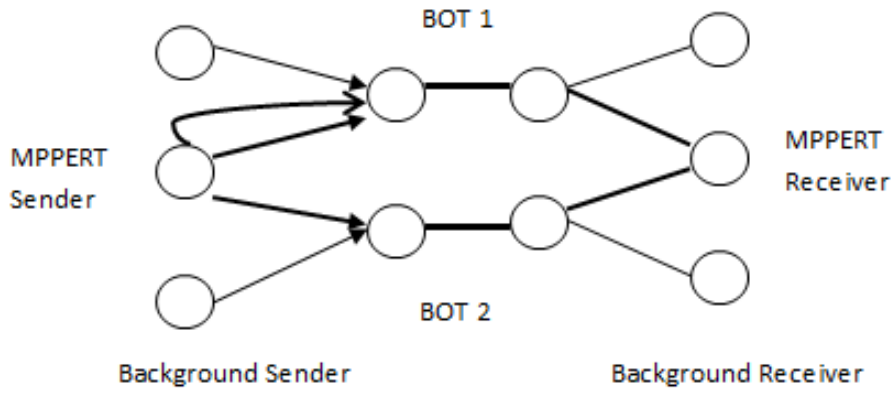

Fig. 10.: Experiment setup for three multipath flows

- Illustrates the correlation comparison for shared/disjoint paths.
- Provides a comparison for coupled subflows sharing a bottleneck link and disjoint subflows under similar condition. (subflow 2 uses disjoint path (bot2) but experiences similar background traffic as in bot1 having both subflows 1 and 3 sharing the link.)
- Measures the time required to correctly estimate the shared links.

C. Homogeneous Environment

We analyze the correlation for both shared and disjoint subflows. The correlation in Fig. 11 and 12 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.

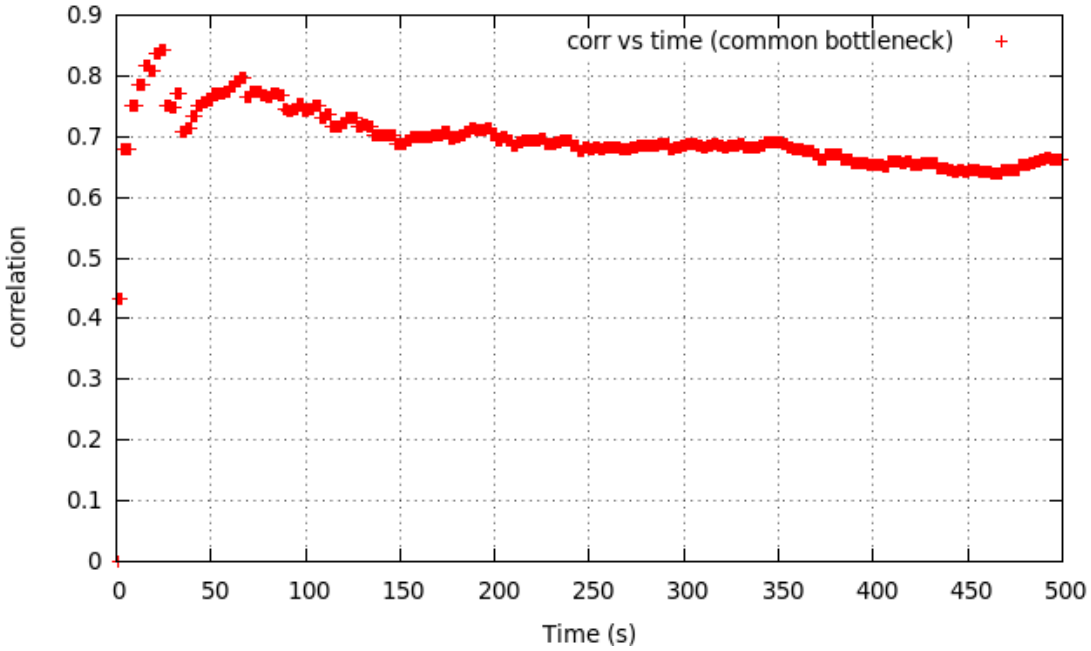

Fig. 11.: Correlation of MPPERT subflows sharing a common bottleneck

The result of congestion window variation for the three subflows as per experiment are shown in Fig. 13. 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. It took close to 10 secs for the correlation metric to decide whether the subflows shared a bottleneck link or are disjoint. 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).

