F-TCP: a Delay-based Protocol with Fair Co-existence

Julianne Sansa-Otim
Kapteyn Institute
University of Groningen,
Postbus 800, 9700 AV Groningen
The Netherlands.
E-mail: sansa@astro.rug.nl

Idris A. Rai
Faculty of Computing and IT,
Makerere University,
P.O. Box 7062, Kampala
Uganda.
E-mail: rai@cit.mak.ac.ug

J. M. van der Hulst
Kapteyn Institute
University of Groningen,
Postbus 800, 9700 AV Groningen
The Netherlands.
E-mail: vdhulst@astro.rug.nl

***Abstract*—Various studies have shown that regular TCP is inefficient in high-speed networks. This paper proposes F-TCP, a delay-based TCP variant, which is able to operate efficiently in high-speed networks. The slowstart phase of F-TCP continues until a threshold determined from probing the available bandwidth. When competing with loss-based flows F-TCP reduces it's window to a value derived from the available bandwidth. More specifically, an adaptive bandwidth share estimation with a delay-sensitive instability measure is employed to guide window backoff when congestion is detected by F-TCP. Using ns-2 simulations we show that F-TCP has good throughput efficiency, intra-protocol fairness and TCP friendliness properties. Our results also illustrate fair co-existence between a delay-based protocol (F-TCP) and a loss-based protocol (regular TCP) thus F-TCP maintains it's fair share of the link. In addition we show that F-TCP avoids self-induced packet losses by using delay as the congestion signal hence zero packet loss is experienced in all the simulations where all the flows are F-TCP. In simulations with regular TCP, packet losses are inevitable since it is loss-based. We therefore conclude that when treated carefully, delay (or RTT) can indeed be a useful congestion signal indicator. Bandwidth estimation is a good indication of available bandwidth for delay-based flows if made aware of the early congestion back off experienced by these flows.**

***Index Terms*—High-speed networks; TCP congestion control; delay-based protocol; bandwidth estimation;**

I. Introduction

Transmission Control Protocol (TCP) has been the most commonly used transport protocol in the Internet since its design days in the 80's. Using TCP, an end-to-end network connection determines how much traffic (the congestion window or $cwnd$) to put on the network with in a certain period (round trip time or RTT). Given the same RTT, connections that have a larger $cwnd$ achieve a higher throughput compared to connections with a smaller $cwnd$. TCP interprets receipt of acknowledgements (ACKs) to indicate that there is more bandwidth on the network and therefore the connection can increase it's $cwnd$. In addition TCP may infer network congestion when packet losses occur or when the RTT increases much more than the round trip propagation delay.

Some of the assumptions made during TCP design about the network conditions have, however, changed drastically over the years and are no longer valid. In particular, the high-speed long distance nature of today's networks poses various challenges to traditional TCP, most importantly being it's slow convergence which leads to link under utilisation. Various high-speed TCP protocols have been proposed to address these challenges. We classify these protocols into three broad categories namely, loss-based, delay-based and loss-delay-based.

Loss-based protocols are those protocols which use only packet loss to infer network congestions. Some of the proposed high-speed protocols in this category include HSTCP [1], STCP [2], H-TCP [3], BI-TCP [4], CUBIC-TCP[5], TCPW [6] and HTCP-BE [7]. Due to increased aggressiveness these high-speed loss-based protocols increase link utilisation. Packet losses, however, also increase leading to more packet re-transmissions, and therefore inefficient usage of link bandwidth.

Loss-delay-based protocols are those protocols which combine both packet loss and delay information to infer congestion. TCP-Illinois [8] and CTCP [9] are two recently proposed loss-delay-based protocols. The evaluation of these two protocols in [10] shows some weaknesses in their behaviour. Particularly, it is shown that both bursty and long-lived traffic adversely affect their performance. In addition [10] shows that both TCP-Illinois and CTCP have poor convergence and RTT fairness properties similar to that of TCP NewReno *i.e.* under common network settings these protocols have low link utilisation just like TCP NewReno. The self-induced packet loss characteristic observed in loss-based protocols also exist in loss-delay-based protocols. Self-induced packet losses are undesirable because they increase inefficient use of network bandwidth as the lost packets need retransmission. The solution to this problem is to use purely delay-based congestion detection.

Delay-based protocols such as PERT[11] and m-PERT [12] on the other hand are those protocols which use only delay or RTT information to infer network congestion. When these protocols operate in a homogeneous environment no packet loss is experienced since they decrease $cwnd$ long before packet loss happens. They, however, have a co-existence challenge when operating in a heterogeneous environment with loss-based flows. Since they back-off much earlier, they can end up with a much lower throughput than their fair share. In Section II we discuss

978-0-7695-3979-9/10 $26.00 © 2010 IEEE
DOI 10.1109/ICN.2010.14

delay-based protocols in a more detailed manner.

Our contribution in this paper is the design of a delay-based protocol, called F-TCP that: 1) is able to co-exist fairly with loss-based protocols through the use of bandwidth estimation, and 2) operates efficiently in high-speed networks. In addition we improve a bandwidth estimation technique referred to as ABSE to accurately estimate available bandwidth for delay-based protocols and we show some limitations of m-PERT at high-speed links.

The paper is organised as follows: in Section II, we discuss the related work. In Section III, we describe the F-TCP algorithm and in Section IV, we present its performance evaluation. Finally, in Section V, we summarise the conclusions and present future work.

