CAPS: Coding-based Adaptive Packet Spraying to Reduce Flow Completion Time in Data Center

Jinbin Hu[§], Jiawei Huang[§], Wenjun Lv[§], Yutao Zhou[§], Jianxin Wang[§], Tian He[‡]
[§]School of Information Science and Engineering, Central South University, Changsha, China 410083
[‡]Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, USA 55455
Email: {jinbinhu,jiaweihuang,wenjunlv}@csu.edu.cn, ytzh06@gmail.com, jxwang@csu.edu.cn, tianhe@umn.edu

***Abstract*—Modern data-center applications generate a diverse mix of short and long flows with different performance requirements and weaknesses. The short flows are typically delay-sensitive but to suffer the head-of-line blocking and out-of-order problems. Recent solutions prioritize the short flows to meet their latency requirements, while damaging the throughput-sensitive long flows. To solve these problems, we design a Coding-based Adaptive Packet Spraying (CAPS) that effectively mitigates the negative impact of short and long flows on each other. To exploit the availability of multiple paths and avoid the head-of-line blocking, CAPS spreads the packets of short flows to all paths, while the long flows are limited to a few paths with Equal Cost Multi Path (ECMP). Meanwhile, to resolve the out-of-order problem with low overhead, CAPS encodes the short flows using forward error correction (FEC) technology and adjusts the coding redundancy according to the blocking probability. The coding layer is deployed between the TCP and IP layers, without any modifications on the existing TCP/IP protocols. The experimental results of NS2 simulation and Mininet implementation show that CAPS significantly reduces the average flow completion time of short flows by ∼30%-70% over the state-of-the-art multipath transmission schemes and achieves the high throughput for long flows with negligible traffic overhead.**

***Index Terms*—Data center; TCP; packet spray; multipath**

I. Introduction

With the rise of data centers, a variety of cloud-based applications such as web search and social networking are deployed by a large number of online service providers like Google and Amazon. To obtain better user experience and financial revenue, how to achieve the low latency and high throughput becomes a crucially important issue.

In recent years, Random Packet Spraying (RPS) [1] is proposed in data center networks (DCNs). As a packet-level transmission scheme on the switch, RPS randomly assigns the packets of every flow to one of the available shortest paths to the destination. RPS splits each flow into packets and spreads the packets to all available paths to achieve high efficiency and easy deployment in the multipath topologies such as Fat-tree and Clos, which are widely used in the large-scale data centers. Recently, the packet spraying technique has already been implemented on the commodity switches (e.g. Cisco [2]).

Unfortunately, RPS does not consider the important traffic characteristic that the short and long TCP flows are mixed in DCNs. The data center traffic can be characterized by heavy-tailed distribution [3], [4], that is, around 90% of data is provided by only around 10% of TCP flows, and about 90% of TCP flows provide only about 10% of data. Furthermore, most short flows belong to the delay-sensitive applications, while the majority of long flows are throughput-sensitive. Since RPS does not provide any differentiation between the short and long flows, it inevitably leads to the negative interaction between the two kinds of flows.

When the packets of both short and long TCP flows are mixed over the multiple paths by RPS scheme, the following two issues come up. The first one is the head-of-line blocking. When the switch buffer is occupied by the packets of long flows, the short flows have to experience large queuing delay, leading to the long-tailed flow completion time (FCT). Secondly, RPS randomly spreads the packets into different paths, possibly resulting in the TCP out-of-order problem.

Coding is a very powerful scheme to address these issues. At the sender, the source packets are encoded and scattered to multiple paths. Though some packets experience the head-of-line blocking or out-of-order, once sufficient encoded packets arrive at the receiver, the original packets can be recovered immediately. We propose a coding-based adaptive packet spraying (CAPS), which successfully integrates coding into packet spraying and efficiently avoids both head-of-line blocking and packet reordering. Moreover, the transparent coding layer only needs to be deployed between the TCP and IP layers at the end hosts, while making no modifications on the existing TCP/IP protocols.

In summary, our major contributions are:

- We conduct an extensive simulation-based study to analyze two key issues with multipath transmission: the short flows experience large FCT due to the head-of-line blocking caused by the long flows, and the coexisting of short and long flows leads to the reordering problem.
- We propose a multipath transport scheme CAPS, which randomly scatters the encoded packets of short flows to all paths and transmits the long flows to a few paths. Specifically, we design the coding layer, which rationally adjusts the number of redundant packets to resolve the out-of-order and head-of-line blocking problems. To improve the link utilization, CAPS swiftly scatters the packets of long flows to the unused paths by short flows.
- By using both NS2 simulations and Mininet implementation, we demonstrate that CAPS performs remarkably better than the state-of-the-art multipath transmission schemes. Especially, CAPS greatly reduces the average FCT (AFCT) of short flows by ∼30%-70% under high workload. Meanwhile, CAPS yields up to ∼45% and ∼35% throughput improvement for long flows over RepFlow and Freeway, respectively.

978-1-5386-4128-6/18/$31.00 ©2018 IEEE

The rest of the paper is organized as following. In Section II and III, we respectively describe our design motivation and overview. In Section IV and V, we introduce packet spraying and coding of CAPS, respectively. We discuss the implementation in Section VI. In Section VII and VIII, we show the Mininet experimental and NS2 simulation results, respectively. In Section IX, we present the related work and then conclude the paper in Section X.

II. Design Motivation

To motivate our design, we investigate the impact of the long flows on the short ones with existing RPS scheme.

A. Head-of-line Blocking

In the data center, almost 90% of TCP flows are less than 100KB [3]. Based on this characteristic, a flow with its data size less than 100KB is considered as a short flow, otherwise, that is a long flow. Since RPS does not isolate the short flows from long ones, the packets of short flows may be spread to the paths occupied by long flows, and have to be queued behind the long flows. Therefore, the short flows may suffer from the heavy head-of-line blocking and long tailed FCT [5], [6].