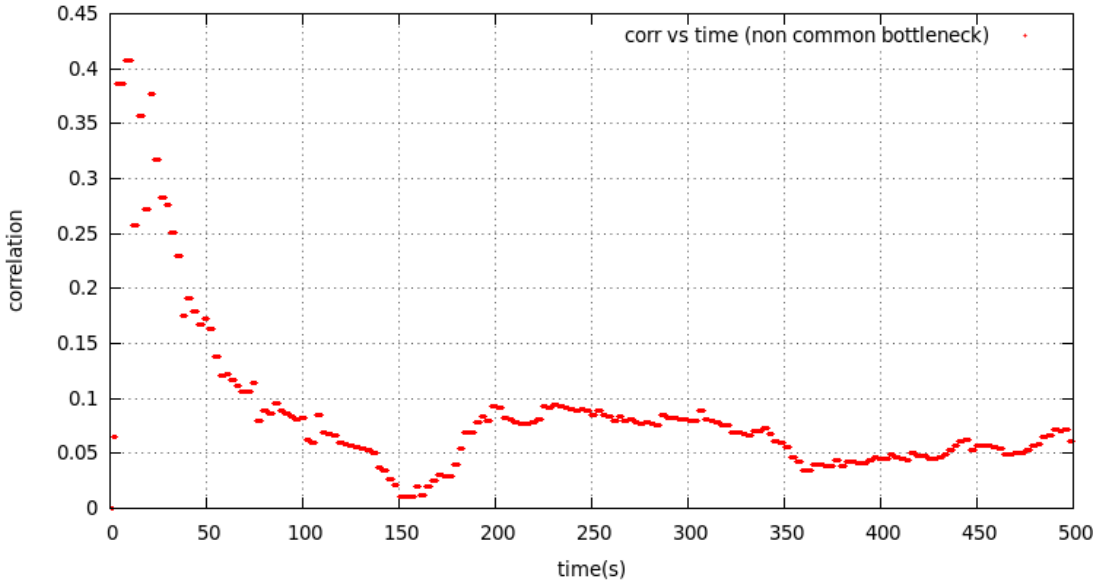

Fig. 12.: Correlation of MPPERT subflows with disjoint paths

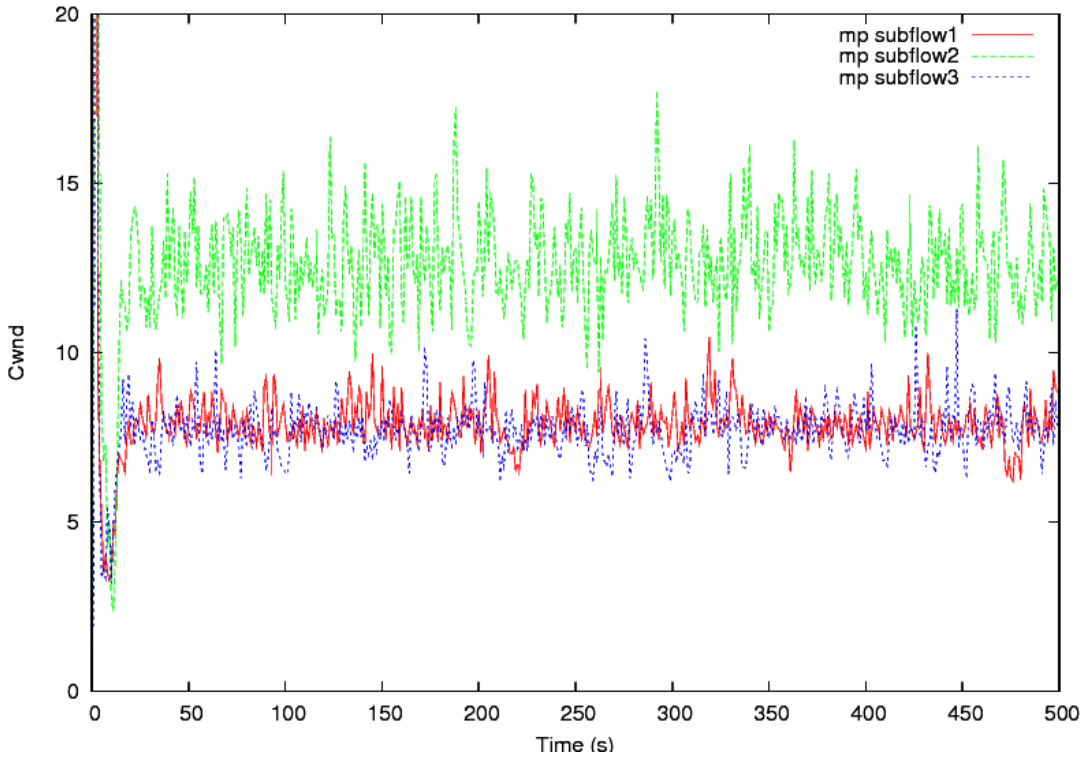

Fig. 13.: MPPERT disjoint/shared path subflow congestion window variation

D. Heterogeneous Environment

In order for PERT to compete with TCP, it has to operate in compete mode. It changes its aggressiveness factor α proportional to the ratio of early response probability and drop probability. In order to make smooth changes, one needs to increase alpha gradually. Increasing alpha steeply, may result in poor correlation as the queue sizes undergo drastic changes. 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. 14,15. Suitable choice of thresholding is important for the correct detection of shared and disjoint links.

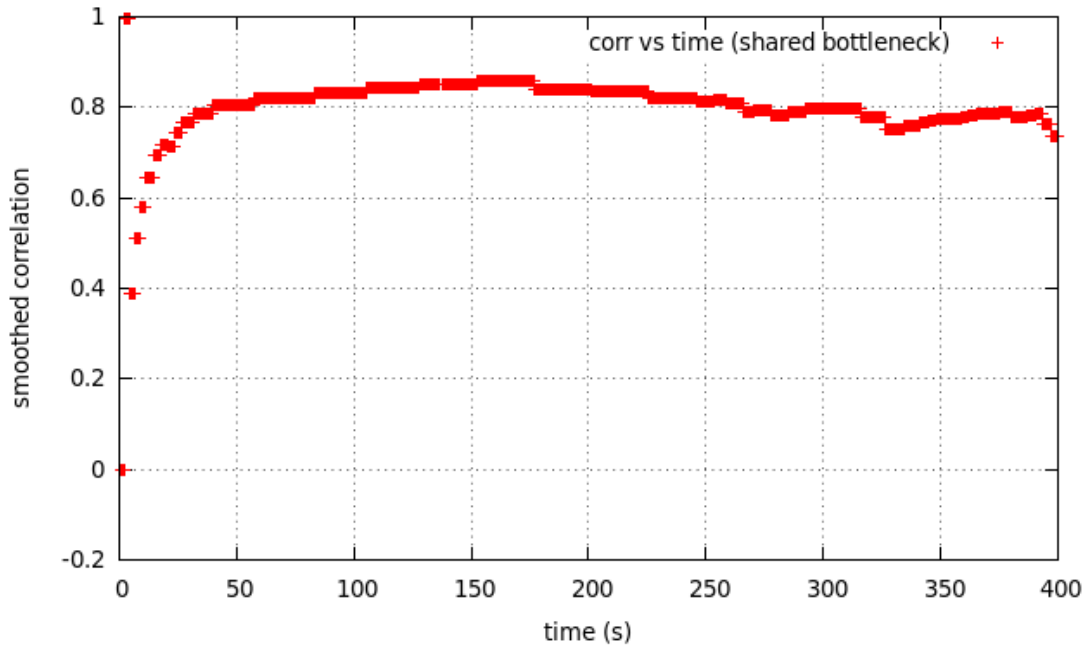

Fig. 14.: Correlation for MPPERT shared bottleneck subflow in heterogeneous 50-50 mix environment

Correlation of subflows sharing a bottleneck link with high probability of loss is lower than the case when probability of loss is much lower. This may sometimes delude in

