A Solution to MPTCP's inefficiencies under the Incast problem for data center networks

Link:

Link to published version in Science Direct

Document Version:

Preprint submitted to Elsevier

A Solution to MPTCP's inefficiencies under the Incast problem for data center networks

Morteza Kheirkhah[a], Myungjin Lee[b]

[a]*Department of Electronic & Electrical Engineering, University College London, UK*
[b]*School of Informatics, University of Edinburgh, UK*

Abstract

In recent years several multipath data transport mechanisms, such as MPTCP and XMP, have been introduced to effectively exploit the path diversity of data center networks (DCNs). However, these multipath schemes have not been widely deployed in DCNs. We argue that two key factors among others impeded their adoption: *TCP incast* and *minimum window syndrome*. First, these mechanisms are ill-suited for workloads with a many-to-one communication pattern, commonly found in DCNs, causing frequent TCP incast collapses. Second, the syndrome we discover for the first time, results in 2-5 times lower throughput for single-path flows than multipath flows, thus severely violating network fairness.

To effectively tackle these problems, we propose AMP: an adaptive multipath congestion control mechanism that quickly detects the onset of these problems and transforms its multipath flow into a single-path flow. Once these problems disappear, AMP safely reverses this transformation and continues its data transmission via multiple paths. Our evaluation results under a diverse set of scenarios in a fat-tree topology with realistic workloads demonstrate that AMP is robust to the TCP incast problem and improves network fairness between multipath and single-path flows significantly with little performance loss.

Keywords: Data Center Network, Multipath Transport, Congestion Control, Flow Scheduling

1. Introduction

Data center is a crucial infrastructure that drives the Internet today. A large-scale data center comprises hundreds of thousands of servers, and hosts a diverse set of online services that require high bandwidth, low latency or both from the network. To meet such requirements, lots of recent advances [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13] have focused on improving TCP congestion control (CC) algorithms, by leveraging path diversity [3, 4, 5, 6], exploiting explicit congestion signals from network switches [1, 5, 7, 13, 8], measuring delays [11, 2], etc.

In this paper we focus on striking a right balance between throughput and latency at transport layer. To that end, one seemingly natural way is to combine a multipath tranport protocol (*e.g.*, MPTCP [3]) and a low-latency tranport protocol (*e.g.*, DCTCP [1]) in that the former usually achieves a high throughput and the latter keeps switch buffer occupancy low by exploiting Explicit Congestion Notification (ECN). Thus, the crux of this idea is to maintain multiple subflows per connection using a multipath mechanism and each subflow runs a low-latency protocol such as DCTCP.

This makes sense because any single transport protocol is difficult to meet high-throughput and low-latency requirements. For example, Equal Cost Multi Path (ECMP) routing would be likely to cause collisons among long-lived DCTCP flows on the same link, which can degrade throughput substantially. In contrast, MPTCP is good at fast load-balancing, overcoming the shortcoming of ECMP. However, it tends to occupy switch buffers aggressively, thus hurting the performance of latency-sensitive short flows.

We conduct such integration, which we call Data Center MultiPath (DCM, in short), and examine a similar existing approach called XMP [5]. We find that both schemes can provide fast load-balancing while keeping switch buffer occupancy low. However, two major challenges —*TCP incast* and *minimum window syndrome*— still render these approaches less practical as a transport protocol for DCNs.

We carefully examine these problems when multipath schemes are in use (§3). Multiple subflows in these MPTCP variants boost the possibility of TCP incast in a many-to-one communication pattern while for example senders and a receiver are co-located in a single rack. Worse, in that setting, network resource competition between ECN-capable multipath (*e.g.*, DCM) and single-path (*e.g.*, DCTCP) flows causes a serious co-existence problem, which we name *minimum window syndrome*. Surprisingly, the syndrome consistently makes multipath flows achieve 2-5× more throughput than single-path flows, thus severely violating fairness among the TCP flows. We find out that using both multiple subflows and small ECN marking threshold is behind the syndrome.

Finally we propose AMP, an adaptive multipath congestion control algorithm that is robust to the TCP incast problem and effectively handles the minimum window syndrome with little performance compromise on both throughput and latency. In addition to good fairness and high performance, we design AMP such that it is simple enough to keep its behavior trace-

able and its overheads low, and can shift traffic quickly from congested paths to less congested paths. AMP requires none of sophisticated mechanisms such as RTT-dependent congestion window (`cwnd`) increase (in MPTCP) and dynamic `cwnd` decrease (in DCTCP).

AMP's approach is simple but effective: it simply transforms a multipath flow into a single-path flow at the onset of the problems. The key in AMP is the early detection of the problems. We leverage the fact that all subflows of a multipath flow have the smallest congestion window value, which is a good indicator that all of the subflows compete with other flows on a single link. If the minimum window state across all subflows remains for a small time period (*e.g.*, 1-3 RTTs), AMP executes this transformation by deactivating all subflows but one. If AMP no longer receives ECN-marked packets for some time period (*e.g.*, 8 RTTs), it reactivates all suspended subflows (§4). Our evaluation shows that this neat technique substantially mitigates TCP incast and improve fairness without any side-effect (§5.2).

AMP also simplifies congestion control operations, which keeps AMP easily traceable and its overheads low. AMP just increases only one window per RTT across all subflows, similar to the behavior of a single-path TCP whereas the other schemes consider RTTs of all subflows to update their `cwnd`. In response to ECN signals, AMP cuts `cwnd` by a constant factor instead of dynamically adjusting it based on the fraction of marked packets (§4.2). Our extensive evaluations in a large-scale fat-tree topology with realistic traffic matrix demonstrate that AMP under incast-like workloads works better than and in other workloads performs as well as the existing solutions, despite its simplicity (§5.3).

Overall, this paper makes three main contributions:

- To the best of our knowledge, we report for the first time that the minimum window syndrome can do exist when ECN-capable multipath and single-path TCPs are deployed in data centers. We carefully examine its root cause.
- We propose AMP[1], an adaptive multipath TCP for data center networks that effectively copes with the TCP incast and minimum window syndrome. AMP is resilient against the incast problem and ensures graceful co-existence with single-path TCP flows.
- We evaluate AMP over a wide variety of scenarios in a large-scale fat-tree topology, and demonstrate that AMP mitigates buffer inflation and achieves higher fairness and comparable performance against existing multipath protocols.

2. Preliminary

In this section we review two multipath mechanisms to facilitate our later discussions: (1) DCM, a new extension of MPTCP that combines the congestion control of MPTCP and DCTCP together; and (2) XMP, an existing proposal.

2.1. DCM

An intuitive and reasonable approach is to combine MPTCP and DCTCP. The main idea is to enable each subflow of MPTCP with the ECN response mechanism of DCTCP. On top of the basic MPTCP algorithm, which swiftly shifts traffic from highly congested to less congested paths, DCM handles ECN-marked packets similar to DCTCP for each subflow. That is, each subflow of DCM adjusts its sending rate in proportion to the extent of congestion, represented by the amount of ECN-marked packets. For instance, when a subflow rapidly reduces its `cwnd` due to receiving a large amount of ECN-marked packets over a few windows of data, DCM moves the traffic from that subflow to other subflows with better network condition (*e.g.* with larger `cwnd` and low RTT). In this way, each subflow of DCM follows DCTCP to reduce its `cwnd` and MPTCP to increase its `cwnd`.

The DCM does, in short:
- For each ACK on subflow s, $w_s \leftarrow w_s + \min(\frac{a}{w_{total}}, \frac{1}{w_s})$
- For each loss, $w_s \leftarrow \frac{w_s}{2}$
- For first marked ACK in a window, $w_s \leftarrow w_s(1 - \frac{\alpha_s}{2})$

w_s is a `cwnd` size of subflow s, w_{total} is $\sum_r w_r$ for all r and a controls the aggressiveness of `cwnd` increase across all subflows. The following formula calculates the value of a:

$$a = w_{total} \frac{\max_r (w_r/rtt_r^2)}{(\sum_r (w_r/rtt_r))^2} \qquad (1)$$

Here $\max_r$ is the maximum value across all subflows. For instance, when an MPTCP flow encounters a path with high RTT and low packet drop probability, it increases its aggressiveness to fully utilize that path. The aggressiveness is also capped by $\frac{1}{w_s}$ to prevent a subflow to increase its `cwnd` more than one segment per RTT, ensuring that a subflow, and in turn, the MPTCP flow, is not harming other competing (possibly, single-path) flows.