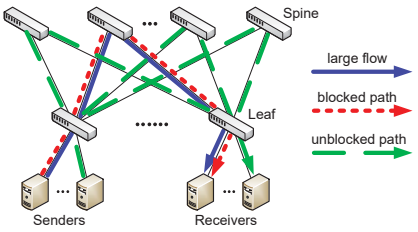


Fig. 1: Leaf-Spine topology

We conduct NS2 simulation test to analyze the head-of-line blocking problem with the leaf-spine topology as shown in Fig. 1. Each sender sends a flow to a single receiver via multiple switches. The buffer size of each switch is 256 packets. There are 40 equal cost paths between the source and destination nodes. The bandwidth of each path is 1Gbps and the round-trip propagation delay is 100μs. In addition, according to the measurement results in [7], the 75th and 95th percentile of number of concurrent large flows sharing a ToR switch are 2 and 4, respectively. In this test, the mixture of 100 short TCP flows (less than 100KB) and 4 long-lived background TCP flows are generated in heavy-tail distribution.

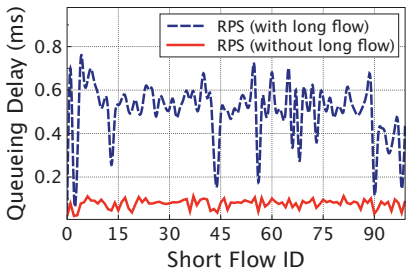


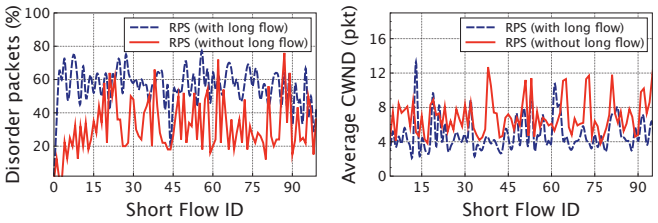
Fig. 2: Average queueing delay

We compare the average queueing delay of short flows under two cases. Firstly, the ToR switch uses RPS to spread all packets of short and long flows to all paths. Secondly, without long flows, only the packets of short flows are spread to all paths. Fig. 2 demonstrates the average queueing delay for the short flows. In the case without the long flows, the short flows do not experience the head-of-line blocking, and thus the average queueing delay of the short flows is reduced significantly by up to around 80%.

B. Out-of-order

Since RPS randomly spreads the packets to all paths, the long flows potentially make the short flows suffer from the out-of-order problem, which means the later-sent packets may be received ahead of the earlier-sent ones. Fig. 3 (a) shows the ratio of the number of disorder packets to all packets. Compared with scenario without the long flows, the ratio of disorder packets of short flows becomes larger when both kinds of flows are mixed.

When the out-of-order event happens, the TCP sender assumes the packets are lost and then cuts its congestion window, resulting in spurious retransmission and even timeout. As shown in Fig. 3 (b), the average congestion windows of the short flows are about 50% smaller than that in the scenario without the long flows.



(a) The percent of disorder packets (b) The average congestion window

Fig. 3: Disorder due to long flows

Without the long flows, both the queueing delay and the number of disorder packets in short flows are significantly reduced. Therefore, as shown in Fig. 4, the FCT of short flows is greatly reduced without the impact of long flows.

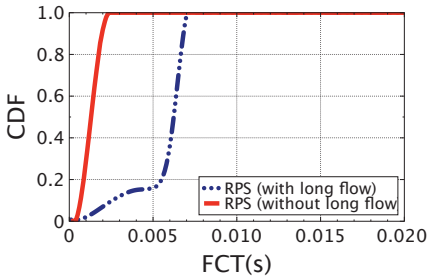


Fig. 4: FCT of short flows

C. Summary

Our analysis of the coexisting short and long flows leads us to conclude that (i) the short flows experience increasing

delay due to the head-of-line blocking once all flows are treated as the same, (ii) the packet reordering due to the long flows seriously enlarges the FCT of short flows. These conclusions motivate us to tackle above problems by designing and implementing a coding-based adaptive packet spraying scheme.

III. Desgin Overview

In this section, we present an overview of CAPS. The two key points of CAPS are using packet spraying and FEC coding to solve the head-of-line blocking and out-of-order problem, respectively. Specifically, on the one hand, when the long flows are limited to a few paths, most packets of the short flows are spread on the paths without blocking and therefore achieve the lower queueing delay. On the other hand, FEC coding eliminates the impact of packets reordering. Even if some encoded packets of the short flows are blocked by the long flows, the original packets can be recovered from the other non-blocked encoded packets. The architecture of CAPS consists of three modules, as shown in Fig. 5.

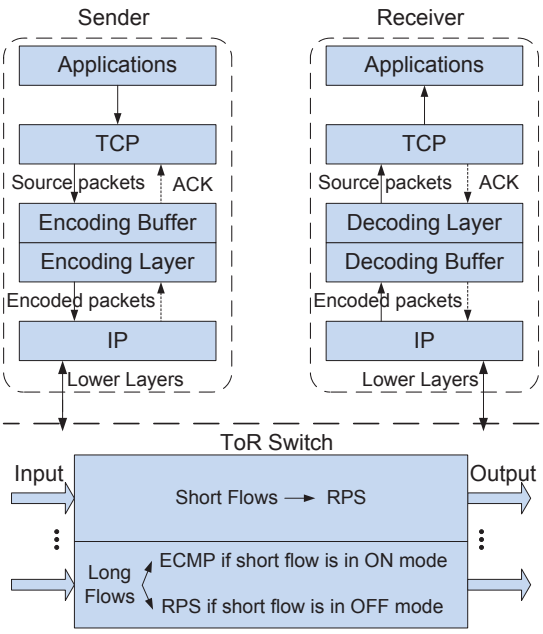


Fig. 5: CAPS Architecture

(1) **Encoding Module:** On the sender side, the encoding module accepts the source packets from the transport layer and caches them into an encoding buffer. Then the sender generates $k+r$ encoded packets (i.e., a coding unit) from k original packets in the encoding buffer, and delivers the $k+r$ encoded packets to the network layer. During the encoding procession, an important task is to dynamically adjust the number of redundant packets r according to real-time state of network traffic. When receiving the ACK packets, the sender removes the ACKed packets from the coding buffer and delivers the ACK packets to the transport layer.