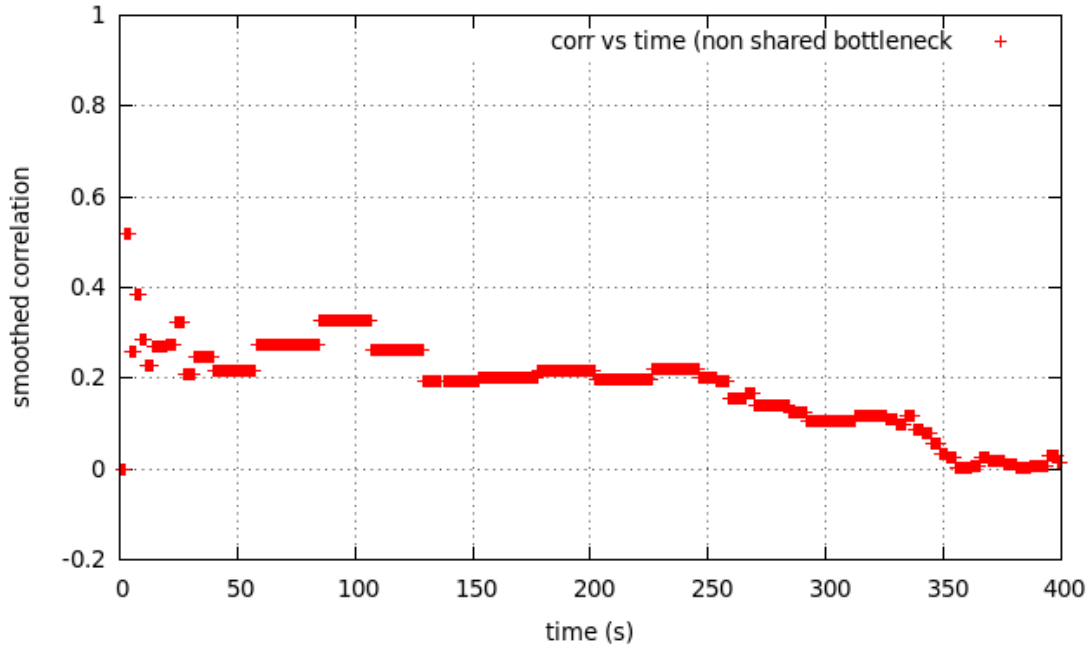

Fig. 15.: Correlation for MPPERT disjoint bottleneck subflow in heterogeneous 50-50 mix environment

concluding shared bottleneck subflows as being disjoint. This may lead to false -ve detection of the shared bottleneck. In order to lower 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 shoots up and detection of shared bottlenecks become easier.

E. Suppressing Shared Subflow

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 bottle-

neck 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.

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, creates 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 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 III and IV show the performance of coupling with correlation based supression. We observe that the bottleneck detection technique successfully classifies the shared sub-

flows 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 $\approx$ .12Mbps) respectively. Table V and VI show 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 $\approx$ .12Mbps) respectively. This provides throughput aggregation over the available disjoint paths.

Table III.: Coupled with correlation based suppression

Starting 6-MPPERT	**6 disjoint 20(10) path**
Avg Background Throughput path1	1965398
Avg Background Throughput path2	1989003
Avg Background Throughput path3	1966071
Avg Background Throughput path4	1953238
Avg Background Throughput path5	1977550
Avg Background Throughput path6	1941901
Multipath subflow 1	503688
Multipath subflow 2	265142
Multipath subflow 3	497288
Multipath subflow 4	626451
Multipath subflow 5	383667
Multipath subflow 6	733090
Multipath total	3009329
Starting 6-MPPERT	**4 disjoint 20(10) path**
Avg Background Throughput path1	1935559
Avg Background Throughput path2	1939981
Avg Background Throughput path3	1930198
Avg Background Throughput path4	1952365
Multipath subflow 1	120768
Multipath subflow 2	120768
Multipath subflow 3	555636
Multipath subflow 4	754368
Multipath subflow 5	844301
Multipath subflow 6	631854
Multipath total	3027615

Table IV.: Coupled with correlation based suppression with 2 disjoint path

Starting 6-MPPERT	**2 disjoint 20(10) path**
Avg Background Throughput path1	1884309
Avg Background Throughput path2	1927330
Multipath subflow 1	834576
Multipath subflow 2	120187
Multipath subflow 3	120270
Multipath subflow 4	120270
Multipath subflow 5	120270
Multipath subflow 6	882036
Multipath total	2197610

Table V.: Coupled/Decoupled switch with correlation based suppression

Starting 6-MPPERT	**6 disjoint 20(10) path**
Avg Background Throughput path1	1834381
Avg Background Throughput path2	1831131
Avg Background Throughput path3	1839459
Avg Background Throughput path4	1835652
Avg Background Throughput path5	1838786
Avg Background Throughput path6	1835869
Multipath subflow 1	1806212
Multipath subflow 2	1839958
Multipath subflow 3	1764238
Multipath subflow 4	1801059
Multipath subflow 5	1770140
Multipath subflow 6	1797568
Multipath total	10779179
Starting 6-MPPERT	**4 disjoint 20(10) path**
Avg Background Throughput path1	1833616
Avg Background Throughput path2	1827141
Avg Background Throughput path3	1838188
Avg Background Throughput path4	1835736
Multipath subflow 1	1579553
Multipath subflow 2	120103
Multipath subflow 3	120103
Multipath subflow 4	1884758
Multipath subflow 5	1774379
Multipath subflow 6	1800976
Multipath total	7279875

Table VI.: Coupled/Decoupled switch with correlation based suppression 2 disjoint path

Starting 6-MPPERT	**2 disjoint 20(10) path**
Avg Background Throughput path1	1842909
Avg Background Throughput path2	1834281
Multipath subflow 1	120103
Multipath subflow 2	120103
Multipath subflow 3	120103
Multipath subflow 4	1249412
Multipath subflow 5	120103
Multipath subflow 6	1811698
Multipath total	3541527

CHAPTER VI

PERFORMANCE COMPARISON MPPERT AND MPTCP

Multipath TCP provides improved reliability, load balancing and mobility by providing multiple concurrent connections between end hosts. It promotes pooling of the network resources into a single logical resource to promote better utilization. This section provides performance comparison and analysis of our new multipath algorithm, MPPERT, with MPTCP [21]. The key criteria for comparison are given below.

1. Queue Management
2. Packet Loss
3. Throughput and subflow traffic Distribution
4. Resource Pooling

A. Queue Management

Drop Tail/Tail Drop Queue management algorithm allows router to buffer maximum possible packets before any packet gets dropped. Congestion occurs when buffer remains continuously full. Tail drop algorithm may not distribute buffer uniformly among the competing flows but maximizes the available resource at the router. RED (Random Early Detection) is an AQM (Active Queue Management) algorithm which monitors queue sizes and drops packets based on statistical probability. RED algorithm provides fairer distribution of queue share to the competing flows. However, by dropping packets early, it may be unable to fully utilize the buffer resources available at the router.

MPPERT is a delay based multipath algorithm that performs AQM at the end hosts.

It proactively responds to an increase in queue size to avoid congestion. This relaxes the need for AQM capability at the routers and provides greater control to the end hosts. MPPERT performs decently with both RED and Tail Drop queue management in all-PERT environment. In a heterogeneous environment, MPPERT pushes most of its packets early in the queue and avoids sending packets when the queue gets full. This helps MPPERT in maintaining the desired throughput and reducing the number of packets lost. We employ experiment setup similar to Fig. 3 with RTT 40ms, buffer size 1BDP and each bottleneck link of 10Mbps capacity carrying 10 background traffic each (case 10(10)+10(10)).

Table VII shows performance comparison of MPTCP and MPPERT flow 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. It also offers higher total system throughput in comparison with MPTCP. Both MPTCP and MPPERT(k=0) flows offer similar throughput distribution for multipath subflows.