α_s is an estimate of the fraction of marked packets on subflow s and is updated once per window of data (roughly an RTT) as follows:

$$\alpha_s = (1 - g)\alpha_s + gF_s \qquad (2)$$

F_s is the fraction of marked packets (in the last window of data) on subflow s; g is a (constant) weight coefficient for exponentially averaging α_s. When $\alpha_s \rightarrow 0$, w_s decreases gently; as $\alpha_s \rightarrow 1$, w_s does more aggressively.

2.2. XMP

XMP is another multipath congestion control algorithm that aims to strike a balance between latency-throughput trade-offs. XMP combines an ECN-based scheme for controlling the buffer occupancy in switches and a rate-based congestion control algorithm for balancing traffic among its subflows.

The XMP does, in short:
- Every window of data on subflow s, $w_s \leftarrow w_s + \delta_s$
- For each loss, $w_s \leftarrow \frac{w_s}{2}$
- For first marked ACK in a window, $w_s \leftarrow w_s(1 - \frac{1}{\beta})$
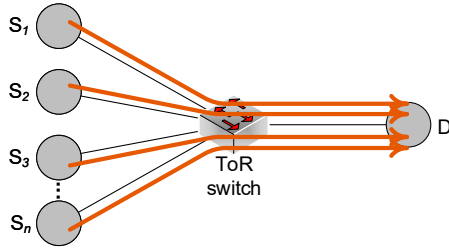

Figure 1: A many-to-one communication scenario over a 10Gbps bottleneck link.
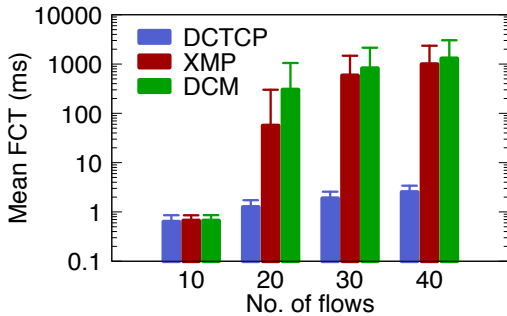

Figure 2: Impact of the TCP incast on different multipath protocols (DCM and XMP) and DCTCP. DCM and XMP use four subflows per connection. File size is 128KB, link capacity is 10Gbps, and switch buffer size is 100 packets. The y-axis is log-scaled.

δ_s dictates the amount of `cwnd` increase for each subflow, calculated once per window of data; and β is a fixed reduction factor (set to 4 in [5]). The value for δ_s is calculated by the following formula:

$$\delta_s = \frac{rtt_s}{rtt_{min}} \times \frac{w_s/rtt_s}{\sum_r (w_r/rtt_r)} \qquad (3)$$

XMP is in principle similar to MPTCP and DCM, but there are differences, too. One of them is that in XMP, network congestion is signaled via packet queuing delay (inferred through RTT) and ECN-marked packets.

3. Issues of MPTCP variants

MPTCP and its ECN-capable variants have not been widely deployed in DCNs. While there may be several other reasons, we identify two key technical issues. First, the ECN-capable MPTCP variants (DCM and XMP) are unable to handle incast-like traffic; many applications (MapReduce [15], Partition/Aggregate [1], etc.) have a many-to-one communication pattern that is prevailing in DCNs. Second, the ECN-capable MPTCP variants fail to gracefully coexist with an ECN-capable single-path TCP such as DCTCP; an MPTCP variant can hurt DCTCP flows' throughput significantly. We call this problem the minimum window syndrome. In what follows, we demonstrate the impact of these two problems via simulation under a simple topology shown in Figure 1.

3.1. TCP incast

TCP incast can happen in applications which have barrier-synchronized workload and a high fan-in communication pat-

tern. A unique characteristic of the barrier-synchronized workload is that traffic of multiple TCP flows to the same destination arrives in a bursty fashion at a bottleneck switch, which has a shallow buffer. This causes bursty packet losses and eventually triggers expensive timeout at TCP senders, which substantially delays the completion of a job [1].

TCP incast is a well-studied topic [1, 10] and for instance DCTCP mitigates the problem using ECN. Unfortunately, the ECN-capable MPTCP variants are still susceptible to the TCP incast even in the help of ECN. To demonstrate that, we create a simulation environment as shown in Figure 1 using NS-3 [16]. The simulation setup is as follows. Every 1 second k number of multipath flows join to a bottleneck link with a fixed interval of $50\mu s$ where $k = 10, 20, 30, 40$ while setting the flow size to 128KB. Each simulation lasts for 20 sec. Each multipath flow has 4 subflows. In the setup, we test DCM and XMP. We also separately run DCTCP as baseline.

Figure 2 shows that DCTCP overall outperforms DCM and XMP. In many cases the average flow completion time (FCT) of DCTCP is almost 1-2 orders of magnitude shorter than that of DCM and XMP; when $k = 30$, the average FCT of DCTCP is about 2ms whereas that of DCM and XMP is over 800ms. Furthermore, the FCT distribution of DCTCP has a narrow standard deviation (*i.e.*, the whisker bar in the graph), but the standard deviation of XMP and DCM is large (less than 1 millisecond for DCTCP vs. above 1 second for DCM and XMP). This means that the other schemes have a long-tailed FCT distribution and make some flows experience much higher FCTs (due to retransmission timeouts).

From these results, it is evident that the multipath variants cannot handle the TCP incast problem. The reason is somewhat obvious. The MPTCP variants maintain 4 subflows. Hence, one multipath flow generates at least 4 packets per RTT. More number of multipath flows implies a sharp increase of the probability of burst packet losses. For example, every RTT 30 multipath flows shown in Figure 2 generate at least 120 packets, which are far exceeding the queue length (*i.e.*, 100 packets in this case) of the bottleneck switch.

Without giving up the benefit of a multipath protocol, a (practical and possibly natural) way to deal with this problem may be allowing both multipath and single-path protocols and letting them share the DCN fabric. The basic idea is to permit DCTCP for latency-sensitive applications and multipath protocols for bandwidth-hungry services. However, keeping graceful co-existence of these two different protocols turns out to be challenging, which we discuss next.

3.2. Minimum window syndrome

In the presence of ECN-capable multipath (*e.g.*, DCM and XMP) and single-path TCP flows (*e.g.*, DCTCP), serious unfairness between them can occur. The key characteristic of this problem is that when all the subflows of DCM or XMP flows compete with DCTCP flows on the same bottleneck link, DCTCP flows obtain 2-5 times less amount of bandwidth and higher queueing delay than they should. We call this co-existence problem the minimum window syndrome.

(a) Normal situation (b) Persistent buffer inflation (c) Minimum window syndrome
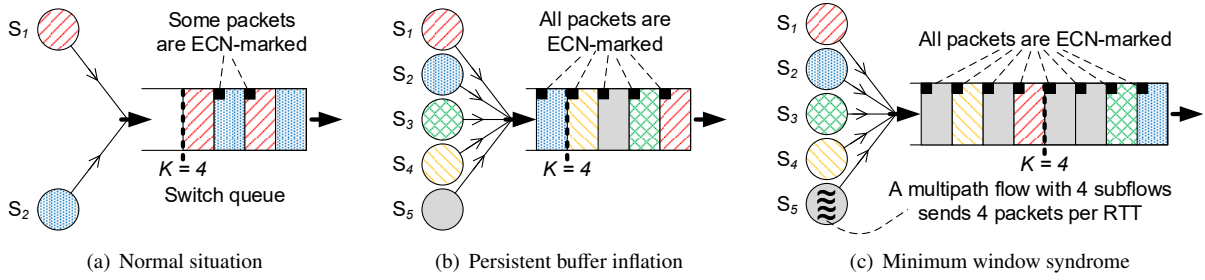
Figure 3: Illustration of the minimum window syndrome. The syndrome leads to severe unfairness and escalates the likelihood and impact of persistent buffer inflation significantly.

Triggering the syndrome. We discuss when the syndrome can occur through examples shown in Figure 3. We assume that network switches ECN-mark packets only if their instant queue length is larger than a marking threshold K. Such switches are widely deployed in data centers. To keep the discussion simple, let us assume $K = 4$ and zero propagation delay. That is, as soon as packet leaves the queue, sender can send a new packet as it receives acknowledgment instantly.
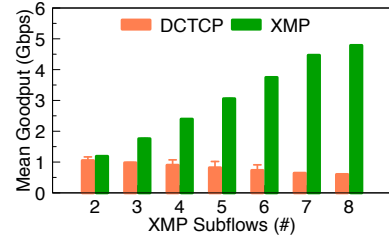
In Figure 3(a), two single-path flows share the bottleneck link fairly by generating on average two packets per RTT (bounded by queuing delay); in other words, cwnd in each flow oscillates between 1-3 packets.

