Software-Defined Traffic Load Balancing for Cost-Effective Data Center Interconnection Service

Young-Jin Kim
Bell Labs, Nokia
Murray Hill, NJ, USA
young.kim@nokia.com

Jesse E. Simsarian
Bell Labs, Nokia
Crawford Hill, NJ, USA
jesse.simsarian@nokia.com

Marina Thottan
Bell Labs, Nokia
Murray Hill, NJ, USA
marina.thottan@nokia.com

***Abstract* — For interconnection between geographically-separated data centers, network carriers typically implement multiple optical paths in a wide-area network. For example, when transmission wavelengths have 100 Gb/s granularity, three 100 Gb/s wavelength paths are provisioned to satisfy a customer demand of 300 Gb/s. Over the multiple provisioned paths, interconnection traffic is typically distributed using per-flow hashing, which results in an uneven distribution of traffic caused by hash collisions. For a relatively-few number of high-bandwidth traffic flows (> 1 Gb/s) between data center locations, per-flow hashing can perform poorly in terms of bandwidth utilization and availability.**

We propose new software-defined traffic load balancer, *SD-TLB*, that performs measurement-based flow distribution over multiple optical paths, with an implicit impairment detection method using per-port statistics on available paths and a flow redistributor that is immediately adjusted to the current network state. While our approach does not provide the same level of protection as 1+1 optical protection, it can provide the necessary redundancy for data center inter-connection at a lower cost. We experimentally implement the *SD-TLB* using ASIC-based switches and open virtual switches interconnected by wavelength-division multiplexed transport network test-bed. The experimental results show that *SD-TLB* outperforms today's hashing–based alternatives in balancing, throughput, and restoration in the presence of outages and impairments and as a result achieves improved cost-efficiency.

***Keywords* — Data Center Optical-Interconnect, Traffic Load Balancing, Outage and Impairment Protection.**

I. INTRODUCTION

Extra-large enterprise networks and cloud providers operating multiple data centers are increasingly exchanging high-volumes of data between their data center (DC) sites for data caching and replication, storage backup, and load balancing. DC interconnects (DCIs) [1] are often connected via metro-optical connections (up to 500 km). As shown in Fig. 1, a single DCI is provisioned on multiple wavelengths (*i.e.*, optical-circuits) over optical transport network (OTN) or a dense wavelength-division multiplex (DWDM) [2].

Network carriers as DCI providers do not typically expose optical path diversity information to their customer DCI users (*i.e.,* DC operators) since they offer DCIs as a service, *e.g.*, 300G bidirectional link service. As described in [3], for a pair of DCs, interconnection traffic is usually distributed over multiple optical-circuit paths using a traffic load balancer (TLB) that relies on *per-flow hashing* such as hash-threshold [5]. It is known that no per-flow hashing scheme, where a packet arriving at a network's ingress node is sent over a path decided by a hashing calculation based on its header information, avoids collisions that occur when

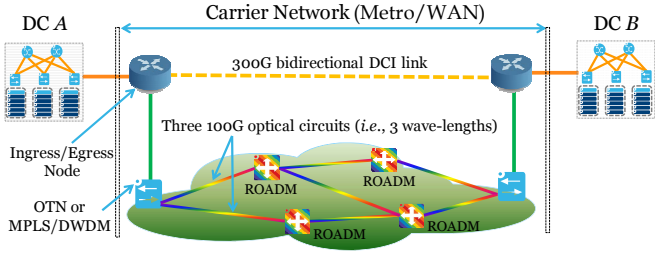

Fig. 1. An example inter-DC connection over a carrier network.

different traffic flows between a given source-destination pair are assigned the same path. With relatively-few high-bandwidth flows between two DC sites [1], per-flow hashing schemes can incur uneven traffic distribution over multiple optical paths resulting in lower bandwidth availability. For a circuit-switched optical path, outage or impairments can cause service degradation (or disruption) when there are no designated protection optical paths. To meet service-level agreements (SLAs) for customers who need high bandwidth availability, network carriers as DCI providers typically provision redundant optical paths for path protection. For example, for a 300G DCI and 100G wavelength granularities, a network carrier establishes six 100G optical paths, three as active paths and three as protection paths, called 1+1 protection. For stringent SLAs, protection paths are exclusively dedicated to the DCI. As a result, customer DCI users pay a high cost for using high-availability DCIs, even though the utilization of DCIs typically does not exceed 50%[1] and high availability of DCIs is not always required.

Motivated by this, we address the problem of building cost-effective DCIs that have the necessary availability for a DCI service at a lower cost. As a solution, we propose a new software-defined traffic load balancer SD-TLB that consists of 1) a round-robin based flow scheduler and 2) a fast self-restoration scheme against outage and impairments, based on measured per-port statistics. SD-TLB was built on the CPU of physical (*i.e.,* ASIC-based forwarding) SDN switches [8], as well as on open virtual switches [9].

Our major contributions are as follows. First, we show through experiments with multiple 10G optical paths that link aggregation group (LAG) [4] (today's TLB industry practice) and OpenFlow group tables [10] (an available

[1] According to [6-7], the DC operators on an average utilize only 20-30% of the DCIs that are over provisioned due to a high peak-to-mean traffic ratio. When network carriers have protection paths to protect the DCIs, the utilization from the view point of carriers can decrease to 10-15%.

