MP-HULA: Multipath Transport Aware Load Balancing Using Programmable Data Planes

Cristian Hernandez Benet
Karlstad University

Andreas J. Kassler
Karlstad University

Theophilus Benson
Brown University

Gergely Pongracz
Networking Research - Ericsson

ABSTRACT

Datacenter networks offer a large degree of multipath in order to provide large bisectional bandwidth. The end-to-end performance is determined by the load-balancing strategy which needs to be designed to effectively manage congestion. Consequently, congestion aware load-balancing strategies such as CONGA or HULA have been designed. Recently, more and more applications that are hosted on cloud servers use multipath transport protocols such as MPTCP. However, in the presence of MPTCP, existing load-balancing schemes including ECMP, HULA or CONGA may lead to suboptimal forwarding decisions where multiple MPTCP subflows of one connection are pinned on the same bottleneck link.

In this paper, we present MP-HULA, a transport layer multi-path aware load-balancing scheme using Programmable Data Planes. First, instead of tracking congestion information for the *best* path towards the destination, each MP-HULA switch tracks congestion information for the *best-k* paths to a destination through the neighbor switches. Second, we design MP-HULA using Programmable Data Planes, where each leaf switch can identify, using P4, which MPTCP subflow belongs to which connection. MP-HULA then load-balances different MPTCP subflows of a MPTCP connection on different next hops considering congestion state while aggregating bandwidth. Our evaluation shows that MP-HULA with MPTCP outperforms HULA in average flow completion time (2.1x at 50% load, 1.7x at 80% load).

CCS CONCEPTS

• **Networks → Programmable networks**;

KEYWORDS

In-Network Load Balancing; Programmable Switches; Network Congestion; Multipath.

ACM Reference Format:
Cristian Hernandez Benet, Andreas J. Kassler, Theophilus Benson, and Gergely Pongracz. 2018. MP-HULA: Multipath Transport Aware Load Balancing Using Programmable Data Planes. In *NetCompute'18: Morning Workshop on In-Network Computing, August 20, 2018, Budapest, Hungary*, Jennifer B. Sartor, Theo D'Hondt, and Wolfgang De Meuter (Eds.). ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3229591.3229596

NetCompute'18, August 20, 2018, Budapest, Hungary

ACM ISBN 978-1-4503-5908-5/18/08…$15.00
https://doi.org/10.1145/3229591.3229596

1 INTRODUCTION

In recent years, many applications have been hosted inside data centers that are increasingly demanding in terms of data transfer. For example, [3] shows that 90% of the traffic in data centers belong to long flows (more than 1MB). In addition, with the recent support for multi-path transport layer inside modern Operating Systems, more and more applications are enabled to use Multipath TCP (MPTCP). Apples Siri is just a prominent example of such Cloud based MPTCP service. Such multipath transport protocols attempt to exploit the path diversity by splitting traffic between multiple subflows which has several benefits. It allows to aggregate capacity of multiple paths, shifts traffic adaptively to less congested sub-flows and improves the fault tolerance to link failures. Consequently, data centers should be designed for multipath transports [20]. With the recent development of stateless MPTCP server load balancers [17], MPTCP is expected to be more prevalent in the data center.

Despite the effort made in the transport layer, today many data centers still use Equal-Cost Multi-Path (ECMP) as a routing strategy, which assigns each flow to one of several least cost paths randomly based on a hash function. Unfortunately, when using MPTCP combined with ECMP, hash collisions may occur resulting in under-utilized links and unbalanced load across multiple paths [20]. Hedera [1] develops a dynamic flow scheduling system using a centralized solution which suffer from high control loop latency. Other load-balancing schemes such as CONGA [2] and HULA [15] propose distributed solutions attempting to cope with the slow decision making of centralized solutions for high volume of data. HULA performs flowlet routing along least congested paths, which are updated based on distributed probing taking advantage of the emerging capabilities provided by data plane programmability such as P4 [4]. Because Conga and HULA are not designed for multipath transport, they cannot exploit the features of multipath transport efficiently thus resulting in low performance.

This paper proposes MP-HULA, a data-plane load-balancing approach that is multi-path transport aware. By using P4, MP-HULA switches parse MPTCP header fields thus being able to associate MPTCP subflows to a MPTCP connection. Instead of tracking only the least utilized path towards each Top-of-Rack (ToR) switch, MP-HULAs adaptive probing mechanism keeps congestion state for the *best-k* next hops per destination. MPTCP sub-flows are then split into flowlets [13], which is a long enough burst of packets of a MPTCP subflow to avoid reordering. The MP-HULA switch then uses the congestion state for *best-k* next hops together with

the MPTCP sub-flowlet information to route different MPTCP sub-flowlet towards different next hops taking into account congestion information. Our evaluation in the NS2 network simulator demonstrates the effectiveness of MP-HULA in reducing flow completion time between 1.7x and 2.1x compared to HULA with MPTCP and uncoupled congestion control. When compared with HULA and TCP, MPTCP combined with MP-HULA reduces average flow completion time (3.4x at 50% load, 2.9x at 80% load).