II. Related Work

This section discusses related work in terms of delay-based TCP protocols, Active Queue Management (AQM) schemes and Bandwidth estimation techniques. Our discussion of delay-based TCPs is because the proposed F-TCP is a delay-based protocol. AQM is introduced and a specific AQM called F-RED is discussed in more detail since the design of F-TCP borrows some ideas from F-RED in order to achieve fair co-existence between F-TCP and regular TCP. Bandwidth estimation and particularly Adaptive Bandwidth Share Estimation (ABSE) is also highlighted because it provides information necessary to emulate the behaviour of F-RED at the end-system level (as required by F-TCP which operates end-to-end).

A. Delay-based TCPs

Delay-based protocols particularly TCP Vegas [13], [14] can be traced back even before the era of high-speed networking. They use variation in RTT to detect network congestion and adjust the $cwnd$ appropriately. FAST TCP [15] is a high-speed variant of TCP Vegas in which absence of congestion implies much faster $cwnd$ increase. While these studies have discussed the issue of end-host delay-based congestion detection, some other studies have raised doubts about the accuracy and effectiveness of delay information [16], [17], [18]. However, a recent study [11] shows that delay information is in fact more accurate than previously believed. Using this observation [11] then proposes Probabilistic Early Response TCP (PERT), which is a delay-based protocol that emulates the behaviour of RED in the congestion response function. PERT is shown to have the following advantages: i) it efficiently maintains low queue lengths; ii) it has almost zero packet loss; and iii) a high degree of fairness in a homogeneous environment. PERT, however, is unable to co-exist with loss-based protocols as it responds to congestion much earlier causing the loss-based protocol flows to have an unfair share of the available bandwidth.

A modified version of PERT (henceforth referred to as m-PERT) that improves the co-existence situation is proposed in [12]. However, [19] shows that m-PERT still does not achieve fair co-existence between the m-PERT (delay-based) and regular TCP (loss-based) flows and also increases overall loss rates. Since both PERT [11] and m-PERT [12] emulate RED, it is useful to re-examine the characteristics of RED and the broader area of Active Queue Management (AQM) in an effort to improve fair co-existence.

B. AQM and F-RED

With AQM, an Internet router is enabled to drop or mark packets before it's queue is full. This is unlike the most popular and historical First In First Out (FIFO) queue management (with droptail queues) which only detects when the queue is full and drops all the incoming packets at that point until the queue clears. RED [20] is one of the most well-known AQMs which probabilistically distributes packet drops to a small subset of connections sharing the router as the queue builds up. In RED all the flows in this subset experience the same loss rate regardless of their bandwidth, therefore unfairness may exist in some cases [21].

1) The F-RED algorithm: F-RED [21] is a variation of RED, which in addition to all the benefits of RED also determines a flow's loss rate in relation to how many packets it has in the queue. F-RED differentiates flows by how many packets they have in the queue versus an average fair packet number that each flow should have. F-RED then ensures that flows with more packets in the queue experience a higher loss rate than flows with fewer packets. In so doing F-RED achieves a higher degree of fairness compared to RED. Just like RED, F-RED adopts the parameters $Qavg$, $Qmin_{thresh}$, $Qmax_{thresh}$ and $pmax$, which it uses to calculate the probability with which to drop an arriving packet.

F-RED additionally introduces five more parameters, namely:

- $minq$, a global constant indicating the minimum number of packets each active flow is allowed to buffer.
- $maxq$, a global variable indicating the maximum number of packets each active flow is allowed to buffer.
- $avgcq$, a global variable estimating the average number of packets each active flow should have in the buffer.
- $qlen$, a per-flow variable indicating a flow's count of it's buffered packets.
- $strike$, a per-flow variable indicating the number of times the flow has failed to respond to congestion notification.

With these parameters, F-RED is able to differentiate three types of traffic (non-adaptive, adaptive-robust and adaptive-fragile) and deals with them accordingly to ensure that fairness is attained.

2) End-host emulation of the F-RED algorithm: To emulate an AQM at the end-host it is important to note that the end-host knows much less about the network compared to the routers in which AQMs are implemented. In PERT, the end-host emulation of RED parameters $Qavg$, $Qmin_{thresh}$ and $Qmax_{thresh}$ are estimated using queuing delay. This is because the end-host has knowledge of flow queuing delay but not the buffer queue size. To emulate F-RED, we first simplify the algorithm described in [21] to only require knowledge of $avgcq$ and $qlen$ in addition

to the RED parameters. This simplified version is not able to isolate non-adaptive flows since $strike$ is not defined. However, since the simplification is for the purpose of emulating the F-RED algorithm at the end-host (with in TCP) this is not an issue since non-adaptive flows are UDP *i.e.* non TCP. At flow level $avgcq$ indicates each flow's fair share, while $qlen$ shows each flow's current contribution to the queue.

Assuming that bandwidth estimation techniques can be used to get an accurate estimate of the flow's available bandwidth, we conjecture that this estimate can be used to emulate $avgcq$. The flow's congestion window, $cwnd$ can then be used to emulate $qlen$. If our assumption about bandwidth estimation is right then emulating F-RED instead of RED addresses the challenge of fair co-existence raised in [19]. Next we review bandwith share estimation in order to establish its reliability in indicating a flow's fair share.