SDN-based TLB) that rely on per-flow hashing are limited in providing high bandwidth availability where each flow on average demands more than 1G bandwidth. In addition, since SD-TLD has awareness of Multipath TCP (MPTCP) [11] that creates multiple TCP sub-flows per TCP connection to allow the use of multiple paths, path collision among MPTCP sub-flows in a TCP connection is avoided. Therefore, we can make TCP traffic flows evenly distributed over multiple optical paths. More importantly, we provide the necessary redundancy for DCIs by exploiting measured per-port statistics to distribute flows, thus mitigating the need for dedicated protection paths. Furthermore, our implementation addresses practical issues such as the limited flow-table size and flow-setup latency inherent to ASIC-based SDN switches [8]. Our experimental setup evaluates SD-TLB in the face of real outage (*i.e*., cable plug-out) and impairment scenarios (*i.e*., high bit error ratio (BER) on an optical path [12]) in a 30G DCI over a 3 x 10G optical network (See Fig. 5).

II. RELATED WORK

Load balancing, a central aspect of traffic engineering that exploits a logical aggregation group combining multiple physical paths (or connections) for higher bandwidth and redundancy against failures, has been the subject of several prior studies [3-7] [10-11] [13] [21-22]. We note that this paper focuses on a new cost-effective TLB for metro and long-haul optical networking rather than for general data-center networking.

Most of the prior studies rely on per-flow hashing (PFH) or per-packet round-robin (PPR) to distribute traffic over a group of combined multiple paths. In PFH, all packets in a flow go through the same path determined by hashing per-packet although multiple paths are available. However, PFH inherently incurs uneven traffic distribution that results in service degradation, *i.e.,* lower bandwidth availability. It can provide statistically good aggregate throughput when there are a large number of flows but can show a high variance of per-flow throughput when there are a small number of high-bandwidth flows, as shown in Sec. 5. With PPR, where the forwarding decision is made in a round-robin fashion, a network node evenly distributes packets over all paths in an aggregation group regardless of the flow a packet belongs to. All the paths are assumed to be of equal cost such as delay and thus need be established over the same fiber cable. 1+1 protection is required for PPR or service disruption occurs when the fiber cable fails.

GMPLS (Generalized Multiprotocol Label Switch) load sharing group [3], LAG [4], and OpenFlow group table [10] use either PFH or PPR; ECMP (Equal-Cost Multi-Path routing) [13] uses PFH; OIF (Optical Internetworking Forum) FlexEthernet super-rate [21] uses only PPR. Note that PPR-based schemes are not compared in the rest of this paper since SD-TLB is effective regardless of whether all paths in an aggregation group are of equal cost or not. ECMP is commonly used for multi-hop packet networks where intermediate nodes between an ingress-egress node pair are Ethernet switches or IP routers where hashing calculation is possible. However, it is not well aligned with optical networking where intermediate optical nodes cannot perform hashing computation and a connection between two packet nodes through optical paths is viewed as a direct link (see Fig.1). In comparison, GMPLS inherently supports optical networking [23]; LAG is a good fit for optical networking because it exploits multiple parallel connections over a single packet-layer hop without hashing at intermediate nodes; OpenFlow group table that provides different group abstractions such as multicasting, multi-pathing, and failover has similar behavior to LAG.

Recent literature on load balancing shows dynamic flow scheduling [6-7] [22] that addresses problems caused by the static hashing of PFH and is implemented using SDN concepts. Large DC operators [6-7], who tried to improve utilization of expensive DCI links (given by network carriers) to slow down the increase of DCI capital expenditure, introduced SDN-enabled ECMP TLBs for inter-DC networking where multiple DC locations are connected to each other through long-haul networks. Recall that network carriers do not expose optical path diversity information to their customers and have their own TLBs (exploiting optical path diversity) for high bandwidth availability of DCI link services (see Fig.1). Thus, their SDN-controlled TLBs have no direct interaction with carriers that provide the DCI links and as a result work over overlay-network graphs that have DC sites as a set of vertices and DCI links as a set of edges. In those works, a centralized global SDN controller, which can monitor the utilization of all DCI links among all DC sites and estimate flow demand, proactively computes a set of rules for ECMP-based flow distribution over multiple DCI links per-site, and installs the rule set to corresponding DC sites that will perform flow distribution. Hedera [22] is a SDN-based dynamic flow scheduler for traffic load balancing in multi-rooted trees such as FatTree [24], a typical DC network topology. Similar to [6-7], it is implemented by a DC-wide global SDN controller that can detect large-size flows and estimate flow demand.

However, prior work [6-7] [22] leveraged by a global SDN controller is limited in support for real-time restoration against failures due to non-negligible delay caused by the physical distances between a global controller and all nodes in a long-haul network, or a high number of nodes in a network. In contrast, for cost-effective DCI traffic distribution where the required availability for DCI links is preserved at a lower cost without 1+1 protection, SD-TLB-enabled nodes supports faster restoration through immediate failure detection and routing-rule computation independent of global SDN controllers. SD-TLB built for TLB in a DCI link (an aggregate of multiple optical paths in a long-haul network) complements the solutions in Refs. [6-7] that describe TLB over multiple DCI links per-DC site with no awareness of the underlying transport-network topology. Compared to Refs. [6-7] [22], our SD-TLB is a *localized* approach for rule computation and a *non-hashing* approach for traffic distribution whereby upon the arrival of a new flow, a SD-

TLB enabled node computes rules locally using optical path information (previously given by a global controller [14]) and taking into consideration the current node state.

