Evaluation of SDN Enabled Data Center Networks Based on High Radix Topologies

Bogdan Andrus[(1)(4)], Victor Mehmeri[(1)], Achim Autenrieth[(4)]

(1) Department of Photonics Engineering, Technical University of Denmark, Kgs. Lyngby, Denmark.
email: bogan@fotonik.dtu.dk
(4) ADVA Optical Networking SE, Fraunhoferstr. 9a, 2152 Martinsried / München. Germany.

Juan José Vegas Olmos[(1)(2)], Idelfonso Tafur Monroy[(1)(3)]

(2) Mellanox Technologies, Denmark.
(3) Department of Electrical Engineering, Technical University of Eindhoven, Netherlands.

***Abstract*— The relevance of interconnects for large future datacenters and supercomputers is expanding as new technologies like Internet of things (IoT), virtual currency mining, High Performance Computing (HPC) and exa-clouds integrate further into data communication systems. The Data Center Network Layer is the workhorse that manages some of the most important business points by connecting the servers between them and delivering high performance to users. Evolution of the networking layer has seen, in addition to improvements of individual link speeds from 10Gb/s to 40Gb/s and even 100Gb/s and beyond, quite important changes in the topological design. Traffic intensive server-centric networks and high performance computing tasks are pushing a shift from the conventional Layer 2 oriented fat tree architectures with multiples tiers towards clos networks and other highly interconnected matrices. Optimal performance and reliability perquisites imposed on the network cannot be fully achieved by solely changing the topological design. A software-centric control of the network enables the use of additional redundant paths not only for increased performance but also reliability concerns. By decoupling the network control from individual devices and centralizing the network intelligence inside a Software Defined Network (SDN) controller, dynamic workloads can easily be accommodated with the deployment of custom modules or applications for traffic management. In this paper, we focus on the innovations for next generation data center networks from a twofold perspective. On the one hand, we evaluate the applicability of new potential interconnection schemes like torus, hypercube, fat tree and jellyfish in regard to identified key metrics such as performance, complexity, cost, scalability and redundancy. Our evaluation comprises of a mathematical interpretation of the graphs with a focus on the abstract metrics (e.g., bisection bandwidth, diameter, port density, granularity etc.) followed by a simulation of the scalable networks in a virtual environment and subject them to various traffic patterns. On the other hand, we introduce an emulated SDN test framework, which decouples the control plane from the interconnection nodes and gives a centralized view of the topology to a controller handling the routing of the internal workflows for the data center. With the use of our SDN enabled testbed we demonstrate and highlight the clear superior performance gain of centralizing the network intelligence inside a software controller, which allows us to apply a custom routing algorithm.**

Keywords- SDN; Data Center topologies; Torus; Hypercube; Jelly Fish; Fat Tree.

I. INTRODUCTION

New technological trends like IoT, virtual currency mining, High Performance Computing (HPC), exa-clouds integrate further and further towards data communication systems making the role of interconnects more important than ever before. The current global evolution of data center traffic is predicted to reach an annual rate of 15.3 zettabytes (ZB) - with a monthly rate of 1.3 ZB - by the end of 2020 [1]. This prediction translates into tripling traffic demands over a period of 5 years spanning from 2015 to 2020 with a compound annual rate of 27%. The distribution of data center related traffic regards the majority of connections established inside the data center with a quota of 77% for server-to-server communication. Major factors influencing such patterns are identified in distributed computing/processing as well as reliable and fast migration of sizable volumes of data across vast domains of physical servers. Such circumstances highlight the importance of the network topology in the process of designing data centers, on account of the fundamental limits (e.g., cost and performance) imposed by the chosen interconnection graph.

Meanwhile, Big Data and Cloud applications, which are subject to exponential growth rates are pressuring Data Center enterprises to drastically improve their infrastructure not just to meet the increased bandwidth demand but also additional QoS perquisites related to certain applications and services. As a consequence, such priorities are placing the boost for network capacity in the top research directions weather they focus on enhancing individual link capacity to 40Gb/s, to 100Gb/s or beyond or by deploying special network topologies and routing structures [2]-[5]. An

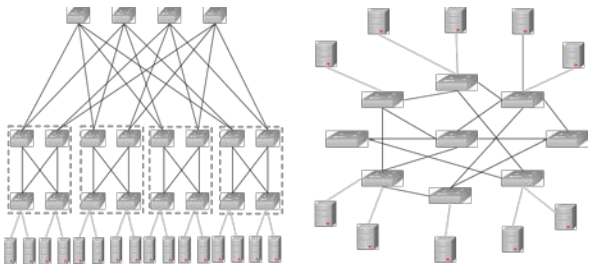

Fig. 1. (a) - Fat Tree (left) and (b) - JellyFish (right) topologies

 ISBN: 978-1-61208-591-3

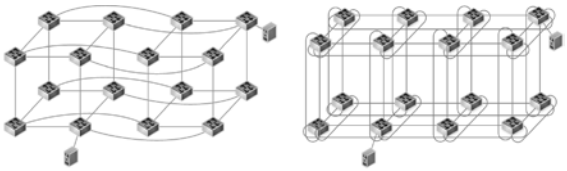

Fig. 2. (a) - Hypercube (left) and (b) - Torus (right) topologies

interconnection matrix is not only an important component in a server-centric Data Center construct but also crucial element for efficient on-chip networks [6]. For this reason, topologies originally adopted in parallel computing platforms and on-chip networks, that present higher interconnectivity between nodes, have gained more and more traction in Data Center network deployments [7].