C. Adaptive Bandwidth Share Estimation (ABSE)

While various bandwidth estimation techniques [6], [22], [23] have been proposed, we explore ABSE [23]. ABSE is attractive because it adapts to changing network congestion levels, flow RTTs and to the rate of change of network conditions. It therefore results in a more accurate estimation of the connection's bandwidth share, when used for loss-based TCP (specifically TCPW [23], [24]). When using ABSE for delay-based TCP, however, we discovered that ABSE under-estimates the available bandwidth. We therefore modify it, inorder to use it in F-TCP (as discussed in Section III-A).

The advantages of delay-based protocols (particularly PERT) highlighted in Section II-A are attractive. The challenge of fair co-existence between delay-based and loss-based protocols, however, needs to be addressed before delay-based protocols can be deployed on the Internet, where loss-based protocols dominate. F-RED has potential to address this challenge and ABSE can be used to provide information for the emulation of F-RED in the end-system. The F-TCP protocol described in the next section is designed with these points in mind.

III. F-TCP Description

F-TCP is designed as a delay-based protocol with two objectives: firstly fair sharing of the available bandwidth between itself (a delay-based TCP) and loss-based TCPs and secondly efficient use of the bandwidth especially when operating in high-speed networks. F-TCP uses bandwidth estimation to set the appropriate $cwnd$ when competing with loss-based TCP. A modified ABSE technique was used in probing the available bandwidth in F-TCP. In this section we start by explaining how the ABSE network instability measure was modified in order for ABSE to estimate an accurate bandwidth share for delay-based F-TCP and then we describe the F-TCP algorithm in details.

A. Delay Sensitive Network Instability Measure

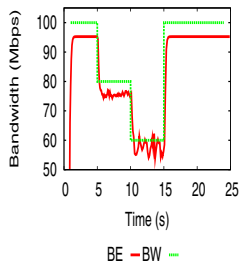
In ABSE [23], [24], the instantenous network instability (U_k) is measured with a constant EWMA filter, as illustrated in Equation 1.

$$U_k = \beta U_{k-1} + (1-\beta)|s_k - s_{k-1}| \quad (1)$$

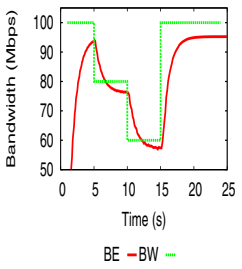
where s_k is the current bandwidth sample, s_{k-1} is the previous bandwidth sample and the weight (β) was set to 0.6 in [23], [25]. After a TCP connection's $cwnd$ decreases due to congestion (detected by either a packet loss or increased delay), the factor $|s_k - s_{k-1}|$ is much larger than it's previous or next values, causing U_k to increase. This is then interpreted as a persistent change resulting in a decrease of the agility filter (τ_k) and consequently leading to the available bandwidth estimate adapting very quickly to the latest (reduced) bandwidth samples. It is desirable for the bandwidth estimation to adapt quickly to reduced bandwidth samples after a packet loss event to quickly relieve congestion and avoid further packet losses. However if congestion was detected due to increased delay then adapting quickly to the reduced bandwidth may lead to the delay-based flow receiving a lower than fair bandwidth share especially when it co-exists with loss-based flows.

We therefore differentiate the network instability caused by response to packet loss from that due to increased delay thus calling it a delay-sensitive network instability measure. If the delay-based flow responds to increased delay then β in Equation 1 is assigned a value of 0.8 so as to place more importance on the previous instability measure. If, however, it experiences a packet loss then $\beta = 0.6$ as in the original ABSE algorithm. We refer to the agility filter calculated from the delay-sensitive network instability measure as the delay-sensitive adaptive agility filter τ'_k.

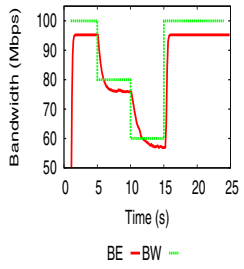
Figure 1 shows the bandwidth estimated by applying a small constant τ, a large constant τ, the ABSE adaptive τ_k and the delay-sensitive adaptive τ'_k. The sampling interval in this experiment is fixed to the inter-arrival time of the last two ACKs. The bottleneck link is set to a bandwidth of 100 Mbps with a one-way delay of 35 ms using a droptail queue whose buffer is set to the bandwidth delay product. The F-TCP flow shares the link with an on-off non-adaptive UDP flow whose data rate is time-varying in such a way that the available bandwidth to the F-TCP is as shown in the curve labelled BW of Figure 1. The results show that in all the four cases of τ, the estimator (labelled BE) tracks the available bandwidth quite well although when full bandwidth is available, all the cases estimate a slightly lower value. It is noted that in case (a) with small τ, the amplitude of oscillation is high and when the available bandwidth decreases, BE under-estimates it. In case of (b) with large τ, the oscillation is eliminated due to stronger filtering but the response to the available bandwidth is slow. In addition when the available bandwidth decreases, BE under-estimates it, just as in (a) above. In case (c) with adaptive τ_k, the estimator has fast response to available bandwidth with less oscillation but just like (a) and (b) when the available bandwidth decreases, BE under-estimates it. In case (d) with delay-sensitive adaptive τ'_k, the estimator has a quite fast response to available bandwidth with low oscillation and unlike (a), (b) and (c) when



