Experiences with MPTCP in an intercontinental OpenFlow network

Pol, R van der; Bredel, M; Barczyk, A; Overeinder, B; van Adrichem, NLM; Kuipers, FA

Publication date
2013

Document Version
Final published version

Published in
Proceedings 29th Trans European Research and Education Networking Conference

Experiences with MPTCP in an intercontinental OpenFlow network

Ronald van der Pol*, Michael Bredel†, Artur Barczyk†, Benno Overeinder‡, Niels van Adrichem§, Fernando Kuipers§

*SURFnet
Radboudkwartier 273
3511 CK Utrecht, The Netherlands
Email: Ronald.vanderPol@SURFnet.nl

†California Institute of Technology, c/o CERN
1211 Geneva, Switzerland
Email: michael.bredel@cern.ch, Artur.Barczyk@cern.ch

‡NLnet Labs
Science Park 400, 1098 XH Amsterdam, The Netherlands
Email: benno@NLnetLabs.nl

§Delft University of Technology
Mekelweg 4
2628 CD Delft, The Netherlands
Email: N.L.M.vanAdrichem@tudelft.nl, F.A.Kuipers@tudelft.nl

Abstract

The size and amount of e-science data sets is growing rapidly. Keeping up with the network demand in order to transfer these data sets over the Internet is a challenge. Single links do not have enough capacity anymore. Therefore we need to install more interfaces in the servers and use all available paths in the network. In this paper we describe two new technologies that help to optimally use the capacity of all multiple paths simultaneously. OpenFlow is used to discover the topology of the network, calculate multiple paths and configure those paths on the OpenFlow network. Multipath TCP (MPTCP) is used on the servers to distribute the load across the paths. We describe the end-to-end goodput measurements we did in our OpenFlow testbed. We show that we reach a much higher throughput with multiple paths compared to a single path.

KEYWORDS

MPTCP, OpenFlow, Multipath, Big data

I. INTRODUCTION

In this paper we present our experiences with multipath TCP (MPTCP) in an intercontinental OpenFlow testbed. We believe that using multipath routes will become an important network concept in the next few years. One of the major reasons is that data sets in e-science are increasing exponentially in size. To transfer these huge data sets we need to make efficient use of all available network capacity. This means using multiple paths simultaneously whenever possible.

Multipath routing can be done at network layer 3 with Equal Cost Multipath (ECMP) routing or at data link layer 2 with protocols like TRILL (IETF RFC 5556) [1] or IEEE 802.1aq (Shortest Path Bridging – P802.1aq-2012 [2]). In all these cases, load balancing across the paths is done based on flows by calculating a hash (based on e.g. Ethernet addresses, IP addresses and TCP/UDP port numbers) of the packets. Each packet of such a flow follows the same path through the network, which prevents out of order delivery within a flow. When the traffic has many different flows the traffic is evenly spread across the various paths by the load balancing mechanisms. But when there are only a few *elephant* flows dominating the traffic, which is typically the case in large data e-science applications, this is not the case. Another disadvantage of hashing is that usually all links get the same percentage of the hash values and therefore all the paths need to have the same capacity.

To gain experience with this concept we set up a testbed in which we transferred data between two servers. The testbed consisted of OpenFlow enabled switches and multiple paths (both link-disjoint and common-link) between the servers. We used the OpenFlow communications protocol to access and configure the switches' forwarding planes. An OpenFlow controller application provisioned multiple paths between the servers and MPTCP was used on the servers to simultaneously send traffic across all those paths. Figure 1 shows the testbed that was used for the SC12 demonstration in Salt Lake City in November 2012.

