Public Review for

Probe-Aided MulTCP: An Aggregate Congestion Control Mechanism

Fang-Chun Kuo and Xiaoming Fu

Despite some views on changing things (e.g. “Flow Rate Fairness: Dismantling a Religion” by Bob Briscoe fairly recently in these very pages), the Internet still runs smoothly because (at least partly) of TCP-friendliness. Actually, some people suggest that it is also possibly that most sites are access link speed limited, and the core networks are over-provisioned, but that is another story.

However, there are certainly links, and indeed end-to-end paths where there is plenty of capacity. These links are still shared, and capacity has to be assigned somehow, but why should it be assigned only in *units of a single TCP flowshare's worth?*

A goal in the original work on MulTCP was to provide a mechanism for single sources to allocate themselves multiple TCP’s worth of capacity, but remain collectively TCP-friendly. This paper extends that work to improve the fairness of MulTCP by provide a *reference* flowshare of a single TCP’s worth, and use this to control the window adjustment for the aggregate rate. The authors explain the algorithm and evaluate it using ns-2, and present the results clearly and effectively. There may be niche applications of this approach directly (as they mention, in storage area networks, or in some overlays), and the algorithm is of interest to those working on TCP-friendly flows of other type (e.g. multimedia).

We might talk about beating weapons of TCP-friendliness into flowshares.

Public review written by
Jon Crowcroft
University of Cambridge

Probe-Aided MulTCP: An Aggregate Congestion Control Mechanism

Fang-Chun Kuo and Xiaoming Fu
Institute for Computer Science, University of Goettingen, Germany
{fkuo,fu}@cs.uni-goettingen.de

ABSTRACT

An aggregate congestion control mechanism, namely Probe-Aided MulTCP (PA-MulTCP), is proposed in this paper. It is based on MulTCP, a proposal for enabling an aggregate to emulate the behavior of multiple concurrent TCP connections. The objective of PA-MulTCP is to ensure the fair sharing of the bottleneck bandwidth between the aggregate and other TCP or TCP-friendly flows while keeping lightweightness and responsiveness. Unlike MulTCP, there are two congestion window loops in PA-MulTCP, namely the probe window loop and the adjusting window loop. The probe window loop constantly probes the congestion situation and the adjusting window loop dynamically adjusts the congestion window size for the arriving and departing flows within the aggregate.

Our simulations demonstrate that PA-MulTCP is more stable and fairer than MulTCP over a wide range of the weight N in steady conditions as well as in varying congestion conditions. PA-MulTCP is also responsive to flow arrival/departure and thus reduces the latency of short-lived transfers. Furthermore, PA-MulTCP is lightweight, since it enjoys above advantages at the cost of only an extra probe window loop, which has a marginal influence on the implementation complexity. Finally, the design of PA-MulTCP decouples the congestion management from the other functionalities in the aggregate flow management. As a result, PA-MulTCP could be potentially applied to a wider range of scenarios, e.g. wireless TCP proxies, edge-to-edge overlays, QoS provisioning and mass data transport.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture and Design—*Network communications*

General Terms

Design

Keywords

TCP, Fairness, Congestion Control, Aggregate Flow Management

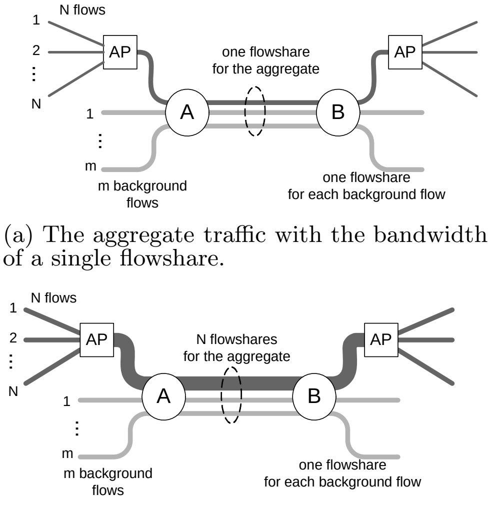
1. INTRODUCTION

End-to-end measurement is performed in many existing stateful Internet transport protocols, e.g. SCTP [27], DCCP [9], TCP. In these protocols *per-flow* states are maintained for managing network resources. However, each flow is unaware of the network information available from other flows due to per-flow management, especially the information from other flows sharing the same bottleneck. As a consequence, each flow has to individually and repeatedly probe the network for discovering its states. This might result in low efficiency of resource utilization.

Aggregate flow management has been proposed for improving network utilization [3,6,7,14,16,22,24,25]. Its main idea is to introduce certain cooperation between flows sharing the same bottleneck [7, 24]. More explicitly, multiple flows sharing the same bottleneck are aggregated into a single flow or a smaller number of flows in an aggregate flow management scheme. Thus a coarser grain of resource management is performed, while details of individual flows are hidden. Aggregate flow management is applied at an *aggregate point* (AP), e.g. an end host or a router, to the flows traversing a bottleneck link. These flows are controlled by the AP jointly, rather than independently. Aggregate flow management has also been applied in a wide range of contexts, such as edge-to-edge QoS overlay services [28], wireless TCP proxies for addressing network heterogeneity [4], a coordination protocol for distributed media applications [20], and an overlay network architecture providing DoS-limiting as well as resilience on network edges [12]. Furthermore, it could be used in modern storage networks in which a huge amount of block data are exchanged over a single TCP connection [26].

It is essential that the aggregate traffic is congestion responsive, although individual flows within the aggregate may use a variety of transport-level protocols, including those without congestion control. With the aid of congestion control, aggregate traffic should achieve high bandwidth utilization, while acquiring a fair share of the bottleneck link bandwidth. Congestion control may be either coupled or decoupled from the other functionalities in a specific aggregate flow management scheme. In this paper we focus on the aggregate congestion control mechanism only, i.e. it is decoupled from the other functionalities. The decoupling nature enables combining the aggregate congestion control mechanism with different functionalities in an aggregate flow management scheme.

One of the most important performance metrics in aggregate congestion control is fairness, i.e. the equal use of resources. *Flowshare* was introduced for characterizing fairness in aggregate congestion control [20]. A flowshare is defined as the bandwidth utilized by a single congestion-controlled flow, i.e. a single TCP connection. In the aggregate flow management schemes proposed in [3,12,16,22],



(a) The aggregate traffic with the bandwidth of a single flowshare.