MPTCP performs much better with RED queue management as per Table VIII. In RED, routers perform AQM and randomly drop packets to lower average queue size. For RED queue management, PERT and MPPERT flows may mistakenly get pushed in compete mode. This unnecessary shift from safe to compete mode may lower the total system throughput in a homogeneous environment.

Table VII.: Performance comparison MPPERT, MPTCP for tail drop queue management

10(10)+10(10)	**MPTCP**	**MPPERT**
Avg Background Throughput path1	982949	949191
Avg Background Throughput path2	981918	947665
Multipath subflow 1	158832	523066
Multipath subflow 2	167892	538326
Multipath total	326724	1061392
Avg Queue Size path 1	32.3946	13.209
Avg Queue Size path 2	32.4967	12.7726
System Throughput	19975394	20029952

Table VIII.: Performance comparison MPTCP, MPPERT for RED active queue management

10(10)+10(10)	**MPTCP**	**MPPERT**
Avg Background Throughput path1	931540	859483
Avg Background Throughput path2	928359	864797
Multipath subflow 1	498711	625143
Multipath subflow 2	528379	638032
Multipath total	1027090	1263176
Avg Queue Size path 1	5.08194	4.97659
Avg Queue Size path 2	4.95652	4.84448
System Throughput	19626080	18505976

B. Packet Loss

MPPERT is a delay based protocol which pro-actively 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 hand, continuously introduces large number of packet losses and retransmissions which decreases total throughput of the system. Fig. 16 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 loosing packets at the bottleneck link.

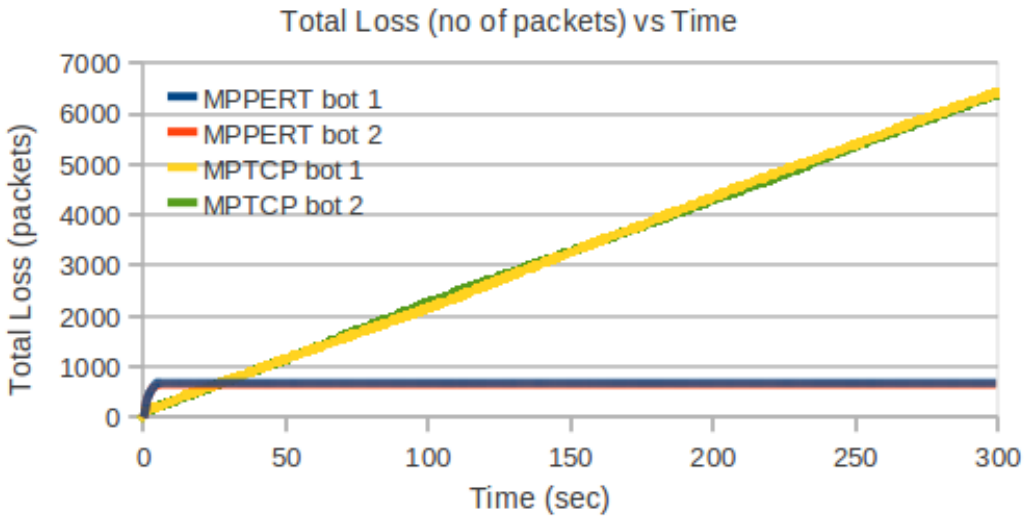

Fig. 16.: MPPERT and MPTCP pathloss comparison at bottleneck 1 and 2

C. Throughput and Subflow Traffic Distribution

We compare total throughput achieved by MPPERT and MPTCP for the cases 10(10)+10(10), 10(10)+10(5) and 10(10)+10(2) as shown in Table IX. 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 IX 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. 17 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, which is much closer to the throughput of the best-single path background flow. Fig. 17(b) suggests that both MPTCP and MPPERT flows can shift traffic from more congested (path 1) to the less congested (path 2) path. The simulation results suggest that a MPPERT flow, though slightly more aggressive, is able to compete and successfully shift traffic off the congested path in a heterogeneous environment.

We observe, from results in Fig. 18, that the total system resource utilization is

lower when all the flows employ TCP. 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 start with 15(1)+20(1) (15Mbps and 20Mbps link with 1 background flow each) and 10(1)+20(1) (10Mbps and 20Mbps link with 1 background flow each) configurations. We then scale the link bandwidth and the background traffic proportionally, maintaining the same available BW, to monitor change in system resource utilization. Fig. 18 shows that the MPPERT system maintains much higher system througput, 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.

D. Resource Pooling

To analyze the resource pooling performance of MPPERT and MPTCP, we compare both algorithms for scenario shown in Fig. 19. Here S2 is a multipath capable flow. Bottleneck links 1,2 are 12Mbps and 18Mbps respectively. Here RTT is 40ms. Ideal resource pooling should distribute the total 30Mbps capacity equally (10Mbps each) among the 3 competing flows. Table X shows that in homogeneous environment, the traffic distribution of flows S1,S2,S3, for MPPERT (k=-0.9), is 99.1%, 95.4% and 95.4% of the ideal 10Mbps each distribution in comparison to 67.4%, 88.75% and 85.04% for MPTCP. Total throughput of the system is about 20% higher for MPPERT (k=-0.9) than with MPTCP. MPPERT (k=0) distributes traffic inversely proportional to path loss, similar to MPTCP. However, it performs slightly better on account of PERT's properties of lower packet losses and smaller queue lengths.

Table IX.: Total throughput and subflow distribution comparison

10(10)+10(10)	**MPTCP**	**MPPERT**
Avg Background Throughput path1	931540	949191
Avg Background Throughput path2	928359	947665
Multipath subflow 1	498711	523066
Multipath subflow 2	528379	538326
Multipath total	1027090	1061392
Multipath throughput distribution	0.485	0.492
System Throughput	19626080	20029952
10(10)+10(5)	**MPTCP**	**MPPERT**
Avg Background Throughput path1	936039	952168
Avg Background Throughput path2	1714053	1762334
Multipath subflow 1	457491	493171
Multipath subflow 2	1116690	1170704
Multipath total	1574181	1663875
Multipath throughput distribution	0.709	0.703
System Throughput	19504836	19997225
10(10)+10(2)	**MPTCP**	**MPPERT**
Avg Background Throughput path1	950425	967880
Avg Background Throughput path2	3482633	3317438
Multipath subflow 1	310018	336126
Multipath subflow 2	2500000	3242388
Multipath total	2810018	3578514
Multipath throughput distribution	0.889	0.906
System Throughput	19279534	19892190