Multipath TCP [11] is anew experimental connection-oriented layer-4 protocol that exploits IP address diversity between two end hosts rather than physical path diversity in networks. MPTCP resiliency can be improved by SD-TLB. MPTCP alone cannot overcome limitations inherited from today's TLB practices and therefore works well together with SD-TLB, as shown in Sec. 5. Binder [15] uses a middle-box approach that relies on MPTCP and logical path diversity rather than physical path diversity. In Binder, all messages are intercepted by a Binder middle box and forwarded to a MPTCP protocol stack before being sent to a counter-Binder middle box by an IP source routing scheme. This approach can be used for low-rate wireless networking, but is not suitable for high-bit rate networking.

III. DESIGN OF SD-TLB

SD-TLB is embedded into ingress nodes in carrier networks. Architecturally, SD-TLB interacts with three other components in the case of ASIC-based SDN switches: the internal open virtual switch (OVS), the physical ASIC forwarding table, and a global network controller [14] as shown in Fig. 2. SD-TLB performs two functions: (1) per-flow scheduling that in a round-robin manner searches for candidate paths with the consideration of current switch states, and (2) outage/impairments detection and recovery.

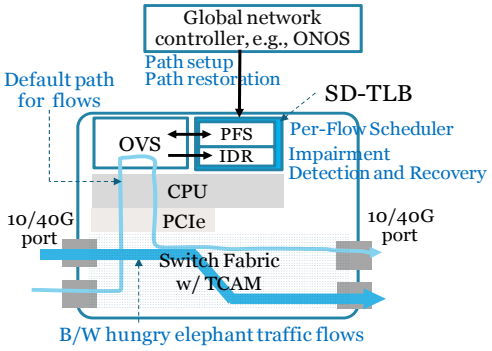

Fig. 2. System architecture of an ASIC SD-TLB enabled switch.

Upon the arrival of a new TCP flow that has no existing routing rule at a SD-TLB switch, a per-flow scheduler (PFS) selects one of N per-destination paths. We assume that there are no anonymous TCP flows at the DCI. The N paths were previously provided to the switch by a global network controller. PFS writes a rule into the flow table of the internal OVS. PFS can evenly distribute TCP flows across N per-destination paths. In the event of a network state change (*e.g.* a change in the number of available paths), the global controller updates the path information. Further, SD-TLB has MPTCP awareness with additional data structures for MPTCP flows that tracks the last chosen MPTCP flow path. Thus, we can ensure collision-free behavior among M MPTCP sub-flows per user traffic flow when $M \leq N$.

SD-TLB protects in-service flows from impairments or outages in the order of 100 ms without requiring expensive dedicated protection paths or time-consuming action by the global network controller. As shown in Fig. 4, our scheme is purely local: if the number of packets received (or sent) in path p during the current time window is substantially less than the average number of packets received or sent in p during k last consecutive time windows (parameters such as k and C can be adjusted through the global controller), an ingress node infers that p has failed or is impaired and takes action by re-routing the flow. It relies on the property of acknowledgement-based TCP congestion control scheme.

Outage: if path p from ingress node A to egress node B has failed, B never receives any packet through path p. Simultaneously, either A never gets an acknowledgement or it consecutively receives duplicate acknowledgements in the case that the flow's forward path is physically different from its backward path. It follows that all TCP flows that traverse p immediately reduce their congestion window size to zero such that A will not send any packets on path p.

Impairment: if path p from node A to node B is severely impaired, B will receive a fraction of all packets injected into p by A as packets in transit are dropped by CRC check failures inside the network. As a result, senders will receive a high number of duplicate acknowledgements. It follows that all TCP flows that traverse p immediately reduce their congestion window size. A will observe a much-smaller number of packets to p, compared to an average number of packets observed in the past.

```
P  := a set of active paths to destination site D
N := a number of active paths to destination site D
I  := the last used path index in P
F_p := a set of flows walking over a path p in P
h  := an identifier per Multipath TCP flow toward D
ML_h := the last used path index by a Multipath TCP flow h
f  := a new flow forwarded for SD-TLB switch
If f is going to walk toward D
   p = P [(I +1) mod N ]
   If f is a new Multipath TCP flow with an identifier h
      ML_h = (I +1) mod N
   elif f is a subsequent Multipath TCP flow with identifier h
      p = P [(ML_h +1) mod N ]
```

Fig. 3. Pseudo code of per-flow scheduler (PFS).

```
C := statistics collection interval.   # e.g., 100 ms
t_c := current time
t_c-1 := t_c - C
W_c := the time window from t_c-1 to t_c
R_c[p]: the number of received packets in path p during W_c
R_h[p]: the average number of received packets in path p
during the last k consecutive windows W_c-1, W_c-2 ,…,W_c-k
T_c[p]: the number of sent packets in path p during W_c
T_h[p]: the average number of sent packets in path p during the
k last consecutive windows W_c-1, W_c-2 ,…,W_c-k
For p in a set of active paths
   If R_c[p] > R_h[p] * threshold1 or T_c[p] > T_h[p] * threshold1
      Revoke p to the set of active paths
      Migrate all flows in the bucket of path p
   If R_c[p] < R_h[p] * threshold2 or T_c[p] < T_h[p] * threshold2
      Remove p from the set of active paths
      Restore migrated flows back to the bucket of path p
```

Fig. 4. Pseudo code of impairment detection and recovery (IDR).

IV. IMPLEMENTATION AND NETWORK TEST BED

SD-TLB forwards high-bandwidth flows in ingress switches in a carrier network. Thus, for implementation and experiments, we use open architecture switch platforms with powerful CPUs and multiple 10G ports, as shown in table 1. We implement SD-TLB with about 1,500 lines of Python code that can execute on the CPU of ASIC-based SDN [8] or on open virtual switches [9].