(b) The aggregate traffic with the bandwidth of multiple flowshares.

Figure 1: Fairness Issue in Aggregate Congestion Control.

one standard TCP congestion window loop [5,15] is used for the aggregate. This results in a single flowshare as shown in Figure 1(a). Both the aggregate and each of the m background flows are allocated $\frac{1}{m+1}$ share of the bottleneck link bandwidth, although there are N flows within the aggregate. This is unfair since the share of the bottleneck link bandwidth allocated to each flow within the aggregate is $\frac{1}{(m+1)N}$, while that allocated to each background flow is $\frac{1}{m+1}$. Ideally as shown in Figure 1(b), N flowshares should be allocated to an aggregate having N flows under a good aggregate congestion control mechanism, i.e. the share of the bottleneck link bandwidth allocated to the aggregate should be $\frac{N}{N+m}$. Therefore, the aggregate congestion control should support multiple flowshares so as to ensure a fair share of the bottleneck link bandwidth for the aggregate.

Multiple flowshares in aggregate congestion control have been adopted in the design of MulTCP [6], Multi-Probe Aggregate TCP (MPAT) [25] and Coordination Protocol (CP) [20]. However, none of them is an elegant solution to the fairness issue. One problem in MPAT is that the information in congestion windows is not integrated, which is ineffective for short-lived transfers and may degrade bandwidth utilization. Another problem in MPAT is that numerous congestion window states have to be maintained. Furthermore, due to the inappropriate congestion adjustment, neither CP nor MulTCP can achieve a fair share when the same bottleneck link is shared with the aggregate and other TCP or TCP-friendly flows.

The novel contribution of this paper is an aggregate congestion control mechanism, namely Probe-Aided MulTCP (PA-MulTCP). It extends MulTCP with an additional probe window, which effectively improves fairness without the need of managing numerous window states. Our simulation results show that with the aid of PA-MulTCP the aggregate can hold its fair share of the bottleneck bandwidth while other TCP flows are also treated fairly. Furthermore, it is also demonstrated that, due to the integrated congestion information, the latency of short-lived transfers is reduced by $25-50\%$ in PA-MulTCP compared to that in MPAT.

The outline of this paper is as follows. Existing techniques supporting multiple flowshares in aggregate congestion control are summarised in Section 2. Since our proposal is based on MulTCP, the mechanism of MulTCP and its limitations are discussed in Section 3. Our design of PA-MulTCP is detailed in Section 4. Our simulation results are presented in Section 5, which validate our design. Finally, our conclusions are provided in Section 6.

2. RELATED WORK

There are several protocols proposed for improving fairness by supporting multiple flowshares, e.g. MulTCP [6], CP [20] and MPAT [25].

The Additive Increase Multiplicative Decrease (AIMD) algorithm [15] is the traditional congestion control mechanism used in standard TCP. It is modified in MulTCP [6] for enabling an aggregate to emulate the behavior of multiple concurrent TCP connections. Nevertheless, it has been demonstrated in [13,25] that beyond small values of N, a MulTCP flow fails to behave like N independent TCP flows. In other words, the throughput of the aggregate in MulTCP does not increase proportionally to the number of flows within the aggregate, especially when $N > 4$. Furthermore, it has been noted in [25] that loss behavior of a single MulTCP flow is quite different from that of N independent TCP flows. This leads to an increasingly unstable window adjustment as N increases. Hence it is not clear if MulTCP is tunable to behave like N TCP flows, when N is larger than four.

In order to avoid intractable tuning for the modified AIMD in MulTCP, MPAT [25] has been proposed for ensuring a fair share among individual flows and decoupling application flows from their congestion windows. The number of active congestion window loops in MPAT is the same as the number of flows within the aggregate. As a result, the more flows a MPAT aggregate manages, the more number of congestion window states are maintained. Furthermore, each congestion window loop in MPAT probes the network condition independently, which results in competition among different congestion window loops. This may degrade performance, especially for the throughput of the short-lived flows [7,24]. For example, web transfers are typically characterized by fragile short-lived TCP flows. If such a short-lived TCP flow joins the aggregate, the newly created congestion window loop does not have enough time for adapting to the network state. Thus the new congestion window loop suffers from longer delay and more packet losses, compared to the other congestion window loops maintained in the aggregate.

In contrast to multiple AIMD window loops in MPAT for ensuring fairness, one TCP-Friendly Rate Control (TFRC) loop [10] is employed in CP [20] for estimating the available bandwidth for a single TCP flow. The total bandwidth for the aggregate having N flows is simply N times the estimated bandwidth for a single flow. Since only one congestion window loop is used in CP, the competition between multiple congestion window loops in MPAT is eliminated in CP. Unfortunately, the bandwidth estimated by TFRC is unreliable [23]. It could be more than 20 times or less than $\frac{1}{10}$ as much as the actual bandwidth for a single TCP flow. This results from the difference in loss event rate, which is incurred by different sending rates and greatly increases the initial throughput difference. Therefore, the TFRC rate adjustment loop used in CP may lead to aggressive behavior or inefficient bandwidth utilization due to an erroneous estimation of the sending rate.

Generally it would be desirable if an aggregate congestion control mechanism supporting multiple flowshares could possess the following properties:

- **Lightweight**. A moderate number of congestion window states should be maintained.
- **Responsiveness**. The congestion control mechanism should respond promptly to changes in congestion conditions. Also when short-lived transfers join the aggregate, it should reduce initialization overheads for obtaining network conditions so that it can promptly reallocated network resources.
- **Fairness**. Ideally the aggregate serving N flows should achieve N flowshares with the aid of aggregate congestion control. It should avoid both aggressive behavior, i.e. the achieved bandwidth is far greater than N flowshares, and inefficient bandwidth utilization, i.e. the achieved bandwidth is far less than N flowshares.

Probe-Aided MulTCP (PA-MulTCP) is proposed to meet these challenges. There are two window loops in PA-MulTCP. One is a modified AIMD congestion window loop, which is similar to that in MulTCP. The other is a probe window loop, which assists the modified AIMD congestion window loop for estimating the congestion window size. The introduction of an extra probe window loop is of great benefit for the aggregate to achieve effective throughput at the cost of only marginal complexity. In contrast to the TFRC rate adjustment loop in CP for estimating the bandwidth of a single TCP flow, more accurate estimation of the TCP flow behavior is obtained by the probe window loop in PA-MulTCP.