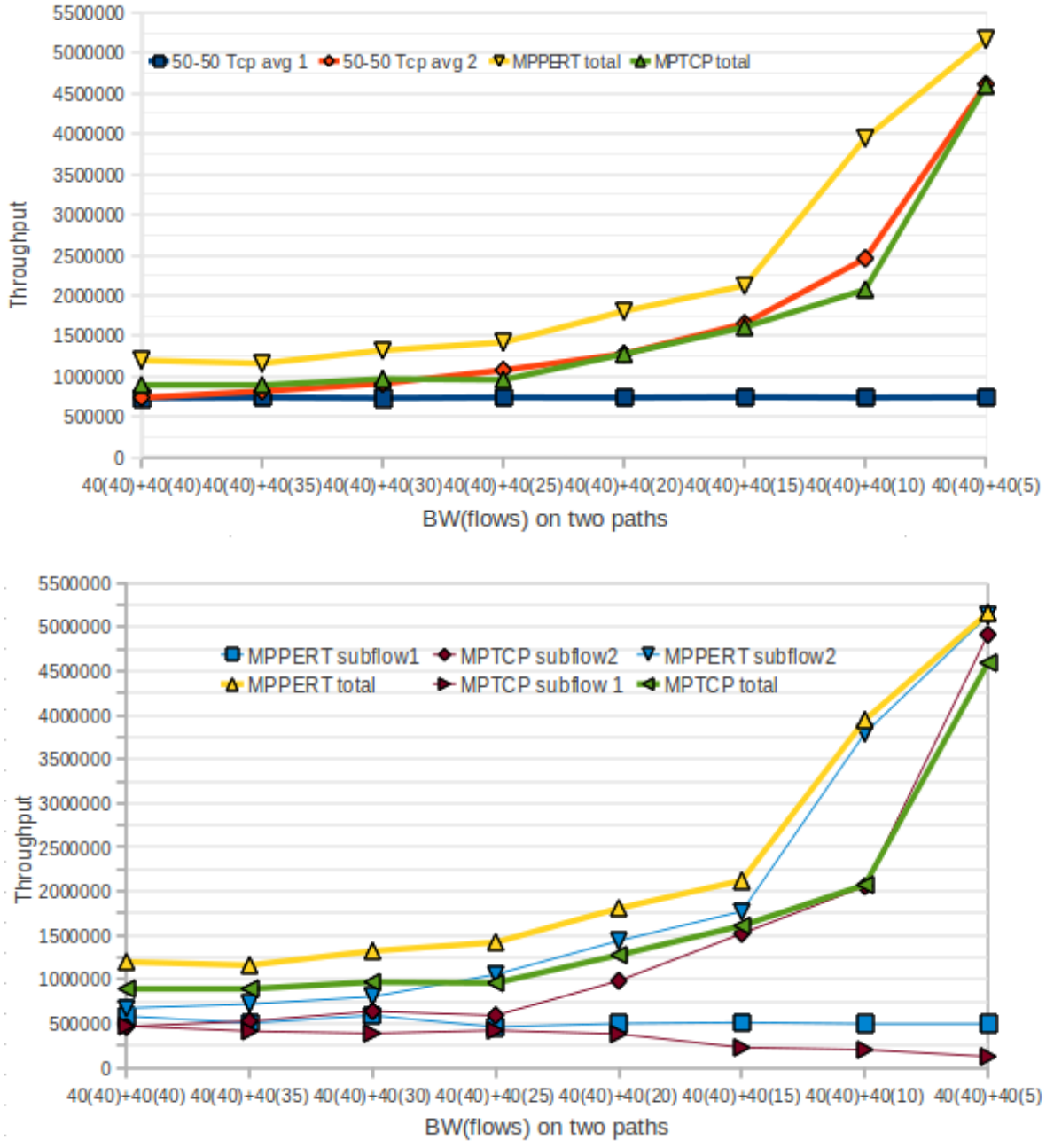

Fig. 17.: Throughput comparison for 2-MPPERT and 2-MPTCP competing in heterogeneous environment

