AMP: An Adaptive Multipath TCP for Data Center Networks

Morteza Kheirkhah
University College London
m.kheirkhah@ucl.ac.uk

Myungjin Lee
University of Edinburgh
myungjin.lee@ed.ac.uk

***Abstract*—MPTCP and its ECN-capable variants such as XMP and DCM have recently been introduced to effectively exploit the path diversity of modern data center networks (DCNs). Although these multipath schemes improve overall network throughput compared to single-path schemes due to their fast, host-based, load balancing ability, they failed to address the following two problems: *TCP incast* and *last hop unfairness*. Firstly, these mechanisms cause frequent TCP incast collapses when used for workloads with a many-to-one communication pattern, commonly found in DCNs. Secondly, the last hop unfairness problem severely violates network fairness as single-path flows achieve 2-5 times less throughput than multipath flows.**

To effectively tackle these problems, we propose the Adaptive MultiPath (AMP) 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 data transmission via multiple paths. Our evaluation results under a diverse set of scenarios in a large-scale fat-tree topology demonstrate that AMP is robust to the TCP incast problem and improves network fairness between multipath and single-path flows significantly with no performance loss.

I. Introduction

Data centers are a crucial component of the Internet today. Large-scale data centers are comprised of hundreds of thousands of servers, and host a diverse set of online services that require high bandwidth, low latency or both from the network. To meet these requirements, a large number of recent advances [1]–[6] have focused on improving TCP congestion control (CC) algorithms, by leveraging path diversity [2]–[4], exploiting explicit congestion signals from switches [1], [4], [6], and measuring delays [7], [8].

In this paper we focus on striking a balance between throughput and latency at the transport layer. To that end, one possible idea is to combine a multipath transport protocol such as MPTCP [2], which usually achieves a high throughput, with a low-latency transport protocol such as DCTCP [1], which keeps switch buffer occupancy low by exploiting Explicit Congestion Notification (ECN). Thus, the crux of this idea is to combine a multipath mechanism that maintains multiple subflows per connection with a low-latency mechanism to govern the behavior of each subflow.

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

We examine the performance of existing works such as eXplicit MultiPath (XMP) [4] and Data Center MultiPath (DCM) [9] (our prior work), which follow such an integration. It turns out that these provide fast load balancing while keeping switch buffer occupancy low. However, they perform poorly during a *TCP incast* incident generated by a many-to-one communication workload commonly observed in DCNs.

Our examination reveals that multiple subflows in multipath flows boost the possibility of TCP incast especially when many senders and a receiver are co-located in a single rack (§III-A). Worse, in that setting, network resource competition between multipath (*e.g.*, XMP) and single-path flows (*e.g.*, DCTCP or XMP using one subflow) causes a serious co-existence problem, which we call *last hop unfairness (LHU)*. In such scenarios flows using multiple paths consistently achieve 2-5 times greater throughput than flows using single-path, thus severely violating fairness among the flows (§III-B). As a result, these problems render ECN-based MPTCP variants less practical as a transport protocol for DCNs.

To overcome these limitations, we propose the Adaptive MultiPath (AMP) congestion control algorithm that is robust to the TCP incast problem and effectively handles the LHU problem with no performance compromise. In addition, the design of AMP is simple with low overhead. The scheme moves traffic quickly from congested paths to less congested ones. AMP does not require hard to trace mechanisms such as RTT-dependent congestion window (`cwnd`) increase, as in standard MPTCP, or dynamic `cwnd` decrease, as in DCTCP.

AMP's approach is simple and effective: it transforms a multipath flow into a single-path flow at the onset of problems. The key in AMP is the early detection of the problem. We leverage the fact that all subflows of a multipath flow have the smallest `cwnd` value, which is a good indicator that all of the subflows are competing 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 (§IV). Our evaluation shows that this neat technique greatly mitigates TCP incast and improves fairness with no side-effects (§V-B).

AMP also simplifies congestion control operations, which keeps AMP easily traceable and its overheads low. AMP just increases one full-sized segment per RTT across all subflows, similar to the behavior of single-path TCP, whereas the other

ISBN 978-3-903176-16-4 © 2019 IFIP

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 (§IV-B). Our evaluations in a large-scale fat-tree topology with widely used traffic matrices demonstrate that AMP under incast-like workloads performs better than the existing solutions, despite its simplicity (§V-C).