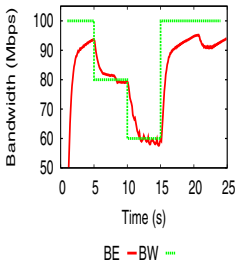
(a) Estimate with $\tau = 0.1$ sec



(b) Estimate with $\tau = 1$ sec

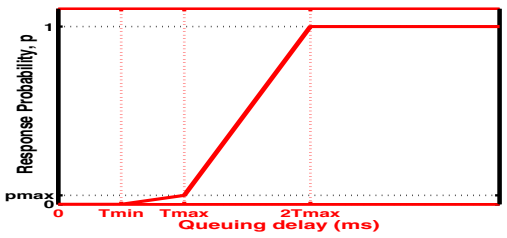


(c) Estimate with Adaptive τ_k

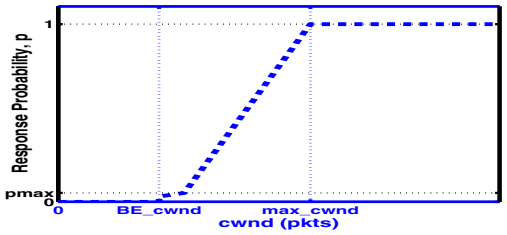


(d) Estimate with Delay-sensitive Adaptive τ'_k

Fig. 1. Bandwidth estimation with different τ



(a) RED emulation



(b) simplified F-RED emulation

Fig. 2. Backoff probability function

the available bandwidth decreases, BE tracks it well. Case (d) therefore most accurately estimates the available bandwidth when there is competition in the link and we therefore use it in delay-based F-TCP. Next we describe the F-TCP algorithm.

B. The F-TCP Algorithm

The proposed F-TCP protocol is characterised with a delay-based congestion response in a probabilistic manner as initially proposed by the PERT algorithm [11], which we modify slightly to emulate a simplified F-RED algorithm. That is, if the observed queuing delay is high, the path is presumed congested. However delay measurements may be noisy, and to mitigate this, a probabilistic response is used. The probabilistic response sets a smaller probability of response when the perceived queuing delay is small, which increases as the queuing delay increases.

Figure 2(a) shows the response probability, p against the queuing delay. Similar to gentle RED, three thresholds are defined, $Tmin$ (emulates $Qmin_{thresh}$), $Tmax$ (emulates $Qmax_{thresh}$) and $2Tmax$ (emulates $2Qmax_{thresh}$). In addition, the maximum response probability $pmax$ used by RED is also defined. These parameters have been previously discussed in Section II-B1. When the queuing delay $\leq Tmin$, $p = 0$. As the queuing delay increases beyond $Tmin$, p increases linearly until $pmax$ at $Tmax$. For $Tmax <$ queuing delay $< 2Tmax$, p increases linearly between $pmax$ and 1. For queuing delay $> 2Tmax$, p stays constant at 1. For parameters $Tmin$, $Tmax$ and $pmax$, we use 5 ms, 10 ms and 0.05 respectively as was used in [12]. To emulate the simplified F-RED algorithm, Figure 2(b) also shows the response probability, p against $cwnd$. Similar to F-RED two dynamic thresholds are defined, namely, BE_cwnd (emulates $avgcq$) and max_cwnd (emulates $maxq$). For $cwnd \leq BE\_cwnd$, $p = 0$ and when $cwnd$ increases beyond BE_cwnd, then p increase is defined by the queuing delay, in such a way that $cwnd = max\_cwnd$ when $p = 1$.

The slow-start threshold ($ssthresh$) is limited by the available bandwidth ($BE_{current}$) and $cwnd$ increases exponentially until it reaches $ssthresh$. Beyond $ssthresh$ if no congestion is detected, $cwnd$ increases linearly just like in regular TCP. When congestion is detected (either based on increasing delay probabilistically or by 3dupacks), $cwnd$ is decreased to a value derived from the bandwidth estimate, $BE_{current}$. When congestion is detected by a timeout of an unacknowledged packet, $cwnd$ is reset to 1 and $ssthresh$ is set proportional to $BE_{current}$. The F-TCP protocol is summarised in Algorithm 1.

IV. Performance Evaluation

In this section, we discuss the simulation setup and performance evaluation of F-TCP. The evaluation is conducted by testing F-TCP performance while examining some of the well known transport protocol metrics such as throughput efficiency, responsiveness, protocol fairness, TCP friendliness, stability and RTT fairness.

A. Simulation setup

The ns-2 simulator [26] is used to evaluate the performance of F-TCP using the dumbbell network topology illustrated in Figure 3, where

- S_1 - S_n are 1 to n senders of flows of F-TCP or other protocols (e.g. regular TCP, UDP or another high-speed TCP)