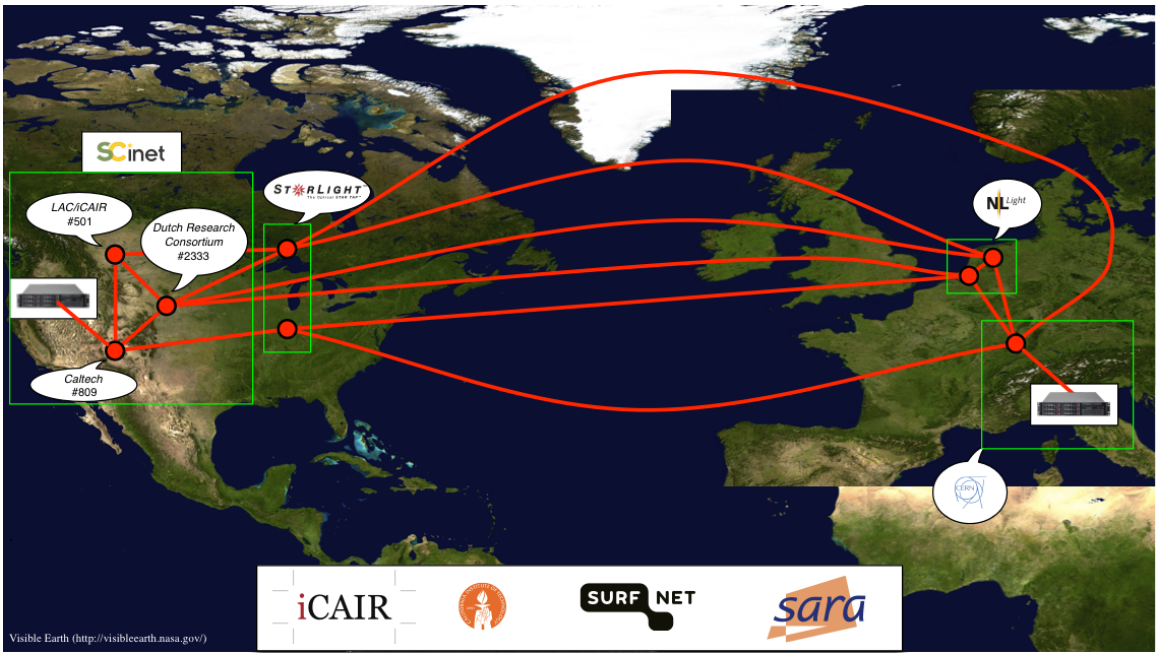

Fig. 1. SC12 Multipath OpenFlow Network Topology.

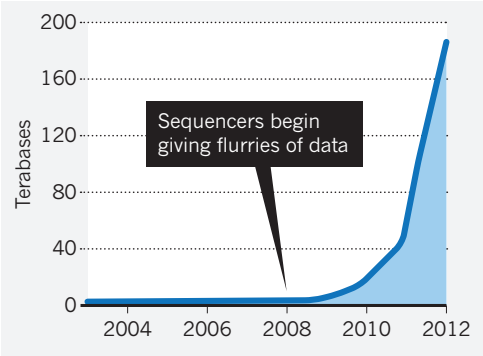

Fig. 2. The amount of genetic sequencing of data stored at the European Bioinformatics Institute doubles in less than a year (source: EMBL-EBI).

II. Challenges when transferring Big Data

The amount and size of data sets in e-science is growing steadily. More and more databases are put online and new data is generated daily. Due to its ever-growing size, transferring these huge data sets over the Internet is a challenge. In the past we have seen a growth factor of 2 per year, although recently the rate of increase in Internet IP traffic is around 1.4 per year (see TERENA Compendium, Section 4.3 [3]).

High energy physics and astronomy are known science disciplines that generate massive data sets. For example at CERN, the Large Hadron Collider experiments generate about 15 petabyte per year. But also recent developments in biology (e.g., systems biology and high-throughput genomics) make that life scientists are starting to generate and analyse huge data sets; and encountering the same challenges in handling, processing and moving information. As prices drops spectacular for automated genome sequencing, small laboratories can become big-data generators (see Fig. 2). And vice versa, laboratories without such equipment can become large data consumers. As an example, a research group is looking for changes in the genome sequence of tumors and want to compare their data with thousands other published cancer genomes. And ideally, they would routinely look at the sequenced cancer genomes. But with the current infrastructure that is impossible as downloading the data is too time consuming [4]. So, network bandwidth not only have to keep up with the data growth, but on top of that new applications also needs increased bandwidth for practical usage in routine tasks.

There are several challenges when these big data sets needs to be transferred via the Internet. We are reaching the limit of what can be sent via a single path. The amount of data that can be sent over a fiber is reaching the physical limit of what theoretically can be sent. Figure 3 shows what impact the Shannon limit has on the amount of data that can be sent over a

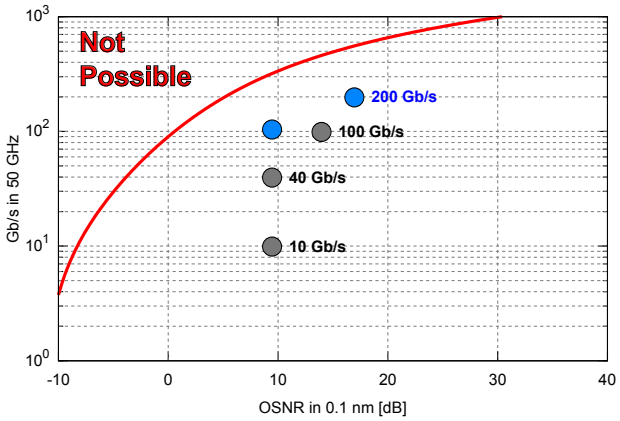

Fig. 3. Optical line systems are reaching the theoretical Shannon limit (source: Ciena).