Overall, this paper makes the following main contributions:

- To the best of our knowledge, we explore for the first time the shortcomings of employing ECN-based variants of MPTCP under incast-like workloads. This includes the discovery of the LHU problem. Note that all multipath schemes considered in this paper are ECN-based (unless otherwise stated) because other schemes (e.g. standard MPTCP) do not co-exist with ECN-based schemes at all [10].
- We propose AMP, an adaptive (ECN-capable) multipath congestion control algorithm for data center networks that effectively copes with the TCP incast and ensures graceful co-existence with single-path 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 schemes. Note that we have implemented AMP on top of our custom implementation of MPTCP in NS-3 [11]. Our code can be found in [12] with required instructions to reproduce all results presented in this paper.

II. Preliminary

A. Data Center TCP

DCTCP strives to maintain a low buffer occupancy in switches. DCTCP employs an ECN-based CC mechanism that adjusts its sending rate in proportion to the extent of congestion, represented by the amount of ECN-marked packets. **In short, the behaviour of DCTCP is:**

- For each ACK, $w \leftarrow w + \frac{1}{w}$
- For each loss, $w \leftarrow \frac{w}{2}$
- For first marked ACK in a window, $w \leftarrow w(1 - \frac{\alpha}{2})$

α is an estimate of the fraction of marked packets and is updated once per RTT as follows:

$$\alpha = (1-g)\alpha + gF \qquad (1)$$

F is the fraction of marked packets; g is a weight coefficient for exponentially averaging α. When $\alpha \rightarrow 0$, w decreases gently; as $\alpha \rightarrow 1$, w does more aggressively.

B. eXplicit MultiPath

XMP is a multipath CC algorithm that aims to strike a balance between latency-throughput trade-offs. XMP combines an ECN-based scheme for controlling buffer occupancy in switches and a rate-based CC algorithm for balancing traffic among its subflows. **In short, the behaviour of XMP is:**

- 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, $w_s \leftarrow w_s(1 - \frac{1}{\beta})$

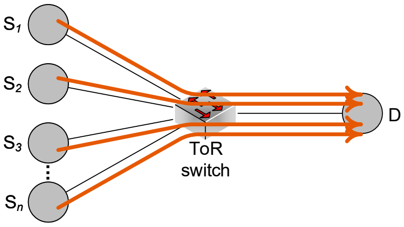


Fig. 1. A many-to-one communication scenario over a 10Gbps link.

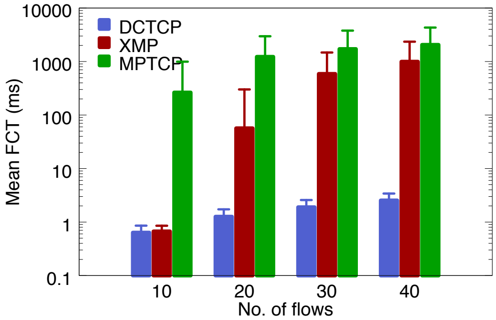


Fig. 2. Impact of the TCP incast on XMP, DCTCP and standard MPTCP. XMP and MPTCP use 4 subflows. File size is 128KB, link rate is 10Gbps, and switch buffer size is 100 packets. The y-axis is log-scaled.

δ_s dictates the amount of cwnd increase for each subflow, calculated once per window of data; and β is a fixed reduction factor. The value for δ_s is calculated by the following formula:

$$\delta_s = \frac{rtt_s}{rtt_{min}} \times \frac{w_s/rtt_s}{\sum_r (w_r/rtt_r)} \qquad (2)$$