(2) **Packet Spraying Module:** At the switch, the packets of the short flows are spread to all output ports using RPS technique which have already been implemented in many commodity switches. On the other hand, the long flows are transmitted by ECMP [8] or RPS respectively according to the ON or OFF mode of short flows.

(3) **Decoding Module:** On the receiver side, the encoded packets from the network layer are cached into the decoding buffer. The original k packets can be decoded from $k+r$ encoded packets and handed over to the upper layer.

IV. Packet Spraying

To reduce the negative impact of short and long flows on each other, CAPS uses different strategies to transmit the two kinds of flows. When the short and long flows are transmitted at the same time, the long flows are limited to only a few paths by ECMP strategy, while the short flows are simply spread to all paths by RPS to make full use of the multiple paths between any given pair of hosts. The details of CAPS operation on ToR switch are described as following.

(1) **Long flows:** For a long flow arriving at the ToR switch with the short flows existing at the same time, it is forwarded to the next hop by ECMP, which is extensively used as the $de\ facto$ routing algorithm. As identified by the TCP 5-tuple, the TCP flows are randomly hashed to their respective paths. Thus, the small number of long flows are limited to a few paths and the head-of-line blocking problem due to the long flows is avoided for the short flows on the other paths. Moreover, since ECMP is a flow-level scheme, the out-of-order problem of long flows is also resolved.

(2) **Short flows:** Once arriving at the ToR switch, the packet of short flows is routed by RPS. Since the packets of short flows are randomly scattered to all available paths to the destination on the packet-level, RPS utilizes all available bandwidth more efficiently than ECMP in terms of the throughput and flow completion time. However, RPS potentially results in the significant packet reordering due to the large queuing delay on the paths with long flows. To overcome the out-of-order problem, we use FEC coding technology to encode the packets of short flows at the sender as illustrated in the following section.

V. Encoding and Decoding

In this section, we firstly give our rational selection for coding algorithm. Then the key point of redundancy optimization is discussed. Finally, we analyze the delay improvement and traffic overhead.

A. Coding Algorithm

Forward Error Correction (FEC) technology [9], [10], [11] effectively mitigates the negative impact of packets blocking and reordering. The reason is that FEC only cares about how many, rather than which encoded packets have been received. FEC codes are mainly divided into two categories, called fixed-rate codes [12] and rateless codes [13]. For rateless codes, redundancy should be adjusted in real-time according to the varying packet loss or blocking probability. In order to avoid unnecessary redundant packets, the receiver sends the feedback information to the sender to stop encoding for the current

coding unit [14], [15], unavoidably increasing the latency and also reducing the robustness of data transmission.

In our CAPS, we use the fixed-rate codes due to the following reasons. Firstly, the fixed-rate coding scheme works well when the blocking rate does not change rapidly. Here, we investigate the probability of a short flow being blocked by long flows (i.e., blocking probability). We conduct a simulation in NS2 with the same settings as described in Section II. As shown in Fig. 6, the traffic of short flows shows ON/OFF mode, in which the packets of short flows start and finish their transmissions during the ON periods. Both the ON and OFF periods follow exponential distribution and the OFF period is much longer than the bursty ON period [3]. During the ON periods of the short flows, the long flows are always existing because the long flows have much larger flow size. This phenomenon means that, for short flows, the blocking probability is fixed during their lifetime.

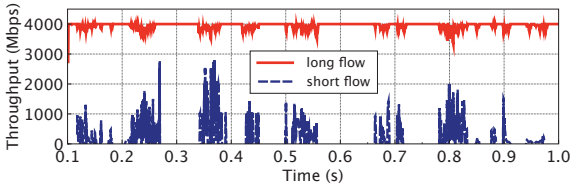


Fig. 6: The traffic mode of short and long flows

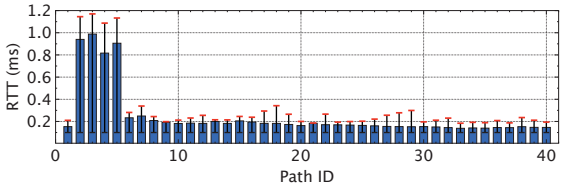


Fig. 7: The RTT of each path

Secondly, the blocking probability can be estimated in advance at the sender based on the measurement of RTT. As shown in Fig. 7, 4 paths with long flows have much larger RTT than the other paths with only short flows. The blocking probability by long flows can be estimated as 0.1, which is the ratio of the number of paths with large RTT to the total number of paths. However, since CAPS randomly spreads packets of the short flows to all paths on the ToR switch, the sender is unable to directly obtain the exact RTT for each path. Here, we utilize the TCP congestion control mechanism. Specifically, when the sender receives the ACK packets, based on the corresponding RTT for each ACK, the blocking probability is calculated as the ratio of the number of ACK packets with large RTT to the total number of received ACK packets. According to the RTT statistics, the empirical threshold for the large RTT is set as 2x average RTT of all packets.

The classical fixed-rate codes include Reed-solomon codes (RS) and Low Density Parity Check Codes (LDPC). Since RS is suitable for the short code unit with total number of bits less than 1000, we choose LDPC because it is more practical to combine multiple packets (i.e., 1500 Bytes for each

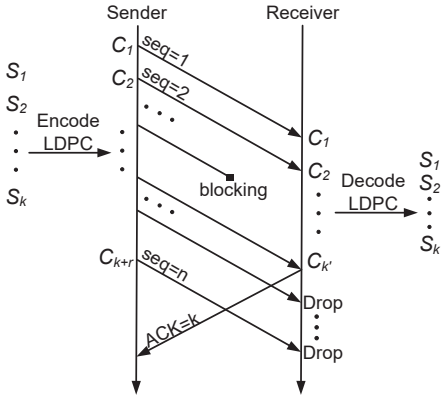


Fig. 8: Data Transmission for a Coding Unit

packets) into a code unit [12]. As shown in Fig. 8, a code unit containing k source packets (S_1, S_2, ... , S_k) is encoded into $k+r$ encoded packets (C_1, C_2, ... , C_{k+r}) at the sender. The value of r can be changed at each encoding unit according to the blocking probability. The receiver performs decoding process to reconstruct the original packets from any set of k' encoded packets, for k' slightly larger than k. In CAPS, k is set to the congestion window size of short flows. Since k' can be approximated by k when the number of bits of a code unit is large (i.e., >1000 bits) [11], for simplicity, we substitute k for k' in the following sections. At the receiver, the subsequent received encoded packets belonging to the same coding unit are dropped directly.