Algorithm 1 The F-TCP algorithm

Initialise $cwnd = 1$, $ssthresh = BDP$, $dupacks = 0$ {BDP is the bandwidth delay product}
On receipt of each ACK:
Estimate the current queuing delay, $\delta_{current}$
Compute the available bandwidth estimate, $BE_{current}$ {Using ABSE with a delay-sensitive instability measure}
$previousBE\_cwnd = BE\_cwnd$
$BE\_cwnd = BE_{current} * minRTT$
if $BE\_cwnd < previousBE\_cwnd$ **then**
  $ssthresh = previousBE\_cwnd$
end if
if $cwnd < BE\_cwnd$ **then**
  $p = 0$ {As in Fig. 2, p Vs $cwnd$}
else
  $p = f(\delta_{current})$ {Just as in Fig. 2}
end if
if ACK is a duplicate **then**
  Increment dupacks by 1
end if
if $cwnd < ssthresh$ **then**
  $cwnd+ = 1$ {Increase exponentially since this is slow-start}
else
  Pick a random number $rand$, uniformly from 0 to 1
  if $rand < p \wedge$ only once per RTT **then**
    $cwnd = BE\_cwnd$
  else
    increment $cwnd$ by $\alpha/cwnd$
    $(\alpha = 1)$
  end if
end if
if $dupacks > 3$ **then**
  $cwnd = BE\_cwnd$
  $ssthresh = BE\_cwnd$
  $dupacks = 0$
end if
if TIMEOUT **then**
  $cwnd = 1$
  $ssthresh = BE\_cwnd$
end if

- D_1 - D_n are the corresponding destinations of flows of F-TCP or other protocols (e.g. regular TCP, UDP or another high-speed TCP)
- R1 and R2 are routers such that R1–R2, the bottleneck link has bandwidth B Mbps which is either 100 Mbps or 1000 Mbps and the one way link delay, Y ms is either 5 ms or 50 ms.
- All the other links *i.e.* S_1–R1, R2–D_1, S_n–R1 and R2–D_n, have bandwidth of A Mbps and one-way delays of X ms.

For most simulations (Sections IV-B - IV-E) the bottleneck link (R1–R2) $B = 100$ Mbps, $Y = 50$ ms, $A = 100$ Mbps and $X = 10$ ms. For these simulations $minRTT$ is 140 ms. For the simulations in Section IV-F, B is either 100 Mbps or 1000 Mbps, $A = B$, Y is 5 ms and X is uniformly distributed between 5 and 30 ms resulting in $minRTT$ between 30 and 130 ms. The droptail queue is used in all the simulations and the queue buffer size is set to $1XBDP$ where BDP is the bandwidth delay product.

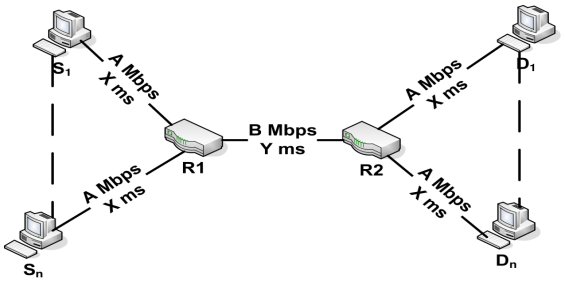
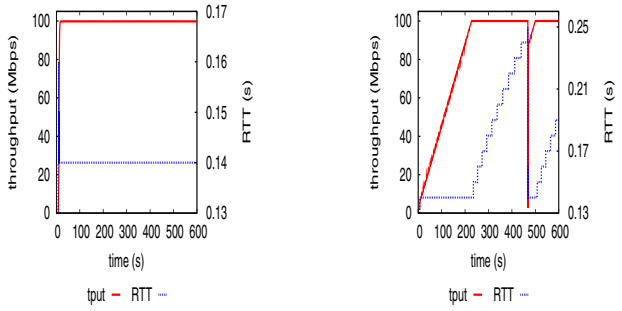


Fig. 3. Dumbell Topology

B. Throughput Efficiency and Responsiveness

To test the throughput efficiency and responsiveness of F-TCP a single flow is simulated, while monitoring its throughput and RTT throughout the simulation. For comparison purpose, the simulation is repeated for regular TCP (TCP NewReno).



(a) F-TCP (b) Regular TCP

Fig. 4. Throughput and RTT for a Single Flow: (a) F-TCP (b) Regular TCP

Figure 4(a) shows that the F-TCP flow very quickly grabs and maintains the full bandwidth, while still keeping its RTT at the minimum of 140 ms. F-TCP is fast in reaching full bandwidth because it stays in the slow-start phase (in which the $cwnd$ increases very fast) until the bandwidth estimate terminates it. F-TCP maintains full bandwidth while keeping low RTT because it keeps track of its queuing delay to limit $cwnd$ to the data rate. The regular TCP flow (Figure 4(b)) on the other hand, slowly reaches the available bandwidth (after 200 s). It is also shown that the TCP flow's RTT continues to increase implying that TCP continues to fill the queue much faster than it is being emptied even after full bandwidth is achieved. This leads to a self-induced loss at ~ 470 s which drives the flow's

bandwidth momentarily below the link bandwidth. From this result we conclude that F-TCP is more efficient and responsive than regular TCP.

C. Protocol Fairness

To test the intra-protocol fairness properties of F-TCP, 20 F-TCP flows are made to share the bottleneck link.

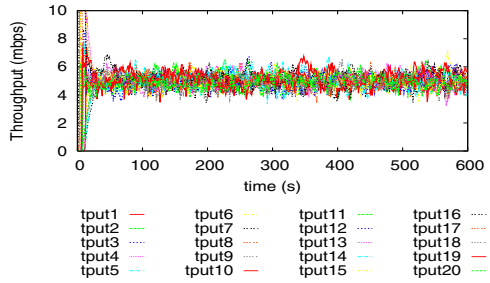


Fig. 5. Throughput of Multiple (20) Flows

Figure 5 shows the throughput of each of the 20 flows for the entire simulation. It can be seen that the 20 flows equally share the bottleneck capacity. Therefore F-TCP is fair (comparable to regular TCP, which is also known to be fair).