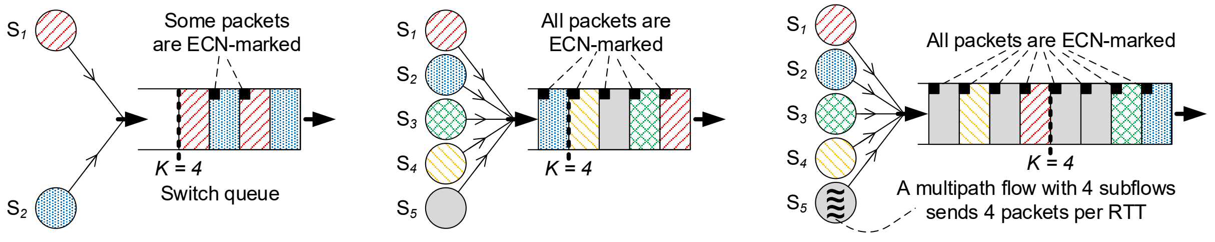
III. Issues of MPTCP variants

A. TCP incast

There are two key issues with an ECN-capable MPTCP variant. Firstly, it is unable to handle incast-like traffic that is prevalent in DCNs; many applications (e.g. MapReduce [13] and Partition/Aggregate [1]) have a many-to-one communication pattern. Secondly, it fails to gracefully coexist with single-path flows such as DCTCP[1]; an ECN-capable MPTCP variant can harm DCTCP flows by causing their throughput to be reduced significantly (which we term LHU). The following sections, we demonstrate the impact of these two problems via simulation under a simple topology as shown in Figure 1.

TCP incast is a well-studied topic [1], [14] and for instance DCTCP mitigates the problem using ECN. Unfortunately, the ECN-capable MPTCP variants are still susceptible to TCP incast, even when making use of ECN. To demonstrate this, we create a simulation environment as shown in Figure 1 using NS-3. The simulation setup is as follows. Every 1 second k multipath flows join a 10Gbps link where $k = 10, 20, 30, 40$ while setting the flow size to 128KB. The switch uses a shallow buffer with size of 100 packets. Each simulation lasts for 20 seconds. Each multipath flow has 4 subflows. We also separately run DCTCP and standard MPTCP as baselines.

[1] A multipath flow that has only one subflow behaves identically to a single-path flow. This implies that it is possible to use a multipath transport protocol within DCNs to deliver short flows via a single subflow and long flows via multiple subflows (see [2], [3] for a detailed discussions about why MPTCP with multiple subflows performs poorly for short flows).


S1
S2
Some packets are ECN-marked
K = 4
Switch queue
S1
S2
S3
S4
S5
All packets are ECN-marked
K = 4
S1
S2
S3
S4
S5
All packets are ECN-marked
K = 4
A multipath flow with 4 subflows sends 4 packets per RTT

(a) Normal situation (b) Persistent buffer inflation (c) Last hop unfairness problem

Fig. 3. Illustration of the last hop unfairness problem. The LHU leads to severe unfairness and escalates the likelihood of persistent buffer inflation significantly.

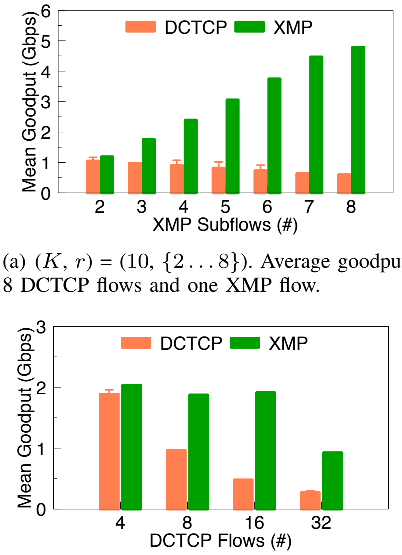
Figure 2 shows that DCTCP outperforms both XMP and standard MPTCP. In many cases the average flow completion time (FCT) of DCTCP is almost 1-2 orders of magnitude less than that of XMP; when $k = 30$, the average FCT of DCTCP is less than 2ms whereas that of XMP and standard MPTCP is over 800ms and 2000ms respectively. Furthermore, the FCT distribution of DCTCP has a narrow standard deviation (as shown by the whisker bars in the graph), but the standard deviation of XMP and MPTCP is large (less than 1 millisecond for DCTCP vs. above 1 second for both XMP and MPTCP). This implies long-tailed FCT distributions with some flows experiencing much higher FCTs due to retransmission timeouts.

From these results, it is evident that MPTCP and its ECN-capable variants cannot handle the TCP incast problem. The multipath flows maintain four subflows. Hence, one multipath flow generates at least four packets per RTT. A greater number of multipath flows implies a sharp increase in the probability of losses due to packet bursts. For example, in Figure 2, 30 multipath flows generate at least 120 packets every RTT, which are far exceeding the queue length of the bottleneck switch.

B. Last Hop Unfairness

We explore the LHU problem through examples shown in Figure 3. We first assume that network switches ECN-marked packets only if their instant queue length is larger than a marking threshold K. Such switches are widely deployed in DCNs. To keep the discussion simple, let us assume $K = 4$ and that there is zero propagation delay, i.e., assume that as soon as a packet leaves the queue the sender receives an instantaneous acknowledgment and can therefore transmit a new packet immediately.