A. Control plane

SD-TLB has a localized and reactive scheme. Thus, as shown in Fig. 3, SD-TLB intercepts TCP messages (SYN only or both SYN and ACK) toward the destinations of interest (i.e. other DCs) whose addresses and routes have been previously provided by a global network controller. For an ASIC SDN switch, all unmatched traffic flows are sent to the internal OVS, as shown in Fig. 2. If the flow is a new TCP flow, it is intercepted[2] by SD-TLB PFS. To implement this, we write rules into OVS flow table before the new flow arrives, e.g., tcp, 10.20.10.1/24, actions=To-PFS. For TCP SYN-only messages, SD-TLB PFS stores the TCP information into its data structure without flow rule generation and then sends it back to the OVS. For TCP SYN and ACK messages, it decides paths (see Fig. 3), creates flow rules, and installs the rules to the OVS that will perform routing using the rules. SD-TLB checks TCP option fields in SYN messages to check whether flows are conventional TCP or MPTCP, and also the associations among sub-flows per Multipath TCP flow. The long delay (i.e., in the order of 10 ms to seconds) for writing into physical flow tables [8] is avoided by writing flow rules into the OVS whose writing delay is known to be small. Elephant flows are later moved to a physical flow table for being handled in a switch ASIC.

B. Data plane

As shown in Fig. 2, flows matched in a physical flow table are routed by the ASIC-based switch and the rest of the flows are sent to an internal OVS. In the OVS, all matched flows are tagged to be routed to their destinations. Unmatched flows (including UDP and ICMP) other than new TCP flows (see Sec. 4.1) can be routed by the internal OVS with no involvement of PFS since OVS supports the feature specified as normal action in OpenFlow [10]. A key advantage of this approach is that such a hierarchical data plane can address the limited physical ASIC flow-table size (*e.g.* 4K flow entries) because OVSes can store large forwarding tables, proportional to the CPU's available random-access memory.

Note that each path is represented by a VLAN [16] identifier in our implementation. Thus, all traffic flows are tagged in an ingress node and untagged in an egress node. In principle, VLAN tags can be replaced with MPLS labels. One benefits of using VLAN tags is interoperability with OTN with only Ethernet awareness in client ports [2]. VLAN tags are also necessary to support large scale flow aggregations, although this may require some coordination with the global controller [14].

C. Network test bed

Fig. 5 shows our transport-network test bed where an ingress node and egress node are connected through carrier-grade MPLS routers, wavelength-division multiplexing (WDM) optical transport shelves, and three 10G paths. The ingress/egress nodes employ SD-TLB or two alternatives, 1) LAG when ASIC-based SDN switches are used as the ingress/egress nodes and 2) OpenFlow multipath group (OFMG) when software switches are used as the ingress/egress nodes. Note that the ASIC-based SDN switches [8] could not be connected to the MPLS/WDM network elements since LAG supported by them cannot be used with the MPLS routers and also they do not have OpenFlow multipath group ability that is specified in OpenFlow 1.1+ (see table 1). Therefore, in the case of ASIC-based SDN switches, our experiments are done using three 10G back-to-back connections between two ASIC-based switches. As shown in Fig. 5, two end-point PCs running Linux Kernel 3.8.20 and equipped with a 2 x 10G dual-port Ethernet card generate and receive TCP traffic flows using Iperf [17].

We keep total throughputs from the ingress node to the egress node below 20 Gb/s although the total bandwidth in the network is 30 Gb/s. This is inspired by the low average link utilization (*i.e..,* 20-30%) typically employed by DC operators [6-7] and by network capacity (*i.e.,* bandwidth) over-provisioned by network carriers.

TABLE I. SWITCHES USED IN IMPLEMENTATION AND EXPERIMENTS

	Software Switch [9]	**ASIC-based Switch [8]**
CPU	Intel Xeon E5-2670 2.6GHz	Intel Core i3 2.0GHz
Memory	64GB RAM	2GB RAM
OS	Linux Kernel 3.8.19	Linux Kernel 2.6.34
Ports	5x10G ports	48x10G ports, 4x40G ports
Switch ASIC	None	Yes with 4K-entries TCAM
OpenFlow	version 1.3+	version 1.0 only

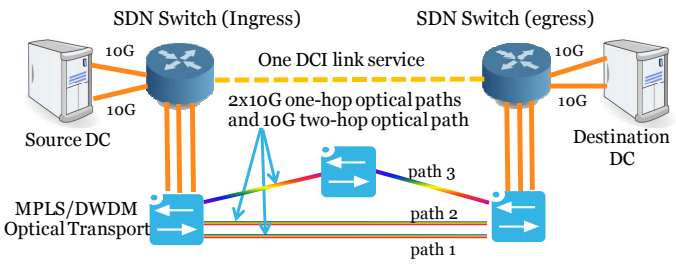

Fig. 5. Network test bed with 3 x 10G paths.

[2] Intercepting TCP SYN messages in SDT-TLB enabled nodes introduces non-negligible additional TCP handshake delay (see Table II). Note that snooping TCP SYN messages can avoid the delay, *e.g.,* tcp 10.20.10.1/24, actions = *To-default-path & To-PFS*. However, it is not implemented since it requires multicast ability that the switches used by us cannot support yet.