2 MULTI-PATH TRANSPORT AWARE LOAD BALANCING IN THE DATA PLANE

A MPTCP connection can be seen as a single socket connection from an application's perspective. It is composed of one or more sub-flows, each one with its own sequence number space and congestion window so that it can adapt to congestion along each path. MPTCP can run multiple sub-flows on a single path. When using ECMP, different sub-flows may be routed over different paths, which may lead to hash collisions and several flows of a single MPTCP connection may end up on the same bottleneck link leading to loss of throughput [23]. The key idea of MP-HULA is to load-balance different sub-flows of a given MPTCP connection over different paths on a per flowlet basis, taking congestion into account.

As in HULA, probes are used to convey congestion information which is used for MPTCP flowlet routing. In contrast to HULA, however, probes are replicated to track the *best-k* paths from each leaf switch throughout the network. This is because we route each sub-flow of a given MPTCP connection over disjoint paths to aggregate bandwidth. In this section, we describe how the multiple sub-flows of a given MPTCP connection are identified and tracked by MP-HULA capable switches, how congestion information is used for congestion aware flowlet routing of multiple MPTCP sub-flows and how the probe replication and processing logic tracks the *best-k* paths using custom data structures expressed in P4 [4].

2.1 MPTCP header processing in P4

In MPTCP, each sub-flow has a port/IP-address that is different from other sub-flows of the same connection. In order to identify, which sub-flow belongs to a given MPTCP connection, ToRs need to parse MPTCP protocol header extensions during connection establishment and sub-flow opening in order to obtain tokens, which uniquely identify the MPTCP connection [7] (Figure 1). This functionality is assigned to the ToR since all MPTCP messages exchanged during connection establishment and sub-flow opening pass through it. As the load-balancing may spread out the packets after the ToR on different paths, other switches might not receive all messages which are required in order to successfully correlate the MPTCP sub-flows. Alternatively, correlating sub-flows to a given MPTCP connection might also be done on the hypervisor or P4 programmable NIC of the sender. A custom register data structure *MPTCP Association Table* (Figure 2) is used by the ToR to store, per sub-flow, the 5-tuple sub-flow Hash, Token A, Token B, Sub-flow counter, Sub-flow number, MPTCP connection ID.

During the three-way handshake, the source sends a SYN with the MP_CAPABLE option set and a random 64-bit key (Key A). The ToR parses the SYN packets sent by the hosts and checks, if the MP_CAPABLE option exists (Figure 1 step 1). Then, it computes

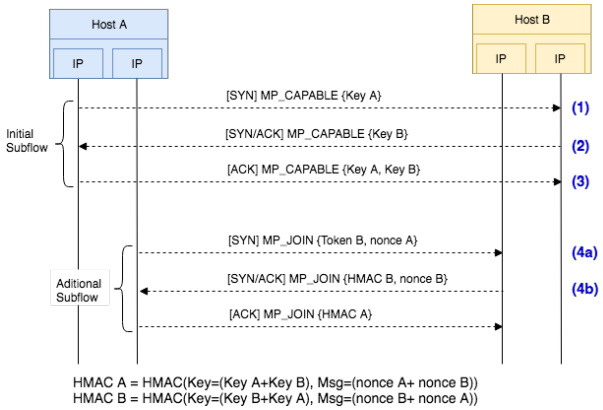

Figure 1: MPTCP sub-flow establishment

a five-tuple hash with the source and destination IP, source and destination port, and the layer 4 protocol. It uses this hash index to store the sender key in an auxiliary table (Figure 2 step (1)), which will be used later to calculate the tokens.

MPTCP Association Table

5-tuple Hash	Token A	Token B	Sub-flow counter	Sub-flow num.	MPTCP ID
HASH 1	A (3a)	B (3b)	~~1~~ 2 (5a)	1	ID1 (5b)
HASH 2 (6a)	A	B	2	2 (6b)	ID1 (6c)

Token table

Token	5-tuple HASH
A (3c)	HASH 1
B (3d) (4)	HASH 1

Auxiliary table for keys

5-tuple Hash	Key A	Key B
HASH 1	A (1)	B (2)

Figure 2: MPTCP association, token and auxiliary tables

The ToRs parse the SYN/ACK messages sent from the receiver which also have the MP_CAPABLE option set in order to extract the receiver key (Key B, step (2)). The hash is computed exchanging sender information for the receiver's (destination IP, source IP, destination port, source port and protocol) and the ToR stores the receiver key B in the auxiliary table (Figure 2 step (2)). Once Key B is inserted in the table, the ToR generate the Tokens A and B and inserts them into the MPTCP Association table (step (3)). The token is a truncation of the 32 most significant bits of the SHA-1 of the key, calculated using external functions in P4. Once step (3) is complete, the entry in the auxiliary table for that hash might be deleted. A new entry is created in the *Token Table* (steps (3c) and (3d)), which is needed in order to associate new sub-flows to a given MPTCP connection. Using the token as an index in the table may lead to a collision risk [7], especially if a large number of sub-flows pass through that ToR from different servers and we suggest to implement the detection and identification function in the hypervisor or NIC of the sender. Alternatively, we could create a token table per incoming port and using another table to relate

the incoming port with port token table, which would consume more memory. The token will be utilized when new sub-flows are created in order to identify the initial sub-flow and thus the MPTCP connection ID.