single fiber. There is a *Not Possible* region which is dependent on the optical signal to noise ratio (OSNR). This OSNR can also be viewed as a measure of distance. Higher OSNR values represent shorter distances. There is a trade-off here between higher bandwidth with shorter reach and lower bandwidth with longer reach. The dots represent various Ciena optical line systems. The two dots closest to the Shannon Limit represent the latest technology. Both 100 Gb/s and 200 Gb/s systems are approaching the limit. Trying to reach the limit even more would result in very expensive line systems and therefore not economically feasible. Experts do not foresee a technology that can solve this problem within the next 3–5 years. Using multiple fibers in parallel is therefore the only solution out of this problem. This means introducing multipathing.

Another reason to employ multipathing is the fact that it takes time until higher speed interfaces for server and switches become affordable. At the moment 10GE and 40GE are affordable, but 100GE is still very expensive. Modern servers can easily fill 10GE and 40GE interfaces. So in order to get the most out of a server multiple interfaces need to be used. This is also a logical step after servers with multiple disks (RAID) and multiple CPUs and cores. There are several technologies for using multiple interfaces simultaneously and load balancing across those interfaces. Link aggregation and equal cost multipath routing (ECMP) are two of them. Both use hashes to distribute the load across the links. Hashing works fine when there are many flows. But when there is only one flow it will be mapped to one link only. Even with a small amount of flows the statistical nature of hashing is not optimal because the hashes map to the same link and the load is not evenly spread. ECMP and link aggregation are load balancing technologies within the network. It is also possible to do load balancing at the end nodes. Multipath TCP (MPTCP) is such a technology. The next section describes how MPTCP works.

III. MPTCP

Multipath TCP is a new approach towards efficient load balancing. Instead of letting the network do the load balancing by using hashes and ECMP, MPTCP is doing the load balancing in the end nodes as part of the TCP process. Multipath TCP (MPTCP) is described in RFC 6182 [5] and the "TCP Extensions for Multipath Operation with Multiple Addresses" internet draft [6]. MPTCP is an active working group in the IETF.

Figure 4 shows how MPTCP works. In an MPTCP enabled kernel the TCP component is replaced by an MPTCP component and a TCP subflow component for each interface. The MPTCP component receives a byte stream from the application (MPTCP uses an unmodified socket API and TCP semantics northbound, so applications do not need to be adapted). The MPTCP component splits the byte stream into multiple segments which are handed to the TCP subflow components. Each subflow behaves as a normal TCP flow to the network. MPTCP can handle paths of different bandwidth because there is a congestion control mechanism across the subflows. This congestion control mechanism takes care that traffic on a congested path is moved to a path with less congestion [7]. It adapts the load balancing according to the load of other traffic on the network.

The MPTCP component implements three functions. First, it takes care of path management by detecting and using multiple paths to a destination. When a connection sets up, the end-points negotiate their alternative IP addresses, which are assumed to be alternative paths. Second, packet scheduling splits the byte stream received from the application in multiple segments and transmits these segments on one of the available subflows. These segments are numbered to enable the receiving MPTCP component to put the segments in the correct order and reconstruct the original byte stream. Finally, there is congestion control across the subflows. This function spreads the load over the subflows. When a subflow becomes congested, traffic is moved to a subflow that is less congested. This function also takes care of retransmissions on another subflow when one of the subflows fails.

According to [7], the MPTCP congestion control mechanism ensures (1) fairness, both over shared bottlenecks as to single