Now suppose that 5 single-path flows compete with each other as illustrated in Figure 3(b). Because $K = 4$, a new arriving packet finds the queue length is always equal to K, meaning that it is the 5th packet in the queue. Thus, all packets across flows are ECN-marked all the time, and each flow is forced to reduce its cwnd to one packet. Even though there is no way to further reduce cwnd (as it is one packet), the number of packets arriving at the queue always exceeds K. This causes *persistent buffer inflation* (also discussed in [17]), but there is no unfairness across flows.

Finally, Figure 3(c) illustrates a case where the single-path flow in S_5 is replaced with one multipath flow having four subflows. Similar to the previous case, all packets across flows are ECN-marked. Even if the cwnd of the single-path flows and all subflows reduces to one packet, the number of packets in flight far exceeds K all the time. However, since all the subflows belong to one multipath flow, the flow ends up sending four times more packets than single-path ones. Furthermore, the syndrome substantially escalates the likelihood and impact of the persistent buffer inflation (see the buffer length twice as large as K in Figure 3(c)), which can potentially harm latency-sensitive short flows.

Conditions for the syndrome. In reality, BDP needs to be considered and is a few tens of packets in DCNs [5, 13]. Thus, to create the syndrome, more than $(BDP + K)$ number of flows are necessary. However, the MPTCP variants set *minimum congestion window size* ($cwnd_{min}$)—an internal constant that governs the minimum number of packets a sender can send regardless of congestion level—to two packets[2]. Thus, the number

(a) $(K, r) = (10, \{2 \ldots 8\})$. Average goodput of 8 DCTCP flows and one XMP flow.
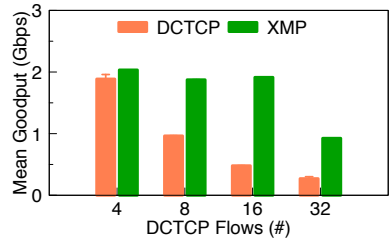
(b) $(K, r) = (20, 4)$. Average goodput of varying number of DCTCP flows and one XMP flow.

Figure 4: The minimum window syndrome under various conditions. K: ECN marking threshold, and r: the number of subflows.

of flows including single-path flows and subflows in multipath flows should be larger than $(BDP + K)/cwnd_{min}$.

Demonstration of the syndrome. Now suppose a setup in Figure 1 where an ECN-enabled switch connects n sending servers and one receiving server. The receiver is equipped with DCTCP, DCM and XMP; server S_1 runs DCM (or XMP) having r subflows, and the remaining $n - 1$ servers with DCTCP ($n \geq 2$). Those n senders send traffic to the receiver.

We do various simulations by varying parameters and study the impacts of the syndrome. We change the simulation duration from 10ms to 1 sec, use 1Gbps and 10Gbps link and test both DCM and XMP. Across these variations, we observe a very similar trend. Thus we only show the results of the 1 sec duration over 10Gbps link using XMP in interest of space. We depict a setting as (K, r) where K is the ECN marking threshold and r is the number of subflows.

Varying number of subflows: Given 8 DCTCP flows and one

---

[2]MPTCP and XMP use two packets to probe congestion level on each path (see a detailed discussion in [18] and Algorithm 1 in [5]). DCTCP also uses

two packets originally, but a recent study proposed to use one packet for the value (see page 11 in [19]) and the DCTCP source was patched accordingly. Unless otherwise stated, we set $cwnd_{min} = 2$ for consistency in this paper.

DCM or XMP flow, we vary the number of subflows from 2 to 8, while setting $K = 10$ and $cwnd_{min} = 2$ as suggested in [5]; thus, the setting is $(10, \{2 \ldots 8\})$.

Figure 4(a) shows that the syndrome begins as soon as the DCM or XMP flow starts to use three subflows or more. When four subflows are used, the XMP flow obtains 2.3× higher goodput than DCTCP flows. The figure clearly demonstrates that the number of subfows is a key factor that triggers the problem. DCTCP flows seem to have no problem in the 2-subflow case. However, the problem recurs when at least about 16 DCTCP flows are in use (not shown for brevity). Worse, using two subflows costs about 10% goodput loss (*e.g.*, 1Gbps out of 10Gbps rate) when compared to using four subflows [5]. Also, a number of subflows (*e.g.*, 8 subflows) are in general beneficial when there exist lots of parallel paths in a large DCN [3]. Thus, using a smaller number of subflows is not a fundamental solution.

Different marking threshold: As a small marking threshold can be a potential cause of the problem, increasing K may be useful. However, this can also introduce an additional delay, which might hurt the flow completion time of latency-sensitive short flows. Nevertheless, we test $K = 20$. With the setting (20, 4), we vary the number of DCTCP flows.

Figure 4(b) shows that increasing the marking threshold marginally alleviates the problem; given 8 DCTCP flows, a goodput gap between DCTCP and DCM or XMP is a factor of two. In contrast, recall that the gap is a factor of 2.3 under the same condition in Figure 4(a). We also tested a case where $cwnd_{min} = 1$ while keeping the setting as (10, 4). This reduced the likelihood of the syndrome, but we observed that a slight increase of the number of DCM or XMP flows (from 1 to 4) triggered the syndrome, when 8 DCTCP flows are given (the exact graph is omitted).

Summary. We obtain two key findings from these results. First, the condition that triggers the minimum window syndrome is relatively simple: the total number of packets in flight from both multipath and single-path flows should exceed BDP plus K frequently. In our setup, BDP is 20 packets. In Figure 4(a), the condition begins to hold when the setting has 3-4 subflows for the XMP flow and 8 DCTCP flows (the average number of packets in flight is about 30-32). Second, tweaking those parameters either alleviates the problem marginally or makes performance loss inevitable.

4. Design

We propose AMP, a multipath congestion control mechanism that coexists well with ECN-capable single-path TCPs and is resilient against TCP incast. In designing AMP, in addition to the obvious objectives—high throughput and low latency, we have the following design objectives:

- *Good fairness:* Multipath and single-path TCP flows should be able to achieve their fair share of bandwidth at a bottleneck link, even in the presence of an incast-like traffic pattern.
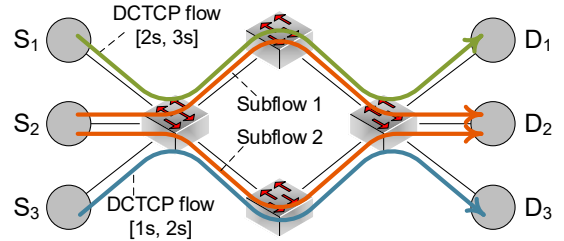

Figure 5: A setup for testing traffic shifting time. An orange line represents a subflow of a multipath flow.

- *Fast traffic shifting:* Multipath flows should be able to avoid congested paths quickly. This especially helps latency-sensitive short single-path TCP flows experience less impact due to congestion.

- *Simplicity:* An algorithm should be kept as simple as possible so that its behaviors are easily analyzed and its overheads are kept low.

To achieve the above objectives, we deliberately test existing solutions: DCM and XMP. In analyzing them, we make several key observations essential for our design.

4.1. Key observations

(1) The number of subflows for a multipath flow should not be static. Multiple subflows are in general beneficial to obtain high throughput. On the contrary, in the presence of the TCP incast and minimum window syndrome, it is effective to have a smaller number of subflows (ideally, one subflow), as discussed in §3. However this costs throughput performance. Thus having the static number of subflows can only achieve either good fairness against single-path flows or high throughput, but not both of them. Thus, the number of subflows should be adjusted adaptively, which can be done by (de)activating subflows in an online fashion. However, it is inappropriate to deactivate subflows incrementally because mitigating the two problems can take too long, which may cause significant queuing delay to latency-sensitive short flows over a longer period of time.

(2) The `cwnd` values in subflows are a cue for the TCP incast and syndrome. Detecting these problems early is key to adjusting the number of subflows. We notice that when these problems are about to occur, subflows are in a unique status where the `cwnd` values across all subflows are always equal to a minimum (*e.g.*, two packets in [5, 3]). This is a good indicator that these problems are in effect because it is unlikely that all subflows of a multipath flow passing through different paths face excessive congestion, especially in a large-scale data center that has 100s of parallel paths between a pair of source and destination.