As identified in [8], one major constraint not tackled by these graphs is the issue of granular scalability or the capability of adding a flexible number of servers or increasing capacity while maintaining structural properties. The number of redundant links, but most importantly their placement in the overall graph raises the network's capability to experience local failures without major impact on operations. The reliability (fault tolerance) of such constructs entails a compromise in terms of additional underutilized spare links or a disproportionate increase of hop count in link failure situations. Furthermore, implications connected to the manual configuration of such networks and potentially complex routing mechanisms not only translate into additional costs related to Capital Expenditure (Capex) but also Operational Expenditure (Opex).

Software Defined Networking (SDN) approach to network management and configuration seeks to bring the flexible programmability needed by real time applications and services, which can considerably reduce the set-up time. One major benefit from adopting an SDN framework relates to lowering Capex and Opex costs. Firstly, low-cost white box switches can be developed by detaching the control plane functionality from all network devices. This step is followed by centralizing their behavior inside a software controller responsible for the management and control operation of the entire network. As such, minimizing the expenses is achieved by replacing the large number of nodes that are capable of supporting complex path computation algorithms with white-box switches that provide fewer features but present a more flexible and reconfigurable alternative. Maintaining a general overview and supervision of every network device inside a dynamic software controller can facilitate overall network management and configuration. Therefore, by automating the manual operations of managing and configuring every network device (also required when scaling such intricate interconnections), Opex oriented costs can also be lowered.

The contributions of this paper can be divided into several sections. In Section 2, we extend our previously presented mathematical analysis [9] of high radix topologies (e.g., Torus, Hypercube) with regard to indications on performance, cost, latency of implementation for new topologies (e.g., Fat Tree, Jellyfish). Section 3 highlights the results and behavior of scaling such topologies in a simulation environment (e.g., [10]) using a random traffic pattern. In Section 4, we present the evaluation testbed for measuring the performance (e.g., network throughput, delay, jitter, and loss rate) of an SDN implementation employing each topology against conventional Spanning Tree Protocol deployment based on our previously published works [11, 12]. Finally, in the Section 5, we present and discuss the results obtained from the simulation and emulation testbeds.

II. BACKGROUND ON NETWORK TOPOLOGIES

A key component in the performance of a Data Center architecture network is the topology. The impact of a topology is not only significant for the global network ratio of performance vs. cost but also for the failure resiliency aspect. Traversing nodes and links incurs energy, and since the number of hops for the various paths is affected by the interconnection implementation, an important role in the energy consumption can also be easily identified.

A hypercube graph, Figure 2(a), is an n-dimensional generalization of a cube also called n-cube and comprises of 2n nodes. One main characteristic is the high connectivity and small diameter however, not very easy to scale due to complexity. A torus topology can be visualized as a three-dimensional mesh in the shape of a rectangular prism with all the nodes on each face having an additional connection to the corresponding nodes on the opposite face, as illustrated in Figure 2(b). Torus based networks are usually employed in top performing supercomputers due to their high radix, relatively low cost and reduced diameter compared to mesh. Another widely deployed Data Center topology is Fat-Tree, Figure 1(a), which is capable of delivering high bisection bandwidth due to its path multiplicity and maintain a low and constant diameter if the number of layers remains constant with scaling. On the other hand, Jellyfish, Figure 1(b), a random graph and multipath based topology, has been proven to be more cost-efficient than a Fat-Tree using identical devices, providing support for 25% more servers running at full capacity [13]. Furthermore, Jellyfish graph provides an attractive solution for a more granular expansion and allows heterogeneity in switch port count, a desired advantage in terms of flexibility.

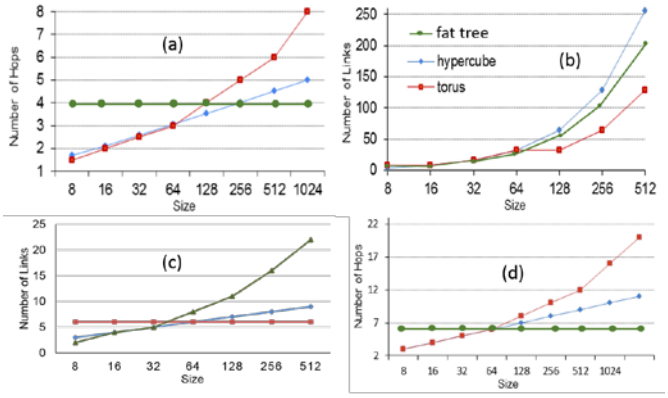

Fig. 3. Average Distance (a), Bisection Bandwidth (b), Node Order (c), Diameter (d)