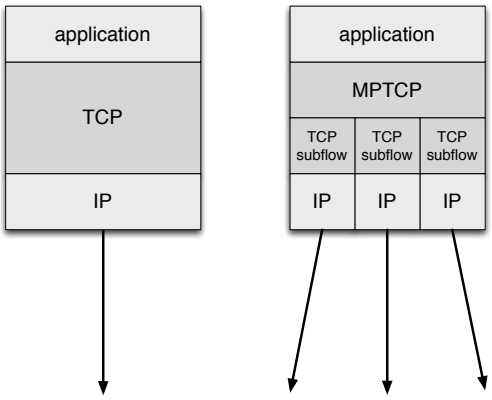

Fig. 4. Traditional TCP vs Multipath TCP

TCP streams, and (2) efficient use of available paths. To do so, every subflow $r \in R$ maintains its own TCP window size to control its congestion rate. When packet loss occurs, it decreases its window size ω_r to $\omega_r/2$, equal to the decrease of a regular single TCP window. However, when a subflow receives an ACK, it will increase its windows size ω_r by $\min_{S \subseteq R \cap r \in S} \frac{\max_{s \in S} \omega_s / RTT_s^2}{(\sum_{s \in S} \omega_s / RTT_s)^2}$. Meaning, that for all possible subsets $S \subseteq R$ including subflow r, it computes the highest congestion window compared to the sum of all window sizes, but normalizes for deviating (especially high) RTTs to react less aggressively on high delay subflows. The subflow increases its window size with the lowest value from the resulting sets to ensure fairness towards single-flow TCP connections.

Measurements in section V show that when subflows are dependent, meaning their paths share one or more bottlenecks, eventually the system converges to use independent paths[1].

IV. Related Work

Much work has been done in the area of multipathing and concurrent multipath routing in particular. A two-fold is found in the way the strategy of multipathing is executed. First, there are implementations (such as ECMP [8]) that are deployed on the network layer, enabling routers to forward packets by load-balancing over multiple paths without involving end-nodes. ECMP, however, does not allow one flow to benefit from multiple paths but divides multiple flows over different paths to prevent packet reordering and large jitter buffers. Second, there are client-server based implementations (such as MPTCP [5] itself) in which additions to the end-nodes of two connections are made to benefit from multiple paths available through multihoming.

CMT over SCTP [9], pTCP [10], M/TCP [11] are end-node-centric alternatives to MPTCP. According to [7], however, those do not provide fairness in comparison to other single-path connections due to uncoupled congestion control between the paths. The different proposals appear equal in terms that they stripe the bytes from the application TCP flow across different TCP subflows, but mainly differ in the way they apply congestion control. Concurrent Multipath Transfer (CMT) over SCTP, however, differs from the other alternatives that it uses the Stream Control Transmission Protocol (SCTP) instead of TCP as its transport layer protocol. SCTP natively supports multihoming and path selection, mainly used for robustness purposes, though does not support concurrent multipath forwarding natively.

Finally, OpenFlow itself is working on a specification to enable multipath utilization from a network controller perspective [12].

V. Measurements

A. GLIF and SC12 Measurements

In 2012 two demonstrations were given, one in October in Chicago during the Global Lambda Integrated Facility (GLIF) meeting and one in November in Salt Lake City during Supercomputing Conference (SC12). During the GLIF meeting we showed data streaming from Geneva to Chicago over multiple 10GE paths. On our servers we used the Linux MPTCP implementation of the IP Networking Lab at the Université Catholique de Louvain in Belgium [13]. Both servers had two 10GE network interface cards each. On these physical interfaces we configured two MAC VLAN virtual interfaces so that we could give each of the four virtual interfaces its own MAC address.

In our testbed the various paths through the network are set up by provisioning forwarding entries on the OpenFlow switches. Each of the four virtual interfaces was mapped to its own path through the OpenFlow network and each path had it own IP

[1]Which in our topology resemble the disjoint paths

subnet assigned to it. The OpenFlow forwarding entries matched on destination MAC and IP address. There was no routing in our testbed, so four subflows (paths) could be used by MPTCP.

During the GLIF demonstration we were able to reach an aggregated end-to-end throughput of around 3 Gbit/s. Later analysis showed that we used an MPTCP segment size of only 1400 bytes while the path MTU was 9000 bytes. This small MPTCP segment size was probably one of the reasons for the relatively low throughput.

During SC12, we showed OLiMPS and MPTCP by streaming from Geneva to Salt Lake City as part of the SCInet Research Sandbox. Figure 1 shows the topology used for the SC12 demonstration, while Fig. 5 shows the live monitoring website. In this experiment, we measured an aggregate of 16 Gbit/s between Geneva and Salt Lake City.