When MPTCP creates a new sub-flow, the sender sends a SYN packet with MP_JOIN containing the token and a generated random number (nonce A), as shown in Figure 1 step (4a). The ToR parses this message and obtains the sent token that is subsequently used to look up the hash of the initial MPTCP sub-flow in the token table, step (4) in Figure 2. The hash index obtained in the token table is used to update the counter of that MPTCP connection in the MPTCP Association table and at the same time, calculate the MPTCP connection ID (step (5a) and (5b)). The MPTCP ID can be created in different ways, however to simplify, we will take the 32-bit Token A and the 32-bit from Token B to generate a unique 64-bit ID, which is stored in step (5b) in the MPTCP Association table. Then, we add a new entry for the new sub-flow together with the 5-tuple hash, its sub-flow number (current state of the counter) and MPTCP connection ID as shown in Figure 2 step (6a), (6b) and (6c). Therefore, the sub-flow counter is used to enumerate the new sub-flows. For example, when the new sub-flow is initiated, the counter increases from one to two as shown in Figure 2 step (5a), which indicates the sub-flow number of the MPTCP Association table step (6b). For security and validation reasons, the nonces (A and B) exchanged during the MP_JOIN process, shown in Figure 1 step (4a) and (4b), could be stored also in an additional auxiliary table to later generate the HMAC A and HMAC B and thus verify the correct establishment of the sub-flows. We assume a secure data center environment and omitted this step.

2.2 Multipath sub-flow association mapping

Although the ToR is able to correlate different sub-flows to a MPTCP connection, upper layer switches might not have information on all token and keys exchanged due to the flowlet switching. Therefore, the ToR needs to augment MPTCP data packets by an additional header which allows each switch to uniquely identify the MPTCP connection and sub-flow, if more than one sub-flow exists. This header is removed by the target ToR switch and intermediate aggregation and core switches use only this additional header for forwarding decisions after correlating MPTCP sub-flows to their respective connection. When the MPTCP ID is created by the ToR and the sub-flow counter is greater than one, Figure 1 step (6b) and (6c), the ToR adds the following header to all MPTCP packets after identifying the correct sub-flow and MPTCP ID:

- **MPTCP_ID (64 bits):** This ID is used to uniquely identify the MPTCP connection.
- **sub-flow_num (4 bits):** identifies the sub-flow number within the MPTCP connection.

We assume that the 64-bit ID can uniquely identify each MPTCP connection within the data center. Likewise, 4-bits are used for the sub-flow index within a given MPTCP connection, which supports 16 possible sub-flows. However, these values can be easily modified.

2.3 Hop-by-hop Probe Processing

The original HULA uses 64 byte probe packets containing the *id* of the originator ToR and a field to aggregate link utilization. The probes update a table that the switches use to store the best next hop for each destination ToR and its respective link utilization. In contrast to HULA, we maintain k tables to store the *best-k* next hop switches and their corresponding link utilization. We now detail, how we modify the HULA probe processing logic to track congestion information for the *best-k* paths along with the next hop information. We do *not* modify the HULA probe packet format.

When a probe packet arrives, the switch calculates the maximum of the utilization field in the packet and the TX link utilization of the packet's input port, *MaxUtil*. *Min_hop_util* calculates the minimum of the lowest link utilization value obtained from the best hop table *k=1*) and *MaxUtil*. Then, *Min_hop_util* is compared with the entries of all *best-k* next hop tables. The tables are updated in order to maintain a sorted array of best *best-k* hops along with their utilization using *if-else* logic. For example, for k = 3, if the second value (e.g. 65%) is lower than the *Min_hop_util* (e.g. 67%), the second value is copied to the k=3 table position and the 2nd entry is replaced with the *Min_hop_util* value. Finally, the utilization field of the probe packet is updated with the new *Min_hop_util* value and replicated to all the ports, except for the one where it was received from, until it reaches the next pod, where it is only replicated to the lower hierarchy switches.