3. MULTCP PROBLEM FORMULATION

Since PA-MulTCP is based on MulTCP, the aggregate congestion control mechanism in MulTCP is overviewed in this section. Its aggressive behavior is illustrated by our simulation results.

3.1 MulTCP

AIMD algorithm is adopted in the TCP congestion control. There are two parameters in AIMD, the additive increase parameter a and the multiplicative decrease parameter b.

Let us denote the AIMD algorithm with the parameters a and b as AIMD(a, b). The congestion window size is decreased to the $(1 - b)$ times of current window size when a loss event is detected, otherwise it is increased by a packets every Round Trip Time (RTT). AIMD$(1, 1/2)$ is used in the standard TCP [2].

Crowcroft and Oechslin proposed MulTCP [6], aiming at proportional multiple flowshares along a congested link by assigning a weight N to the aggregate. AIMD$(N, \frac{1}{2N})$ is used in MulTCP, instead of AIMD$(1, 1/2)$ in the standard TCP, for emulating the behavior of N concurrent TCP connections using the same congestion window loop. The rationales behind AIMD$(N, \frac{1}{2N})$ are:

- **Increase parameter** $a = N$. The congestion window size of a single TCP flow is increased by one packet per RRT in the standard TCP. Accordingly, the congestion window size of N virtual TCP flows is increased by N packets per RRT in MulTCP.
- **Decrease parameter** $b = \frac{1}{2N}$. If one packet is lost, only one of the N TCP flows halves its congestion window size W in the standard TCP. Accordingly, the congestion window size becomes $(N - 1)W + \frac{1}{2}W = (1 - \frac{1}{2N})NW$ in MulTCP.

For more general MulTCP [19], the parameter pair (a, b) can be tuned for generating a desired sending rate, as long as some relationship between a and b is satisfied. This relationship can be derived from following sending rate equations for TCP-like protocols using AIMD(a, b) [29]:

$$W = \sqrt{\frac{2a}{b(2-b)p}} \ , \tag{1}$$

$$T = \frac{2-b}{2\tau_{rtt}} W \ , \tag{2}$$

where W is the congestion window size at the end of the *congestion epoch*[1], T is the sending rate, τ_{rtt} is RTT and p is the loss event rate in steady-state. Substituting $a = 1$ and $b = 1/2$ into Equations (1) and (2), we have the throughput of a standard TCP flow:

$$T_t = \frac{1}{\tau_{rtt}} \sqrt{\frac{3}{2p_t}}, \tag{3}$$

where T_t and p_t are the throughput and the loss rate of a standard TCP flow, respectively. Ideally, the throughput of a MulTCP aggregate having N flows, or a MulTCP(N) flow for short, should be N times of that of a standard TCP flow:

$$T_m = \frac{N}{\tau_{rtt}} \sqrt{\frac{3}{2p_m}}, \tag{4}$$

where T_m and p_m are the throughput and the loss rate of an MulTCP(N) flow, respectively. Assuming that $p_m = p_t$, the relationship between a and b can be derived from Equations (1)-(4) as follows:

$$b = \frac{2a}{3N^2 + a} \ . \tag{5}$$

This relationship offers a wide range of possible values for a and b for achieving various desired transient behaviors.

3.2 Loss Rate Behavior

The intention of MulTCP is to emulate the behavior of N concurrent TCP connections. Unfortunately, it has been demonstrated in [25] that a MulTCP(N) flow is more aggressive than N TCP flows. In other words, the throughput of a MulTCP(N) flow is higher than N times of that of a standard TCP flow. This is caused by lower loss rate in MulTCP. From Equations (3) and (4), it is straightforward that $T_m > NT_t$ holds if $p_m < p_t$.

It has been reported that any two competing flows sharing a same bottleneck link will experience different loss event rates if their sending rates are significantly different from each other [23]. Since the sending rate of a MulTCP(N)

[1] A congestion epoch is defined as the time interval between two successive loss event indications. It begins with a congestion window having its size of $(1-b)W$ packets, increased by a packet per RTT up to a congestion window having its size of W packets.

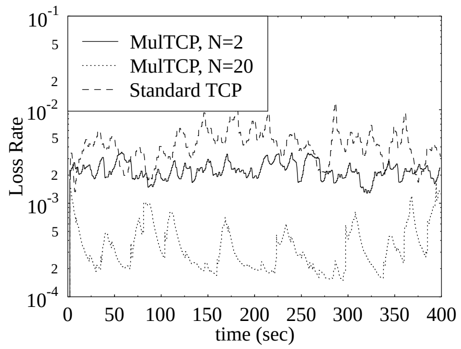


Figure 2: Loss rate experiencing by TCP and MulTCP with $N = 2$ and 20.

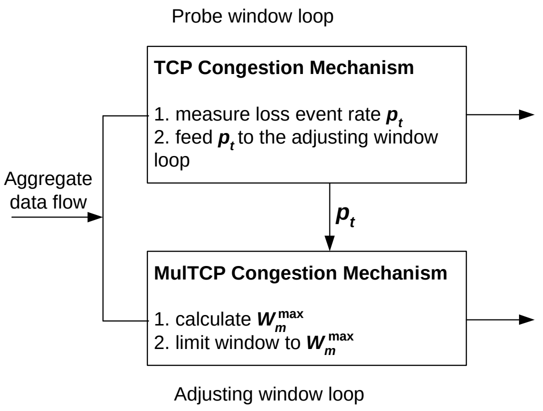

Probe window loop
TCP Congestion Mechanism
1. measure loss event rate p_t
2. feed p_t to the adjusting window loop
Aggregate data flow
p_t
MulTCP Congestion Mechanism
1. calculate W_m^{max}
2. limit window to W_m^{max}
Adjusting window loop

Figure 3: Window loop scheme in PA-MulTCP

flow is larger than that of a single TCP flow when $N > 1$, its loss rate might differ from that of a single TCP flow. This conjecture is further confirmed by our simulation implemented on ns-2 [1].