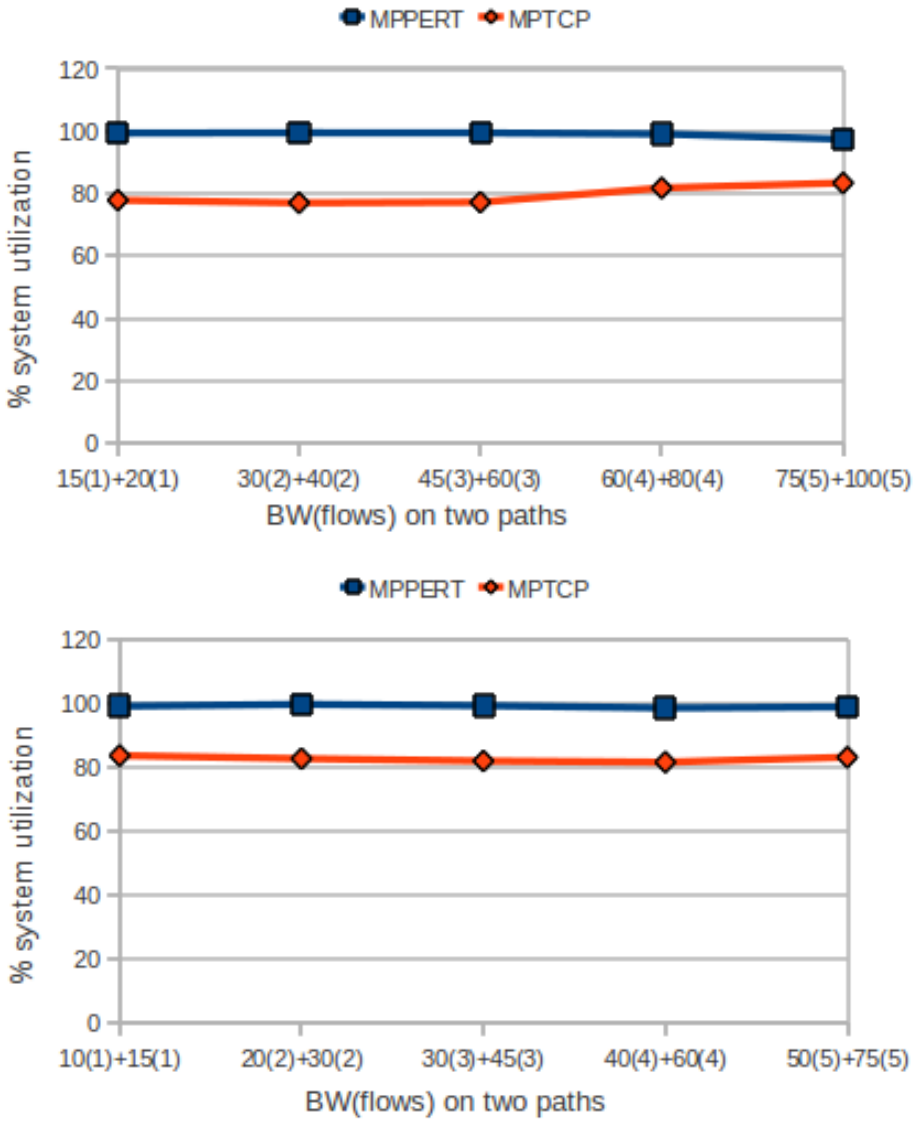

Fig. 18.: System resource utilization comparison for 2-MPPERT and 2-MPTCP

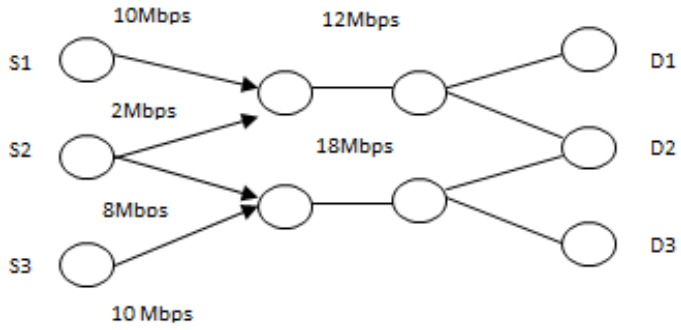

Fig. 19.: Experiment setup MPPERT, MPTCP resource pooling comparison

Table X.: Resource pooling comparison MPPERT and MPTCP for figure 19

	MPTCP	**MPPERT k=0**	**MPPERT k=-0.9**
Throughput S1	6739865	8451906	9910109
Throughput S2 path 1	3044750	3481350	2055812
Throughput S2 path 2	5830170	6861631	7489080
Throughput S2 Total	8874920	10342981	9544893
Throughput S3	8503904	10241246	9541485
System Throughput	24118689	29036133	28996487

CHAPTER VII

RESPONSIVENESS OF MPPERT

In order to comprehensibly assess the performance and traffic shift of MPPERT algorithms, we need to monitor the performance in various scenarios such as (a) rapid increase/decrease of available BW, (b) high and low link bandwidths, (c) with small duration flows. We categorize these scenarios into four major test cases, for 2-MPPERT flow in a configuration similar to Fig. 3.

1. Available BW of one subflow is 4 times the other.
2. Available BW of one subflow is 2 times the other. Decrease available BW on path1 (without changing path2) by increasing the background flows, to make available BW on both the paths equal at T=300s.
3. Available BW of one subflow is 2 times the other. Increase available BW of path2 (without changing path 1) to make both of them equal at T=300.
4. Available BW of path1 equals path2. Increase available BW on one path, at T=300s, while keeping other the same.

Case 1

Configuration : Bottleneck link capacity 40Mbps, path1 :10 PERT background flow, path2 : 40 PERT background flows.

For the given configuration, one subflow has 4 times the available BW of the other. We observe from Fig. 20 that the throughput for both k=-1/2, 1 is higher for the less congested path1 but the traffic shift is more in case of k=-1/2. This happens because changing k values from k=1 to k=-1, increases the rate with which lower

path loss (higher available BW) window grows. Thus, as we decrement k, window increase factor becomes higher and even with the same number of acknowledgements, window grows faster to achieve higher throughput.

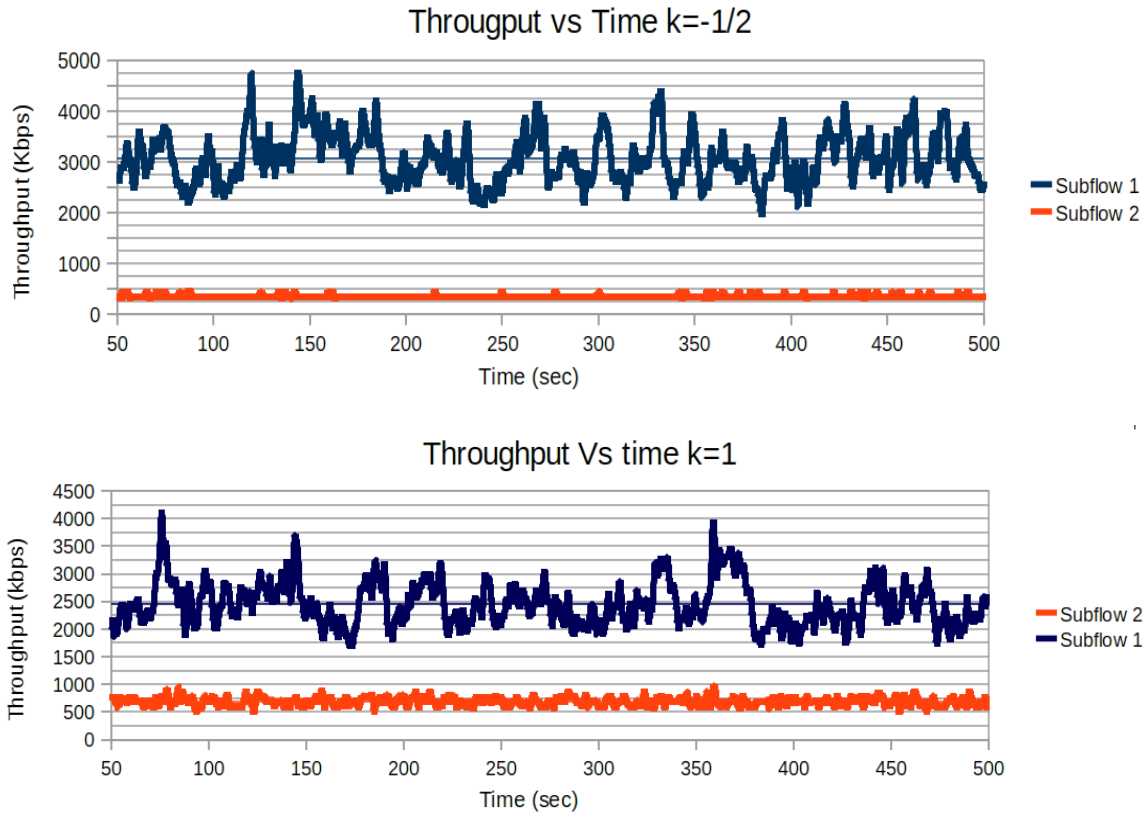

Fig. 20.: MPPERT throughput vs time for long duration subflow

Cases 2

Configuration : Bottleneck link capacity 40Mbps, path1 :15 and 30 PERT background flows at T=0 and T=300 respectively, path2 :30 PERT background flows.