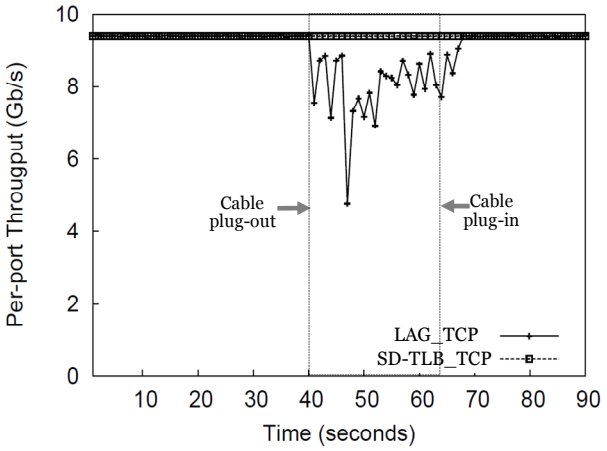

Fig. 6. Per-port aggregate throughput with 4 TCP flows/port using LAG and SD-TLB.

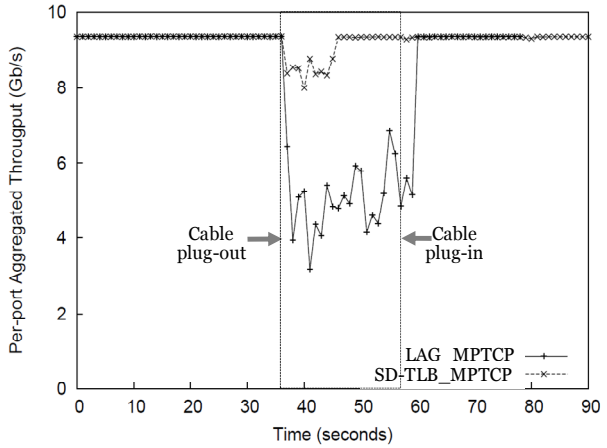

Fig. 7. Per-port aggregate throughput with 4 MPTCP flows/port using LAG and SD-TLB.

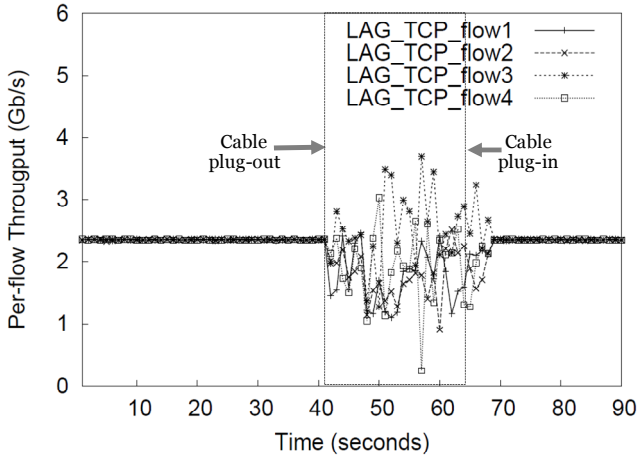

Fig. 8. Per-flow throughput with 4 TCP flows/port using LAG.

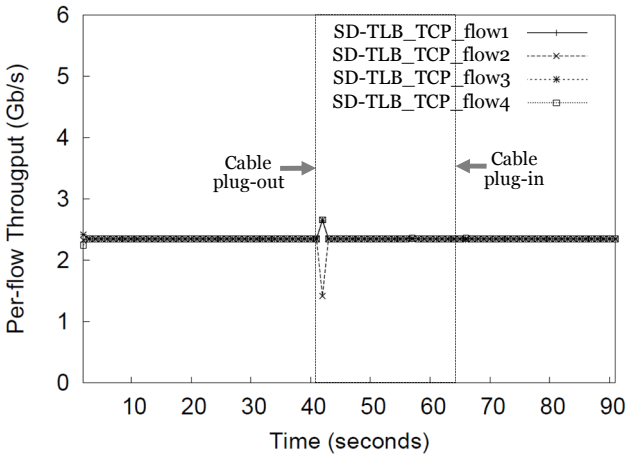

Fig. 9. Per-flow throughput of 4 TCP flows/port using SD-TLB.

V. Experimental Results

In the experiments, we measure the following metrics: aggregate throughput, balancing among in-service flows and availability during outages or impairments. With cable plug-out and plug-in at the ingress node (*i.e.* the outage case) or optical-layer impairments, we have compared how SD-TLB and the alternatives (LAG or OFMG) impact conventional TCP [18] traffic and Multipath TCP [19] traffic. The MPTCP was set to create two TCP sub-flows per user flow using the *ndiffports* option [20]. Note that few experimental results with MPTCP are shown in this paper due to space constraints. To create impairments, we use a fast semiconductor optical amplifier that periodically extinguishes the optical signal for an adjustable time [12] on path 2 before the optical receiver (Fig. 5). To measure per-flow throughputs among competing flows, a source PC simultaneously generates 3 or 4 flows per 10G port. Thus, we have 6 or 8 flows over the 2 ports for conventional TCP.

For the SD-TLB impairment detection recovery, per-port statistics are collected every 200 ms. For checking if end users can experience service degradation for cable plug-out or optical-layer impairments, we measured per-port throughput and per-flow throughput at 2x10G Ethernet card of a destination PC (see Fig. 5).

A. Outage with ASIC-based SDN switches

Using two ASIC SDN switches described in Table 1, we compared SD-TLB and LAG. ECMP was not compared in this paper since it assumes one IP address per-path (by contrast SD-TLB, LAG and OFMG all need single IP address for all multiple paths), and as a result needs at least three nodes to implement multiple paths between two nodes. In addition, we could not use OFMG with the ASIC-based SDN switches that presently support only OpenFlow 1.0. Fig. 6 shows 19 Gb/s TCP traffic (9.5 Gb/s per-port) transmitted between two ASIC-based SDN switches interconnected with 3 x 10G back-to-back connections.