D. TCP Friendliness

To evaluate TCP friendliness a simulation where an F-TCP flow shares the bottleneck with a regular TCP flow is set

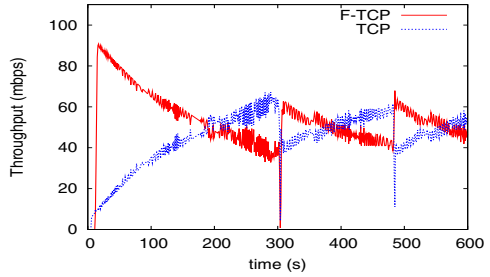
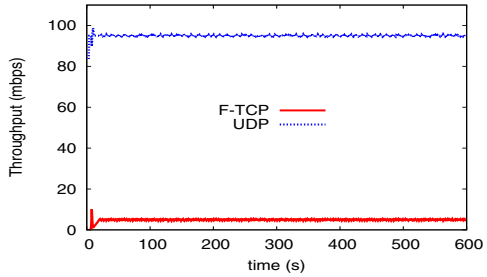


Fig. 6. Throughput for 1 F-TCP & 1 Regular TCP

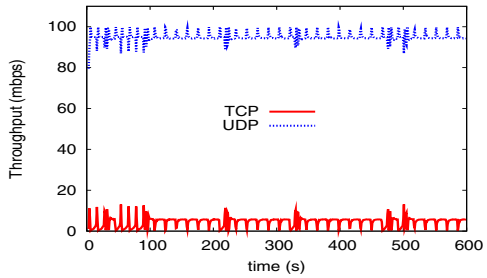
Simulation results in Figure 6 show the throughput of both flows. It can be seen that F-TCP quickly grabs the available bandwidth, which it reduces gradually as TCP increases. TCP is characterised with increasing its throughput until the queue is full and packet loss is experienced. In this simulation two packet loss events are seen at $\sim$ 305 s and $\sim$ 480 s. Both TCP and F-TCP experience packet loss since the queue is full, however, immediately after the packet loss event, F-TCP recovers quickly as it grabs the available bandwidth while TCP returns to its conservative increase. From this result we conclude that F-TCP has both good TCP friendliness and throughput efficiency properties.

E. Interaction with UDP traffic

UDP traffic is known to use as much bandwidth as its sending rate and is non-responsive to congestion. To compare UDP interaction with F-TCP versus UDP interaction with TCP, we simulate a UDP flow of data rate set to 95 Mbps sharing the (100 Mbps) link with either an F-TCP or TCP flow.



(a) F-TCP and UDP



(b) TCP and UDP

Fig. 7. Throughput of a heavy UDP flow with: (a) F-TCP and (b) Regular TCP

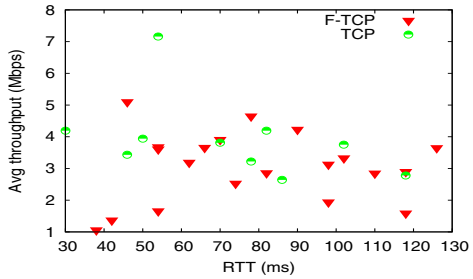
The simulations results in Figure 7 show that F-TCP utilises the bandwidth left over by the UDP traffic more efficiently than TCP. This is because the F-TCP flow settles at the available bandwidth, while the TCP throughput keeps increasing to values greater than the available bandwidth causing congestion for itself as well as the UDP flow.

F. Co-existence with loss-based flows in comparison with m-PERT

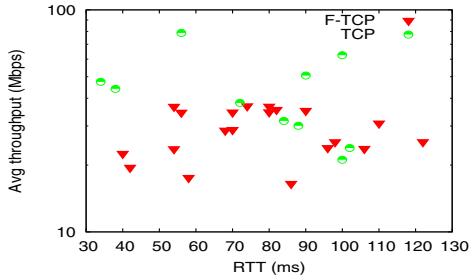
To evaluate the co-existence properties of F-TCP with loss-based regular TCP, a simulation in which 20 flows of F-TCP together with 10 flows of regular TCP sharing the bottleneck is run. For comparison purpose the same simulation is repeated, where m-PERT flows replace the F-TCP flows.

Figure 8 shows 20 flows of F-TCP competing with 10 flows of regular TCP while Figure 9 shows 20 flows of m-PERT competing with 10 flows of regular TCP. This experiment is done for both a 100 Mbps and 1 Gbps bottleneck link. The results show that F-TCP provides fairer co-existence than m-PERT for both low-speed and high-speed links.

The behaviour of m-PERT in the high-speed bottleneck is surprising as the m-PERT flows get a very low throughput. We

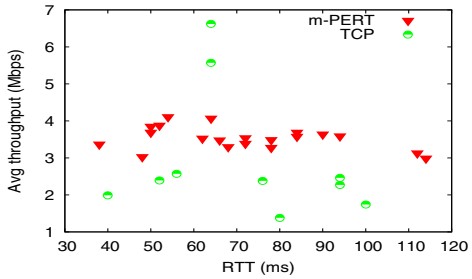


(a) 100 Mbps bottleneck

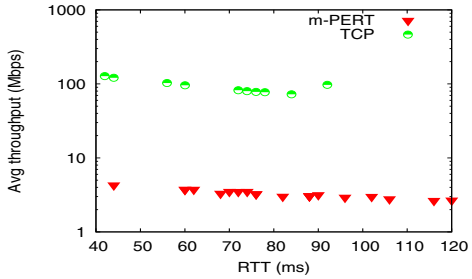


(b) 1 Gbps bottleneck

Fig. 8. Throughput of 20 F-TCP & 10 Regular TCP; (a) 100 Mbps bottleneck & (b) 1 Gbps bottleneck



