Prioritized Adaptive Max-Min Fair Residual Bandwidth Allocation for Software-Defined Data Center Networks

Andrew Lester
School of Information Technology
Illinois State University
aeleste@ilstu.edu

Yongning Tang
School of Information Technology
Illinois State University
ytang@ilstu.edu

Tibor Gyires
School of Information Technology
Illinois State University
tbgyires@ilstu.edu

***Abstract*—Modern data center networks commonly adopt multi-rooted tree topologies. Equal-Cost Multi-Path (ECMP) forwarding is often used to achieve high link utilization and improve network throughput. Meanwhile, max-min fairness is widely used to allocate network bandwidth fairly among multiple applications. However, today's data centers usually host diverse applications, which have various priorities (e.g., mission critical applications) and service level agreements (e.g., high throughput). It is unclear how to adopt ECMP forwarding and max-min fairness in the presence of such requirements. We propose Prioritized Max-Min Fair Multiple Path forwarding (PMP) to tackle this challenge. PMP can optimally allocate current available bandwidth to maximally satisfy user demands. When predefined application requirements are available, PMP can prioritize current demands and allocate available bandwidth accordingly. Our performance evaluation results show that PMP can improve application throughput 10-12% on average and increase overall link utilization especially when the total demanded bandwidth is close or even exceeds the bisectional bandwidth of a data center network.**

Keywords*- *SDN, max-min fair, scheduling.

I. Introduction

In recent years, data centers have become critical components for many large organizations [1] [2] [19]. A data center (DC) refers to any large, dedicated cluster of computers that is owned and operated by a single authority, built and employed for a diverse set of purposes. Large universities and private enterprises are increasingly consolidating their IT services within on-site data centers containing a few hundred to a few thousand servers. On the other hand, large online service providers, such as Google, Microsoft, and Amazon, are rapidly building geographically diverse cloud data centers, often containing more than 10,000 servers, to offer a variety of cloud-based services such as web servers, storage, search, on-line gaming. These service providers also employ some of their data centers to run large-scale data-intensive tasks, such as indexing Web pages or analyzing large data-sets, often using variations of the MapReduce paradigm.

Many data center applications (e.g., scientific computing, web search, MapReduce) require substantial bandwidth. With the growth of bandwidth demands for running various user applications, data centers also continuously scale the capacity of the network fabric for new all-to-all communication patterns, which presents a particular challenge for traditional data forwarding (switching and routing) mechanisms. For example, MapReduce based applications, as a currently adopted default computing paradigm for big data, need to perform significant data shuffling to transport the output of its map phase before proceeding with its reduce phase. Recent study shows the principle bottleneck in large-scale clusters is often inter-node communication bandwidth. Traffic pattern study [17] showed that only a subset (25% or less) of the core links often experience high utilization.

Modern data center networks commonly adopt multi-rooted tree topologies [1] [2] [19]. ECMP is often used to achieve high link utilization and improve network throughput. Meanwhile, max-min fairness is widely used to allocate network bandwidth fairly among multiple applications. Many current data center schedulers, including Hadoops Fair Scheduler [26] and Capacity Scheduler [25], Seawall [24], and DRF [27], provide max-min fairness. The attractiveness of max-min fairness stems from its generality. However, today's data centers usually host diverse applications, which have various priorities (e.g., mission critical applications) and service level agreements (e.g., high throughput). It is unclear how to adopt ECMP forwarding and max-min fairness in the presence of such requirements. We propose Prioritized Max-Min Fair Multiple Path forwarding (PMP) to tackle this challenge. PMP can optimally allocate current available bandwidth to maximally satisfy user demands. When predefined application requirements are available, PMP can prioritize current demands and allocate available bandwidth accordingly.