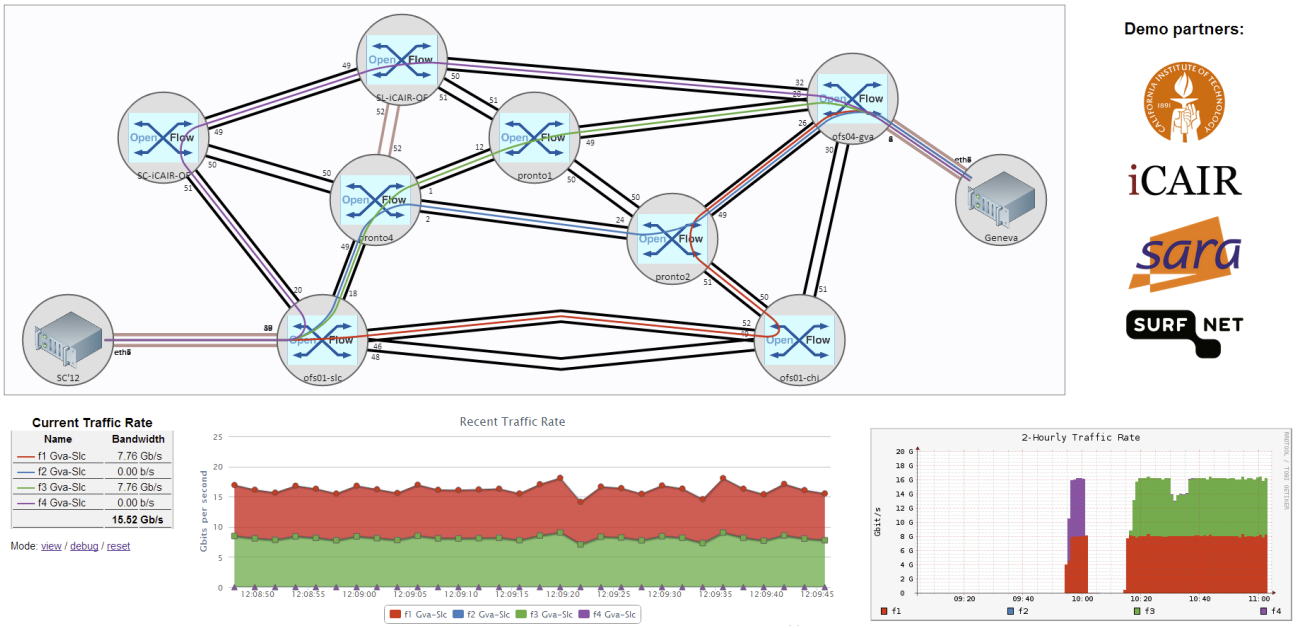

Fig. 5. Live monitoring website at SC12

In a different setup, we have streamed data between Geneva and Amsterdam, and by using four paths we were able to reach an aggregated end-to-end throughput of around 13 Gbit/s. Each of the four subflows is mapped to one of the four paths and has its own colour in the graphs (see Fig. 6). When the stream was started all four subflows were used. After a while, MPTCP only used the red and green subflows, and neglected the path between the two left-most OpenFlow switches. These two active subflows are the only two link disjoint paths between the servers, so it makes sense that MPTCP would eventually use only these paths. The congestion control algorithm that is explained in Section III manages the subflows and balances the load over the available paths. Figure 7 shows how initially all flows are used and after some time only the two link disjoint paths. Figure 8 show the subflow usage after some time.