(3) Adaptive cutback of `cwnd` at subflow slows down traffic shifting. One of differences between DCM and XMP is the response mechanism to ECN-marked packets. In DCM a subflow cuts its `cwnd` in proportion to the fraction of marked packets over a window (see Eq. (2)); whereas in XMP a subflow decreases its window by a constant factor β (see §2.2). To understand the effect of this difference, we modify MPTCP to
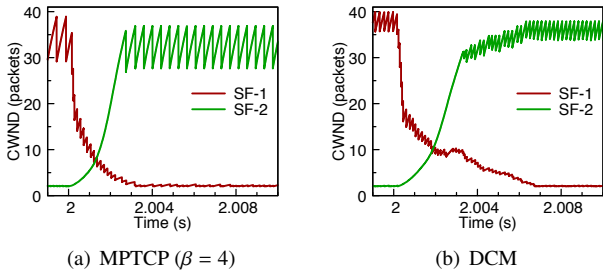
5

Figure 6: Traffic shifting times of MPTCP and DCM. MPTCP finishes its traffic shifting at 2.003s and DCM does at 2.007s; DCM is 4ms slower than MPTCP.

reduce the `cwnd` of subflow by a constant factor (we use $\beta = 4$) when it sees ECN-marked packets and examine traffic shifting times for MPTCP with $\beta = 4$ and DCM.

Given a topology shown in Figure 5, a multipath flow (DCM or MPTCP) with two subflows begins to traverse from S_2 to D_2 at 0s. Then, S_3 sends traffic to D_3 using DCTCP within interval (1s, 2s) and another DCTCP flow from S_1 to D_1 for (2s, 3s). At 2s, a multipath flow is sending its entire traffic through the upper path and we plot how `cwnd` of each subflow varies within interval (1.999s, 2.01s) after the second DCTCP flow appears on the upper path. Figure 6 shows that MPTCP finishes traffic shifting about 4ms faster than DCM.

The reason is because a DCTCP subflow in DCM conservatively reduces `cwnd` based on the fraction of marked packets. Hence, even if there exists a congestion-free path, a DCM flow shifts its traffic slowly. In contrast, with a constant factor (*e.g.*, $\beta = 4$), MPTCP is aggressive enough to make a subflow on the congested path quickly reduce its window, thereby achieving faster traffic shifting than DCM. This conservative nature of DCTCP perfectly makes sense if a flow traverses one path only. However, because the subflows of a multipath flow travel through multiple different paths in general, it is more appropriate to get rid of traffic from the congested path rather than to withstand against congestion.

(4) RTT measurements of subflows are unnecessary for updating their `cwnd`. Interestingly, both DCM and XMP rely on RTT measurements in increasing `cwnd` of subflows. DCM inherits MPTCP's design principles, one of which targets to address the RTT mismatch issue [18] that can occur when there are paths with high RTT and low loss probability and paths with low RTT and high loss probability. However, higher RTT typically means large queuing delay and hence high loss probability in DCNs because DCNs usually have a symmetrical structure where all paths between a pair of servers have the same length. Thus, DCNs have no paths that cause the RTT mismatch problem.

Moreover, ECN tends to equalize RTTs throughout the data center network when network switches react to instant queue length with a small ECN marking threshold [1, 5]. Assuming 5-hop paths with 10Gbps links, 10 packets of marking threshold and 1500B packets, a maximum RTT difference is just about $108\mu s$. In average cases, as the utilization of network links increases, the RTT difference will become even smaller. Thus, differentiating the sending rate of each subflow based on such

---

Algorithm 1: Pseudocode of AMP

```
 1  /* Subflow suppression/release         */
 2  SuppressSubflows(nRound)
 3      nSF = 0      /* counter for subflows */
 4      for subflow s ∈ [1,...,n] do
 5          if w_s = cwnd_min then nSF ← nSF + 1
 6      end
 7      if nSF = n then nRound ← nRound + 1
 8      else nRound ← 0
 9      if nRound < γ then return
10      for subflow s ∈ [2,...,n] /* at γ rounds  */
11      do
12          active_s ← false
13      end
14  ReleaseSubflows(ACK, nRound)
15      if ACK.marked then nRound ← 0
16      else nRound ← nRound + 1
17      if nRound < τ then return
18      for subflow s ∈ [2,...,n] /* at τ rounds  */
19      do
20          active_s ← true
21      end
22  /* RTT-agnostic CWND increase           */
23  IncreaseCWND(s, w_total)
24      /* For each ACK of subflow s          */
25      w_s ← w_s + 1/w_total
26  /* Constant factor CWND decrease to ECN  */
27  RespondToECN(s)
28      /* For the first marked ACK of subflow s
           per window                         */
29      w_s ← max (w_s(1 − 1/β), cwnd_min)
30  /* Response to duplicate ACKs           */
31  DecreaseCWND(s)
32      w_s ← max (w_s/2, cwnd_min)
```

a small RTT difference would not bring much benefit. Even in a case that a path is highly congested, sources can quickly identify it with ECN signals and do traffic shifting accordingly.

4.2. AMP algorithm

We now discuss the exact algorithm of AMP designed with the above four observations. AMP mainly consists of three components: (i) subflow suppression/release, (ii) constant factor decrease of congestion window, and (iii) RTT-agnostic congestion window increase.

The subflow suppression/release is a key mechanism that ensures graceful coexistence between multipath and single-path flows. The second component enables fast traffic shifting. The final part, as its name suggests, excludes RTT measurements, without any performance penalty, from the part of increasing `cwnd`, which overall makes our algorithm simple. Algorithm 1 shows the pseudocode of AMP, that we explain next in detail.

Subflow suppression/release (SSR). The SSR mechanism permits detection of cases where all subflows belonging to an AMP

flow struggle at the same bottleneck link due to congestion. A representative example is a many-to-one communication pattern (*e.g.*, incast) where multiple flows (and subflows) compete for bandwidth at a last mile hop (*i.e.*, ToR switch). Upon detection, AMP transforms its flow to a single-path flow. Once congestion disappears, AMP converts its flow from a single-path flow to a multipath one.

Subflow suppression consists of two steps: detection and suppression. (1) At detection step, AMP checks whether the `cwnd` of all its subflows has been equal to a minimum window size for γ number of consecutive RTTs (lines 3-9 in Algorithm 1). (2) At suppression step, if the previous detection condition is met, AMP deactivates all its subflows except for the initial one by resetting `active` flag (lines 11-13).

AMP conducts subflow release similarly. If the initial subflow does not receive any more marked packets for τ number of consecutive RTTs (lines 15-17), AMP reactivates all those inactivated subflows (lines 19-21). When releasing the subflows, AMP sets `active` flag for each subflow.

Overall, while it is a simple heuristic, SSR ensures fairness between multipath and single-path flows at a shared bottleneck link. It also helps to accommodate more senders during an incast-like episode or to reduce the chance of costly timeouts. We demonstrate SSR's efficacy in §5.2.

RTT-agnostic congestion window increase. As discussed in §4.1, employing an ECN-based congestion control tends to equalize RTTs in DCNs. The difference in RTTs for paths is at most K packets where K is a small marking threshold at switches (say, 10 packets). In addition, the RTT mismatch problem does not exist in DCNs, either. Based on these insights, for each non-duplicate ACK of subflow, we simply increase its `cwnd` by $1/w_{total}$ (line 25 in Algorithm 1) where w_{total} is the total window size across all subflows. This ensures that AMP can only increase one segment per RTT across all subflows, preserving network fairness with single-path flows at bottleneck links [20, 18].

The amount of `cwnd` increase of AMP also strikes a right balance. Given an congestion control algorithm C, let the amount of `cwnd` increase of a subflow per ACK be C_{inc}. For instance, the amount, $1/w_{total}$, is AMP_{inc}.

Now suppose RTT difference among all subflows is negligible. Then, Eq. (1) for DCM reduces to $a \approx w_{max}/w_{total}$ where w_{max} is the maximum window size across all subflows. The increasing amount per ACK is then about $w_{max}/(w_{total})^2$ which we call DCM_{inc}. In case of XMP, Eq. (3) reduces to $\delta_s \approx w_s/w_{total}$. Note that δ_s is the amount of `cwnd` increase per RTT in XMP. Since w_s is the current window size of subflow s, the subflow would receive w_s number of ACKs. Thus, for every ACK, XMP increases `cwnd` of a subflow by $1/w_{total}$, which is XMP_{inc}. Putting it together, we have

$$DCM_{inc} \le AMP_{inc} \approx XMP_{inc}$$

Note that if w_{max} approaches w_{total}, $DCM_{inc} \approx AMP_{inc}$. Looking at these relationships among three algorithms, the increment is comparable across all of them, but AMP's algorithm is much simpler than the other two.

	cwnd increase	Response to ECN	SSR
AMP	$w_s + 1/w_{total}$	$w_s(1 - 1/\beta)$	✔
DCM	$w_s + \min(\frac{a}{w_{total}}, \frac{1}{w_s})$ a as in Eq. (1)	$w_s(1 - \alpha_s/2)$ α_s as in Eq. (2)	✘
XMP	$w_s + \delta_s$ δ_s as in Eq. (3)	$w_s(1 - 1/\beta)$	✘

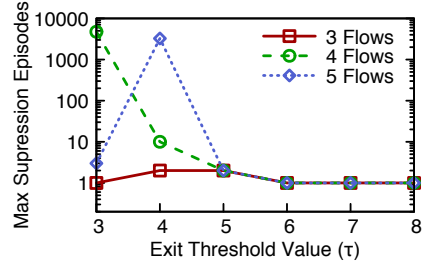Table 1: Summary on AMP, DCM and XMP.