At ~41 sec, we have unplugged one of the optical interconnection cables. Almost no throughout reduction is observed with SD-TLB since it quickly redistributes traffic flows on the failed path to other paths. In comparison, LAG has a considerable throughput reduction, even though 20G of 30G network capacity remains. Similar results also are observed in experiments using MPTCP, as shown in Fig. 7. In addition, LAG shows fluctuations in per-flow throughput during the outage due to competition among flows caused by uneven distribution during outage, as shown in Fig. 8. In contrast, SD-TLB quickly responds to the outage with no instability in per-flow throughputs (Fig. 9).

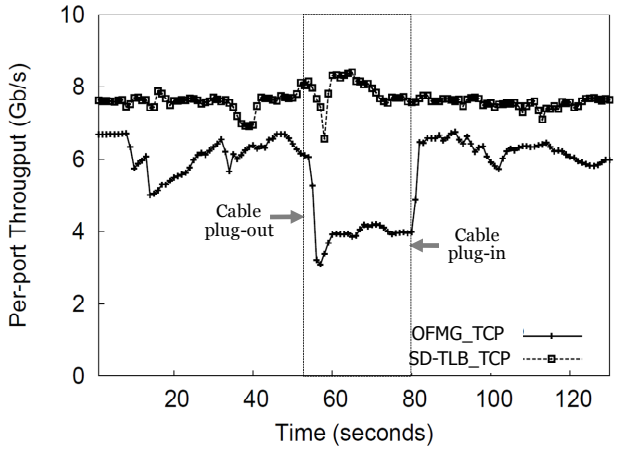

Fig. 10. Per-port aggregate throughput with 3 TCP flows/port using OFMG and SD-TLB.

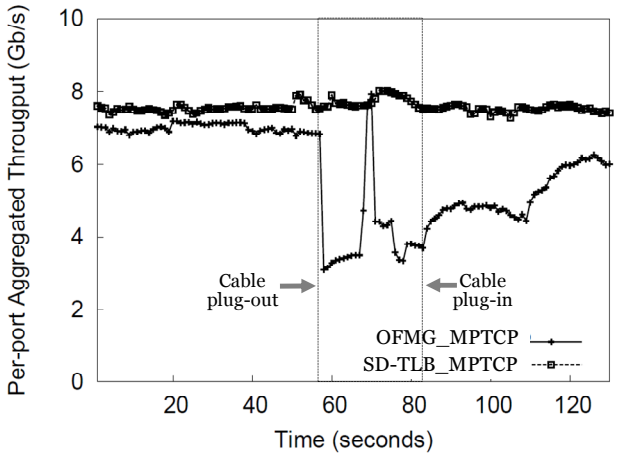

Fig. 11. Per-port aggregate throughput with 3 MPTCP flows/port using OFMG and SD-TLB.

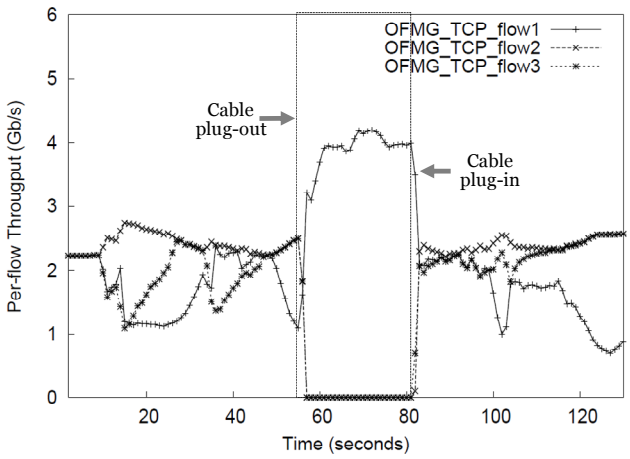

Fig. 12. Per-flow throughput with 3 TCP flows/port for OFMG.

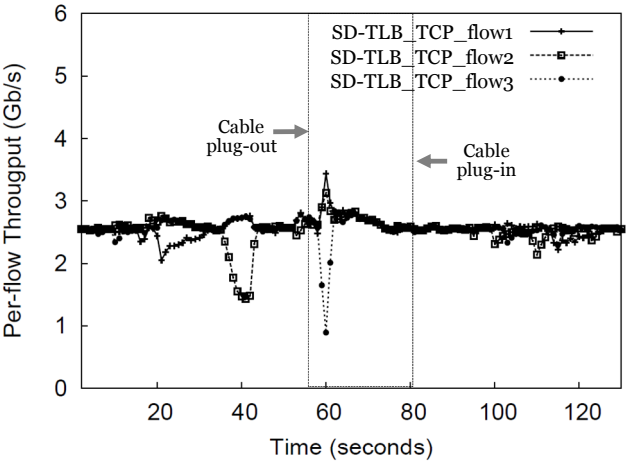

Fig. 13. Per-flow throughput of 3 TCP flows/port for SD-TLB.

B. Outage with the network test bed

In the network test bed, we use OVS ingress/egress nodes [9] that are interconnected by 3 x 10G paths in the packet over WDM transport network (see Fig. 5). Note that OVSes cause more throughput variability compared to ASIC-based switches. Since the OVS nodes support OFMG, which can be carried over the transport network, we compare OFMG and SD-TLB. Fig. 10 and 11 show outage effect via the per-port aggregate throughput with 3 TCP flows/port and with 3 MPTCP flows/port respectively.