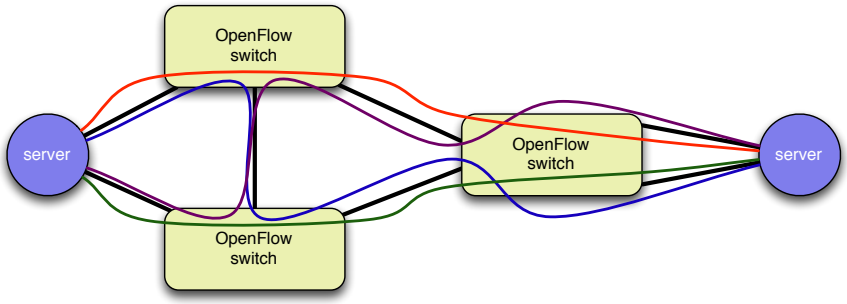

Fig. 6. Four paths between Geneva and Amsterdam

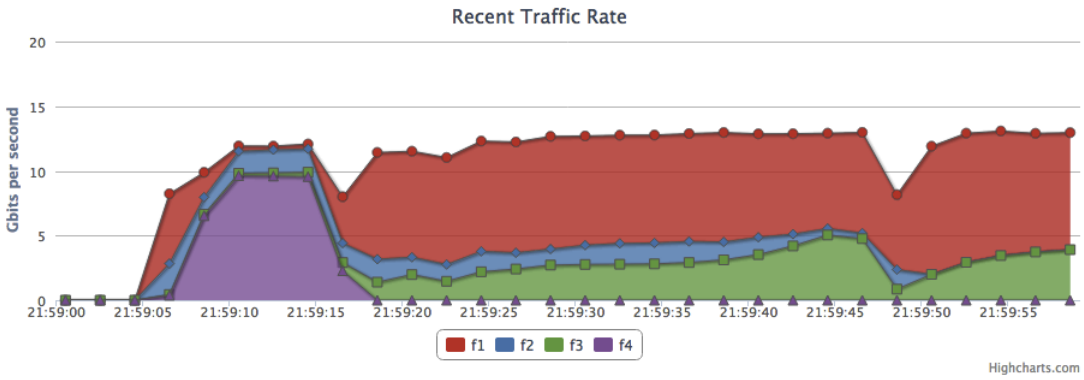

Fig. 7. Streaming between Geneva and Amsterdam, initial phase

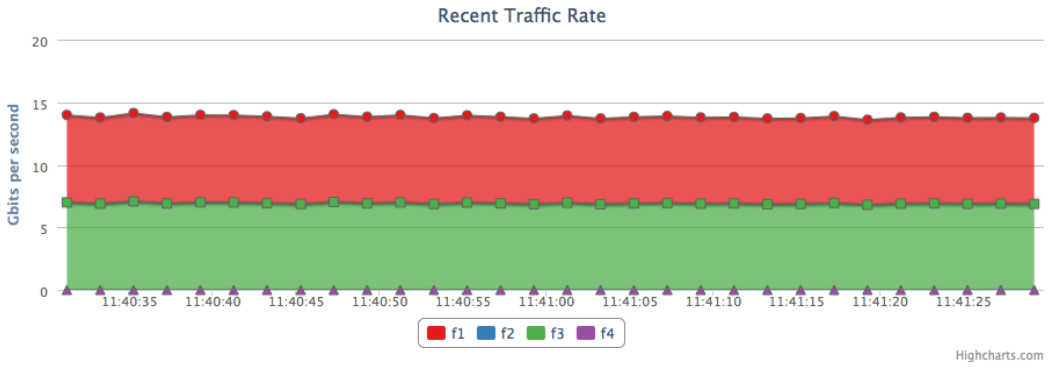

Fig. 8. Streaming between Geneva and Amsterdam, stable phase

B. SURFnet–US LHCNet Testbed

The SURFnet–US LHCNet testbed is an OpenFlow network infrastructure collaboration between Amsterdam (SURFnet/AMS) and Geneva (US LHCNet/GVA). In the current setup, the servers and OpenFlow switches have 10 GE interfaces, and the LAN/WAN links are 10 GE fibers. Figure 9 shows the interconnection topology of the AMS–GVA servers and OpenFlow switches in the testbed. A server and two OpenFlow switches are located at both datacenters in Amsterdam and Geneva. Each servers is connected with two OpenFlow switches in the datacenter, and between the two OpenFlow switch pairs, two 10 GE long distance links connecting Amsterdam and Geneva.

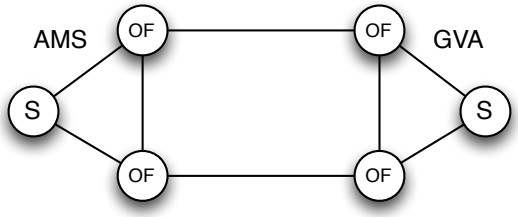

Fig. 9. SURFnet–US LHCNet testbed with OpenFlow switches and 10 Gb/s links.

The testbed allows for studying multipath networking and load balancing data flows using MPTCP and OpenFlow switches. In particular, path discovery and load balancing of an OpenFlow controller in combination with MPTCP congestion control mechanism appears a new and challenging research subject. The dynamics of both layers in balancing traffic over available links has to be in concert, while separation of layers make they are not aware of each other. Before evaluating the combination MPTCP and OpenFlow technology, we run a series of measurements with default Linux IP stack and static OpenFlow routes between AMS and GVA. The testbed has two disjoint paths through different equipment (besides the OpenFlow switches, different switch fabric for WAN links are deployed).