B. Redundancy Optimization

In the encoding operation, the coding redundancy affects both the decoding delay and traffic overhead. It is hard to balance these two aspects. For instance, in order to speed up the decoding operation, the sender should send more redundant packets, which unavoidably brings about unnecessary traffic overhead and limits the transmission rate of the sender by itself. However, if too less redundant packets are transferred, the decoding speed is limited because the receiver has to wait for enough encoded packets for decoding operation. In brief, the coding redundancy should be elaborately adjusted to achieve good tradeoff between the decoding delay and traffic overhead. We give the redundancy optimization as following.

Let n_L and n denote the number of ACK packets with large RTT and the total number of received ACK packets, respectively. Then we get the blocking probability p_B of a short flow blocked by long flows as

$$p_B = \frac{n_L}{n}. \tag{1}$$

A code unit has k source packets and r redundant packets. That is, k source packets are encoded into $k+r$ encoded packets. To guarantee that at least k encoded packets reach the receiver without blocking, the following Equation (2) should be satisfied

$$(1-p_B)\times(k+r)\geq k. \tag{2}$$

To reduce the traffic overhead, with Equation (1) and (2), the number of redundant packets r for each k source packets is set as

$$r = \frac{k}{1 - p_B} - k. \tag{3}$$

Fig.9 shows the number of redundant packets r with increasing p_B. For the higher blocking probability or larger coding unit, the sender uses more redundant packets to compensate the blocked packets and achieve the high successful decoding probability.

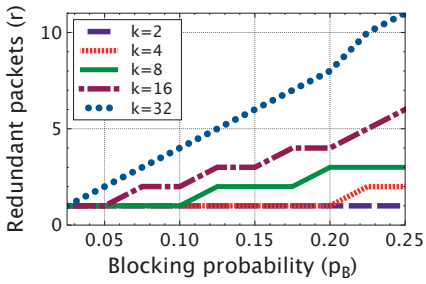


Fig. 9: Number of redundant packets r

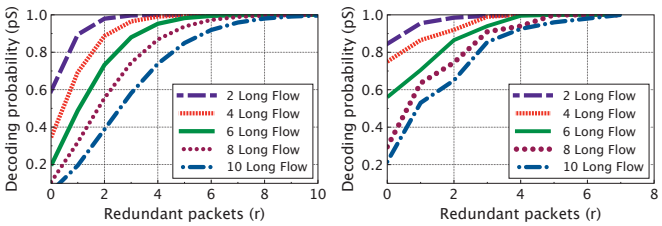
Unfortunately, since the short flows are randomly spread to all paths, the decoding probability of short flows is not 100%, because some unlucky packets may be blocked or even dropped on the paths with long flows. Here, we analyze the successful decoding probability p_S of short flows.

Within $k + r$ encoded packets, supposing the number of blocked packets by long flows is j, we obtain the probability of any blocked j encoded packets out of $k + r$ packets as

$$p_B(j) = C_{k+r}^{j} \times {p_B}^{j} \times (1 - p_B)^{k+r-j}. \tag{4}$$

The receiver can successfully decode the original packets only when the number of blocked packets is no larger than the number of redundant packets r. Then the successful decoding probability p_S is computed as

$$p_S = \sum_{j=0}^{j \leqslant r} p_B(j) = \sum_{j=0}^{j \leqslant r} C_{k+r}^{j} \times {p_B}^{j} \times (1 - p_B)^{k+r-j}. \tag{5}$$



(a) Numeric comparison (b) NS2 simulation

Fig. 10: p_S with varying r and n_L

The numeric comparison of the successful decoding probability p_S is shown in Fig. 10 (a). The number of paths and the coding unit size k are set as 40 and 10, respectively. In the numeric analysis, we simply substitute the ratio of the number of long flows' paths to the total number of paths for the blocking probability. With the increasing number of redundant packets r, the successful decoding probability becomes larger due to more unblocked packets. Furthermore, it is much easier to decode successfully with a smaller number of long flows. For 2 long flows, the successfully decoding probability reaches almost 1 when r is 2. We also conduct the simulation experiments in NS2 with the same settings in Section V.A. Fig.10 (b) shows the test result, which is consistent with the trend of numeric analysis.

C. Delay Performance Analysis

In our designed CAPS, the long flows are limited to a few paths when the short flows are transmitted. For the short flows, once enough number of encoded packets arrive, the original packets can be recovered immediately by the receiver, thus greatly reducing the transfer delay. However, the redundancy still leads to the buffering delay for decoding operation at the receiver. Here, we analyze the delay performance of short flows without and with coding.

(i) $Without\ Coding$

Let RTT_L and RTT_S represent the maximum values of the RTT of paths with and without long flows, respectively. Given the blocking probability p_B for each packet, the probability for all k packets transmitted though the paths without long flows is $(1 - p_B)^k$. Then the probability for at least a packet experiencing the paths with long flows is $1-(1-p_B)^k$. Finally, we obtain the average delay d_{nc} of k packets without coding as

$$d_{nc} = (1 - p_B)^k \times RTT_S + (1 - (1 - p_B)^k) \times RTT_L. \tag{6}$$

(ii) $With\ Coding$

In the decoding operation, the receiver should buffer k encoded packets before reconstructing the original packets, introducing the extra buffering delay d_B. Since the k packets are sent back-to-back by the sender, the ith packet has to wait for the rest $k - i$ packets, with the buffering time as $\frac{(k-i)MSS}{C}$. Here, we use C and MSS to denote the bottleneck link capacity and the size of a TCP segment, respectively. The average buffering delay for waiting any set of k encoded packets within a coding unit is calculated as

$$d_B = \sum_{i=1}^{k} \frac{(k-i)MSS}{C} \times \frac{1}{k} = \frac{(k-1)MSS}{2C}. \tag{7}$$

Though introducing the buffering delay, the coding operation helps the short flows to avoid the impact of blocked packet by the long flows. The delay improvement is analyzed as following.