2.4 MPTCP Flowlet Routing

As in HULA, the load-balancing granularity is the flowlet in order to avoid packet reordering. All packets belonging to the same flowlet are routed over the same path *p1* over the same next hop until the flowlet gap f_g has expired. Once this timer expires, a new flowlet is created in the *Flowlet table*, updating the time stamp. Depending on the network conditions, when the new flowlet is created, it can be assigned to a new path *p2*. In our case, as we want to aggregate the bandwidth across multiple MPTCP sub-flows, when the flowlet belonging to an MPTCP sub-flow for a given MPTCP connection expires, we check if there is another flowlet, which belongs to the same MPTCP connection, assigned to the best available path. If this is the case, we do not send this new flowlet over the best next hop because that may lead to situations, where multiple flows of the same MPTCP connection share the same bottleneck link. Instead, we send it over the best next hop not used by another sub-flow of the same MPTCP connection. However, other schemes are also possible in case more sub-flows are opened as alternative next hops are available. The first sub-flow will always be routed over the least congested next hop, which is important for short flows.

2.5 Path selection and MPTCP association under partial information

When an MPTCP connection has more than one sub-flow, the ToR adds the header described in Section 2.2 to all packets belonging to any of these sub-flows. MP-HULA switches at higher level (e.g. core, aggregation) perform a 5-tuple hash and check if the header is present in the packet. If this header does not exist, we use the same technique as HULA and therefore the packet is sent on the best-next hop. In case more than one sub-flow has been opened by MPTCP, the ToR attaches the additional header to each packet which contains the MPTCP_ID and sub-flow_num. The *Flowlet table* stores those

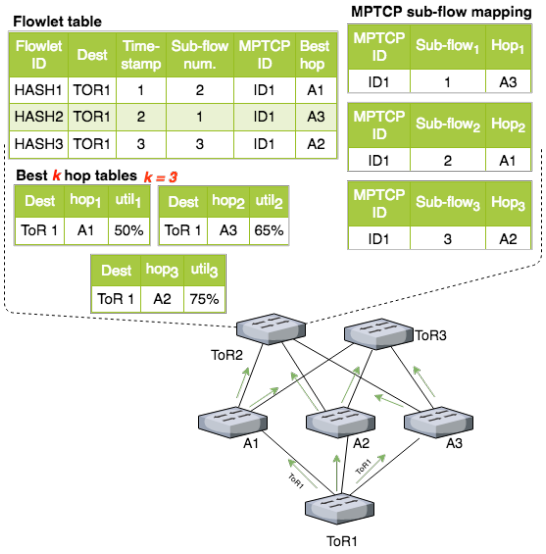

Figure 3: Best k next hop switch and flowlet tables

values as illustrated in Figure 3. Then, the switch obtains the best-next hop from the *best-k*-hop tables, hop_1, checks the MPTCP sub-flow mapping tables if this best-next hop has already been assigned previously to any other sub-flow of this MPTCP_ID. If this best-next hop does not exist in the mapping tables, this best-next hop is assigned and stored in the Flowlet table as the best hop. Otherwise, if hop_1 is found, the hop_1 is discarded and we identify the second best-next hop, hop_2. These steps are repeated until hop_k in case of finding the selected best-hop in any of the MPTCP sub-flow mapping tables. If in the last step, i.e. the hop_k, the best-hop is found again in the MPTCP sub-flow mapping tables, the hop_1 is assigned as the best hop for that flowlet implementing a round-robin scheme but other schemes are possible, too.

2.6 Feasibility of MP-HULA in P4

As Hula, implementing MP-HULA in P4 requires stateless (header field reading and writing) and stateful (record and manipulate congestion and forwarding state) operations. In addition, we require external function support from P4 (which can be implemented using extern block [26]) to implement the SHA-1 algorithm to obtain the token. However, in our case, the SHA-1 function is called only once per MPTCP connection and implementing the SHA-1 in FPGA adds a small overhead [16, 19]. We can outsource the SHA-1 computation to crypto acceleration blocks of P4 enabled NICs[1] or switches. MP-HULA requires more congestion state compared to HULA to track k paths and requires forwarding state per MPTCP sub-flowlet (instead of per flowlet). Note, that k allows to make a trade-off between the path diversity, memory consumption and processing overhead at the switch which is subject to future study.

Parsing State: MP-HULA requires state information to correlate the MPTCP sub-flow to the MPTCP connection using the MPTCP Association table (19 bytes per sub-flow), Token table (6 bytes per sub-flow) and Auxiliary table for keys (10 bytes per sub-flow), see Figure 2. For example, when processing 100K MPTCP connections, 6 sub-flows, in total 21 MB are required. This state can be outsourced to the Hypervisor/programmable NIC at the server.

Forwarding State: MP-HULA requires per sub-flow forwarding state implemented by the Flowlet table (25 bytes per MPTCP sub-flowlet), Best hop tables (16 bytes per ToR and table) and MPTCP sub-flow mapping (13 bytes per sub-flow) as shown in 3. Assuming 10K ToRs, 6 sub-flows, 4 best paths, 100K MPTCP connections, the memory requirement is around 25 MB.