Figure 7: The impact of τ (i.e., the exit threshold) on the number of suppression episodes.

Constant factor decrease of congestion window. In AMP a subflow responds to ECN signals once every window of data (i.e., approximately an RTT) by reducing its `cwnd` with a constant factor β, as depicted at line 29 of Algorithm 1. The parameter β should be determined such that a link is fully utilized. In other words, a queue should not be completely drained due to `cwnd` reduction. In [5], this problem of choosing β is formulated as follows:

$$\frac{BDP + K}{\beta} \le K,$$

Note $\beta \ge 2$; otherwise, it reduces `cwnd` more aggressively than a standard TCP. We choose β using this formula. For instance, consider a DCN where each link has 1Gbps speed and RTT is about $250\mu s$ [1] (i.e., BDP is about 20 packets). If we set $K = 10, \beta \ge 3$. Since computing BDP even for other link speed (e.g., 10Gbps) is easy, it is straightforward to set β after K is first determined.

Summary. Table 1 highlights key mechanisms of AMP, DCM and XMP. From the table, we see that AMP is much simpler than other solutions, easing the tuning of AMP. A key differentiator is the subflow suppression/release mechanism that mitigates the TCP incast and minimum window syndrome.

5. Evaluation

In this section we evaluate AMP via extensive simulations using NS-3 [14]. For comparison, we use DCTCP, DCM and XMP[3]. We first study how to tune the parameters of AMP. We then examine AMP under a few basic scenarios. In particular, we will answer robustness of AMP against the TCP incast, its

---

[3]We do not use any ECN-incapable TCP because it does not coexist with ECN-capable TCPs at all [19].
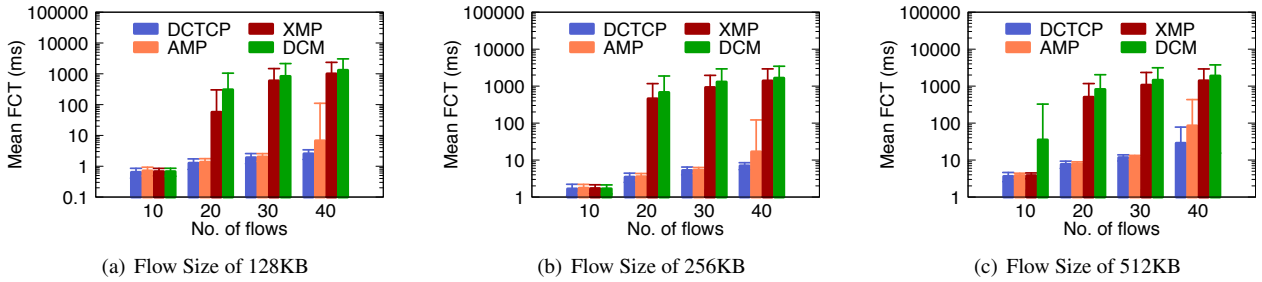
Figure 8: Impact of the TCP incast on different multipath protocols. A multipath protocol (AMP, XMP and DCM) is only used to transfer the incast traffic. A whisker bar denotes standard deviation. The y-axis is log-scaled.

effectiveness to the minimum window syndrome, and its speed in traffic shifting. We finally study the overall performance of AMP under a large-scale fat-tree topology that represents a realistic data center network.

Basic configuration. Throughout our simulations, the following parameters are used without any change: (i) a link rate of 10Gbps, (ii) a link delay of $2\mu s$, (iii) an MSS of 1400 bytes, (iv) a maximum queue size of 100 packets, and (v) $\beta = 4$ for AMP and XMP. We also tested AMP over 1Gbps settings and observed that the trends were similar to those of 10Gbps settings. We only show the results under the 10Gbps settings in interest of space.

We set a default value for each of the following parameters: (i) the number of subflows per multipath flow = 4, (ii) the minimum congestion window size, $cwnd_{min} = 2$ packets, and (iii) the ECN marking threshold, $K = 10$ packets. When necessary (*e.g.*, for further analysis), we change their values.

Evaluation metrics. We have four key metrics: Jain's fairness index [21], goodput, flow completion time (FCT) and job completion time (JCT). We define JCT as a time period until all flows in a job finish their transmission from its beginning.

5.1. Parameter tuning

The subflow suppression/release (SSR) mechanism has two parameters: γ to begin the subflow suppression process and τ to finish it. We empirically determine γ and τ.

First, setting γ is relatively easy; we test different γ values (1-10 RTTs) in the presence and absence of the TCP incast and minimum window syndrome. If there indeed exist the two problems in the network, it is important to begin the suppression process early enough to alleviate their impact quickly. When $\gamma \geq 3$ (in RTTs), AMP reacts these problems slowly. For instance, under the same setting for the TCP incast shown in Figure 2, average FCT of AMP, when $\gamma = 3$, is an order of magnitude higher than that of DCTCP. When $\gamma = 1$, there is a chance of false alarm. We find AMP performs best when $\gamma = 2$, which we use by default.

Second, setting τ (*i.e.*, the exit threshold) should be more cautious. The risk involved in selecting τ is oscillation. If τ is too small, AMP will repeatedly begin and end the suppression process. The frequent oscillation may be synchronized across AMP flows, which subsequently causes faster queue build-up due to traffic bursts when all suspended subflows across flows

are reactivated simultaneously. This may make all incoming packets ECN-marked, which in turn leads to the repetition of the whole suppression process by suspending all subflows.

To find a suitable value for τ, we conduct simulations while varying τ and using 3-5 AMP flows under the topology shown in Figure 1. Figure 7 depicts how many suppression episodes happen across different τ values and the number of flows. Ideally, there must be only one episode. However, when $\tau <$ 6, there are more than one episode; moreover, the count of episodes varies a lot when the number of AMP flows is different. When $\tau \geq 6$, the SSR mechanism becomes stable (meaning that there is only one episode), and the median queue length is just about 10 packets. Therefore, we set $\tau = 8$ as default (to be conservative).

5.2. Microbenchmarking

Robustness against the TCP incast. Multipath congestion control mechanisms usually work poorly when they are used for traffic that is short-lived and has a high fan-in pattern (*e.g.*, TCP incast). To understand how well AMP tolerates such a traffic pattern, we use the same simulation setup used in §3.1. That is, there is no mix of single-path and multipath flows; we use multipath protocols only to transfer high fan-in short-lived traffic. This time we vary file size from 128KB to 1MB. DCTCP is used again as baseline.

AMP is as good as DCTCP apart from a case of 40 flows, in which AMP performs slightly worse than DCTCP (Figure 8). However, AMP outperforms XMP and DCM; in most cases the average FCT of AMP is almost 1-2 orders of magnitude shorter than that of XMP and DCM. For instance, Figure 8(a) shows that when the number of flows is 30 and flow size is 128KB, the FCT of AMP is about 2ms and that of XMP and DCM is over 800ms. In addition, AMP has a narrow standard deviation in its FCT distribution, but XMP and DCM have a large standard deviation (1-2ms for AMP vs. 1 second for XMP and DCM). This confirms that AMP presents a stable FCT performance even under various TCP incast scenarios. Note that the y-axis of the graph is presented in log scale.