When at least k encoded packets are transferred through the paths without long flows and successfully recovered at the receiver, the total delay of the successful decoding includes both RTT_S and the buffering delay d_B. The probability for this case is the successful decoding probability p_S. When more

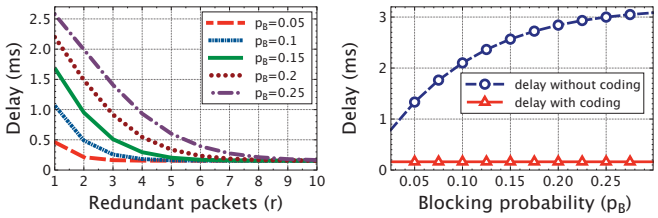
than r packets go through the paths with long flows, since the buffering delay for decoding operation is much smaller than RTT_L, we only consider RTT_L for the blocked packets. Therefore, the average coding delay d_c for a coding unit is calculated as

$$d_c = p_S \times (RTT_S + d_B) + (1 - p_S) \times RTT_L. \quad (8)$$

Taking Equation (3), (5) and (7) into (8), we obtain

$$d_c = RTT_L + (RTT_S - RTT_L + \frac{(k-1)MSS}{2C}) \times \sum_{j=0}^{j \leqslant r} C_{k+r}^{j} \times {p_B}^{j} \times (1 - p_B)^{k+r-j}. \quad (9)$$

Fig. 11 (a) shows the coding delay d_c decreases as the redundant packets increasing with a certain blocking probability p_B. The bottleneck link capacity C is 1Gbps and the size of a TCP segment MSS is 1500 Bytes. k is set as 10. Based on the measurement result of the maximum RTT on the paths with and without long flows, RTT_S and RTT_L are set to 0.1ms and 3ms, respectively. When the blocking probability p_B is increased, the corresponding values of d_{nc} and d_c are shown in Fig. 11 (b). It is clear that d_{nc} is much greater than d_c, which means the total delay is significantly reduced by the coding operation.



(a) Increasing r (b) Increasing p_B

Fig. 11: Coding delay with increasing r and p_B

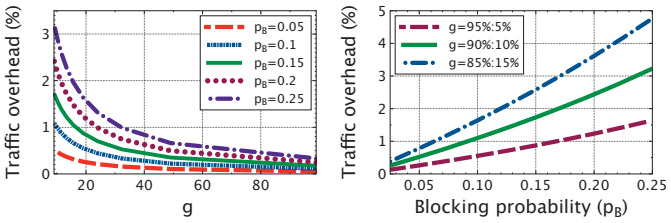
D. Traffic Overhead Analysis

The coding operation adds more redundant packets to the network system, bringing about the traffic overhead. However, our CAPS design only encodes the short flows, which account for about 10% of the total network traffic. The traffic overhead of CAPS is limited. Here, we analyze the traffic overhead of CAPS.

Supposing the ratio of total packets of long flows to short flows is g and the total packets of short flows is s, according to the Equation (3), the load overhead η is calculated as

$$\eta = \frac{\frac{s}{k} \times r}{s + s \times g + \frac{s}{k} \times r} = \frac{p_B}{1 + g \times (1 - p_B)}. \quad (10)$$

In Fig. 12 (a), the ratio of total packets of long flows to short flows g is set to from 90%:10% to 99%:1%, which is heavy tailed distribution as illustrated in Section I. The results show that the traffic overhead increases with larger g, but is always less than 3.5%. In Fig. 12 (b), it is clear that the traffic



(a) Increasing g (b) Increasing p_B

Fig. 12: Traffic overhead with increasing g and p_B

overhead under the increasing blocking probability is less than 5%, that is small enough to be negligible.

VI. Implementation

We implement the design of CAPS with two key points. The first one is to guarantee the throughput of long flows. When long flows are transmitted on a few paths, the short flows on most paths will not be blocked, while the throughputs of the long flows will be unavoidably decreased. Furthermore, unlike short flows, the throughput is much more important than FCT in view of the long flows. Thus, it is necessary to alleviate this performance impairment of long flows.

Fortunately, we can take advantage of the ON/OFF traffic pattern of short flows to adaptively adjust the number of paths for long flows to deal with rapid changes of network dynamics and make full use of available multiple paths. As shown in Fig. 6 in Section V.A, the short flows are only transmitted during the ON periods, leaving the unused paths during the long OFF periods. Therefore, to ensure the throughput of long flows without making damage to the short flows meanwhile, we adopt different strategies to control the long flows according to the traffic mode of short flows.

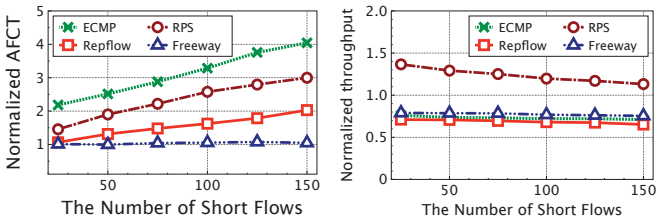
Specifically, CAPS samples the short flows periodically at the switch. When none packet of short flows is received during the sampling interval T, the mode of short flows is set as OFF and vice versa. Here, the sampling interval T is set as 500μs [16], which is the general inactivity gap between two bursts of packets in short flows. If the short flows are in OFF mode, the long flows will be spread to all paths with RPS to achieve high throughput. Otherwise, the long flows are limited to a few paths with ECMP to avoid the impact on the short flow.

The other key consideration of our CAPS is that, in most cases, it is hard to obtain accurate flow size information at the start of a flow. For example, the partial results for online query responses are typically transferred when they are generated, instead of waiting for the end of the query execution. Thus under these situations, CAPS needs to work in the dark even without prior knowledge. In the absence of prior knowledge, CAPS considers all flows as short flows in the beginning, and scatters all packets to all paths. When the amount of data belonging to one flow is larger than the threshold for large flows (i.e., 100KB) [3], [17], [18], the flow is distinguished as a long flow and then transmitted in the different way. The experimental results in Section VIII show that CAPS works well in the dark.