As per the configuration, we start subflow1 with twice the available BW of subflow 2. We increase the background traffic on path 1, at T=300s, to make available BW of both the subflows equal. We try to analyze how quickly the subflow settles down to the ideal distribution. In the present case, it takes less than 10 sec to settle down to the steady state value. The simulations in Fig. 21 suggest that as we vary k from 1 to -1, the amount of traffic shift increases towards the higher available BW path. In effect, it also becomes more sensitive to small fluctuations in the traffic.

Cases 3

Configuration : Bottleneck link capacity 40Mbps, path1 :15 PERT background flow, path2 :30 and 15 PERT background flows at T=0 and T=300 respectively.

Earlier scenario increased traffic on the less congested path, to equalize available BW of both the flows. On the other hand, the following test decreases the congestion on the more congested path (path2), to equalize available BW of both the flows. This would require the path having higher throughput share, subflow 1, to give up throughput and push more traffic on subflow 2. This is apparent in the result shown in Fig. 22.

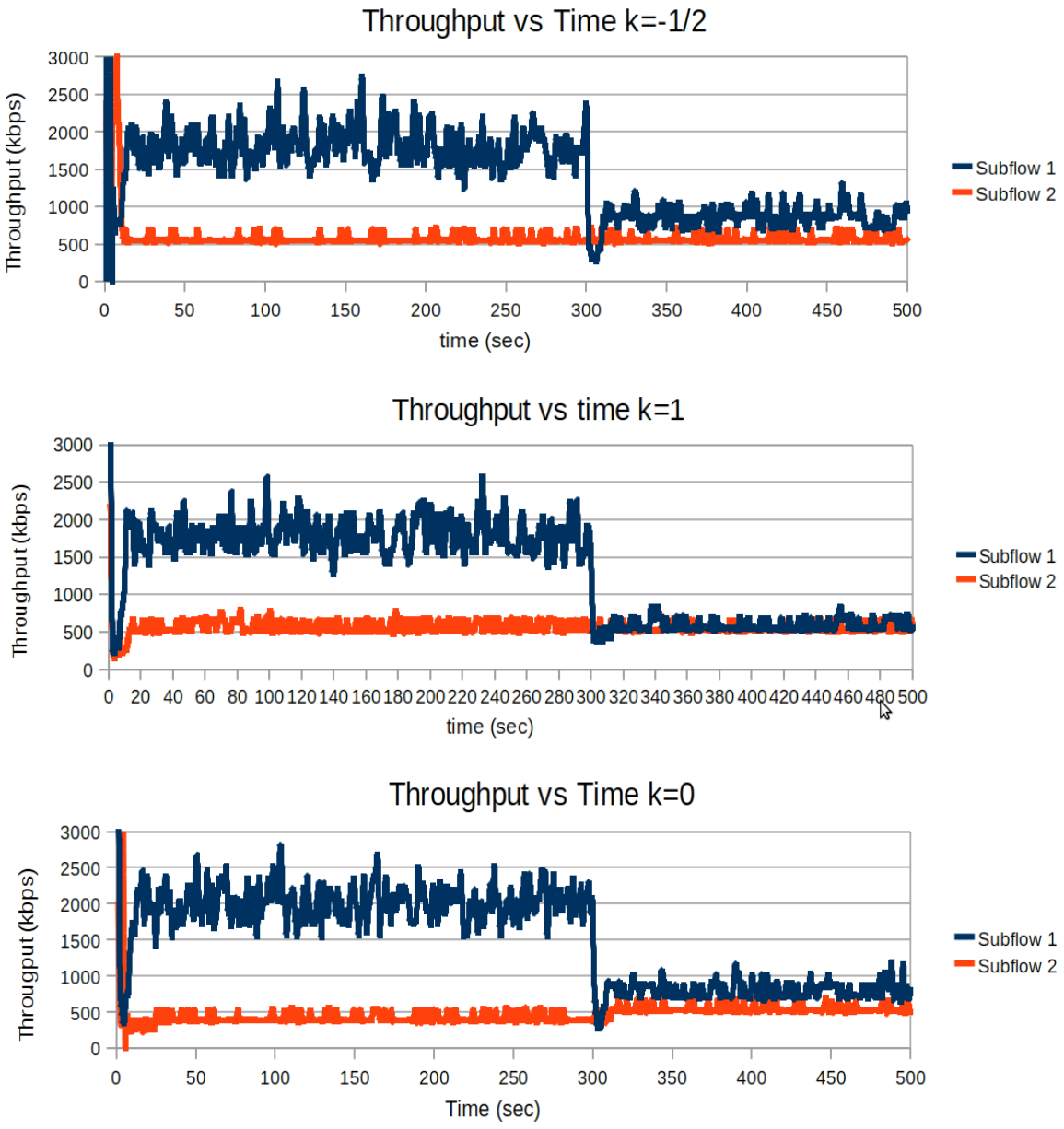

Fig. 21.: MPPERT responsiveness to increase in background traffic for high BW path

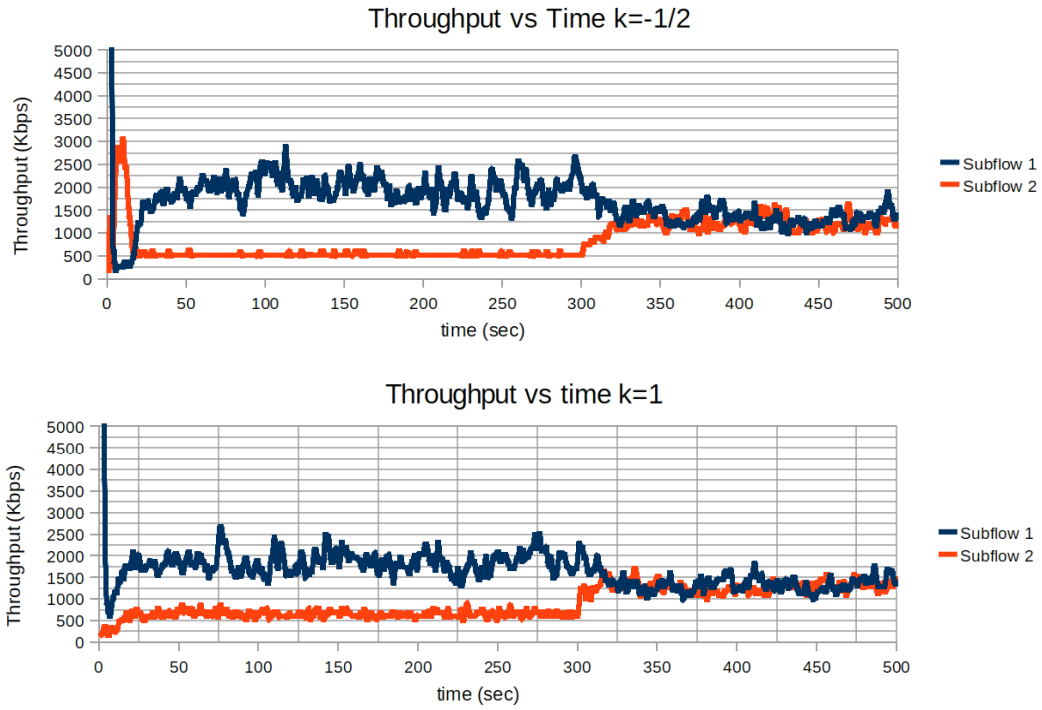

Fig. 22.: MPPERT responsiveness to decrease in background traffic for low available BW path

Cases 4

Configuration : Bottleneck link capacity 40Mbps, path1 :30 PERT background flow, path2 :30 and 15 PERT background flows at T=0 and T=300 respectively.