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); i.e., 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 newly 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. This causes *persistent buffer inflation* (also discussed in [15]), but there is no unfairness across flows. Finally, Figure 3(c) shows a case where the single-path flow in S_5 is replaced with one multipath flow having 4 subflows. Similar to the previous case, all packets across flows are constantly ECN-marked, and thus the cwnd of the all flows and subflows reduces to one packet. But, since all the subflows belong to one multipath flow, the flow ends up sending four times more packets than single-path ones. Furthermore, the LHU 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.



(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.

Fig. 4. The last hop unfairness under various conditions. K: ECN marking threshold, and r: the number of subflows.

In reality, the BDP is a few tens of packets in DCNs [4], [6]. Thus, to create the LHU problem, $(BDP+K)/cwnd_{min}$ flows is sufficient. Given that the BDP and K are small and the minimum congestion window size ($cwnd_{min}$) is set by default to two packets for MPTCP and TCP variants,[2] a small number of flows could easily trigger the LHU problem.

To demonstrate the LHU problem, assume a setup as in Figure 1 where an ECN-enabled switch connects n sending servers and one receiving server. The receiver is equipped with DCTCP and XMP; server S_1 runs XMP having r subflows, and the remaining $n-1$ servers with DCTCP ($n \geq 2$). All n senders send traffic to the receiver.

We have undertaken various simulations to study the impact

[2] MPTCP and XMP use two packets to probe congestion level on each path (see a detailed discussion in [16] and Algorithm 1 in [4]). DCTCP also uses two packets originally, but a recent study proposed to use one packet for the value (see page 11 in [10]) and the DCTCP source was patched accordingly. Unless otherwise stated, we set $cwnd_{min} = 2$ for consistency in this paper.

of LHU. We alter the simulation duration from 10ms to 1 sec and use 1Gbps and 10Gbps links. Across these variations, we observe very similar trends and so, due to space limitations, we concentrate on the results of our tests of 1 sec duration over a 10Gbps link. 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 XMP flow, we vary the number of subflows from 2 to 8, while setting $K = 10$ and $cwnd_{min} = 2$ as suggested in [4]; thus, the setting is (10, $\{2 \ldots 8\}$). Figure 4(a) shows that the LHU begins as soon as the XMP flow starts to use three subflows or more. When 4 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). Using two subflows costs about 10% goodput loss (*e.g.*, 1Gbps out of 10Gbps rate) when compared to using four subflows [4]. Also, a number of subflows (*e.g.*, 8 subflows) are in general beneficial when there are a large number of parallel paths in a large DCN [2]. 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 may 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 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 slightly reduced the likelihood of the LHU, but we observed that a slight increase of the number of XMP flows (from 1 to 4) triggered the LHU, when 8 DCTCP flows are given (the exact graph has been omitted for brevity).

Summary. We obtain two key findings from these results. Firstly, the total number of packets in flight from both multipath and single-path flows should exceed the 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). Secondly, tweaking those parameters either alleviates the problem marginally or makes performance loss inevitable.

IV. Design

We propose AMP, an multipath congestion control mechanism that coexists well with ECN-capable single-path flows 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:

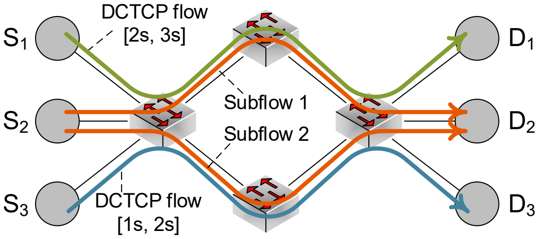


Fig. 5. A setup for testing traffic shifting time. An orange line represents a subflow of a multipath flow.

- *Fairness:* Multipath and single-path 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.
- *Fast traffic shifting:* Multipath flows should be able to avoid congested paths quickly. This especially helps latency-sensitive short 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.

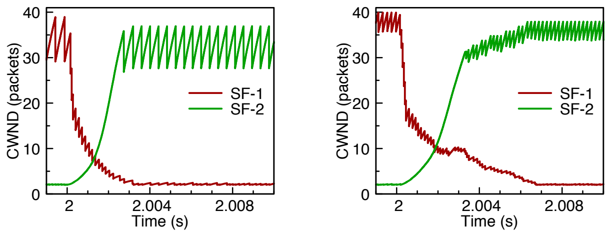
To achieve the above objectives, we study XMP (an existing ECN-capable scheme). In analyzing XMP, we make several key observations essential for the design of AMP.