VII. Testbed Evaluation

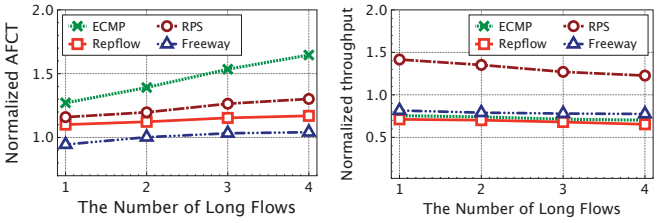
In this section, we use a realistic Mininet, implementation [18], [19], [20] in a small-scale testbed to test CAPS's broad applicability and effectiveness. We implement CAPS on Mininet, a network emulation system based on Linux kernel using virtualization. Mininet's virtual hosts, switches, links and controllers are real components running on the standard Linux kernel, and for the most part their behavior is similar to the discrete hardware elements.

In this test, CAPS is implemented in Mininet 2.3.0 on a Ubuntu kylin 16.04. The test topology is leaf-spine network shown in Fig. 1. The total number of the equal cost paths between the leaf and spine switch is 20. We set the link bandwidth to 20Mb and delay to 1ms [18]. POX is installed as the controller on switches to support ECMP and RPS. The buffer size at switches is 256 packets. The default numbers of short flows and long flows are 100 and 4 [21], respectively. The sizes of short flows are randomly distributed within 100KB. The sizes of large flows are larger than 10MB [22]. The overall traffic obeys heavy tailed distribution as illustrated in [3] .



(a) AFCT of short flows (b) Throughput of long flows

Fig. 13: Varying the number of short flows



(a) AFCT of short flows (b) Throughput of long flows

Fig. 14: Varying the number of long flows

We normalize the results of ECMP[8], RPS [1], RepFlow [18] and Freeway [23] to that of CAPS. Then we compare the performances of them with varying number of short or long flows. RepFlow simply replicates each short flows to reap multi-path diversity to minimize the flow completion time. Freeway adaptively partitions the paths into low latency paths and high throughput paths respectively for the short and long flows to alleviate the impact of long flows to the short ones.

In Fig. 13 (a), the normalized AFCT of ECMP, RPS, RepFlow and Freeway is larger than 1, meaning that CAPS achieves the smaller AFCT of short flows. Specifically, CAPS reduces the AFCT of short flows by ∼50%-70%, ∼40%-60%, ∼30%-40% and ∼1%-5% with increasing the number of short flows over ECMP, RPS, RepFlow and Freeway, respectively. This is because CAPS is not affected by the head-of-line blocking or packets reordering, only requires to receive sufficient packets regardless of which path they come from. Since ECMP and RPS are agnostic to the short and long flows, the short flows suffer from the long queuing delay and long-tailed FCT. The performance of RepFlow is worse than CAPS, because RepFlow only replicates, but does not isolate the short flows from the long flows on different paths. Moreover, the traffic overhead of RepFlow limits its performance improvement under the heavy load.

Fig. 13 (b) shows the normalized throughputs of long flows with increasing number of short flows. CAPS improves the throughputs of long flows by ∼30%-40%,∼25%-45%, ∼25%-35% over ECMP, RepFlow and Freeway, respectively. The reason is that ECMP, RepFlow and Freeway employ only one path to transfer each long flow, resulting the low utilization on the multiple paths. CAPS and RPS work on the packet-level and flexibly scatter the packets to all paths, obtaining the higher throughputs of long flows. Since CAPS sprays the packets of long flows to all paths only during the OFF periods of short flows, the throughputs of long flows in CAPS are lower than that of RPS, showing the tradeoff between the delay gain of short flows and the throughput loss of long flows.

Fig. 14 (a) shows the normalized AFCT with varying number of long flows. Compared with the other schemes, CAPS achieves the better performance. However, with the increasing number of long flows, much more traffic is injected into the network system. Moreover, more long flows lead to larger blocking probability and thus more redundant packers after the coding operation, making the network congestion heavier. Thus, the performance improvement for short flows becomes less compared with the case of increasing only the number of short flows. Fig. 14 (b) shows that CAPS obtains the higher throughputs of long flows compared with ECMP, RepFlow and Freeway.

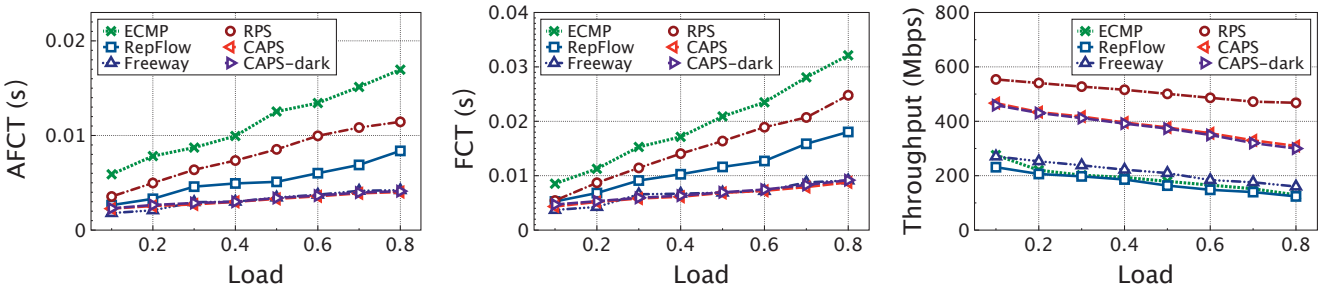
VIII. Simulation Evaluation

To evaluate the performance of CAPS in the large-scale scenarios, we conduct the NS2 simulation tests in the web search [7] and data mining [24] application scenarios. In the web search scenario, 30% of flows larger than 1MB provide more than 95% bytes. In the data mining scenario, ∼ 3.6% flows larger than 35MB provide 95% bytes, while around 80% of flows are less than 100KB.