(a) 100 Mbps bottleneck



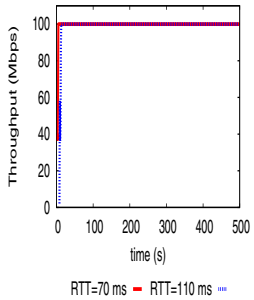
(b) 1 Gbps bottleneck

Fig. 9. Throughput of 20 m-PERT & 10 Regular TCP; (a) 100 Mbps bottleneck & (b) 1 Gbps bottleneck

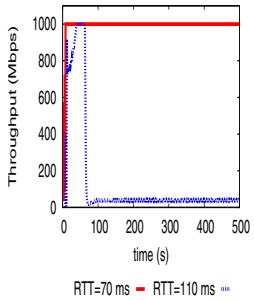
further investigate this behaviour by setting a simulation of one m-PERT flow of RTT either 70 ms or 110 ms on a link of either 100 Mbps or 1 Gbps. Figure 10 shows that m-PERT performs well at low bandwidth for both low and high RTTs, however, at high bandwidth, m-PERT performance suffers for high RTTs. The parameters $Tmin$ and $Tmax$, previously discussed in Section III have to be adjusted upwards to 20ms and 40ms respectively for m-PERT to be efficient for high RTT flows on a high-speed link. It is unknown how these parameter changes would affect performance under other network settings. We note that m-PERT was reported to work well in high-speed networks [12], the simulation done to reach this conclusion was of a single m-PERT flow of $RTT = 70$ ms. We see similar result in Figure 10(b) for the 70 ms curve. However when the RTT of the m-PERT flow is increased to 110 ms the m-PERT flow is unable to sustain the full bandwidth, unless $Tmin$ and $Tmax$ are adjusted. This limitation of m-PERT in high-speed networks is also seen when there are multiple m-PERT flows of different RTTs sharing a high-speed link with TCP flows (Figure 9(a)). F-TCP flows of high RTT do not suffer on high-speed links because F-TCP adjusts the $cwnd$ (and therefore the throughput) to a value derived from the available bandwidth.

To summarise this section, the queue length (in packets) and the packet loss fraction (packets lost/ packets sent) of the bottleneck link in each of the experiments presented previously is given in Table I

From the table it can be seen that simulations involving only F-TCP flows or F-TCP and UDP have a low average queue and



(a) 100 Mbps bottleneck



(b) 1 Gbps bottleneck

Fig. 10. Throughput of an m-PERT flow with low RTT and high RTT; (a) 100 Mbps bottleneck & (b) 1 Gbps bottleneck

zero packet loss. While those with regular TCP, have a higher average queue size and average packet loss. F-TCP inherits low queue and zero packet loss properties from the PERT [11], [12] algorithm by design.

V. Conclusion and Future Work

So far, we have analysed F-TCP, which uses delay as a congestion signal, whose slowstart phase continues until a threshold determined from probing the available bandwidth and which reduces its $cwnd$ when competing with loss-based flows to a value suggested by the available bandwidth. F-TCP shows good throughput efficiency, intra-protocol fairness and TCP friendliness properties. It avoids self-induced packet losses by

Experiment	AvgQueue (pkts)	AvgPktLoss
1 F-TCP	0.6	0
1 TCP	327.4	$4.5 * 10^{-8}$
20 F-TCP	45.5	0
1 F-TCP & 1 TCP	674.5	$7.5 * 10^{-7}$
1 F-TCP & 1 UDP	39.9	0
1 TCP & 1 UDP	562.8	$1.2 * 10^{-3}$
20 m-PERT & 10 TCP (100 Mbps link)	1267.9	$1.9 * 10^{-3}$
20 F-TCP & 10 TCP (100 Mbps link)	1392.6	$8.2 * 10^{-4}$
20 m-PERT & 10 TCP (1 Gbps link)	10114.6	$1.4 * 10^{-6}$
20 F-TCP & 10 TCP (1 Gbps link)	12733.1	$1.4 * 10^{-5}$

TABLE I
BOTTLENECK AVERAGE QUEUE SIZE AND PACKET LOSS RATE OF THE DIFFERENT EXPERIMENTS

using delay as the congestion signal hence zero packet loss is experienced in all the simulations where all the flows are F-TCP. In simulations with regular TCP, packet losses are inevitable since it is loss-based. We ensure fair co-existence between F-TCP (a delay-based protocol) and regular TCP (a loss-based protocol) by employing adaptive bandwidth share estimation with a delay-sensitive instability measure to guide backoff when congestion is experienced.

Delay (RTT) is indeed a useful congestion signal indicator when treated carefully. Bandwidth estimation is a good indication of available bandwidth for delay-based flows if it is made aware of the early congestion back off experienced by these flows. The following areas need further investigation.