Processing at ToR/Hypervisor: In addition to calculating the SHA-1, the ToR/Hypervisor need to insert/remove the header to identify the MPTCP connection (Section 2.2) per packet, which can be done at line rate similar to INT-operations. In addition, it requires additional parsing operations related to MPTCP in order to create state for the MPTCP association, token and auxiliary tables, which is required for each sub-flow opening.

Processing at aggregation/core: In addition to parsing the additional header to correlate the MPTCP subflows, all switches (including the ToRs) need to process the Flowlet table, MPTCP sub-flow mapping and best hop tables. Since there is no for-loop in P4, all search operations must be implemented using if-else, which makes the search in the MPTCP sub-flow mapping table complex for a high number of sub-flows. The same complexity is required when ordering the best hop tables for a large k.

3 EVALUATION AND RESULTS

We evaluate MP-HULA using packet level simulation in NS2. We use a 3-tier Fat-Tree topology (Figure 4) with two Core switches (C1, C2) connecting two pods with two Aggregation switches and ToRs using 40Gbps links. 16 compute nodes are connected to each ToR using 10Gbps links. We use two different workloads to generate traffic based on the traces from [15], which emulate Web-search and data mining jobs. The compute nodes run a client-server application to initiate a TCP or MPTCP connection, which generates traffic using flow size distribution obtained from the CDF samples of the workload traces. Each of the 16 clients (located in pod 1) selects randomly any of the 16 servers (from pod2). The flow inter-arrival pattern is modeled by an exponential Poisson process. We scale the request rate to increase the load from 10% to 90%.

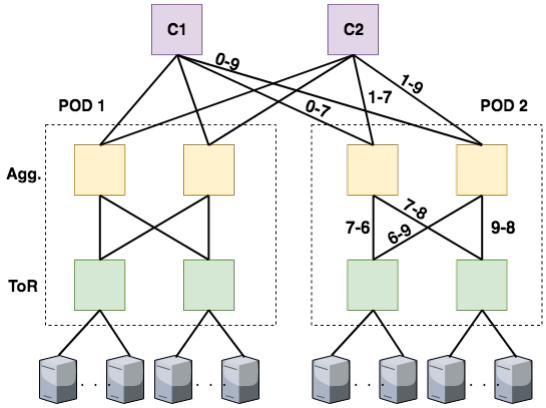

Figure 4: Evaluation Topology

[1] e.g. using https://www.netronome.com/media/redactor_files/WP_NFP4000_TOO.pdf

We compare MP-HULA against ECMP and HULA [15], which forwards all packets in a flowlet towards a given destination ToR only over the single best next hop. We use a flowlet inter-packet gap of 100μs, which is in the order of the network RTT to minimize packet re-ordering at the receiver as in [2, 15]. The probing interval is set to 200μs as in [15]. For MPTCP, we use two sub-flows per connection and we evaluate uncoupled and coupled [21] congestion control. The parameter that determines the number of *best-k* next hops is set to 2. This paper does not intended to optimize the k parameter but rather intends to demonstrate the benefits of the proposed approach and the relevance of this parameter. A k value lower than the number of sub-flows may imply a reduction in the bandwidth aggregation. This is because some sub-flows can result in the same best-hop, and therefore are more likely to travel through the same bottleneck. In addition, another aspect which must not be neglected when selecting the k parameter is the number of output ports in the switch to send the flowlets.

Figure 5 shows the average FCT for the web-search workload using coupled congestion control. Using MPTCP with MP-HULA (MPTCP-COUPLED_MP-HULA) performs significantly better than all of the other schemes because of the large flow sizes in the trace which enable MPTCP to take advantage of the diverse paths. Similarly, MP-HULA performs best also for the small mice flows from the same web-search trace (<100KB transfer) as can be seen from Figure 6. This is also, because the larger elephant flows finish earlier taking advantage of the multipath (omitted due to space constraints). Using uncoupled congestion control reduces flow completion time even more (3.4x at 50% load, 2.9x at 80% load) because of the more aggressive congestion control (Figure 7). This shows that MP-HULA benefits both, coupled and uncoupled congestion control schemes for MPTCP and reduces flow completion time significantly compared to just using HULA and MPTCP. For all the schemes evaluated, ECMP performs the worst when combined with MPTCP due to hash collisions leading to poor link utilization.

To evaluate MP-HULA under link failures, we use an asymmetric topology, where we bring down one of the links that connect the core switches to the aggregation switches reducing bisectional bandwidth. As we can see from Figure 8, again, MP-HULA outperforms all other schemes using the web-search trace showing its effectiveness to shift traffic away from congested paths even in asymmetric topologies. Interestingly, MPTCP with ECMP performs very poor because of the congestion unaware nature of ECMP that routed the traffic towards the core where only one link was available. Figure 9 shows the average link utilization (e.g. 0.72 = 72%) for the web-search workload using MPTCP uncoupled congestion control for different links in the topology, shown in Figure 4. As can be seen, MP-HULA with MPTCP leads to higher link utilization compared to HULA (when run with TCP or MPTCP) because it spreads different MPTCP sub-flowlets more equally on different next hops leading to lower overall flow completion time.