In our simulation, MulTCP is implemented based on the standard SACK TCP [18], which is available in the ns-2 distribution. A network topology having only one congested bottleneck link is used, which has 50ms delay, 400Mbps bandwidth and drop-tail queue management. The buffer size is set to the delay-bandwidth product. A MulTCP(2) flow and a MulTCP(20) flow competing with 200 standard TCP flows are simulated over a shared bottleneck link. The simulation time is 400 seconds. The window sizes of the standard TCP flows and the MulTCP aggregates are large enough so as not to impose any constraint. The packet size is 1500 bytes.

As depicted in Figure 2, MulTCP(N) flows experience lower loss event rates than a standard TCP flow, which is similar to the results in [23]. Therefore it may be inferred that the inconsistency in loss rate is the reason for MulTCP's aggressive behavior.

4. PROBE-AIDED MULTCP DESIGN

As observed in the last section, a lower loss rate in MulTCP would result in more aggressive behavior. The rationale behind PA-MulTCP is that if MulTCP's congestion control mechanism is aware of the loss rate of a standard TCP flow, it may appropriately adjust its congestion window in order to achieve the desired throughput.

This idea is realized in PA-MulTCP by maintaining two congestion window loops, namely the probe window loop and the adjusting window loop, as shown in Figure 3. However, these two AIMD congestion window loops are maintained inside only one connection, rather than two separate connections, i.e. one standard TCP connection and one MulTCP connection. On the one hand, the probe window loop acts as a loss event rate detector, which estimates the loss event rate experienced by a standard TCP flow. The probe window loop uses the TCP-friendly congestion window adjustment [29], so that it experiences a loss event rate which approximates to that experienced by a single standard TCP flow. On the other hand, the adjusting window loop is a throughput controller. It adjusts its congestion window size with a weight N, which is similar to that in MulTCP, but under a constraint on its maximum window size. The maximum window size is calculated based on the loss event rate detected by the probe window loop. Note that the real data are sent through both window loops for minimizing the impacts caused by the extra probe window loop. The packet assignment algorithm is presented later in Section 4.3.

PA-MulTCP is designed to improve efficiency and fairness of the aggregate congestion control, while achieving the desired throughput given the weight N for an aggregate. As that in MulTCP, the aggregate congestion control in PA-MulTCP is decoupled from the other control and management functionalities of individual flows within the aggregate. Therefore, it is left entirely to the application itself or the aggregate flow management scheme to decide how the available bandwidth is actually divided among flows. This offers the potential to apply PA-MulTCP to various scenarios, such as mobile/wireless TCP proxies, edge-to-edge overlays, QoS provisioning and mass data transfers in storage networks.

4.1 Probe Window Loop

The loss event behavior of the probe window loop is critical to PA-MulTCP, since the maximum window size in the adjusting window loop is calculated from the loss event rate measured by the probe window loop. In theory, the parameter pair (a, b) for the probe window loop can be arbitrarily chosen as long as it is *TCP-friendly*, i.e. satisfying [29]:

$$b = \frac{2a}{3 + a} \ . \tag{6}$$

More details can be found in our report [17], where the effect of different values for the parameter pair (a, b) in the probe window loop was extensively investigated. As suggested in [17], AIMD$(1, 1/2)$ is used for the probe window loop in this paper.

The measurement of loss rate is also important to the probe window loop, since it determines the accuracy that the probe window loop models a standard TCP flow. It has been demonstrated that loss rate should be measured as *loss event fraction*, where a *loss event* consists of one or more packets dropped within a single RTT [10]. The average loss interval method proposed in [10] is used in PA-MulTCP for calculating the loss event rate. A loss event rate is calculated by constructing a loss history and identifying loss events. These events are then converted to a loss event rate.

But the loss event rate is calculated on the sender's side in PA-MulTCP, rather than on the receiver's side as implemented in [10]. The reason is that TCP packet losses can be detected by the sender in two ways, through either timeouts or reception of *triple-duplicate* acknowledgements, i.e. four ACKs with the same sequence number. More detailed discussion on the calculation of loss event rate can be found in [10].

4.2 Adjusting Window Loop

According to Equation (5), there are a wide range of possible candidates for (a, b) in the adjusting window loop for achieving the desired transient behavior. It has been shown that a smaller value of a may help mitigate the aggressive behavior in MulTCP [19]. More details can be found in our report [17], where the effect of different values for the parameter (a, b) pair in the adjusting window loop was extensively investigated. As suggested in [17], AIMD($\sqrt{N}$, $\frac{2}{1+3N^{1.5}}$) is used for the adjusting window loop in this paper.

Another challenge in the design of the adjusting window loop is how to adjust the congestion window size, given the loss rate estimated by the probe window loop. By rewriting Equation (1), the congestion window size in MulTCP, W_m, can be expressed as:

$$W_m = \sqrt{\frac{2a}{b(2-b)p_m}} \ . \tag{7}$$

However, if MulTCP could experience the same loss rate in standard TCP, the desired window size W_m^* in MulTCP should be:

$$W_m^* = \sqrt{\frac{2a}{b(2-b)p_t}} \ . \tag{8}$$

Since the loss rate in MulTCP is lower than that in standard TCP, i.e. $p_m < p_t$, the window size in MulTCP is larger than the desired window size, i.e. $W_m > W_m^*$.

It has been shown in [21] that when a constraint on the maximum window size W_m^{max} is imposed on the window adjustment, the windows size W_m would approximate the maximum window size W_m^{max}, i.e. $W_m \approx W_m^{max}$, provided that the original unconstrained window size is larger than the maximum window size. More detailed discussions about the effect of constraints on the congestion window size can be found in [21]. Inspired by this, we simply set the maximum window size W_m^{max} to the desired window size W_m^*, i.e.

$$W_m^{max} = W_m^* = \sqrt{\frac{2a}{b(2-b)p_t}} \ . \tag{9}$$

Provided that the parameter pair (a, b) satisfies Equation (5), the throughput in the adjusting window loop approaches N times of that in a standard TCP flow:

$$T_m \approx \frac{2-b}{2\tau_{rtt}} W_m^{max} = NT_t. \tag{10}$$

Both the parameter pair (a, b) and the maximum window size W_m^{max} in the adjusting window loop are updated immediately, when a flow arrives or leaves in PA-MulTCP. If the current slow-start threshold (ssthresh) is larger than W_m^{max}, it has to be reduced to W_m^{max} as well.