A. 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. In the presence of the TCP incast and last hop unfairness, it is effective to have a smaller number of subflows (ideally, one subflow), as discussed in §III. 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 last hop unfairness. 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 [2], [4]). This is a good indicator that these problems are in effect because it is less likely 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. We examine traffic shifting times for MPTCP when it is integrated with an ECN scheme in two scenarios: (1) MPTCP-FIX: a subflow reduces the cwnd by a constant factor (we use $\beta = 4$) when it sees an ECN-marked packet; and (2)



(a) MPTCP-FIX ($\beta = 4$) (b) MPTCP-ADP

Fig. 6. Traffic shifting times of MPTCP-FIX and MPTCP-ADP. MPTCP-FIX finishes its traffic shifting at 2.003s and MPTCP-ADP does at 2.007s; MPTCP-ADP is 4ms slower than MPTCP-FIX.

MPTCP-ADP: a subflow cuts its `cwnd` in proportion to the fraction of ECN-marked packets over a window (identical to DCTCP). Given a topology shown in Figure 5, a multipath flow (MPTCP-FIX or MPTCP-ADP) 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 with MPTCP-FIX traffic shifting about 4ms faster than MPTCP-ADP.

The reason is because a MPTCP-ADP subflow conservatively reduces `cwnd` based on the fraction of marked packets. Hence, even if there exists a congestion-free path, a MPTCP-ADP flow shifts its traffic slowly. In contrast, MPTCP-FIX is aggressive enough to make a subflow on the congested path quickly reduce its window, thereby achieving faster traffic shifting than MPTCP-ADP. The conservative nature of MPTCP-ADP that is inherited from DCTCP perfectly makes sense if a flow traverses single 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`. XMP relies on RTT measurements in increasing `cwnd` of subflows. It also inherits MPTCP's design principles, one of which targets to address the RTT mismatch issue [16] 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. DCNs thus have no paths that cause the RTT mismatch issue.

Moreover, ECN tends to equalize RTTs throughout a DCN when switches react to instant queue length with a small marking threshold [1], [4]. 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 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.

Algorithm 1: Pseudocode of AMP

```
/* Subflow suppression/release */
SuppressSubflows($nRound$)
    nSF = 0    /* counter for subflows */
    for $subflow\ s \in [1, \dots, n]$ do
        if $w_s = cwnd_{min}$ then nSF ← nSF + 1
    end
    if $nSF = n$ then nRound ← nRound + 1
    else nRound ← 0
    if $nRound < \gamma$ then return
    for $subflow\ s \in [2, \dots, n]$ /* at $\gamma$ rounds */
     do
        $active_s$ ← false
    end
ReleaseSubflows($ACK, nRound$)
    if $ACK.marked$ then nRound ← 0
    else nRound ← nRound + 1
    if $nRound < \tau$ then return
    for $subflow\ s \in [2, \dots, n]$ /* at $\tau$ rounds */
     do
        $active_s$ ← true
    end
/* RTT-agnostic CWND increase */
IncreaseCWND($s, w_{total}$)
    /* For each ACK of subflow $s$ */
    $w_s \leftarrow w_s + 1/w_{total}$
/* Constant factor CWND decrease to ECN */
RespondToECN($s$)
    /* For the first marked ACK of subflow $s$ per window */
    $w_s \leftarrow \max\ (w_s(1 - 1/\beta), cwnd_{min})$
/* Response to duplicate ACKs */
DecreaseCWND($s$)
    $w_s \leftarrow \max\ (w_s/2, cwnd_{min})$
```

B. 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 less complex compared to existing schemes. 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, SSR ensures fairness between multipath and single-path flows at a shared bottleneck link. It also helps to handle more senders during an incast-like episode or to reduce the chance of costly timeouts. We show SSR's efficacy in §V-B.

RTT-agnostic congestion window increase. As discussed in §IV-A, 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 [16], [17].

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. (2) for XMP 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

$$\text{AMP}_{inc} \approx \text{XMP}_{inc}$$

Looking at these relationships among two algorithms, the increment is comparable across both of them, but AMP's algorithm is much simpler than XMP.