Figure 10 shows the average FCT for the data-mining workload using coupled congestion control for MPTCP. Using MPTCP with HULA (MPTCP-COUPLED_HULA) performs better than TCP with HULA (1_TCP-Hula). In addition, we can see how using MP-HULA (MPTCP_MP-HULA) improves performance by using different paths for each of the sub-flowlets. However, because the data-mining workload is comprised of mostly small flows, the benefits of MP-HULA is not so pronounced.

4 RELATED WORK

Centralized Algorithms: Hedera [1], B4 [12] or Planck [22] are centralized approaches, where the controller uses global information in order to route the flows. Under varying traffic patterns and many small flows, common to data centers, those schemes have a high cost and scalability limitations. [18] has a slightly different design goal to influence endpoints congestion control decisions based on centralized control and relies on e.g. ECMP routing.

Host-based: MPTCP [20] is a host-based load balancing scheme which splits an end-to-end connection into multiple TCP sub-flows. It uses end-to-end feedback to shift packets to less congested subflows. Typically, MPTCP is augmented with ECMP routing, which uses random hashing of subflows onto paths leading to hash collisions and low performance. In addition, MPTCP may suffer from incast [2]. Due to the importance of MPTCP for data center networking, many drawbacks have been addressed e.g. in [6, 24, 28]. NDP [10] is a novel transport layer stack for low latency and high throughput data centers which requires additional router support but does not have multipath transport support.

In-Network distributed: Distributed load balancing schemes either use local link utilization (Drill [9]) or global congestion state (Conga [2], Hula [15]) to route packets. While local state has difficulties to react to asymmetric links, approaches that require complete global state, e.g. [2], have scalability issues. In general, all those approaches do not consider the properties of multipath transport protocols and may lead to poor capacity aggregation performance.

Edge based: Schemes as Presto [11], Juggler [8], LetItFlow [25], Clove [14], Hermes [27]) do not need to modify the router or the end-host. Instead, they run in the virtual switch on each server, rely on support from network feedback (e.g. [27]) and consequently need to update each server software.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we design MP-HULA, a scalable, multi-path transport protocol aware load-balancing scheme designed for emerging programmable data planes such as P4. Based on periodic probing messages, switches track the utilization and next hop for the *k-best* paths for a given destination. MP-HULA splits up the MPTCP flows into flowlets and decides locally, which next hop to use for a given MPTCP flowlet based on congestion state and information to which MPTCP connection the flowlet belongs to. MP-HULA capable switches parse transport layer protocol headers in order to correlate MPTCP sub-flows to MPTCP connections to avoid hash collisions. Our evaluation shows that MP-HULA exploits the transport layer multipath characteristics of MPTCP, reducing the flow completion times compared to other load balancing schemes.

As future work, we intend to implement and evaluate MP-HULA on P4 capable hardware and evaluate the performance with more path diversity and different settings. We will also extend the load-balancing features to consider different flow priorities and other emerging multipath transport protocols such as Multipath QUIC [5], which will require less state management due to the presence of the path ID of each subflow in an unencrypted header.