4.3 Packet Assignment

The packet assignment algorithm for mapping a sequence number to a congestion window loop is also important in PA-MulTCP. It affects the number of loss events that are experienced by both the probe window loop and the adjusting window loop. Two assignment algorithms are considered in this paper.

- Consecutive Assignment. There are two congestion window loops in PA-MulTCP, which are denoted as i and j, respectively. A group of consecutive sequence numbers are assigned to the same window loop. For example, k packets with consecutive sequence numbers $[s, s+1, ..., s+k-1]$ are sent to loop i, until loop i is closed. Then the next packet with the sequence number $s+k$ is sent to the other loop j.
- Interleaving Assignment. Sequence numbers are interleaved and sent to two window loops alternately. In other words, if the packet with the sequence number s is sent to loop i, the next packet with the sequence number $s+1$ is sent to the other loop j if loop j is open. Otherwise packet $s+1$ is sent to loop i.

The choice of the packet assignment algorithm might need to take into account the influence of drop-tail queue management. Usually multiple packets are lost when the queue overflows. Typically the congestion window size is halved in most TCP implementations, when several losses are experienced. Therefore, consecutive assignment might be more appropriate in PA-MulTCP for reducing the correlation between two congestion window loops. In other words, with the consecutive assignment the probe window loop might be less influenced by the adjusting window loop. This ensures that the probe window loop experiences a loss event rate, which is similar to that experienced by a standard TCP flow sharing the same bottleneck link.

5. EVALUATION

Our simulation results are provided in this section. They are performed on ns-2 [1]. PA-MulTCP is implemented based on MulTCP, which is further based on the standard SACK TCP. A network topology having only one congested bottleneck link is used, which has 50ms delay and drop-tail queue management[2]. The buffer size is always set to the delay-bandwidth product. The access links are properly provisioned so that all packet drops/delays due to congestion occur only at the bottleneck link. Unconstrained sending and receiving windows are used on both ends of TCP and MulTCP connections, so that flow control is not involved before congestion control. The packet size is 1500 bytes.

Our simulations are performed under both stable conditions and transient congestion. The advantages of PA-MulTCP are illustrated from three aspects, namely fairness and stability in steady states, adaptivity in transient congestion, as well as impact of short-lived transfers.

[2] It has been demonstrated [13] that RED [11] could considerably improve fairness between MulTCP flows and standard TCP flows. Nevertheless, currently drop-tail is the most commonly used queue management scheme on the Internet, and this may remain true in the foreseen future. Therefore, drop-tail is used in our simulations for ensuring that PA-MulTCP provides fair sharing in the drop-tail environment.

5.1 Fairness and Stability in Steady States

As demonstrated in [25], MulTCP cannot provide fair sharing and stable connections for the aggregate. We will show that PA-MulTCP improves both fairness and stability in steady states, compared to MulTCP.

In our simulations, either a PA-MulTCP(N) or a MulTCP(N) aggregate competes with m standard TCP flows over a shared bottleneck link, where N spans from 2 to 30 while m satisfies $m + N = 200$. The bandwidth of the bottleneck link is set to 200 and 400 Mbps, respectively. The simulation time is 400 seconds. Note that the first 100 seconds are not taken into account in the results, since only the performance in steady states is concerned in this subsection. All results are obtained by averaging over 20 runs of the simulation.

We define *fairness index* as the performance metric for evaluating fairness of the long-term sending rate. It is the average throughput of N flows within the aggregate divided by the average throughput of m background standard TCP flows:

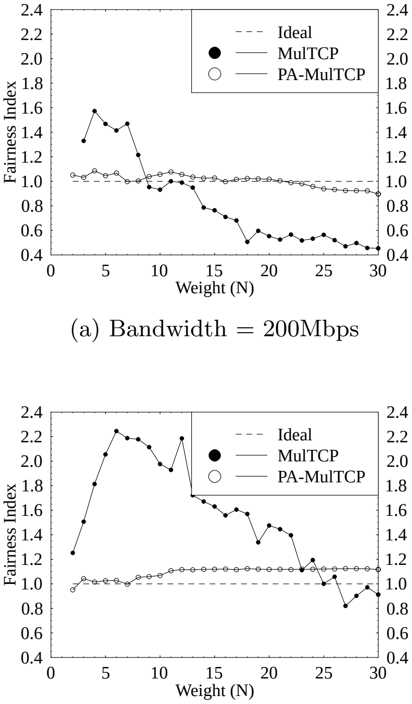
$$\text{Fairness Index} = \frac{\dfrac{T}{N}}{\dfrac{1}{m}\displaystyle\sum_{i=1}^{m} T_{tcp_i}}\ , \qquad (11)$$

where T is the throughput of the aggregate and T_{tcp_i} is the throughput of the ith background standard TCP flow. If the fairness index equals 1, it means that the aggregate fairly shares the bandwidth with the other background TCP flows. If the fairness index is less than 1, the aggregate loses the bandwidth to the background TCP traffic; otherwise, the aggregate is more aggressive than the background TCP flows.

Figure 4 depicts the fairness indexes of both MulTCP and PA-MulTCP when the bandwidth of the bottleneck link is 400Mbps. The dotted line represents ideal fair sharing between the aggregate and the standard TCP flows, i.e. fairness index equals 1. As shown in Figure 4(b), the fairness index of MulTCP is larger than 1 when $N \leq 25$, i.e. the MulTCP aggregate is more aggressive than the background TCP flows. On the other hand, the MulTCP aggregate loses bandwidth to background traffic when $N > 25$, i.e. the fairness index is less than 1. And as depicted in Figure 4(a), the MulTCP aggregate loses bandwidth to background traffic when $N > 10$. This is due to an increased number of timeouts, induced by MulTCP's aggressive control loop for a large N. By contrast, PA-MulTCP maintains fair sharing between the aggregate and the background TCP traffic when N ranges from 2 to 30, regardless whether the bottleneck bandwidth is 200 or 400 Mbps. This infers that a PA-MulTCP aggregate can more precisely hold its fair share when sharing the same bottleneck link with other standard TCP flows. This attributes to the congestion window constrained by the maximum window size.