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 [4], this problem of choosing β is formulated as follows:

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

Note $\beta \geq 2$; 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 $K = 10$, $\beta \geq 3$. Since computing BDP even for other link speed (e.g., 10Gbps) is easy, it is straightforward to set β after K is first determined.

V. Evaluation

We evaluate AMP via extensive simulations using our custom-written simulator in NS-3.19. We have implemented MPTCP, AMP, XMP, DCTCP, ECN, and ECMP in this simulator. Our source code can be found in [12].

In this section, 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 and its effectiveness to the last hop unfairness. We finally study the overall performance of AMP under a large-scale fat-tree topology that represents a realistic data center network. For comparison, we use DCTCP and XMP.

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 (with various queue sizes) and observed that the trends were similar to those of 10Gbps settings. We only show the results under the 10Gbps settings (with a maximum queue size of 100 packets) in interest of space.

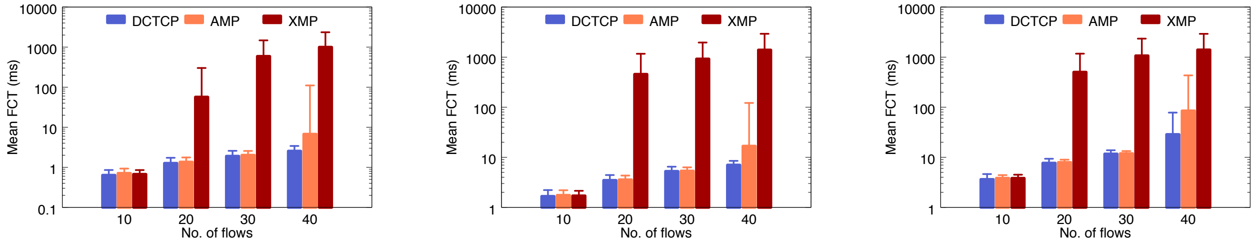
We set a default value for each of the following parameters: (i) the number of subflows per multipath flow = 4, (ii) the minimum congestion window size, $cwnd_{min}$ = 2 packets, and (iii) the ECN marking threshold, K = 10 packets. When necessary (*e.g.*, for further analysis), we change their values.

Evaluation metrics. We have four key metrics: Jain's fairness index [18], 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.

A. 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 last hop unfairness. If there indeed exist the two problems in the network, it is important to begin the suppression process



(a) Flow Size of 128KB (b) Flow Size of 256KB (c) Flow Size of 512KB

Fig. 7. Impact of the TCP incast on different multipath protocols. A multipath protocol (AMP and XMP) is only used to transfer the incast traffic. A whisker bar denotes standard deviation. The y-axis is log-scaled.

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. 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. If τ is too small, AMP will repeatedly begin and end the suppression process. This 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 (3-8 RTTs) and using 3-5 AMP flows under various topologies, including the one shown in Figure 1. We find SSR becomes fully stable across all tests when $\tau \geq 6$, with the median queue length of about 10 packets. We thus set $\tau = 8$ as default (to be conservative). The exact graph has been omitted for brevity, but it is available in [9].

B. 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) [3]. To understand how well AMP tolerates such a traffic pattern, we use the same simulation setup used in §III-A. 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 outperforms XMP (Figure 7); in most cases the average FCT of AMP is almost 1-2 orders of magnitude shorter than that of XMP. For instance, Figure 7(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 is over 800ms. In addition, AMP has a narrow standard deviation in its FCT distribution, but XMP has a large standard deviation (1-2ms for AMP vs. 1 second for XMP). 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. AMP also performs as good as DCTCP. This implies that it might be attractive to use AMP with multiple subflows for delivering both short and long flows in DCNs. Despite this finding, AMP can still be utilized as an unified transport protocol in DCNs if it delivers short flows with only one subflow and long flows with multiple subflows [2], [3].

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 7(a), we observe that AMP has no timeout during the simulation whereas XMP faces up to 10 timeouts (with 7 timeouts at 90th percentile). Notably, 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 two multipath approaches; even the SSR mechanism is not triggered as the flow size is too small. Thus, if the flow size is at least as large as 128KB, our approach works better than XMP as we observe a similar trend for a flow size of 1MB (the plot has been omitted for brevity).