The disruptive Software-Defined Networking (SDN) technology shifts today's networks that controlled by a set of vendor specific network primitives to a new network paradigm empowered by new programmatic abstraction. OpenFlow provides a protocol such that the logical centralized controller can exploit forwarding tables on SDN switches for programmatic multi-layer forwarding flexibility. One of the fundamental transformations that flow based forwarding presents is the inclusion of multi-layer header information to make forwarding match and action logic programmatically. Programmatic policy is vital to manage the enormous combinations of user requirements. For example, an SDN controller can flexibly define a network flow using a tuple as (incoming port, MAC Src, MAC Dst, Eth Type, VLAN ID, IP Src, IP Dst, Port Src, Port Dst, Action).

The rest of the paper is organized as the following. Section II discusses the related research work. Section III formalizes the problem and describes our approach. Section IV presents our simulation design and results, respectively. Finally, Section V concludes the paper with future directions.

II. Related Work

Current large data center networks connect multiple Ethernet LANs using IP routers and run scalable routing algorithms over a number of IP routers. These layer 3 routing algorithms allow for shortest path and ECMP routing, which provide much more usable bandwidth than Ethernets spanning tree. However, the mixed layer 2 and layer 3 solutions require significant manual configuration.

The trend in recent works to address these problems is to introduce special hardware and topologies. For example, PortLand [2] is implementable on Fat Tree topologies and requires ECMP hardware that is not available on every Ethernet switch. TRILL [3] introduces a new packet header format and thus requires new hardware and/or firmware features.

There have been many recent proposals for scale-out multi-path data center topologies, such as Clos networks [4], [5], direct networks like HyperX [6], Flattened Butterfly [8], DragonFly [8]. etc.,, and even randomly connected topologies have been proposed in Jellyfish [9].

Many current proposals use ECMP-based techniques, which are inadequate to utilize all paths, or to dynamically load balance traffic. Routing proposals for these networks are limited to shortest path routing (or K-shortest path routing with Jellyfish) and end up underutilizing the network, more so in the presence of failures. While DAL routing [6] allows deroutes, it is limited to HyperX topologies. In contrast, Dahu [19] proposes a topology-independent, deployable solution for non-minimal routing that eliminates routing loops, routes around failures, and achieves high network utilization.

Hedera [10] and MicroTE [13] propose a centralized controller to schedule long lived flows on globally optimal paths. However they operate on longer time scales and scaling them to large networks with many flows is challenging. Techniques like Hedera, which select a path for a flow based on current network conditions, suffer from a common problem: when network conditions change over time the selected path may no longer be the optimal one. While DevoFlow [14] improves the scalability through switch hardware changes, it does not support non-minimal routing or dynamic hashing. Dahu can co-exist with such techniques to better handle congestion at finer time scales.

MPTCP [11] proposes a host based approach for multi-path load balancing by splitting a flow into multiple sub flows and modulating how much data is sent over different subflows based on congestion. However, as a transport protocol, it does not have control over the network paths taken by subflows. Dahu [19] exposes the path diversity to MPTCP and enables MPTCP to efficiently utilize the non-shortest paths in a direct connect network. There have also been proposals that employ variants of switch-local per-packet traffic splitting [20].

Traffic engineering has been well studied in the context of wide area networks. TeXCP [21], and REPLEX [22] split flows on different paths based on load. However, their long control loops make them inapplicable in the data center context that requires faster response times to deal with short flows and dynamic traffic changes. FLARE [7] exploits the inherent burstiness in TCP flows to schedule “flowlets” (bursts of packets) on different paths to reduce extensive packet reordering.

III. Prioritized Adaptive Max-Min Fair Bandwidth Allocation

While ECMP is often used to achieve high link utilization, max-min fairness is widely used to allocate network bandwidth fairly among multiple applications. However, today’s data centers usually host diverse applications, which have various priorities (e.g., mission critical applications) and service level agreements (e.g., high throughput). It is unclear how to adopt ECMP forwarding and max-min fairness in the presence of such requirements. We propose Prioritized Max-Min Fair Multiple Path forwarding (PMP) to tackle this challenge. In the following, we first formalize the problem, and then present how PMP works.