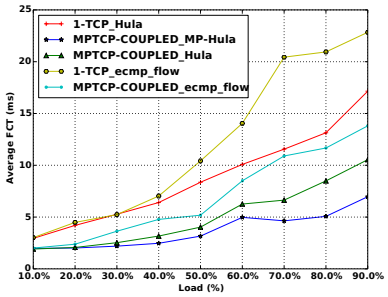

Figure 5: Average FCT using MPTCP-Coupled for web-search traffic

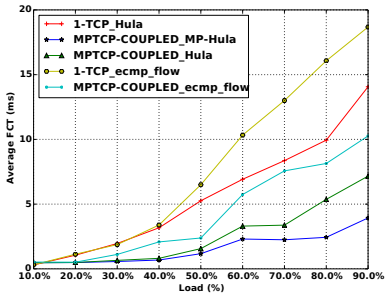

Figure 6: Average FCT for mice flows (<100KB), coupled, web-search

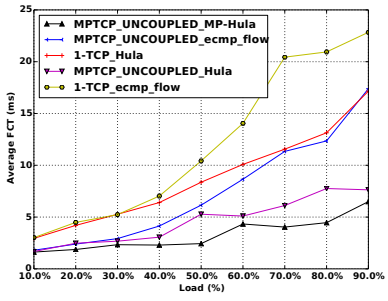

Figure 7: Average FCT using MPTCP-Uncoupled for web-search traffic

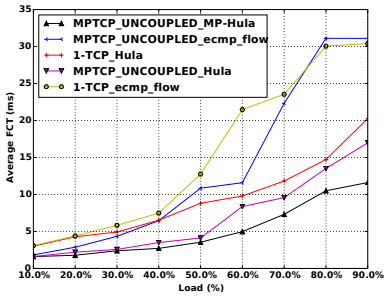

Figure 8: Asymmetric topology for web-search

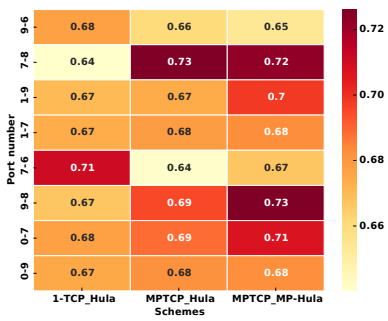

Figure 9: Link utilization (web-search)

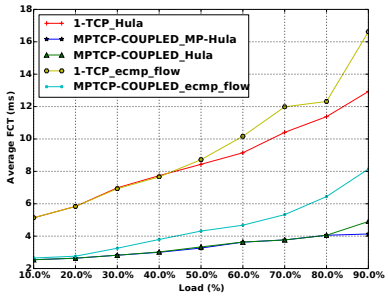

Figure 10: Average FCT using MPTCP-Coupled for data-mining traffic

ACKNOWLEDGMENTS

The authors would like to thank Ricardo Santos and Jonathan Vestin for their valuable feedback. The authors would also like to thank the anonymous referees for their valuable comments and helpful suggestions. This work is supported by the Knowledge Foundation of Sweden through the Profile HITS under Grant No.: 20140037.

REFERENCES

[1] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, and Amin Vahdat. 2010. Hedera: Dynamic flow scheduling for data center networks. In *Nsdi*, Vol. 10. 19–19.

[2] Mohammad Alizadeh, Tom Edsall, Sarang Dharmapurikar, Ramanan Vaidyanathan, Kevin Chu, Andy Fingerhut, Francis Matus, Rong Pan, Navindra Yadav, George Varghese, and others. 2014. CONGA: Distributed congestion-aware load balancing for datacenters. In *ACM SIGCOMM Computer Communication Review*, Vol. 44. ACM, 503–514.

[3] Theophilus Benson, Aditya Akella, and David A Maltz. 2010. Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 267–280.

[4] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. 2014. P4: Programming Protocol-independent Packet Processors. *SIGCOMM Comput. Commun. Rev.* 44, 3 (July 2014), 87–95. DOI: http://dx.doi.org/10.1145/2656877.2656890

[5] Quentin De Coninck and Olivier Bonaventure. 2017. Multipath QUIC: Design and Evaluation. In *Conext'17*. See also http://www.multipath-quic.org.

[6] Gregory Detal, Christoph Paasch, Simon van der Linden, Pascal MÃĹrindol, Gildas Avoine, and Olivier Bonaventure. 2013. Revisiting Flow-Based Load Balancing: Stateless Path Selection in Data Center Networks. *Computer Networks* 57, 5 (April 2013).

[7] Alan Ford, Costin Raiciu, Mark Handley, and Olivier Bonaventure. 2013. *RFC 6824: TCP extensions for multipath operation with multiple addresses*. Technical Report.

[8] Yilong Geng, Vimalkumar Jeyakumar, Abdul Kabbani, and Mohammad Alizadeh. 2016. Juggler: A Practical Reordering Resilient Network Stack for Datacenters. In *Proceedings of the Eleventh European Conference on Computer Systems (EuroSys '16)*. ACM, New York, NY, USA, Article 20, 16 pages. DOI: http://dx.doi.org/10.1145/2901318.2901334

[9] Soudeh Ghorbani, Zibin Yang, P. Brighten Godfrey, Yashar Ganjali, and Amin Firoozshahian. 2017. DRILL: Micro Load Balancing for Low-latency Data Center Networks. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*. ACM, New York, NY, USA, 225–238. DOI: http://dx.doi.org/10.1145/3098822.3098839

[10] Mark Handley, Costin Raiciu, Alexandru Agache, Andrei Voinescu, Andrew W. Moore, Gianni Antichi, and Marcin Wójcik. 2017. Re-architecting Datacenter Networks and Stacks for Low Latency and High Performance. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*. ACM, New York, NY, USA, 29–42. DOI: http://dx.doi.org/10.1145/3098822.3098825

[11] Keqiang He, Eric Rozner, Kanak Agarwal, Wes Felter, John Carter, and Aditya Akella. 2015. Presto: Edge-based Load Balancing for Fast Datacenter Networks. *SIGCOMM Comput. Commun. Rev.* 45, 4 (Aug. 2015), 465–478. DOI: http://dx.doi.org/10.1145/2829988.2787507

[12] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, and others. 2013. B4: Experience with a globally-deployed software defined WAN. In *ACM SIGCOMM Computer Communication Review*, Vol. 43. ACM, 3–14.