To test and assess the two paths in the testbed, we run a series of iperf measurements between Amsterdam and Geneva. The first measurement is a UDP bandwidth test, to make sure that no packet loss is seen on the links. With a bandwidth limit of 9 Gbit/s (option of iperf), we see a perfect 9.05 Gbit/s on the two paths in both directions AMS–GVA (see Table Ia). In a next series of measurements, we evaluated the TCP bandwidth of the two paths. In the AMS–GVA direction, we see a close to maximum bandwidth performance of 9.35 Gbit/s for both paths. In the opposite direction from GVA to AMS, we observed about a 2.5 Gbit/s bandwidth drop to 7.88 Gbit/s and 7.65 Gbit/s respectively (see Table Ib). We can not explain the differences in bandwidth for both directions, but using different firmware versions on the OpenFlow switches resulted in changes in stability and performance. This technology is still evolving and improving on high-bandwidth requirements.

direction	path #	bandwidth
AMS → GVA	path 1	9.05 Gb/s
	path 2	9.05 Gb/s
GVA → AMS	path 1	9.05 Gb/s
	path 2	9.05 Gb/s

(a) UDP bandwidth with iperf and "-b 9G" option.

direction	path #	bandwidth
AMS → GVA	path 1	9.35 Gb/s
	path 2	9.35 Gb/s
GVA → AMS	path 1	7.88 Gb/s
	path 2	7.65 Gb/s

(b) TCP bandwidth with iperf and 10 GE paths.

TABLE I
UDP AND TCP BANDWIDTH OVER 10 GE PATHS.

In Table II and Fig. 10 the results of MPTCP are shown. Both servers in Amsterdam and Geneva run an MPTCP enabled Linux 3.5 kernel. Similar to the TCP measurements, iperf is used for bandwidth tests. The average aggregate bandwidth measured over two minutes run is about 16 Gbit/s (see Table II). In Fig. 10 one can see the course of bandwidth over time. The MPTCP bandwidth has a slow start, it takes almost 10 seconds to attain its stable bandwidth of about 17 Gbit/s (hence the 1 Gbit/s difference with the result in Table II).

direction	path #	bandwidth
AMS → GVA	path 1 & 2	15.7 Gb/s
GVA → AMS	path 1 & 2	– Gb/s

TABLE II
MPTCP BANDWIDTH WITH IPERF OVER TWO 10 GE PATHS.

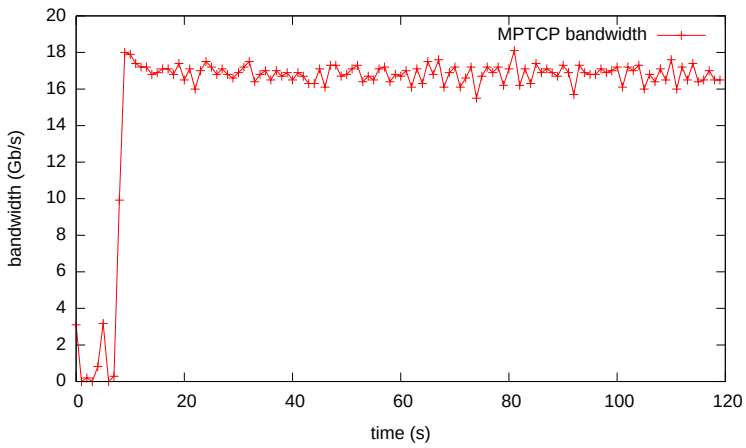

Fig. 10. MPTCP bandwidth over time.

VI. CONCLUSION

Multipath TCP and OpenFlow technology has the potential to exploit path diversity between two endpoints. The path diversity not only improves stability (alternative path in case of link failures), but the MPTCP/OpenFlow combination can also be used to bundle paths and balance traffic over the paths, resulting in higher attained bandwidth between the endpoints. The demonstrations and measurements performed in the past year shows the availability of the technology and the viability of the approach.

For stability in operational environments, work has to be done on OpenFlow traffic load balancing like the OLiMPS controller. Tools like these can exploit the path diversity by dynamically directing traffic over the available links to optimize throughput. On the MPTCP part, work needs to be done on MPTCP congestion avoidance algorithms. In particular, the massive data flows (elephant flows) over long distance links have different requirements than is accommodated by current congestion avoidance algorithms. The high bandwidth and long distance (with relative large RTTs) makes existing MPTCP congestion avoidance very sensitive to packet loss. Future work will be focusing on evaluation of appropriate algorithms for this class of data traffic patterns.

Acknowledgements

The work of SURFnet is made possible by the GigaPort program of the Dutch Government and the Géant3 project of the European Commission. Caltech research in this area is supported by the DOE Office of Advanced Scientific Computing Research (OASCR). The authors like to thank Christoph Paasch for helping with MPTCP kernel tuning of the Linux servers in the testbed.