The SSR mechanism in AMP mitigates the possibility of buffer overflow significantly, thus that of the expensive TCP timeout. When the number of flows is 30 in Figure 8(a), we observe that AMP has no timeout during the simulation whereas XMP and DCM face up to 10 and 16 timeouts respectively (with 7 timeouts at 90th percentile for both schemes). Notably,

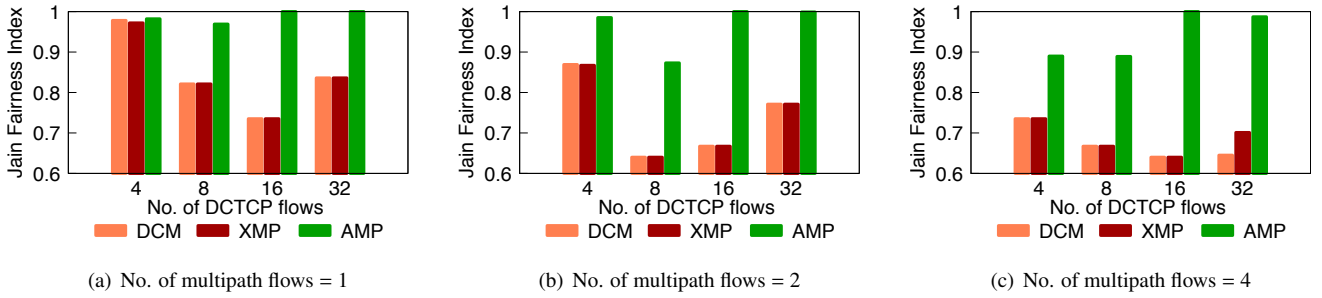(a) No. of multipath flows = 1   (b) No. of multipath flows = 2   (c) No. of multipath flows = 4

Figure 9: Fairness obtained when a multipath scheme (AMP, XMP and DCM) competes with DCTCP flows under the minimum window syndrome. Each multipath flow generates 4 subflows. AMP outperforms XMP and DCM.

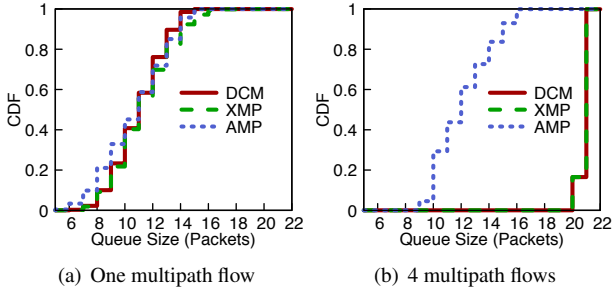(a) One multipath flow   (b) 4 multipath flows

Figure 10: Queue length distribution. (a) no syndrome: 4 DCTCP flows and one multipath flow. (b) intensive syndrome: 4 DCTCP flows and 4 multipath flows.

(a) AMP   (b) XMP

Figure 11: Traffic shifting speed.

when flow size is smaller than 128KB (*e.g.*, 64KB), all multipath schemes work as well as DCTCP and there is little difference among the three multipath approaches; even the SSR mechanism is not triggered at all as the flow size is too small. Thus, if the flow size is at least as large as 128KB, our approach would work better than others as we observe a similar trend for a flow size of 1MB (graph omitted).

Effectiveness to the minimum window syndrome. We use the topology shown in Figure 1 and test the impact of the syndrome on different schemes while varying the number of DCTCP flows and multipath flows. All flows arrive at 0 sec and end at 1 sec.

Figure 9 shows that in almost all cases AMP outperforms the other two schemes. As the number of multipath flows increases, we find the syndrome aggravates fairness even in the presence of a small number of DCTCP flows (cf., two cases of 4 DCTCP flows between Figures 9(a) and 9(c)). Note that the syndrome itself is weak in some cases (*e.g.*, given less than 8 flows in Figure 9(a) and 4 flows in Figure 9(b)); thus marginal difference in fairness is observed among the three schemes.

Since the syndrome causes persistent buffer inflation, we examine queue length. From Figure 10, we make two observations. When there is no syndrome (Figure 10(a)), the queue length distributions across AMP, XMP and DCM are similar. On the other hand, when 4 DCM and XMP flows are used (Figure 10(b)), the queue length is more than 20 packets (100% inflation at median) all the time. On the contrary, the queue length difference of AMP is just about 2 packets (at median, 10 packets in Figure 10(a) and 12 packets in Figure 10(b)). If the intensity of the syndrome grows, the queue length will be-
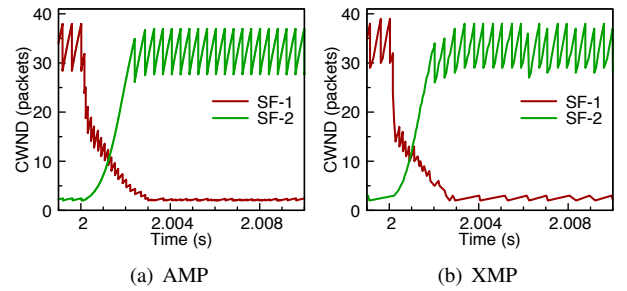
come more inflated accordingly. In general AMP can mitigate the persistent buffer inflation better than other schemes even if the syndrome is more intensive.

The SSR mechanism is key to high performance in both fairness and delay. When we disable SSR, all three approaches present equally poor performance.

Traffic shifting speed. We now evaluate how quickly AMP shifts traffic. We run the same simulation done in §4.1 under the setup in Figure 5. Recall that DCM's traffic shifting time is about 7ms. Figure 11 shows that both AMP and XMP achieve similar traffic shifting time (about 3ms), reassuring DCTCP's slowness in shifting traffic and suggesting that a fixed amount of congestion window reduction to ECN signals is suitable for multipath congestion control mechanisms.

5.3. Large-scale simulation

We now study the overall performance of AMP with different workloads in a realistic data center setup. As many data center networks employ a multi-rooted tree topology [22, 23, 24], we use a 3-tier fat-tree topology that has 128 servers, 32 ToR, 32 aggregate and 16 core switches. ECMP routing is employed to select a path on a per-flow basis.

Incast with background traffic. We aim to examine the performance of a high fan-in workload (*i.e.*, incast traffic) in the presence of background traffic. Specifically, we use DCTCP to generate the incast traffic and a multipath protocol for the background traffic. Note that this scenario is different from one in §5.2 where a multipath protocol is used to transfer the incast traffic.

Setting: We consider a scenario where a client makes parallel reads in a cluster filesystem in the presence of background traffic. We model this as a unit of job: a client sends a 2KB request
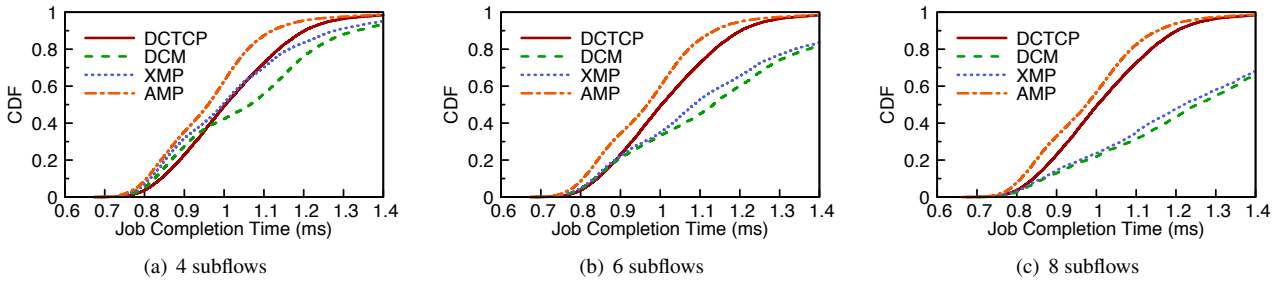
(a) 4 subflows   (b) 6 subflows   (c) 8 subflows

Figure 12: Job completion time of incast workloads. DCTCP is used for generating incast traffic and background traffic is generated by using AMP, DCM, XMP and DCTCP (baseline) separately. A key in the legend denotes the protocol name used for background traffic.

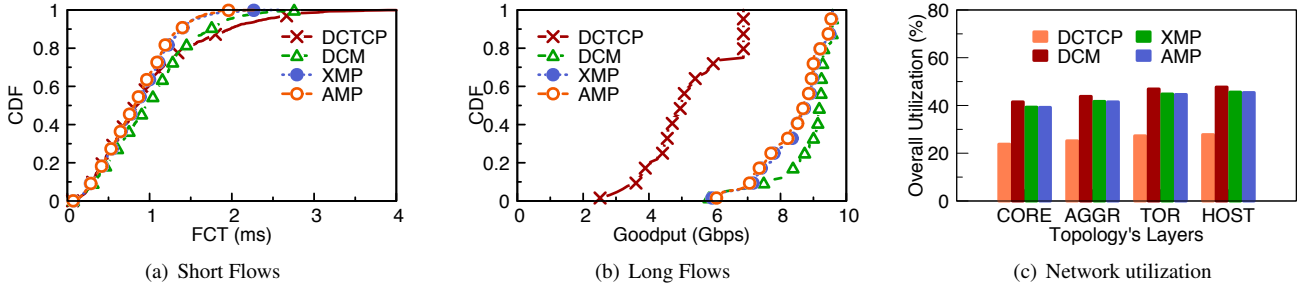(a) Short Flows   (b) Long Flows   (c) Network utilization

Figure 13: FCT, goodput and network utilization performance. Short flows are only generated by DCTCP, and each protocol in the legend is used for long flows. Thus, in (a), FCT is for short DCTCP flows given a different protocol for long flows. Similarly, (b) presents goodput of each protocol used for long flows.

to 10 servers, each of which in turn sends back a 64KB block of response data to the client. One job ends after receiving all blocks. Thereafter a new job begins. There are 8 parallel jobs, and clients and servers in each job are randomly selected. Each host sends a long flow to a randomly selected host to generate traffic on background. The flow size is determined by a Pareto distribution with shape parameter of 1.5 and mean of 192MB. Once a long flow ends, a new one begins immediately. A simulation continues until 1000 long flows are completed.