A. Problem Formalization

Consider a data center network with K-ary fat-tree topology as shown in Fig.1, composed of a set of core switches S_c, a set of aggregation switches S_a, a set of edge switches S_e, and a set of hosts H. Each switch has k-port. There are k pods. Each pod contains $k/2$ aggregation switches and $k/2$ edge switches. In each pod, each k-port edge switch is directly connected to $k/2$ hosts and $k/2$ aggregation switches. The i^{th} port of each core switch $s_i \in S_c(i \in [1,(k/2)^2])$ is connected to pod i [2]. We assume all links (e.g., L_1 in Fig.1) have the same bandwidth for both uplink (e.g., L_1^u) and downlink (e.g., L_1^d) connections.

Recent study [17] showed that less than 25% of the core links have been highly utilized while packet losses and congestions may still often occur. In this paper, we only focus on inter-pod network traffic that requires bandwidth from core links. We denote all links between aggregation and core layers as a set L_{ac}, all links between edge and aggregation layers as a set L_{ea}, and all links between application server and edge layers as a set L_{se}. Generally,

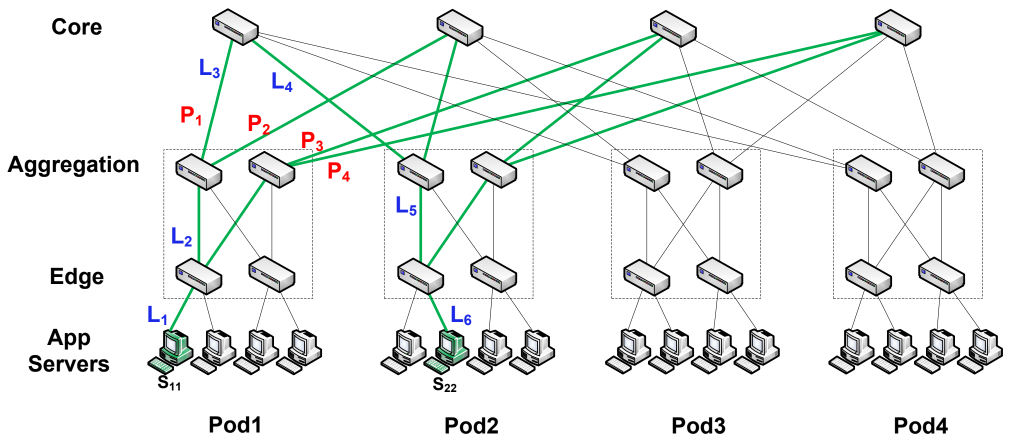


Figure 1: Fat tree topology.

in a network with K-ary fat-tree topology , there are k paths between any two hosts from different pods.

A network task T_i is specified by a source and destination hosts (e.g., S_{11} and S_{22}) and the expected traffic volume. We also consider each task with different priority level w_i. Here, $w_i \in [1, m]$ with the lowest and highest priority levels as 1 and m, respectively. A network scheduler modular (also simply referred to as scheduler) on an SDN controller needs to decide how to allocate available bandwidth to maximally satisfy the application requirements. We define a valid Network Path Assignment PA_i for a given task T_i is a set of paths and their corresponding allocated bandwidths connecting the source to the destination (e.g., a subset of $\{P_1, P_2, P_3, P_4\}$), in which each path consists of a list of directional links (e.g., $P_1 = \{L_1^u, L_2^u, L_3^u, L_4^d, L_5^d, L_6^d\}$) connecting the source to the destination hosts. Here, $L_1^u, L_6^d \in L_{se}$; $L_2^u, L_5^d \in L_{ea}$; $L_3^u, L_4^d \in L_{ac}$.