Here, we analyze how fast a subflow adjusts to the increase in available BW. Fig. 23 suggests that for k=-1/2, the steady state traffic shift is much larger compared to k=1. MPPERT (k=-1/2) also increases its window size faster (as per two flow analysis) compared to MPPERT(k=1), making it more responsive to congestion.

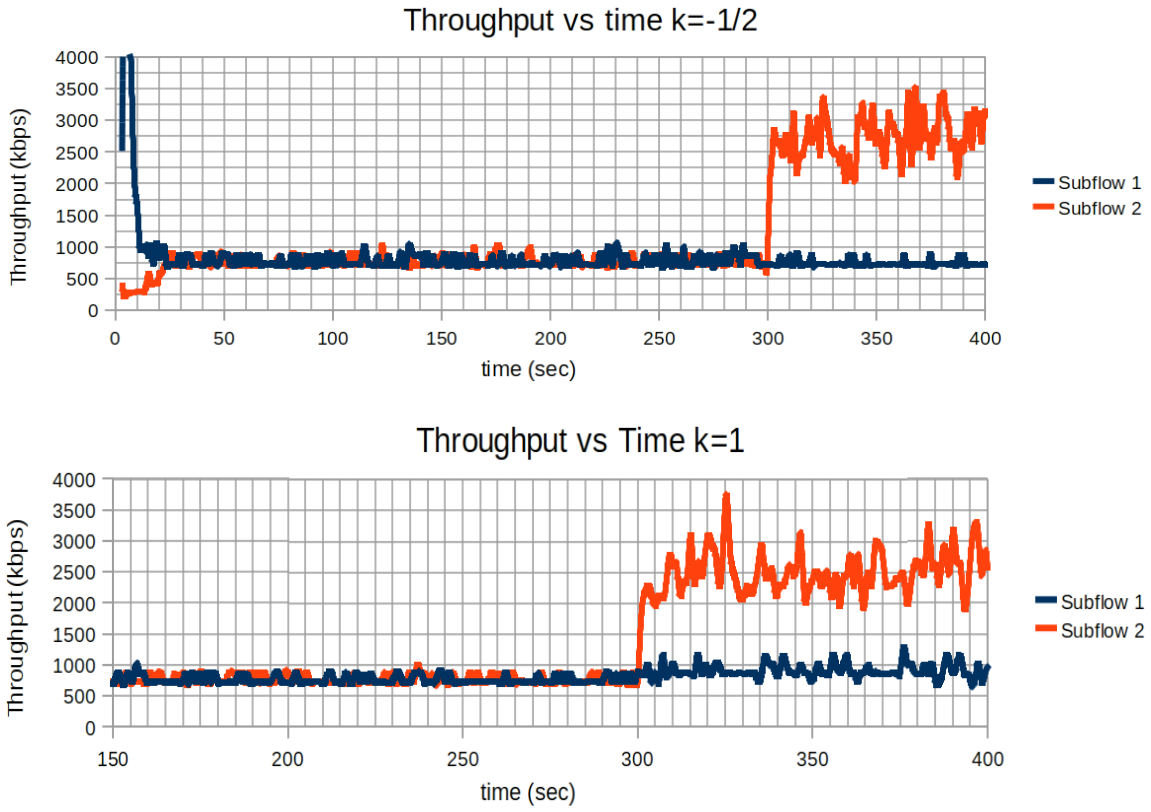

Fig. 23.: MPPERT responsiveness to decrease in background traffic

Selection of parameter 'k' affects the responsiveness of MPPERT. As we vary parameter 'k' from 1 to -1, the sensitivity to congestion increases. This improves both rate

and amount of traffic shift achieved by a MPPERT flow. However, performing floating point operations in the kernel is generally avoided. This would suggest choosing k=0 would present a suitable trade off between the computational requirement and amount of traffic shift achieved by a MPPERT flow.

CHAPTER VIII

CONCLUSION AND FUTURE WORK

In this thesis, we have proposed a new multipath algorithm 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 flappiness, resource pooling, system throughput and packet loss etc. Simulations suggest that MPPERT provides performance boost from single-path PERT. It moreover provides higher connection reliability by using multiple subflow connections between end-hosts.

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.

Increasing the number of subflows in MPPERT would increase the computational requirements. The thesis 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.

In the future, it is desired to deploy MPPERT in various datacenter topologies to

monitor performance enhancements over single path TCP. It will also be helpful in predicting suitable number of subflows for large scale networks. As internet has evolved over the decades, it is also desired to study interactions between middleboxes and end hosts in terms of multipath connection negotiations, additions/deletions of new subflows etc. which are pivotal in commercial deployment of any algorithm.

REFERENCES

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

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

[7] F.P. Kelly, A.K. Maulloo, D.K.H. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237-252, 1998.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

[16] P. Key, L. Massoulie, D. Towsley, "Combining multipath routing and congestion control for robustness," in *Proc. IEEE Conference on Information Sciences and Systems*, 2006, pp. 345-350.

[17] Laws, C.N, "Resource pooling in queueing networks with dynamic routing," *Advances in Applied Probability*, vol. 24, no. 3, pp. 699-726, 1992.

[18] P. Key, L. Massoulie, D. Towsley, "Path selection and multipath congestion control," in *Proc. 26th IEEE International Conference on Computer Communications*, 2007, pp. 143-151.

[19] 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.

[20] 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.

[21] 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.

VITA

Ankit Singh received his B.Tech. degree in electronics and communication engineering from Indian Institute of Technology, Guwahati in 2008 and his M.S. in Computer Engineering from Texas A&M, College Station, Texas, in 2012. His research interests are in the areas of network congestion protocols and wireless ad hoc networks. He has been a recipient of a merit scholarship from the Reserve Bank of India during 2004-2008. Prior to arriving at Texas A&M, he worked in the Networx group, CenturyLink, India 2008-2010. He can be contacted at the following address: Department of Electrical and Computer Engineering, Texas A&M, 52B WERC, College Station, TX 77843-3128.

The typist for this thesis was Ankit Singh.