Results: Figure 12 presents job completion times of short DCTCP flows. Notice from the figure that a key in the legend is the protocol name used for long flows. We plot, as a baseline, the case where DCTCP is also used for long flows. Overall, we make two observations.

First, AMP does not harm short DCTCP flows even if multipath flows use as many as 8 subflows. The results in Figure 12 show that the AMP case (*i.e.*, AMP is used for long flows) obtains slightly better JCT performance than the baseline case across all scenarios.

Second, more number of subflows in DCM or XMP makes the JCT of short DCTCP flows grow quickly. When XMP is used for long flows, the 90th percentile JCT is 1.2ms in the 4-subflow case (Figure 12(a)), 1.5ms in the 6-subflow case (Figure 12(b)), and 1.8ms in the 8-subflow case (Figure 12(c)). In contrast, the 90th percentile JCT is 1.1ms in case where AMP even has 8 subflows, thus reducing JCT by 0.6ms (39% improvement) compared to the corresponding XMP case.

Summary: From the above observations, we conclude that our SSR mechanism reduces buffer inflation effectively and hence makes competing short DCTCP flows finish faster. While we recommend 4 subflows per multipath flow, AMP may safely support up to 8 subflows.

General workload. We now study interaction between short and long flows. Our goal here is to confirm that, despite its simplicity, AMP works as well as other schemes and its SSR mechanism brings no harm.

Setting: 50% of the servers run long flows, and the remaining servers generate short flows scheduled by a Poisson flow arrival with rate $\lambda = 256$ flows/s. Those long flows last for 10 sec to increase chance of saturating the network. The size for short flow is chosen between 1KB and 1MB at uniformly random. We only present results of cases where short flows use DCTCP and long flows use a multipath protocol because other combinations (*e.g.*, DCTCP for long flows and a multipath protocol for short flows) that we tested make no significant difference in performance compared to a base case where both short and long flows use DCTCP only.

We use permutation traffic matrix that has been used in many previous works [25, 5, 3, 26, 27, 28]. Specifically, a host establishes at most two connections: one for receiving traffic and the other for sending traffic. For sending traffic, the host chooses its receiver at random.

Results: Figure 13(a) shows the FCT results of different schemes. A key in the legend denotes a protocol used for long flows. We observe that short DCTCP flows achieve the best FCT result when AMP and XMP are used for long flows. The worst FCT performance was observed when DCTCP is used for long flows because DCTCP suffers from poor ECMP load-balancing. In case of long DCM flows, DCTCP flows work reasonably well and about 10% of the short DCTCP flows obtain longer FCTs than DCTCP flows competing with long AMP flows. This is because DCM's traffic shifting is slower than that of AMP, thus causing a high queuing delay.

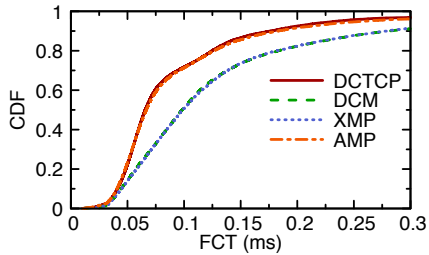The long flows of AMP and XMP show little difference in

Figure 14: The FCT of short DCTCP flows with data mining workload used in pFabric [29]. If a flow size < 100KB, the flow is short; otherwise, it is considered long. The short and long flows coexist. A key in the legend denotes a protocol used for long flows.

goodput. Interestingly, DCM achieves the best performance, which is because it trades (queuing) delay for goodput. Again, using DCTCP for long flows yields the worst goodput performance due to the same reason in the FCT case.

Figure 13(c) shows the mean network utilization at all layers of the fat-tree topology. As expected, the multipath schemes perform equally well because they balance their load among multiple paths.

We also examine those schemes in a fat-tree topology with 128 servers using a realistic data mining workload [29] and under an intensive condition that an average inter-flow arrival time in each server is about $780\mu s$. Figure 14 depicts the FCT of short DCTCP flows when another type of protocol is used for long flows. When AMP is used for long flows, the 90th percentile FCT of short DCTCP flows is 0.18ms, but the corresponding FCT for XMP and DCM is 0.27ms and 0.28ms respectively (~55% improvement over the two schemes). We observe a similar level of improvement at the 99th percentile: 0.78ms for AMP, 1.35ms for XMP and 1.41ms for DCM. Note that the goodput distribution of long flows across all schemes is almost identical (graph omitted).

Summary: Our results suggest that ECN alone can signal network congestion fast enough for multipath congestion control and considering RTT as part of an ECN-based multipath congestion control brings little benefit in the current setting of DCNs. The results also confirm that a fixed amount of congestion window reduction to ECN signals enables faster traffic shifting than adjusting the window dynamically. Thus AMP is light and as good as other solutions.

6. Related Work

Pathological congestion events. TCP incast [30] and TCP outcast [31] are well-known pathological TCP problems in DCNs. The TCP incast is a congestion collapse incident for TCP flows that belong to barrier-synchronized workloads with a high fan-in traffic pattern. A bursty packet arrival overflows shallow switch buffer, leading to expensive TCP timeouts. In contrast, in the TCP outcast, when a few flows from one input port compete for an output port with many flows from another input port, the few flows are penalized more severely. These problems are fundamentally different from the minimum window syndrome studied in this work. While those problems require

switch buffer overflow, the syndrome does not; but it results in severe unfairness between ECN-capable single-path and multipath TCP flows.

Multipath congestion control. A transport layer protocol that exploits multiple paths between source and destination has been an active area of research [32, 12, 33, 3, 5, 34, 35]. MPTCP [3] divides a TCP flow into multiple subflows. Since those subflows may take different paths, MPTCP shifts traffic between its subflows to avoid congested paths. XMP [5] is similar to MPTCP, but it leverages ECN to maintain low buffer occupancy. MMPTCP [4] uses a packet scattering technique to improve delay performance of short flows while it acts as a regular MPTCP for long flows. Unfortunately, these schemes fail to handle TCP incast and minimum window syndrome. On the contrary, AMP is simpler and handles those problems better than these schemes.

ECN-based congestion control. In data centers, many ECN-based proposals adopt instant queue length based ECN. As one of the earlier works in this category, DCTCP [1] reacts to the extent of congestion estimated from the fraction of marked packets. D2TCP [7] and L2DCT [8] build upon DCTCP; D2TCP focuses on decreasing the likelihood of missed deadlines for TCP flows, and L2DCT aims to reduce FCT for short flows. ECN*[13] proposes dequeue marking to improve the performance of both short and long flows. Since a small threshold is used in all of these schemes, they can be prone to the minimum window syndrome when ECN-capable multipath protocols are deployed together.

Delay-based congestion control. Delay-based congestion control mechanisms had continuous attention in the past for the Internet [36, 37, 38] and wireless networks [39, 40]. Lately, refreshed interest in those mechanisms has grown in the context of DCNs [11, 2]. DX [11] and TIMELY [2] measure queuing delays at the microsecond granularity, and use the measurements to keep buffer occupancy low. These approaches are single-path protocols; they may suffer from poor load-balancing of ECMP as DCTCP does. Thus, it would be of interest to extend AMP for delay-based schemes.

Scheduling, prioritization and load balancing. A large body of work focuses on scheduling and prioritization [29, 41, 42, 28, 43, 44], or load balancing [45, 46] to support low latency in DCNs. For scheduling and prioritization, some rely on priority queuing with multiple queues [43, 44]; others conduct decentralized [29, 28] or centralized scheduling [41]; one combines different strategies adopted in prior works [42]. Load balancing schemes [45, 46] break a flow down into small groups of packets, which are in turn distributed across multiple paths. In general, these approaches may be useful to mitigate the TCP incast and minimum window syndrome.