We use *Coefficient of Variation* (CoV) as the performance metric for measuring stability of the long-term sending rate. It is the throughput variation experienced by the aggregate over time. Let T_i be the sending rate of an aggregate during the i-th time interval. Then CoV can be expressed as:

$$CoV = \frac{\sqrt{\dfrac{1}{k}\displaystyle\sum_{i=1}^{k}(T_i - \overline{T})^2}}{\overline{T}}\ , \qquad (12)$$



(a) Bandwidth = 200Mbps

(b) Bandwidth = 400Mbps

Figure 4: Fairness index under different weight N.

where $\overline{T}$ is the throughput averaged over time and k is the number of time intervals. The time interval is 0.1 second in our simulations.

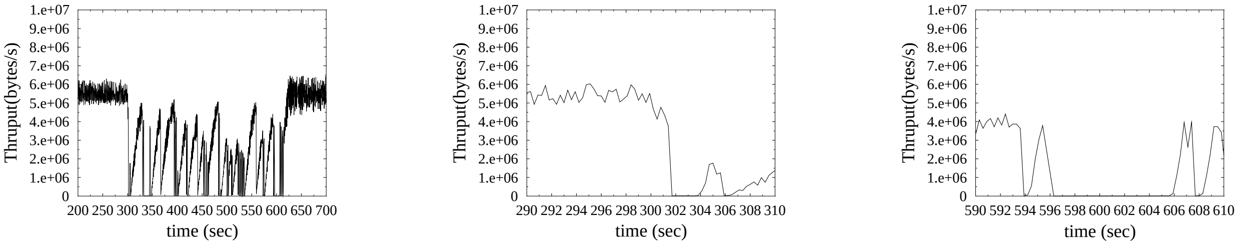
Figure 5 shows that the CoV of both MulTCP and PA-MulTCP. The CoV is higher in both schemes, either when the bottleneck bandwidth is smaller or the weight N is larger. Nevertheless, PA-MulTCP always achieves lower CoV than MulTCP over the range of the measured weight N, as shown in Figure 5. This is probably because that PA-MulTCP less likely results in bursts of losses than MulTCP, when there is either a larger weigh N or smaller network bandwidth. Frequent bursts of losses may incur frequent timeout events, which makes MulTCP more difficult for the congestion window to stay close to its average size.

In summary, the above results demonstrate that PA-MulTCP is more stable and fairer than MulTCP over a wide range of the weight N at the cost of an extra probe window loop.

5.2 Adaptivity in Transient Congestion

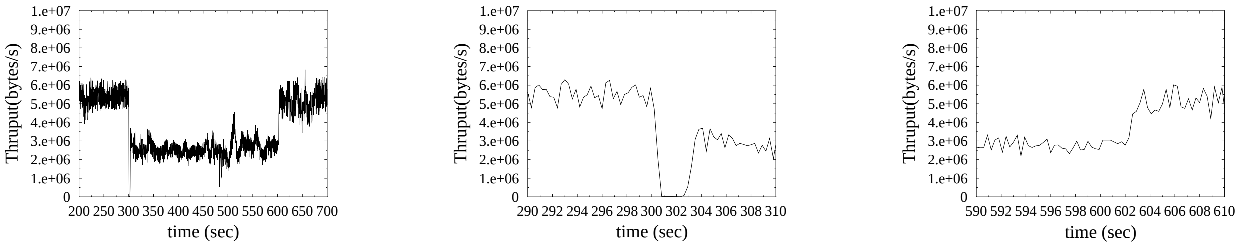
A good aggregate congestion control mechanism should be robust to changes in congestion conditions. We follow a procedure for observing responsiveness and aggressiveness of both MulTCP and PA-MulTCP to these changes. In this procedure, the network congestion level is increased first, then it is decreased later. We will show that PA-MulTCP is able to detect changes and adjust its window promptly when the congestion situation changes.

In our simulations, either a PA-MulTCP(10) or MulTCP(10) aggregate competes with 20 standard TCP flows. The bottleneck link has 120Mbps bandwidth and 50ms delay. The simulation time is 900 seconds. Additional congestion traf-



(a) UDP traffic introduced at t=300s and then removed at t=600s. (b) UDP traffic is introduced at t=300s. (c) UDP traffic is removed at t=600s.

Figure 6: Behavior of MulTCP(N=10) with increased/decreased congestion traffic induced by UDP flows. The backgound traffic is 20 standard TCP flows and the bottleneck bandwidth is 120Mbps.



(a) UDP traffic is introduced at t=300s and then removed at t=600s. (b) UDP traffic is introduced at t=300s. (c) UDP traffic is removed at t=600s.

Figure 7: Behavior of PA-MulTCP(N=10) with increased/decreased congestion traffic induced by UDP flows. The background traffic is 20 standard TCP flows and the bottleneck bandwidth is 120Mbps.

fic is introduced by a Constant Bit Rate (CBR) flow in the background between t=300 and 600 seconds.

As shown in Figures 6(b) and 7(b), the bandwidth in PA-MulTCP and MulTCP is reduced in about 7-8 RTTs (700-800ms) and 17-18 RTTs (1700-1800ms), respectively. It is straightforward that PA-MulTCP responds more promptly to the increased congestion level than MulTCP. Although both MulTCP and PA-MulTCP over-react to the increased congestion, i.e. both sending rates are dropped to almost zero, PA-MulTCP resumes the desired throughput in a shorter time than MulTCP.

As seen in Figure 7(c), after the background CBR traffic is removed, PA-MulTCP immediately utilizes the released bandwidth and claims its fair share of the released bandwidth. On the contrary, it takes longer time for MulTCP to react, as shown in Figure 6(c), because it is not stable and experiences a lot of timeouts.

Furthermore, as depicted in Figure 6(a) MulTCP is unstable when the network is congested, since it suffers from frequent timeouts. This instability prevents its connection from reaching steady states. By contrast, during the time of greater congestion PA-MulTCP not only achieves the desired throughput promptly, but also maintains stability as shown in Figure 7(a).