There is a variety of applications on a data center network, which have different service requirements regarding throughput, packet loss, and delay. For our analysis, we characterize the applications' requirements through their priority levels, which can be the output of some utility function. Priorities can offer a basis for providing application and business oriented service to users with diverse requirements. We consider a model where the weight associated with the different priority classes is user-definable and static. Users can freely define the priority of their traffic, but are charged accordingly by the network. We aim to study the bandwidth-sharing properties of this priority scheme. Given a set of network tasks $T = \{T_i\}$ $(i \geq 1)$ and their corresponding priority levels $K = \{K_i\}$, we consider a Network Path Assignment problem is to find a set of path assignment $PA = \{PA_i\}$ to satisfy the condition of Prioritized Max-Min Fairness.

Definition 1. Prioritized Max-Min Fairness A feasible path assignment PA^x is "prioritized max-min fair" if and only if an increase of any path bandwidth within the domain of feasible bandwidth allocations must be at the cost of a decrease of some already less allocated bandwidth from the tasks with the same or higher priority level. Formally, for any other feasible bandwidth allocation scheme PA^y, if $BW(PA^y_{T_i}) > BW(PA^x_{T_i})$, then it decreases the allocated bandwidth of some other path with the same or higher priority level. Here, $BW(PA^y_{T_i})$ is the total allocated bandwidth for the task T_i in the bandwidth allocation scheme PA^y.

Definition 2. Saturated Path A path P_i is saturated if at least one bottleneck link L_j exists on the path P_i. A link is bottlenecked if the total assigned bandwidth on this link from the given tasks is more than or equal to the maximum bandwidth of the link. Formally, a bottleneck link is the one that $\sum_i BW_{T_i}(L_j) \geq BW_{max}(L_j)$.

B. The Algorithm of Multi-Level Progressive Filling

The network tasks can be dynamically and continuously generated, and submitted to the scheduler. In PMP, the scheduler can periodically query all network switches to collect current link utilizations. Once a new task list received, the scheduler will use a practical approach called "progressive filling" [23] provisioning available bandwidth that results in a prioritized max-min fair allocation following the priority order from the highest to the lowest priority level. The idea is shown in Fig. 2: The scheduler starts with all provisioned bandwidth equal to 0 and increases all bandwidths together at the same pace for the tasks with the same priority level until one or several saturated paths are found. The bandwidth for the corresponding tasks that use these paths are not increased any more and the scheduler continue increasing the bandwidth for other tasks on the same priority level. All the tasks that are stopped have a saturated path. The algorithm continues until it is not possible to increase the bandwidth for the tasks at certain priority level. Then, the algorithm moves to the next

```
Input: A list of tasks $\{T_i\}$; current link utilization $U(L_j)$
Output: Path assignment $PA$ with $PA_i$ for each task $T_i$

 1: Sort $\{T_i\}$ based on their priority levels $K_i$
 2: Start from the highest priority $W = m$ /*$m$ is the
    highest priority level*/
 3: for all $T_i != \emptyset$ $(PL(T_i) = W)$ do
 4:     /*The function $PL()$ returns the priority level of a
        given task*/
 5:     Find all paths for each task $T_i$
 6:     Assign a unit bandwidth (UB) to the least utilized
        path for each task /*we choose UB = 100Kbps*/
 7:     $PA_i \leftarrow \{T_i, \{P_i\}\}$
 8:     $PA \leftarrow PA \cup \{PA_i\}$
 9:     if A path $P$ is saturated and $P \in APL(T_i)$ then
10:         $APL(T_i) \leftarrow APL(T_i) - P$
11:     end if
12:     if $APL(T_i) == \emptyset$ then
13:         Remove $T_i$
14:     end if
15:     if $(\{T_i\} == \emptyset)$ and $(W > 1)$ then
16:         $W = m - 1$
17:     end if
18: end for
19: return $PA$
```

Figure 2: Multi-Level Progressive Filling Algorithm

priority level and repeats the same bandwidth provisioning operations until all tasks are assigned to some paths. The algorithm terminates because the total paths and tasks are finite. When the algorithm terminates all tasks have been served at some time and thus have a saturated path. By Definition 1 the allocation is max-min fair for the tasks at the same priority level.