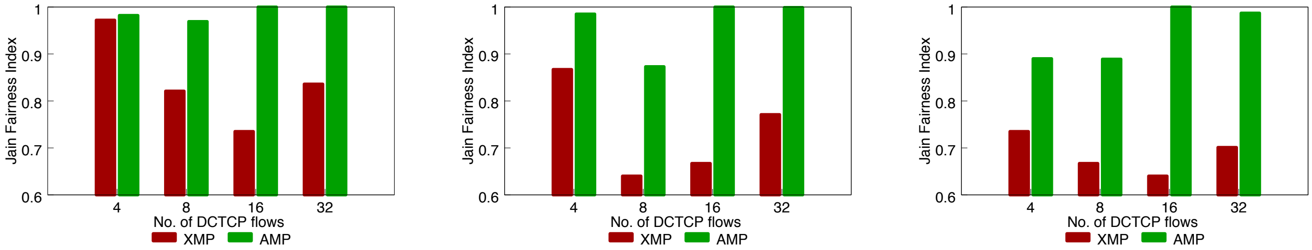
Effectiveness to the last hop unfairness. We use the topology shown in Figure 1 and test the impact of the LHU on different schemes while varying the number of DCTCP flows and multipath flows. All flows arrive at 0 sec and end at 1 sec.

Figure 8 shows that in almost all cases AMP outperforms XMP. As the number of multipath flows increases, we find the LHU aggravates fairness even in the presence of a small number of DCTCP flows.

Since the LHU causes persistent buffer inflation, we examine queue length. From Figure 9, we make two observations. When there is no LHU (Figure 9(a)), the queue length distributions across AMP and XMP are similar. On the other hand, when 4 XMP flows are used (Figure 9(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 9(a) and 12 packets in Figure 9(b)). If the intensity of the LHU grows, the queue length will become more inflated accordingly. Overall AMP can mitigate the persistent buffer inflation better than XMP even if the LHU is more intensive.

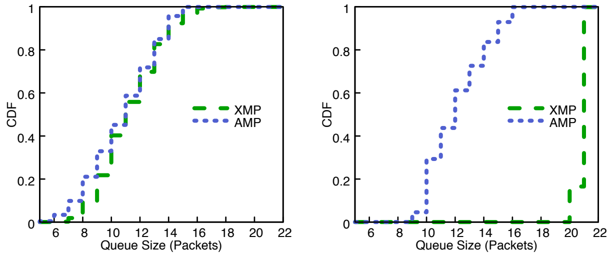
C. Large-scale simulation

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 [19], [20], 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.



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

Fig. 8. Fairness obtained when a multipath scheme competes with DCTCP flows under the last hop unfairness. Each multipath flow has 4 subflows.



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

Fig. 9. Queue length distribution. (a) without LHU: 4 DCTCP flows and one multipath flow. (b) with LHU: 4 DCTCP flows and 4 multipath flows.

Incast with background traffic. We aim to examine the performance of a high fan-in workload in the presence of background traffic. Specifically, we use DCTCP to generate the incast traffic and a multipath protocol for the background traffic.[3] Note that this scenario is different from one in §V-B 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 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 decided by a Pareto distribution with shape parameter of 1.5 and mean of 192MB. Once a long flow ends, a new one begins. A simulation continues until 1000 long flows are completed [4].

Results: Figure 10 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 10 show that the AMP case (*i.e.*, AMP is used for long flows) obtains better JCT performance than the baseline case across all scenarios. Second, more number of subflows in 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 10(a)), 1.5ms in the 6-subflow case (Figure 10(b)), and 1.8ms in the 8-subflow case (Figure 10(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.

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 [2], [3]. 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 [2]–[4], [21], [22]. 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.

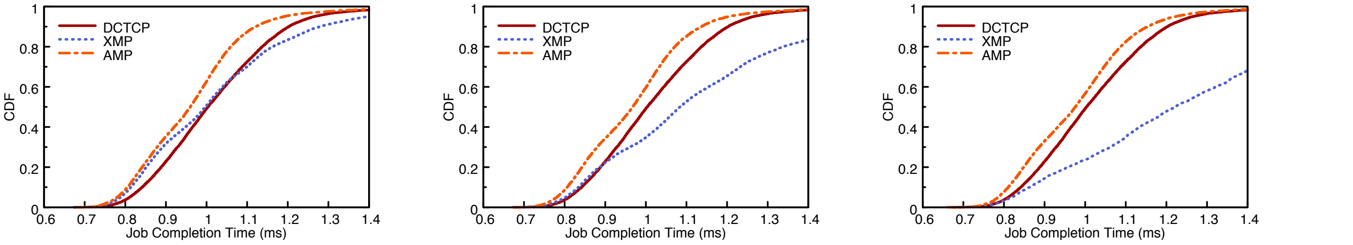
Results: Figure 11(a) shows the FCT results of different schemes. A key in the legend denotes a protocol used for long flows. We observe that short DCTCP flows achieve the best FCT result when AMP and XMP are used for long flows. The worst FCT performance was observed when DCTCP is used for long flows because DCTCP suffers from poor ECMP load balancing.