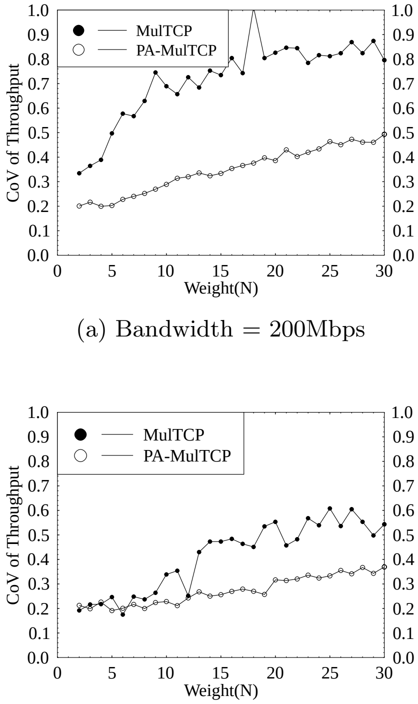
5.3 Impact of Short-Lived Transfer

Nowadays Internet traffic is heavily represented by short-lived connections [8]. Therefore it is important to investigate the impact of such flows on the aggregate congestion control. Since both MulTCP and PA-MulTCP use one adjusting window loop as the throughput controller, they may behave similarly on short-lived transfers. Hence we will compare the impact of short-lived transfer on PA-MulTCP to that on MPAT in this subsection.

Separate window loops are used independently in MPAT for ensuring fairness. Unfortunately this results in competition among different congestion window loops. By contrast, the congestion information is integrated in both MulTCP and PA-MulTCP by using one adjusting window loop as the throughput controller. Such integration is particularly effective for short-lived flows joining the aggregate, since it alleviates the initialization overhead for obtaining the network condition. We will show that PA-MulTCP supports short-lived transfer better than MPAT.

In our simulations, either a PA-MulTCP(4) or MPAT(4) aggregate competes with 20 standard TCP flows. The bottleneck link has 5Mbps bandwidth and 50ms delay. The setting of the initial ssthresh in the window loop varies from 4 to 64 KB[3]. An additional flow is introduced into the aggregate at t=50 seconds, while the transfer size varies from 10 to 400 KB. The completion time of a file transfer in the newly arriving flow is used as the performance metric for measuring the impact of short-lived transfer. For each simulation configuration, the average completion time over 20 runs of the simulation is reported. In addition, the overheads

[3] Currently, the initial ssthresh in TCP is set to an arbitrary default value, ranging from 4 to 64 KB, which depends on the implementation of operating systems.



(a) Bandwidth = 200Mbps

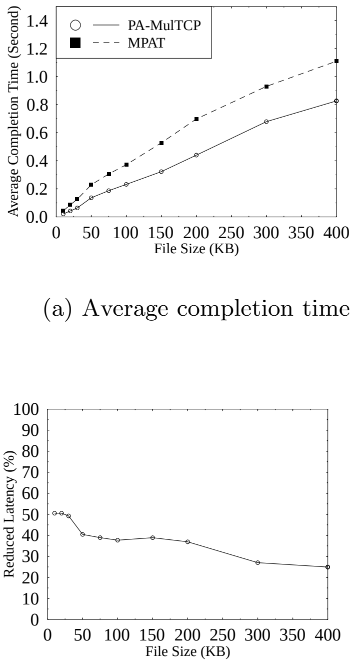
(b) Bandwidth = 400Mbps

Figure 5: CoV of throughput under different weight N.

of handshake at flow arriving the aggregate is removed. We believe it is necessary for us to better understand the performance impact of the slow start procedure in the window loop.

Figure 8 compares the average completion time of file transfers between PA-MulTCP and MPAT, when the transfer size varies from 10 to 400 KB while the initial ssthresh is fixed at 32 KB, which is a default value widely used in many TCP implementations. As shown in Figure 8, PA-MulTCP significantly reduces the average completion time by $25 - 50\%$, compared to MPAT. The improvement decreases as the transfer size increases. This follows the intuition that small transfers benefit more from avoiding the slow start penalty than large ones.

Figure 9 compares the average completion time of file transfers between PA-MulTCP and MPAT, and the transfer size is fixed at 30 and 400 KB, respectively. The initial ssthresh varies from 4 to 64 KB. The AIMD algorithm is adopted in both PA-MulTCP and MPAT for adjusting their window loops. The setting of the initial ssthresh in the AIMD algorithm affects the performance in the slow-start phase. If the ssthresh is high, the exponentially increased window loop generates lots of packets in a short time, causing multiple losses and timeouts. On the other hand, a low initial ssthresh causes premature termination of the slow start phase, resulting in poor startup utilization. Since the performance of short-lived transfers is dominated by the slow-start phase, it is necessary to investigate the influence of the initial ssthresh setting on short-lived transfers. As shown in Figure 9, the performance of short-lived transfers in MPAT is affected by the initial setting of ssthresh, i.e. the higher the initial ssthresh, the shorter the average completion time. This is because that MPAT creates a new window loop for the arriving flow and initiates the new window loop from the slow start phase. However, in PA-MulTCP, an adjusting window loop is maintained through the lifetime of the aggregate, which avoids the slow start penalty when a new flow arrives. As a result, the performance of short-lived transfers in PA-MulTCP does not vary with the different settings of the initial ssthresh.



(a) Average completion time

(b) Reduced Latency

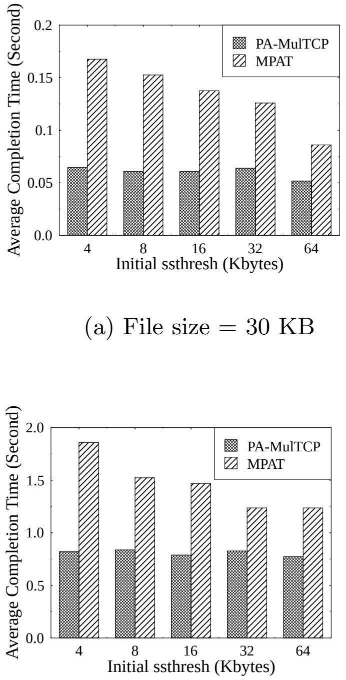
Figure 8: File transfer performance for different transfer sizes. The bottleneck bandwidth is 5 Mbps.

6. CONCLUSION

Single flowshare in traditional aggregate congestion control mechanism results in unfairness between the aggregate and the background flows, which are sharing the same bottleneck. Although MulTCP, MPAT and CP have been proposed for supporting multiple flowshares, none of them can achieve fairness while remaining lightweight and responsive. In this paper PA-MulTCP is proposed for meeting these challenges.