IV. Evaluation

A. Data Center Network Traffic Pattern

Several recent studies [16]–[18] have been conducted in various data center networks to understand network traffic patterns. The studied data center networks include university campus, private enterprise data centers, and cloud data centers running Web services, customer-facing applications, and intensive Map-Reduce jobs. The studies have shown some interesting facts: (1) The majority of the traffic in data center networks is TCP flows. (2) Most of the server generated traffic in the cloud data centers stays within a rack, while the opposite is true for campus data centers. (3) At the edge and aggregation layers, link utilizations are fairly low and show little variation. In contrast, link utilizations at the core network are high with significant variations over the course of a day. (4) In some data centers, a small but significant fraction of core links appear to be persistently congested, but there is enough spare capacity in the core to alleviate congestion. (5) Losses on the links that are lightly utilized on the average can be attributed to the bursty nature of the underlying applications run within the data centers.

B. Methodology and Metrics

In our experiments, we simulate a data center with a fat-tree topology. We implemented PMP based on RipL [28], a Python library that simplifies the creation of data center code, such as OpenFlow network controllers, simulations, or Mininet topologies. We compared PMP scheduler with a commonly used randomization based scheduling method.

In our evaluation, we use three different priority policies for a mixture of traffic patterns: (1) high priority for long TCP flows with the total data size between $1MB$ and $100MB$; (2) high priority for short TCP flows with the total data size between $10KB$ and $1MB$; (3) high priority for random selected flows including both short and long ones referred to as mixed TCP flows.

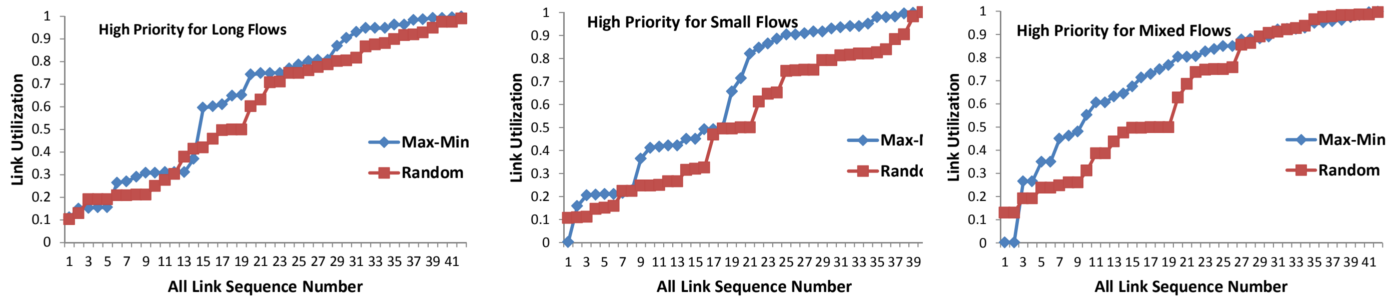
We focus on two performance metrics: (1) Link Utilization that demonstrates how effectively the scheduler utilizes the network bandwidth. Intuitively, when there are high bandwidth demands from user applications, the overall link and path utilizations should be kept in high. (2) Network throughput that shows how efficiently the network serves different applications.