We use the leaf-spine topology with 24 ToR switches, each of which connects to 36 hosts. The whole network has 864 hosts and 12 core switches. There are 12 equal cost paths between any pair of hosts. All flows are generated between random pairs of hosts following a Poisson process with load varying from 0.1 to 0.8 to thoroughly evaluate CAPS's performance. We also test the performance of CAPS without prior knowledge of flow size.

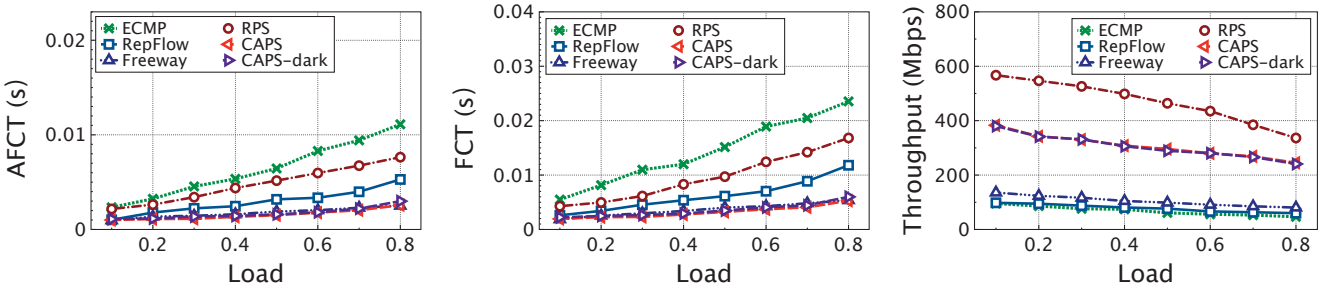
A. Delay Performances of Short Flows

Here, we focus on the flow completion time of short flows with the web search and the data mining workload as shown in Fig. 15 and Fig. 16, respectively. Fig. 15 (a) and 16 (a)



(a) AFCT of short flows (b) FCT of the 99th percentile short flows (c) Throughput of long flows

Fig. 15: Web search application



(a) AFCT of short flows (b) FCT of the 99th percentile short flows (c) Throughput of long flows

Fig. 16: Data mining application

show the average flow completion time, while Fig. 15 (b) and 16 (b) gives the 99th percentile FCT, presenting the tail FCT.

We observe that CAPS improves the AFCT and tail FCT significantly compared to the other schemes except for Freeway, especially in high workloads. For example, under the web search workload, CAPS reduces the AFCT of short flows by by ∼60%-70%, ∼50%-60% and ∼40%-50% for loads from 0.3 to 0.8 over ECMP, RPS, and RepFlow, respectively.

The results demonstrate the advantage of CAPS in the typical multi-path topology. CAPS avoids the impact of out-of-order and head-of-line blocking problem due to the mixture of short and long flows. When the workload becomes high, more long flows occupy more paths, with the result that more short flows hit the long tail. CAPS is able to get enough gain by adaptively adjusting the redundancy of short flows and limiting the number of long flows' paths. For the other schemes, since most short flows experience large queueing delay, the delay performance is degraded. Only the result of Freeway is close to CAPS, because Freeway dynamically separates the short and long flows.

Moreover, though CAPS-dark has not any prior knowledge of the flow size, the impact of long flows on the short flows is negligible when the long flows are regarded as short flows in the beginning, because the number of long flows for each ToR switch is very small (i.e., less than 4) and the threshold for large flows in CAPS-dark is only 100KB. Thus, CAPS-dark achieves almost the same performance as CAPS.

We also find that the short flows in the data mining workload has lower AFCT than those in the web search workload. The reason is that the data mining workload has more obvious boundary between the vast majority of short flows and a few long flows, while in the web search workload there are many medium flows between 100KB and 1MB. These medium flows lead to larger queueing delay.

B. Throughput Performances of Long Flows

We test the throughputs of long flows. As shown in Fig. 15 (c) and Fig. 16 (c), RPS obtains the highest throughput, because it spreads the packets of long flows to all paths. ECMP, RepFlow and Freeway do not effectively utilize all available multiple paths. As a packet-level scheme, CAPS flexibly adjusts the number of paths for long flows according to the ON/OFF mode of short flows. When the short flows are in OFF mode, CAPS scatters the packets of long flows to all paths, significantly improving the throughput performance of long flows compared to the flow-level schemes. Moreover, since the threshold for large flows in CAPS-dark is 100KB, which is much smaller that the long flow size, CAPS-dark has the similar performance to CAPS.

IX. Related Works

Recent data center architecture uses a large number of commodity switches organized as multi-rooted tree with multiple paths from the sender to the receiver. To leverage these multiple paths, ECMP [8] is widely used in current fabrics due to its simplicity. However, the hash collisions problem in ECMP leads to the traffic imbalance if a few long flows exist. To address this shortcoming, the centralized flow

scheduling architecture named Hedera [25] uses the global first fit algorithm to schedule elephant flows to uncongested paths. MultiPath TCP (MPTCP) [26] protocol splits the long TCP flows in subflows across available paths. Furthermore, FMTCP [9] is proposed to use network coding to alleviate the impact of path diversity. Compared with these flow-based traffic splitting schemes, RPS works on the packet-level. RPS randomly spreads all packets along different paths, achieving better load balance and network utilization. However, RPS easily brings about the TCP out-of-order problem, potentially triggering the suboptimal performance.

Various proposals target differentiating short and long flows to provide low latency for short flows. L^2DCT [27] achieves the LAS scheduling discipline at the sender in a distributed way. According to the flow size that has been sent, L^2DCT distinguishes the long and short flows and assigns higher bandwidth to the short flows, thereby reducing the average flow completion time. Based on the multi-path diversity, many multipath transmission schemes are proposed. RepFlow [18] simply replicates each short flow to exploit multiple paths diversity to minimize flow completion times. Based on the RPS and MPTCP, MMPTCP [28] spreads the packet of short flows to reduce the FCT and transmit the long lows by MPTCP to improve their throughputs. Freeway [23] adaptively isolate the short and long flows on different transmission paths to reduce the impact of two types of flows.