In the case of OFMG, the port aggregate throughput drops by ~50% whereas the SD-TLB throughput remains nearly constant. In Fig. 12 and 13 showing the per-flow throughput of TCP flows, we can see if traffic distribution among flows is balanced for OFMG and SD-TLB. SD-TLB has more balanced traffic distributions, especially when the cable is unplugged. An observation different to Sec. 5.1 is that TCP flows in a failed path are service disrupted with OFMG during outage, as shown in Fig. 12. The two nodes used for Fig. 6-9 simultaneously detect outages in a directly-connected link and avoid such a service disruption. In this test where two nodes are connected through a network, failures in either of nodes or inside the network cannot be detected and thus either or both of nodes using OFMG do not move flows to different paths. In comparison, SD-TLB can effectively deal with these cases (Fig. 13).

C. Impairment with the network test bed

We also compare the performance of OFMG and SD-TLB for the case of an impaired link when the OVS ingress /egress nodes are interconnected by the transport network. During the impairment, the optical link has a post forward-error correction BER of ~10^{-6}, causing the throughput of flows on the errored path to drop by over 90%. In this case, SD-TLB can sense the drop in throughput with its per-port statistics monitoring capability, and moves flows away from the impaired path.

We observe from Fig. 14 that for TCP flows, SD-TLB shows a steady per-port aggregate throughput and recovery from service degradation caused by impairments, and by contrast OFMG does not recover from drop in throughput during the impairment period. In addition, we have observed highly-unbalanced per-flow traffic distribution in the case of OFMG (see Fig.15) and well-balanced per-flow traffic distribution for SD-TLB (see Fig. 16).

D. Overhead of SD-TLB

SD-TLB introduces an additional TCP handshake delay since for each new flow SD-TLB performs TCP intercepts

twice, PFS twice, and a rule insertion. As shown in Table 2, we have measured the TCP handshake delay at a source PC. Round trip times (RTTs) are about 1 ms in our test bed (see Fig. 5) and about 0.2 ms through two ASIC-based switches connected back-to-back. From Table 2, we see that it takes about 10 ms to write a rule into a ASIC-based switch flow table. A detailed study of rule insertion was described in Ref. [8]. Compared to other SDN-enabled approaches [6-7] [22], our SD-TLB has no delay caused by communication between SDN switches and external global SDN controllers for flow rule setup as SD-TLB is a local controller collocated with a SDN switch.

We also have the possibility to reduce delays by implementation with the *C/C++*, a compiled language that is known to be generally one to two orders of magnitude faster than Python, an interpreted scripting language that we used for implementing SD-TLB. However, the delay reduction effort may be constrained by lower-power CPUs typically used for ASIC-based switches. The reduced delay may still introduce non-negligible overhead for short-lived flows. The high TCP handshake delay mostly stems from the interception of TCP SYN messages for SD-TLB PFS (described in Sec. IV.A). As described in footnote 2, if SD-TLB enabled nodes do not *intercept* but *snoop* TCP SYN messages, we can keep the messages from being delayed by the completion of SD-TLB PFR processing. That is, an incoming TCP SYN message is forwarded over a default path with no delay even while SD-TLB PFR is performed. As a result, with TCP SYN snooping, SD-TLB may show a comparable TCP handshake delay against alternatives.

TABLE II. TCP HANDSHAKE DELAY

	Without SD-TLB	**With SD-TLB**
Test-bed	1.3 ms	100~150 ms
Back-to-back connection	0.25 ms	70 ms for writes to a RAM table 84 ms for write to a TCAM

TABLE III. CPU LOAD FOR 100MS STATISTICS COLLECTION AND 8 FLOWS/S

	Software Switch [9]	**ASIC-based Switch [8]**
CPU Usage	Less than 3%	Less than 5%

VI. CONCLUSION

In this work, we have demonstrated a software-defined traffic load balancing that delivers high quality DCI service in the face of outages and impairments without expensive optical protection. The cost advantage is obtained through the awareness of optical path diversity and better traffic load distribution. Improved bandwidth availability of SD-TLB is achieved by leveraging the intrinsic behavior of TCP. We remark that the SD-TLB approach is stateless and therefore scales well and can be deployed over existing carrier networks. In addition, the SD-TLB based DCI implementation fits well with the future evolution of carrier networks to large-scale distributed DC networks.

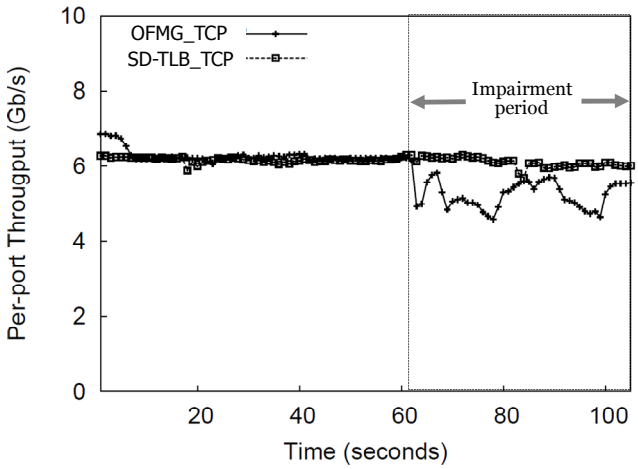

Fig. 14. Per-port aggregate throughput with 3 TCP flows/port for OFMG and SD-TLB.