C. Link Utilization

We created 16 test scenarios to evaluate PMP with different inter-pod traffic patterns. We ran 5 tests for each scenarios. In all test scenarios, the test traffic traversed all edge, aggregation, and core links. The results of multiple test runs from the same test scenario present similar results. In the following, we only report the result of one test run for each test scenario that created traffic between two pods in both directions. Under the same three different priority policies, Fig.3(a)$\sim$(c) shows the overall path utilization; Fig.4(a)$\sim$(c) shows the aggregation link utilizations; and Fig.5(a)$\sim$(c) shows the core link utilizations. Comparing to the randomization based scheduler, our algorithm 2 achieves high utilization on path level, aggregation and core link levels by: (1) dynamically observing all link utilization status, and (2) progressively filling the jobs of the same priority with the available bandwidth with the max-min fairness. The average gain on utilization is approximately improved from 59% to 66%. Note that with the increase of link utilization, idle bandwidth can be effectively utilized by demanding network applications, which can correspondingly improve their performance by reducing their network latencies.

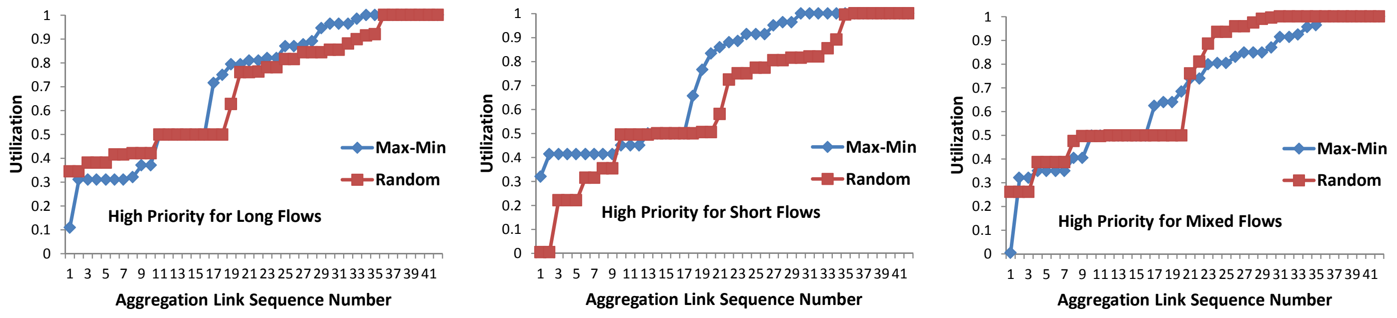
D. Network Throughput

Once the overall utilization can be increased, we expect that the overall application throughput should also be improved. The experiment results presented some interesting



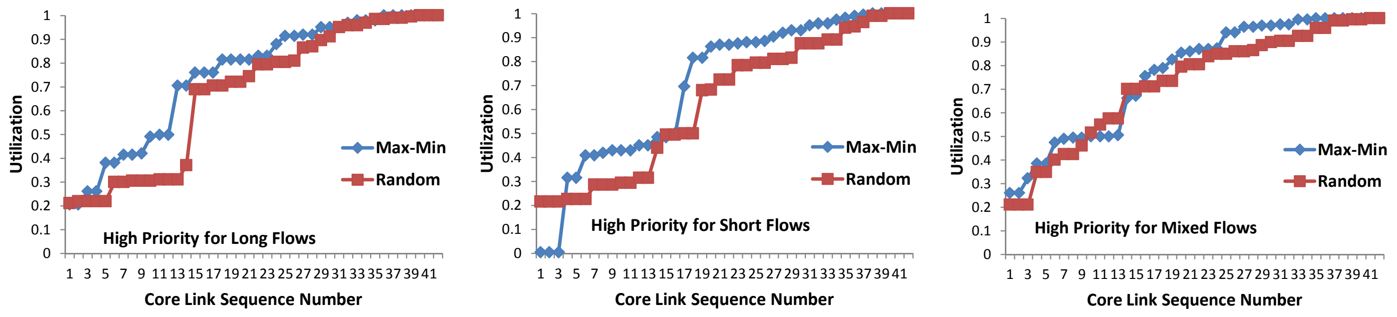
(a) (b) (c)

Figure 3: Path utilization with high priority for (a) long flows (b)short flows (c)mixed flows



(a) (b) (c)