7. Conclusion

In this paper we presented that existing multipath congestion control mechanisms fail to handle (1) the TCP incast problem that causes temporal switch buffer overflow due to synchronized traffic arrival; and (2) the minimum window syndrome

that causes persistent buffer inflation and serious unfairness. To overcome the limitation of the existing solutions, we proposed AMP that adaptively switches its operation between a multiple-subflow mode and single-subflow mode. Our extensive evaluation results showed that AMP is simple yet effective to those problems and in general works well, which makes deploying AMP in data centers attractive.

References

[1] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, M. Sridharan, Data Center TCP (DCTCP), in: ACM SIGCOMM, 2010.

[2] R. Mittal, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, D. Zats, TIMELY: RTT-based Congestion Control for the Datacenter, in: ACM SIGCOMM, 2015.

[3] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, M. Wischik, D.and Handley, Improving Datacenter Performance and Robustness with Multipath TCP, in: ACM SIGCOMM, 2011.

[4] M. Kheirkhah, I. Wakeman, G. Parisis, MMPTCP: A Multipath Transport Protocol for Data Centers, in: IEEE INFOCOM, 2016.

[5] Y. Cao, M. Xu, X. Fu, E. Dong, Explicit multipath congestion control for data center networks, in: ACM CoNEXT, 2013.

[6] G. Chen, Y. Lu, Y. Meng, B. Li, K. Tan, D. Pei, P. Cheng, L. Luo, Y. Xiong, X. Wang, Y. Zhao, Fast and Cautious: Leveraging Multi-path Diversity for Transport Loss Recovery in Data Centers, in: USENIX ATC, 2016.

[7] B. Vamanan, J. Hasan, T. Vijaykumar, Deadline-aware Datacenter TCP (D2TCP), in: ACM SIGCOMM, 2010.

[8] A. Munir, I. A. Qazi, Z. A. Uzmi, A. Mushtaq, S. N. Ismail, M. S. Iqbal, B. Khan, Minimizing flow completion times in data centers, in: IEEE INFOCOM, 2013.

[9] S. Lee, M. Lee, D. Lee, H. Jung, B. Lee, TCPRand: Randomizing TCP payload size for TCP fairness in data center networks, in: IEEE INFOCOM, 2015.

[10] H. Wu, Z. Feng, C. Guo, Y. Zhang, ICTCP: Incast congestion control for TCP in data-center networks, IEEE/ACM Transactions on Networking (ToN) 21 (2) (2013) 345–358.

[11] C. Lee, C. Park, K. Jang, S. Moon, D. Han, Accurate latency-based congestion feedback for datacenters, in: USENIX ATC, 2015.

[12] M. Zhang, J. Lai, A. Krishnamurthy, L. L. Peterson, R. Y. Wang, A transport layer approach for improving end-to-end performance and robustness using redundant paths., in: USENIX ATC, 2004.

[13] H. Wu, J. Ju, G. Lu, C. Guo, Y. Xiong, Y. Zhang, Tuning ECN for Data Center Networks, in: ACM CoNEXT, 2012.

[14] M. Kheirkhah, I. Wakeman, G. Parisis, Multipath-TCP in ns-3, in: Workshop on NS3, 2014.

[15] J. Dean, S. Ghemawat, MapReduce: Simplified Data Processing on Large Clusters, in: USENIX OSDI, 2004.

[16] Network Simulator 3, https://www.nsnam.org/, last checked: 2017-01-27.

[17] M. Alizadeh, A. Javanmard, B. Prabhakar, Analysis of DCTCP: Stability, Convergence, and Fairness, in: ACM SIGMETRICS, 2011.

[18] D. Wischik, C. Raiciu, A. Greenhalgh, M. Handley, Design, Implementation and Evaluation of Congestion Control for Multipath TCP, in: USENIX NSDI, 2011.

[19] G. Judd, Attaining the Promise and Avoiding the Pitfalls of TCP in the Datacenter, in: USENIX NSDI, 2015.

[20] F. Kelly, T. Voice, Stability of End-to-end Algorithms for Joint Routing and Rate Control, SIGCOMM Comput. Commun. Rev. 35 (2) (2005) 5–12.

[21] R. Jain, A. Durresi, G. Babic, Throughput Fairness Index: An Explanation, aTM Forum/99-0045 (1999).

[22] M. Al-Fares, A. Loukissas, A. Vahdat, A Scalable, Commodity Data Center Network Architecture, in: ACM SIGCOMM, 2008.

[23] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, S. Sengupta, VL2: A Scalable and Flexible Data Center Network, in: ACM SIGCOMM, 2011.

[24] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, et al., Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network, ACM SIGCOMM Computer Communication Review 45 (4) (2015) 183–197.

[25] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, A. Vahdat, Hedera: Dynamic Flow Scheduling for Data Center Networks, in: USENIX NSDI, 2010.

[26] P. Costa, H. Ballani, D. Narayanan, Rethinking the Network Stack for Rack-scale Computers, in: USENIX HoCloud, 2014.

[27] M. Kheirkhah, MMPTCP: A Novel Transport Protocol for Data Centre Networks, Ph.D. thesis, University of Sussex (2016).

[28] P. X. Gao, A. Narayan, G. Kumar, R. Agarwal, S. Ratnasamy, S. Shenker, pHost: Distributed Near-optimal Datacenter Transport Over Commodity Network Fabric, in: ACM CoNEXT, 2015.

[29] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, S. Shenker, pFabric: Minimal Near-optimal Datacenter Transport, in: ACM SIGCOMM, 2013.

[30] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, B. Mueller, Safe and Effective Fine-grained TCP Retransmissions for Datacenter Communication, in: ACM SIGCOMM, 2009.

[31] P. Prakash, A. Dixit, Y. C. Hu, R. Kompella, The TCP Outcast Problem: Exposing Unfairness in Data Center Networks, in: USENIX NSDI, 2012.

[32] H.-Y. Hsieh, R. Sivakumar, pTCP: An end-to-end transport layer protocol for striped connections, in: IEEE ICNP, 2002.

[33] J. R. Iyengar, P. D. Amer, R. Stewart, Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths, IEEE/ACM Transactions on networking (ToN) 14 (5) (2006) 951–964.

[34] M. Kheirkhah, I. Wakeman, G. Parisis, Short vs. Long Flows: A Battle That Both Can Win, in: ACM SIGCOMM, 2015.

[35] A. Dixit, P. Prakash, Y. C. Hu, R. R. Kompella, On the Impact of Packet Spraying in Data Center Networks, in: IEEE INFOCOM, 2013.

[36] L. S. Brakmo, L. L. Peterson, TCP Vegas: End to end congestion avoidance on a global Internet, IEEE Journal on selected Areas in communications 13 (8) (1995) 1465–1480.

[37] K. Tan, J. Song, Q. Zhang, M. Sridharan, A compound TCP approach for high-speed and long distance networks, in: IEEE INFOCOM, 2006.

[38] D. X. Wei, C. Jin, S. H. Low, S. Hegde, FAST TCP: motivation, architecture, algorithms, performance, IEEE/ACM Transactions on Networking (ToN) 14 (6) (2006) 1246–1259.

[39] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, R. Wang, TCP westwood: Bandwidth estimation for enhanced transport over wireless links, in: ACM MobiCom, 2001.

[40] C. P. Fu, S. C. Liew, TCP Veno: TCP enhancement for transmission over wireless access networks, IEEE Journal on selected areas in communications 21 (2) (2003) 216–228.

[41] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, H. Fugal, Fastpass: A Centralized Zero-queue Datacenter Network, in: ACM SIGCOMM, 2014.

[42] A. Munir, G. Baig, S. M. Irteza, I. A. Qazi, A. X. Liu, F. R. Dogar, Friends, Not Foes: Synthesizing Existing Transport Strategies for Data Center Networks, in: ACM SIGCOMM, 2014.

[43] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, H. Wang, Information-agnostic Flow Scheduling for Commodity Data Centers, in: USENIX NSDI, 2015.

[44] M. P. Grosvenor, M. Schwarzkopf, I. Gog, R. N. M. Watson, A. W. Moore, S. Hand, J. Crowcroft, Queues Don'T Matter when You Can JUMP Them!, in: USENIX NSDI, 2015.

[45] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. T. Lam, F. Matus, R. Pan, N. Yadav, G. Varghese, CONGA: Distributed Congestion-aware Load Balancing for Datacenters, in: ACM SIGCOMM, 2014.

[46] K. He, E. Rozner, K. Agarwal, W. Felter, J. Carter, A. Akella, Presto: Edge-based Load Balancing for Fast Datacenter Networks, in: ACM SIGCOMM, 2015.