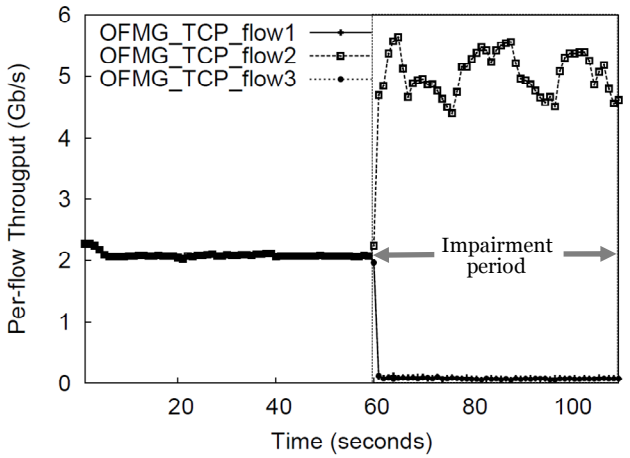

Fig. 15. Per-flow throughput in the presence of impairments for OFMG.

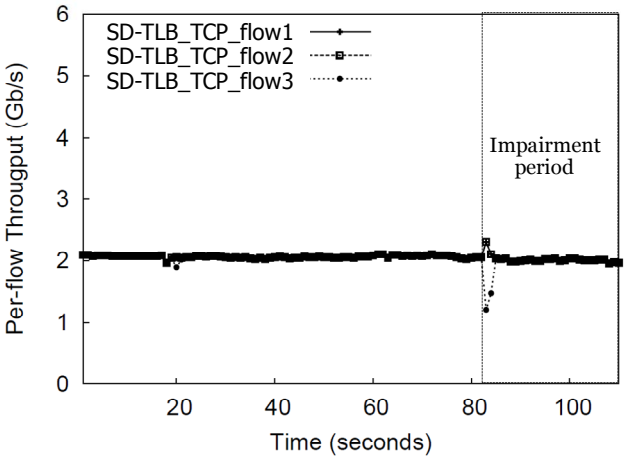

Fig. 16. Per-flow throughput in the presence of impairments for SD-TLB.

REFERENCES

[1] A. Mahimkar, A. Chiu, R. Doverspike, M. D. Feuer, P. Magill, E. Mavrogiorgis, J. Pastor, S. L. Woodward, and J. Yates, Bandwidth on Demand for Inter-Data Center Communication, ACM HotNets-X, November 2011.

[2] Juniper Networks, Converged Packet Transport: Evolution of Core Network: from Circuit to Packet, white paper, 2013.

[3] E. Mannie, Generalized Multi-Protocol Label Switching (GMPLS) Architecture, IETF RFC 3945, October 2004.

[4] Link Aggregation, IEEE 802.1AX - 2008, November 2008.

[5] C. Hopps, Analysis of an Equal-Cost Multi-Path Algorithm, IETF RFC 2992, November 2000.

[6] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, B4: Experience with a Globally-Deployed Software-Defined WAN, ACM SIGCOMM 2013, August 2013.

[7] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, Achieving High Utilization with Software-Driven WAN, ACM SIGCOMM 2013, August 2013.

[8] K. He, J. Khalid, A. Gember-Jacobson, S. Das, C. Prakash, A. Akella, L. E.Li, and M. Thottan, Measuring Control Plane Latency in SDN-Enabled Switches, ACM/USENIX SOSR2015, June 2015.

[9] Open Virtual Switch, http://openvswitch.org/.

[10] Open Networking Foundation (ONF), OpenFlow Switch Specification Version 1.3.1, September 2012.

[11] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, TCP Extensions for Multipath Operation with Multiple Addresses, IETF RFC 6824, January 2013.

[12] J.E. Simsarian, G. Atkinson, K. Carduck, K. Guan, Y.-J. Kim, M. Thottan, and P.Winzer, Cross-Layer Aware Software Defined Networking in an IP Over Optical Transport Network, OSA Photonic Networks and Devices, July 2014.

[13] Virtual Bridged Local Area Networks - Amendment: Equal Cost Multiple Paths (ECMP), IEEE 802.1Qbp - 2014, April 2014.

[14] B. Lantz, B. O'Connor, J. Hart, P. Berde, P. Radoslavov, M. Kobayashi, T. Koide, Y. Higuchi, M. Gerola, W. Snow, and G. Parulkar, ONOS: Towards an Open, Distributed SDN OS, ACM HotSDN2014, August 2014.

[15] L. Boccassi, Binder: a System to Aggregate Multiple Internet Gateways in Community Networks, ACM MobiCom Workshop LCDNet, September 2013.

[16] Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks, IEEE 802.1Q - 2011, August 2011.

[17] Iperf, https://github.com/esnet/iperf.

[18] H. Sangtae, I. Rhee, and L. Xu, CUBIC: a New TCP-friendly High-Speed TCP variant, ACM SIGOPS Operating Systems Review, Vol. 42, No. 5, 2008.

[19] Q. Peng, A. Walid, J.H. Hwang, and S. Low, Multipath TCP Algorithms: Theory, Design and Implementation, IEEE/ACM Transactions on Networking, January 2016.

[20] Multi-Path TCP Linux Kernel Implementation Version 0.90, http://www.multipath-tcp.org.

[21] Optical Internetworking Forum (OIF), Flex Ethernet Implementation Agreement 1.0, March 2016.

[22] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, Hedera: dynamic flow scheduling for data center networks, Proceedings of USENIX conference on Networked Systems Design and Implementation, April, 2010.

[23] Zhaoming Li, Design and Deployment of a GMPLS Control Plane in IP Optical Networks, The City University of New York Ph.D dissertation, 2007.

[24] M. Al-Fares, A. Loukissas, and A. Vahdat, A Scalable, Commodity Data Center Network Arrchitecture, ACM SIGCOMM 2008, August, 2008.