1) F-TCP's dynamics in terms of stability as well as its equilibrium properties need to be examined. Fluid models can be used for this study as was done for some high-speed TCP variants in [27]. These fluid models will be used to provide insight about boundary conditions of F-TCP.
2) Comparing F-TCP performance with other high-speed TCP protocols such as CUBIC, TCP-Illinois and C-TCP.
3) F-TCP's behaviour when there are multiple bottlenecks also needs to be studied.

REFERENCES

[1] S. Floyd, "Highspeed TCP for large congestion windows," Internet Engineering Task Force, RFC 3649, 2003. [Online]. Available: www.rfc-editor.org/rfc/rfc3649.txt [Last accessed on January 14, 2010]

[2] T. Kelly, "Scalable TCP: Improving performance in highspeed wide area networks," *ACM SIGCOMM Computer Communications Review*, vol. 33, no. 2, 2003.

[3] R.N.Shorten and D.J.Leith, "H-TCP: TCP for high-speed and long-distance networks," *Proc. PFLDnet, Argonne*, 2004.

[4] L. Xu, K. Harfoush, and I. Rhee, "Binary Increase Congestion Control for Fast Long-Distance Networks," in *Proc. IEEE INFOCOM*, 2004.

[5] I. Rhee and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant," *Proc. PFLDnet*, 2005.

[6] C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, "TCP Westwood: End-to-End Bandwidth Estimation for Enhanced Transport over Wireless Links," in *Proceedings of MOBICOM 2001*, 2002, pp. 467–479.

[7] J. Sansa, A. Szomoru, and J. M. V. der Hulst, "An evaluation study of data transport protocols for e-vlbi," *International Journal of Computing and ICT Research*, vol. 1, no. 1, pp. 68–75, 2007.

[8] S. L., T. B., and S. R., "TCP-Illinois: A Loss and Delay-Based Congestion Control Algorithm for High-Speed Networks," in *Proc. First International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS), Pisa, Italy*, October.

[9] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "Compound TCP: A scalable and TCP-friendly congestion control for high-speed networks," *Proc. PFLDnet*, 2006.

[10] D. J. Leith, L. L. H. Andrew, T. Quetchenbach, R. N. Shorten, and K. Lavi, "Experimental evaluation of delay/loss-based TCP congestion control algorithms," *Proc. PFLDnet*, 2008.

[11] S. Bhandarkar, A. L. N. Reddy, Y. Zhang, and D. Loguinov, "Emulating AQM from End Hosts," in *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2007, pp. 349–360.

[12] K. Kotla and A. Reddy, "Making a Delay-Based Protocol Adaptive to Heterogeneous Environments," in *Proc. 16th IEEE International Workshop on Quality of Service, (IWQoS 2008), Enschede, The Nethalands*, June 2008.

[13] L. S. Brakmo, S. O'Malley, and L. L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," in *Proceedings of the SIGCOMM '94 Symposium*, August, pp. 24–35.

[14] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1465–1480, 1995.

[15] C. Jin, D. Wei, and S. Low, "FAST TCP: Motivation, Architecture, Algorithms, Performance," in *Proc. IEEE INFOCOM*, 2004.

[16] J. Martin, A. Nilsson, and I. Rhee, "Delay-based congestion avoidance for TCP," *IEEE/ACM Trans. Netw.*, vol. 11, no. 3, pp. 356–369, 2003.

[17] R. S. Prasad, M. Jain, and C. Dovrolis, "On the effectiveness of delay-based congestion avoidance," *Proc. PFLDnet*, 2004.

[18] J. K. S. Rewaskar and D. Smith, "Why dont delay-based congestion estimators work in the real-world?" Department of Computer Science, UNC Chapel Hill, Tech. Rep. TR06-001, 2005.

[19] L. Budzisz, R. Stanojevi, A. Schlote, R. Shorten, and F. Baker, "On the fair coexistance of loss- and delay-based TCP," in *Proc. 17th IEEE International Workshop on Quality of Service, (IWQoS 2009), Charleston, South Carolina*, July 2009.

[20] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, 1993.

[21] D. Lin and R. Morris, "Dynamics of Random Early Detection," *SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 4, pp. 127–137, 1997.

[22] M. Gerla, B. K. F. Ng, M. Y. Sanadidi, M. Valla, and R. Wang, "TCP Westwood with Adaptive Bandwidth Estimation to Improve Efficiency/Friendliness Tradeoffs," *COMPUTER COMMUNICATIONS*, vol. 27, pp. 41–58, 2003.

[23] R. Wang, M. Valla, M. Y. Sanadidi, and M. Gerla, "Adaptive Bandwidth Share Estimation in TCP Westwood," in *Proc. IEEE Globecom*, 2002.

[24] R. Wang, M. Valla, M. Sanadidi, and M. Gerla, "Using adaptive rate estimation to provide enhanced and robust transport over heterogeneous networks," in *Proceedings of the International Conference on Network Protocols*. IEEE, 2002.

[25] M. Kim and B. Noble, "Mobile network estimation," in *Proceedings of the 7th ACM Conf. Mobile Computing and Networking (MobiCom'01)*, pp. 298–309.

[26] The ns-2 Simulator. [Online]. Available: www.isi.edu/nsnam/ns [Last accessed on January 14, 2010]

[27] J. Sansa, P.-T. de Boer, I. A. Rai, and J. M. van der Hulst, "Fluid Flow Models of High-Speed TCP variants," 2010, to be submitted for publication. [Online]. Available: www.astro.rug.nl/šansa/files/ffm_revision16.pdf [Last accessed on January 25, 2010]