In contrast with the multipath transmission schemes on the flow-level, our solution CAPS works through a different perspective: we isolate the two kinds of flows on the packet-level to avoid the head-of-line blocking and introduce the FEC coding to solve the TCP out-of-order for short flows. Meanwhile, CAPS flexibly switches the packets of long flows to idle paths to obtain high throughput.

X. Conclusion

To mitigate the negative impact of short and long flows on each other in data center networks, we propose CAPS, a coding-based adaptive packet spraying design that reduces the flow completion time for short flows and guarantees the throughputs for long flows. CAPS utilizes the FEC code to encode only the short flows and spreads the packets of short flows to all equal cost multipath. CAPS limits the long flows when coexisting with the short flows to avoid the head-of-line problem and scatters the packets of long flows to the unused paths by the short flows to achieve high throughput. We evaluate CAPS with both NS2 simulations and small-scale Mininet testbed. The results indicate that CAPS significantly reduces the AFCT by ∼30%-70% for short flows and achieves high throughput for long flows with negligible traffic overhead.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (61572530, 61629302, 61502539 and 61420106009) and CERNET Innovation Project (Grant No. NGII201601130).

References

[1] A. Dixit, P. Prakash, Y. C. Hu, and R. R. Kompella. On the impact of packet spraying in data center networks. In Proc. IEEE INFOCOM, 2013.
[2] Per packet load balancing. http://www.cisco.com/c/en/us/td/docs/ios/12_0s/ feature/guide/pplb.pdf?dtid=osscdc000283
[3] T. Benson, A. Akella, and D. Maltz. Network traffic characteristics of data centers in the wild. In Proc. IMC, 2010.
[4] L. Chen, K. Chen, W. Bai, et al. Scheduling mix-flows in commodity datacenters with karuna. In Proc. ACM SIGCOMM, 2016.
[5] J. Perry, A. Ousterhout, H. Balakrishnan, et al. Fastpass: A centralized "zero-queue" datacenter network. In Proc. ACM SIGCOMM, 2014.
[6] G. Chen, Y. Lu, Y. Meng, et al. Fast and cautious: leveraging multi-path diversity for transport loss recovery in data centers. In Proc. USENIX NSDI, 2016.
[7] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. Data center TCP (DCTCP). In Proc. ACM SIGCOMM, 2010.
[8] K. He, E. Rozner, K. Agarwal, W. Felter, J. Carter, and A. Akella. Presto: Edge-based load balancing for fast datacenter networks. In Proc. ACM SIGCOMM, 2015.
[9] Y. Cui, X. Wang, H. Wang, et al. FMTCP: A fountain code-based multipath transmission control protocol. In Proc. ICDCS, 2012.
[10] C. Jiang, D. Li, M. Xu. LTTP: An LT-code based transport protocol for many-to-one communi-cation in data centers. IEEE Journal on Selected Areas in Communications, 32(1):52-64, 2014.
[11] J. K. Sundararajan, D. Shah, M. Medard, M. Mitzenmacher, and J. Barros. Network coding meets TCP. In Proc. IEEE INFOCOM, 2009.
[12] D. Declercq, M. Fossorier. Decoding algorithms for nonbinary LDPC codes over GF (q). IEEE Transactions on Communications, 55(4):633-643, 2007.
[13] M. Luby. LT codes. In Proc. IEEE Conf. Foundations of computer science, 2002.
[14] O. C. Kwon, Y. Go, Y. Park, et al. MPMTP: Multipath multimedia transport protocol using systematic raptor codes over wireless networks. IEEE Transactions on Mobile Computing, 14(9):1903-1916, 2015.
[15] S. Ahmad, R. Hamzaoui, M. Al-Akaidi. Adaptive unicast video streaming with rateless codes and feedback. IEEE Transactions on Circuits and Systems for Video Technology, 20(2):275-285, 2010.
[16] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. T. Lam, F. Matus, R. Pan, N. Yadav, and G. Varghese. Conga: Distributed congestion-aware load balancing for datacenters. In Proc. ACM SIGCOMM, 2014.
[17] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, and H. Wang. Information-agnostic flow scheduling for commodity data centers. In Proc. USENIX NSDI, 2015.
[18] H. Xu, B. Li. RepFlow: Minimizing flow completion times with replicated flows in data centers. In Proc. IEEE INFOCOM, 2014.
[19] N Handigol, B. Heller, V. Jeyakumar, B. Lantz, N. McKeown. Reproducible network experiments using container-based emulation. In Proc. ACM CoNEXT, 2012.
[20] A. Khurshid, W. Zhou, M. Caesar, et al. Veriflow: Verifying network-wide invariants in real time. In Proc. USENIX NSDI, 2013.
[21] J. Huang, T. He, Y. Huang, J. Wang. ARS: Cross-Layer adaptive request scheduling to mitigate TCP incast in data center networks. In Proc. IEEE INFOCOM, 2016.
[22] A. R. CurtisR, W. Kim, P. Yalagandula. Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection. In Proc. IEEE INFOCOM, 2011.
[23] W. Wang, Y. Sun, K. Salamatian, et al. Freeway: Adaptively isolating the elephant and mice flows on different transmission paths. In Proc. ICNP, 2014.
[24] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, P. Patel, and S. Sengupta. VL2: A scalable and flexible data center network. In Proc. ACM SIGCOMM, 2009.
[25] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic flow scheduling for data center networks. In Proc. USENIX NSDI, 2010.
[26] C. Raiciu, S. Barre, C. Pluntke, et al. Improving datacenter performance and robustness with multipath TCP. In Proc. ACM SIGCOMM, 2011.
[27] A. Munir, I. A. Qazi, Z. A. Uzmi, A. Mushtaq, S. N. Ismail, M. S. Iqbal, and B. Khan. Minimizing flow completion times in data centers. In Proc. IEEE INFOCOM, 2013.
[28] M. Kheirkhah, I. Wakeman, G. Parisis. MMPTCP: A multipath transport protocol for data centers. In Proc. IEEE INFOCOM, 2016.