The long flows of AMP and XMP show little difference in goodput. Again, using DCTCP for long flows yields the worst goodput performance due to the same reason in the FCT case.

Figure 11(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.

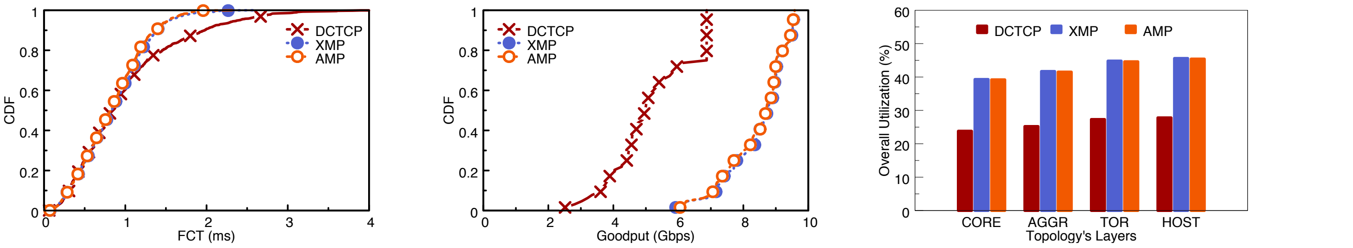
We also test those schemes with realistic web search and data mining workloads [23] and under a more intensive and dynamic condition with an order of magnitude shorter flow arrival rate in a 4:1 oversubscribed fat-tree topology. We make

[3] We also examined scenarios where AMP with one subflow generates the incast traffic and we observed that the trends were identical to that of DCTCP.



(a) 4 subflows (b) 6 subflows (c) 8 subflows

Fig. 10. Job completion time of incast workloads. DCTCP is used for generating incast traffic and background traffic is generated by using AMP, 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

Fig. 11. FCT, goodput and network utilization performance. Short flows are 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.

similar observations across all cases (an extended version of this paper is available in [9] that includes all these results).

VI. Conclusion

In this paper we presented that existing ECN-capable 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 last hop unfairness 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. Finally, AMP does not require any changes in the network (e.g., at the network switches) and at the MPTCP receiver, it only requires small modifications at the MPTCP sender.

Acknowledgment

This research was partially supported by the H2020 5G-MEDIA project under Grant 761699.

References

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

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

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

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

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

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

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

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

[9] M. Kheirkhah and M. Lee, "AMP: A Better Multipath TCP for Data Center Networks," Tech. Rep., 2019. [Online]. Available: https://arxiv.org/pdf/1707.00322.pdf

[10] G. Judd, "Attaining the Promise and Avoiding the Pitfalls of TCP in the Datacenter," in *USENIX NSDI*, 2015.

[11] M. Kheirkhah, I. Wakeman, and G. Parisis, "Multipath-TCP in ns-3," *CoRR*, 2014. [Online]. Available: http://arxiv.org/abs/1510.07721

[12] https://github.com/mkheirkhah/amp, last checked: 2019-03-22.

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

[14] H. Wu, Z. Feng, C. Guo, and Y. Zhang, "ICTCP: Incast congestion control for TCP in data-center networks," in *ACM CoNEXT*, 2010.

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

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

[17] F. Kelly and T. Voice, "Stability of End-to-end Algorithms for Joint Routing and Rate Control," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 2, pp. 5–12, 2005.

[18] R. Jain, A. Durresi, and G. Babic, "Throughput Fairness Index: An Explanation," 1999, aTM Forum/99-0045.

[19] M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," in *ACM SIGCOMM*, 2008.

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

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

[22] M. Kheirkhah, "MMPTCP: A Novel Transport Protocol for Data Centre Networks," Ph.D. dissertation, University of Sussex, 2016.

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