Our ns-2 simulation results have demonstrated that PA-MulTCP can is more stable and fairer than MulTCP over a wide range of the weight N. Moreover, when the congestion condition changes, PA-MulTCP can adjust the bandwidth promptly and maintain proportional bandwidth within the aggregate. Finally, PA-MulTCP integrates congestion information, and thus reduces the latency of short-lived transfers by $25 - 30\%$, compared to MPAT which manages a number of independent congestion windows.

PA-MulTCP is designed for decoupling the aggregate congestion control from the control and management function-



(a) File size = 30 KB

(b) File size = 400 KB

Figure 9: File transfer performance achieved for different values of the initial ssthresh (slow start threshold). The bottleneck bandwidth is 5 Mbps.

alities of individual flows. Hence, PA-MulTCP could be potentially applied to a wider range of scenarios, such as mobile/wireless TCP proxies, edge-to-edge overlays, QoS provisioning and mass data transport in storage networks. As expected, there would be a number of other issues deserving further investigation in these scenarios. For example, PA-MulTCP would need to work with the per-flow congestion control mechanisms in interconnected network segments, when applied in edge-to-edge overlays.

7. ACKNOWLEDGMENTS

We would like to thank Jon Crowcroft for his insightful comments and suggestions during the development of this work. We also thank anonymous reviewers for their valuable feedback.

8. REFERENCES

[1] NS simulator, version 2.29. http://nsnam.isi.edu/nsnam/.

[2] M. Allman, V. Paxson, and W. Stevens. *TCP Congestion Control*. Internet Engineering Task Force, April 1999. RFC 2581.

[3] H. Balakrishnan, H. Rahul, and S. Seshan. An Integrated Congestion Management Architecture for Internet Hosts. In *ACM SIGCOMM*, September 1999.

[4] R. Chakravorty, S. Katti, J. Crowcroft, and I. Pratt. Flow Aggregation for Enhanced TCP over Wide-Area Wireless. In *IEEE Conference on Computers and Communications (IEEE INFOCOM 2003)*, March 2003.

[5] D. Chiu and R. Jain. Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks. *Journal of Computer Networks and ISDN*, 17:1–14, June 1989.

[6] J. Crowcroft and P. Oechslin. Differentiated end-to-end Internet services using a weighted proportional fair sharing TCP. In *ACM SIGCOMM Computer Communication Review archive*, volume 28, pages 53 – 69, July 1998.

[7] L. Eggert, J. Heidemann, and J. Touch. Effects of Ensemble-TCP. *ACM Computer Communication Review*, 30(1):15–29, January 2000.

[8] A. Feldmann, J. Rexford, and R. Cáceres. Efficient policies for carrying Web traffic over flow-switched networks. *IEEE/ACM Transactions on Networking*, 6(6):673–685, 1998.

[9] S. Floyd, M. Handley, and E. Kohler. *Problem Statement for the Datagram Congestion Control Protocol (DCCP)*. Internet Engineering Task Force, March 2006. RFC 4336.

[10] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-Based Congestion Control for Unicast Applications. In *ACM SIGCOMM*, August 2000.

[11] S. Floyd and V. Jacobson. Random Early Detection gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1:397–413, August 1993.

[12] X. Fu and J. Crowcroft. GONE: an Infrastructure Overlay for Resilient, DoS-Limiting Networking. In *ACM NOSSDAV 2006*, May 2006.

[13] P. Gevros, F. Risso, and P. T. Kirstein. Analysis of a Method for Differential TCP Service. In *IEEE Global Internet Symposium*, December 1999.

[14] J. Touch. *TCP Control Block Interdependence*. Internet Engineering Task Force, April 1997. RFC 2140.

[15] V. Jacobson. Congestion Avoidance and Control. In *ACM SIGCOMM '88*, pages 314–329, Stanford, CA, August 1988.

[16] H. T. Kung and S. Y. Wang. TCP Trunking: Design, Implementation and Performance. In *IEEE International Conference on Network Protocols*, page 222, 1999.

[17] F.-C. Kuo and X. Fu. Probe-Aided MulTCP: An Aggregate Congestion Control Mechanism. Technical Report IFI-TB-2007-01, Institute for Informatics, University of Goettingen, April 2007.

[18] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. *TCP Selective Acknowledgment Options*. Internet Engineering Task Force, October 1996. RFC 2018.

[19] M. Nabeshima. Performance Evaluation of MulTCP in High-Speed Wide Area Networks. *IEICE Transactions on Communications*, E88-B:392–396, January 2005.

[20] D. Ott, T. Sparks, and K. Mayer-Patel. Aggregate congestion control for distributed multimedia applications. In *IEEE INFOCOM '04*, March 2004.

[21] J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe. Modeling TCP Throughput: A Simple Model and its Empirical Validation. In *ACM SIGCOMM conference on Applications, technologies, architectures, and protocols for computer communication*, pages 303–314, 1998.

[22] P. Pradhan, T. cker Chiueh, and A. Neogi. Aggregate TCP congestion control using multiple network probing. In *20th International Conference on Distributed Computing Systems (ICDCS'00)*, pages 30–37, April 2000.

[23] I. Rhee and L. Xu. Limitations of Equation-based Congestion Control. In *ACM SIGCOMM*, pages 49–60, August 2005.

[24] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan. Detour: a Case for Informed Internet Routing and Transport. *IEEE Micro*, 19(1):50–59, January 1999.

[25] M. Singh, P. Pradhan, and P. Francis. MPAT: aggregate TCP congestion management as a building block for Internet QoS. In *IEEE International Conference on Network Protocols*, pages 129– 138, Oct. 2004.

[26] SNIA IP Storage Forum White Paper. iSCSI Technical White Paper. http://www.snia.org/tech_activities/ip_storage/iSCSI_Technical_whitepaper.PDF.

[27] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. *Stream Control Transmission Protocol*. Internet Engineering Task Force, October 2000. RFC 2960.

[28] L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz. OverQoS: An Overlay Based Architecture for Enhancing Internet QoS. In *Networked Systems Design and Implementation (NSDI)*, March 2004.

[29] Y. R. Yang and S. S. Lam. General AIMD congestion control. In *IEEE International Conference on Network Protocols*, page 187, 2000.