Figure 4: Aggregation link utilization with high priority for (a) long flows (b)short flows (c)mixed flows



(a) (b) (c)

Figure 5: Core link utilization with high priority for (a) long flows (b)short flows (c)mixed flows

results as shown in Fig.6(a)∼(c). When we emulate more realistic application scenarios, where short and long TCP flows are randomly mixed together, our PMP scheduler obviously outperforms the performance of the random scheduler with about 10-12% improvement. In the scenario of the different policies favoring either short or long flows, our scheduler adopts max-min fairness, and thus, the average throughput has been improved from $2.52Mbps$ in the random scheduler and to $3.46Mbps$ in the max-min scheduler.

V. Conclusion

The role of the data center network is becoming ever more crucial today, which is evolving into the integrated platform for next-generation data centers. Because it is pervasive and scalable, the data center network is developing into a foundation across which information, application services and all data center resources, including servers, storage are shared, provisioned, and accessed. Modern data center networks commonly adopt multi-rooted tree topologies. ECMP is often used to achieve high link utilization and improve network throughput. Meanwhile, max-min fairness is widely used to allocate network bandwidth fairly among multiple applications. However, today's data centers usually host diverse applications, which have various priorities (e.g., mission critical applications) and service level agreements (e.g., high throughput). It is unclear how to adopt ECMP forwarding and max-min fairness in the presence of such requirements. We propose Prioritized Max-Min Fair Multi-

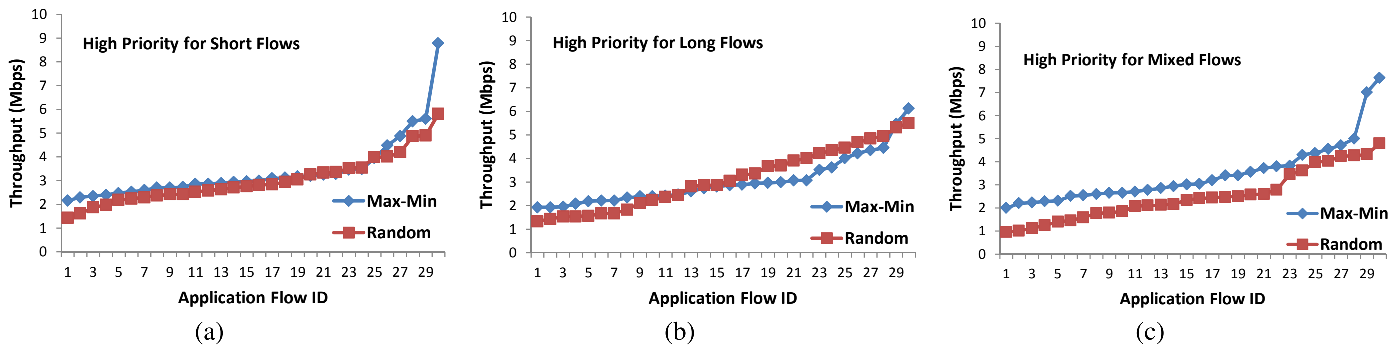


Figure 6: Throughput with high priority for (a) long flows (b)short flows (c)mixed flows

ple Path forwarding (PMP) to tackle this challenge. PMP can prioritize current demands and allocate available bandwidth accordingly. Our performance evaluation results show that PMP can improve application throughput 10-12% on average and increase overall link utilization especially when the total demanded bandwidth close or even exceed the bisectional bandwidth of a data center network.

REFERENCES

[1] M. Al-Fares, A. Loukissas, and A. Vahdat, A Scalable, Commodity Data Center Network Architecture. In SIGCOMM, 2008

[2] R. N. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, PortLand: A scalable fault-tolerant layer 2 data center network fabric. In SIGCOMM, 2009

[3] R. Perlman, Rbridges: Transparent routing. In INFOCOMM, 2004.