References

[1] J. Touch and R. Perlman, "Transparent interconnection of lots of links (TRILL): Problem and applicability statement," IETF, RFC 5556, May 2009.
[2] "IEEE Approved Draft Standard for Local and Metropolitan Area Networks: Bridges and Virtual Bridged Local Area Networks – Amendment 9: Shortest Path Bridging," IEEE P802.1aq/D4.6, Mar. 2012.
[3] "TERENA compendium of national research and education networks in Europe," 2012. [Online]. Available: www.terena.org/compendium
[4] V. Marx, "Biology: The big challenges of big data," *Nature*, vol. 498, no. 7453, pp. 255–260, Jun. 2013.
[5] A. Ford, C. Raicu, M. Handley, S. Barre, and J. Iyengar, "Architectural guidelines for multipath TCP development," IETF, RFC 6182, Mar. 2011.
[6] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP extensions for multipath operation with multiple addresses," IETF, RFC 6824, Jan. 2013.
[7] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in *Proceedings of the 8th USENIX conference on Networked systems design and implementation*. USENIX Association, 2011, pp. 8–8.
[8] C. E. Hopps, "Analysis of an equal-cost multi-path algorithm," IETF, RFC 6182, 2000.
[9] J. Iyengar, K. Shah, P. Amer, and R. Stewart, "Concurrent multipath transfer using SCTP multihoming," *SPECTS 2004*, 2004.
[10] H.-Y. Hsieh and R. Sivakumar, "pTCP: An end-to-end transport layer protocol for striped connections," in *Network Protocols, 2002. Proceedings. 10th IEEE International Conference on*. IEEE, 2002, pp. 24–33.
[11] K. Rojviboonchai and A. Hitoshi, "An evaluation of multi-path transmission control protocol (M/TCP) with robust acknowledgement schemes," *IEICE transactions on communications*, vol. 87, no. 9, pp. 2699–2707, 2004.
[12] "Multipath proposal." [Online]. Available: http://www.openflow.org/wk/index.php/Multipath_Proposal
[13] "Multipath TCP – Linux kernel implementation." [Online]. Available: http://mptcp.info.ucl.ac.be/

Vitae

Ronald van der Pol is a network researcher at SURFnet. He has been working in the field of Education and Research Networks for more than twenty years. His former employers include the VU University in Amsterdam, NLnet Labs and SARA. His current interests are in new network technologies and how these can be applied to next generation networking. In recent years he worked on monitoring and management of optical transport networks, carrier Ethernet, end-to-end performance of demanding applications and OpenFlow. He holds masters degrees in both Physics and Computer Science and is co-author of several IPv6 RFCs.

Michael Bredel is a network research engineer at Caltech based at CERN. He received a Diploma of Electrical Engineering from the Technische Universität Darmstadt, Germany and a Ph.D. from the Leibniz Universität Hannover, Germany. His current research activities are related to network measurements, network management, software defined networks and OpenFlow. His particular interests are new technologies that can improve the movement of Big Data, like for the Large Hadron Collider.

Artur Barczyk is working since 2007 in the Caltech HEP Networking group as a senior research engineer. He is the technical team lead in US LHCNet, the transatlantic, mission-oriented network for LHC data. Artur holds a PhD in Particle Physics.

Benno Overeinder is a senior research engineer at NLnet Labs. Previously, he has worked at the University of Amsterdam and VU University Amsterdam on research in computational science, parallel and grid computing, and distributed systems. At NLnet Labs, his research interests include scalability, stability, robustness, and security of the Internet infrastructure, in particular related to DNS and routing. Benno holds a PhD in Computer Science.

Niels van Adrichem is a PhD student in the Network Architectures and Services group at Delft University of Technology (TUDelft). His research revolves around Future Internet Architectures with a specialization towards Information Centric Networking, Software Defined Networking and Optical Networks.

Fernando Kuipers is associate professor in the Network Architectures and Services group at Delft University of Technology (TUDelft). He received the M.Sc. degree in Electrical Engineering at TUDelft in June 2000 and subsequently obtained his Ph.D. degree (with honors) in 2004 at the same faculty. His research interests mainly revolve around network algorithms and cover Routing, Quality of Service, Network Survivability, Optical Networks, and Content Distribution. His work on these subjects has resulted in over 40 publications in refereed journals and conferences and include distinguished papers at IEEE INFOCOM 2003, Chinacom 2006, IFIP Networking 2008, IEEE FMN 2008, IEEE ISM 2008, and ITC 2009.