[13] Srikanth Kandula, Dina Katabi, Shantanu Sinha, and Arthur Berger. 2007. Dynamic Load Balancing Without Packet Reordering. *SIGCOMM Comput. Commun. Rev.* 37, 2 (March 2007), 51–62. DOI: http://dx.doi.org/10.1145/1232919.1232925

[14] Naga Katta, Aditi Ghag, Mukesh Hira, Isaac Keslassy, Aran Bergman, Changhoon Kim, and Jennifer Rexford. 2017. Clove: Congestion-Aware Load Balancing at the Virtual Edge. In *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT '17)*. ACM, New York, NY, USA, 323–335. DOI: http://dx.doi.org/10.1145/3143361.3143401

[15] Naga Katta, Mukesh Hira, Changhoon Kim, Anirudh Sivaraman, and Jennifer Rexford. 2016. Hula: Scalable load balancing using programmable data planes. In *Proceedings of the Symposium on SDN Research*. ACM, 10.

[16] Ritu Kaur Makkad and Anil Kumar Sahu. 2016. Novel design of fast and compact SHA-1 algorithm for security applications. In *Recent Trends in Electronics, Information & Communication Technology (RTEICT), IEEE International Conference on.*

IEEE, 921–925.

[17] Vladimir Olteanu and Costin Raiciu. 2016. Datacenter Scale Load Balancing for Multipath Transport. In *Proceedings of the 2016 Workshop on Hot Topics in Middleboxes and Network Function Virtualization (HotMIddlebox '16)*. ACM, New York, NY, USA, 20–25. DOI: http://dx.doi.org/2940147.2940154

[18] Jonathan Perry, Hari Balakrishnan, and Devavrat Shah. 2017. Flowtune: Flowlet Control for Datacenter Networks. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. USENIX Association, Boston, MA, 421–435. https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/perry

[19] Tang Qiong and Ye Jianwu. 2012. Implementation and Optimization of the High Performance SHA-1 Model Based on FPGA. In *Computer Science & Service System (CSSS), 2012 International Conference on*. IEEE, 687–690.

[20] Costin Raiciu, Sebastien Barre, Christopher Pluntke, Adam Greenhalgh, Damon Wischik, and Mark Handley. 2011. Improving Datacenter Performance and Robustness with Multipath TCP. In *Proceedings of the ACM SIGCOMM 2011 Conference (SIGCOMM '11)*. ACM, New York, NY, USA, 266–277. DOI: http://dx.doi.org/10.1145/2018436.2018467

[21] Costin Raiciu, Mark Handley, and D. Wischik. 2011. *RFC 6356: Coupled Congestion Control for Multipath Transport Protocols*. Technical Report.

[22] Jeff Rasley, Brent Stephens, Colin Dixon, Eric Rozner, Wes Felter, Kanak Agarwal, John Carter, and Rodrigo Fonseca. 2014. Planck: Millisecond-scale monitoring and control for commodity networks. In *ACM SIGCOMM Computer Communication Review*, Vol. 44. ACM, 407–418.

[23] Marcus Sandri, Alan Silva, Lucio A Rocha, and Fabio L Verdi. 2015. On the benefits of using multipath tcp and openflow in shared bottlenecks. In *Advanced Information Networking and Applications (AINA), 2015 IEEE 29th International Conference on*. IEEE, 9–16.

[24] Jiyan Sun, Yan Zhang, Xin Wang, Shihan Xiao, Zhen Xu, Hongjing Wu, Xin Chen, and Yanni Han. 2017. DCˆ 2-MTCP: Light-Weight Coding for Efficient Multi-Path Transmission in Data Center Network. In *Parallel and Distributed Processing Symposium (IPDPS), 2017 IEEE International*. IEEE, 419–428.

[25] Erico Vanini, Rong Pan, Mohammad Alizadeh, Parvin Taheri, and Tom Edsall. 2017. Let It Flow: Resilient Asymmetric Load Balancing with Flowlet Switching. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. USENIX Association, Boston, MA, 407–420. https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/vanini

[26] Han Wang, Robert Soulé, Huynh Tu Dang, Ki Suh Lee, Vishal Shrivastav, Nate Foster, and Hakim Weatherspoon. 2017. P4fpga: a rapid prototyping framework for p4. In *Proceedings of the Symposium on SDN Research*. ACM, 122–135.

[27] Hong Zhang, Junxue Zhang, Wei Bai, Kai Chen, and Mosharaf Chowdhury. 2017. Resilient Datacenter Load Balancing in the Wild. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*. ACM, New York, NY, USA, 253–266. DOI: http://dx.doi.org/10.1145/3098822.3098841

[28] Jia Zhao, Jiangchuan Liu, Haiyang Wang, and Chi Xu. 2017. Multipath TCP for datacenters: From energy efficiency perspective. In *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*. IEEE, 1–9.