[4] 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 Proc. of ACM SIGCOMM, 2009.

[5] V. Liu, D. Halperin, A. Krishnamurthy, and T. Anderson, F10: A Fault-Tolerant Engineered Network. In NSDI, 2013

[6] J. H. Ahn, N. Binkert, A. Davis, M. McLaren, and R. S. Schreiber, HyperX: Topology, Routing, and Packaging of Efficient Large-Scale Networks. In Proc. of SC, 2009.

[7] J. Kim, W. J. Dally, and D. Abts, Flattened butterfly: A Cost-efficient Topology for High-radix networks. ISCA, 2007.

[8] J. Kim, W. J. Dally, S. Scott, and D. Abts, Technology-Driven, Highly-Scalable Dragonfly Topology. In Proc. of ISCA, 2008

[9] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, Jellyfish: Networking Data Centers Randomly. In NSDI, 2012.

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

[11] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, Data center TCP (DCTCP). In Proc. of ACM SIGCOMM, 2010.

[12] T. Benson, A. Akella, and D. A. Maltz, Network Traffic Characteristics of Data Centers in the Wild. IMC, 2010.

[13] T. Benson, A. Anand, A. Akella, and M. Zhang, MicroTE: Fine Grained Traffic Engineering for Data Centers. In Proc. of ACM CoNEXT, 2011.

[14] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, DevoFlow: Scaling Flow Management for High-Performance Networks. In Proc. of ACM SIGCOMM, 2011.

[15] OpenFlow Switch Specification (Version 1.1), http://www.openflow.org/documents/openflow-spec-v1.1.0.pdf. (retrieved: Sept. 2013)

[16] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, The Nature of Data Center Traffic: Measurements and Analysis. In IMC, 2009.

[17] T. Benson, A. Akella, and D. A. Maltz, Network Traffic Characteristics of Data Centers in the Wild. In IMC, 2010.

[18] G. Wang, D. G. Andersen, M. Kaminsky, M. Kozuch, T. S. E. Ng, K. Papagiannaki, M. Glick, and L. Mummert, Your data center is a router: The case for reconfigurable optical circuit switched paths. In Hotnets, 2009.

[19] S. Radhakrishnan, M. Tewari, R. Kapoor, G. Porter, and A. Vahdat, Dahu: Commodity Switches for Direct Connect Data Center Networks. In Proceedings of the 9th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS13), October 2013

[20] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. Katz, DeTail: Reducing the Flow Completion Time Tail in Datacenter Networks. In SIGCOMM, 2012.

[21] S. Kandula, D. Katabi, B. Davie, and A. Charny, Walking the Tightrope: Responsive Yet Stable Traffic Engineering. In SIGCOMM, 2005.

[22] S. Fischer, N. Kammenhuber, and A. Feldmann, REPLEX: Dynamic Traffic Engineering Based on Wardrop Routing Policies. In CoNEXT, 2006.

[23] A. Ghodsi, M. Zaharia, S. Shenker and I. Stoica, Choosy: Max-Min Fair Sharing for Datacenter Jobs with Constraints, EuroSys 2013.

[24] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, I. Stoica, and S. Shenker, Dominant resource fairness: Fair allocation of multiple resource types. In NSDI, 2011

[25] Hadoop Capacity Scheduler. hadoop.apache.org/common/docs/r0.20.2/capacity_scheduler.html. (retrieved: Sept. 2013)

[26] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling. In EuroSys 10, 2010.

[27] A. Shieh, S. Kandula, A. Greenberg, C. Kim, and B. Saha, Sharing the data center network. In NSDI, pages 2323, 2011.

[28] M. Casado, D. Erickson, I. A. Ganichev, R. Griffith, B. Heller, N. Mckeown, D. Moon, T. Koponen, S. Shenker, and K. Zarifis, Ripcord: A modular platform for data center networking. UC, Berkeley, Tech. Rep. UCB/EECS-2010-93