Some relevant mathematical parameters by which topologies can be characterized and compared in a preliminary network design stage have been identified in

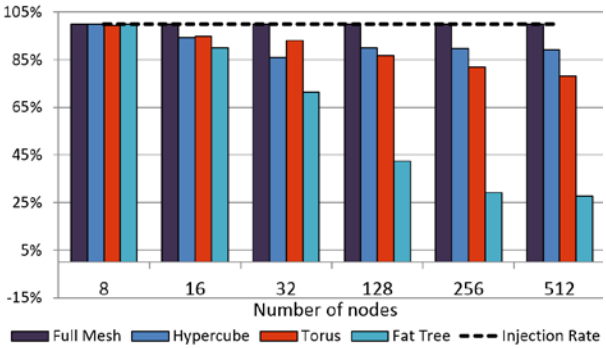

Fig. 4. Average throughput per node

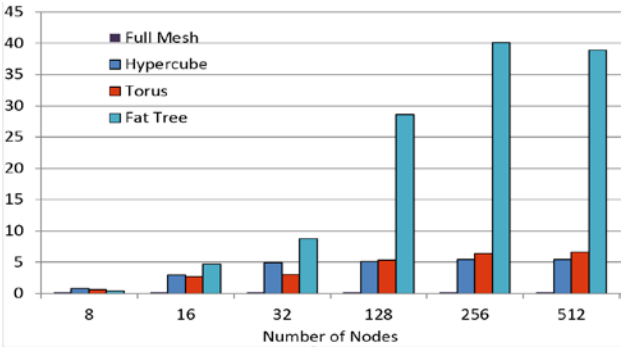

Fig. 5. Average delay per flow

previous research [14]. Such parameters can serve as the building blocks for clarifying certain prospects, in relation to scalability for each network but also in relation to each other.

Bisection bandwidth (Figure 3(b)), the bandwidth sum of all the links across a cut through the middle of the network, gives the link density and bandwidth indications that can be achieved by a certain implementation strategy. Even though the order of growth is similar in torus, hypercube and fat tree, the bisection reaches larger values for hypercube and fat tree. This asset of superior bandwidth comes with a setback related to cost and complexity of wiring on the hypercube side. However, this difference is almost inexistent for 64 nodes and below.

The longest path between any two nodes calculated on the shortest path tree (diameter), is arguably a sign of packet latency, as seen in Figure 3(d). Another similar guideline is average distance in the network, highlighted in Figure 3 (a), which also supports latency in relation to communication patterns. While a similar tendency is observed for both hypercube and torus having a logarithmic and a linear increase, respectively, a 3 layer (e.g., edge, aggregation, core) fat tree maintains a constant diameter if the number of layers is unchanged when expanding.

Node order represents the number of interconnection links (ports) required for each switch and relates to network throughput, however, implementing a system with a high node order implies an increased execution complexity cost. Like in the previous cases, for graphs up to 64 nodes the

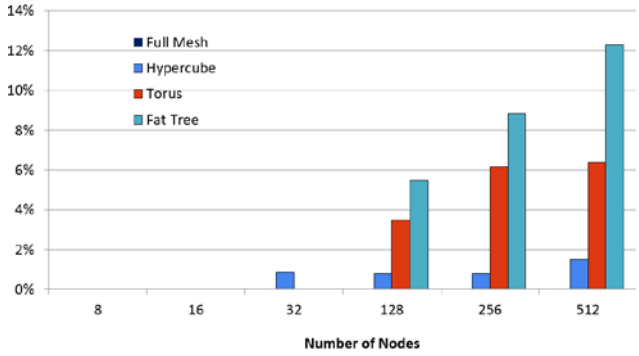

Fig. 6. Average loss rate per flow

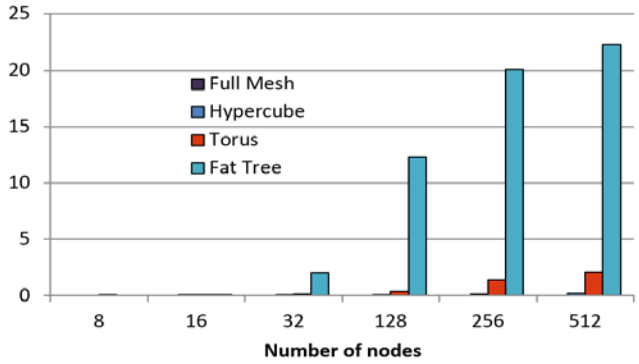

Fig. 7. Average jitter per flow

differences are relatively small between interconnections. By observing the evolution of this parameter, Figure 3(c), we note that the torus network can scale up and maintain a constant node degree. However, the fat tree is characterized by a higher increase rate in switch-to-switch port count when graphs scale.

The next section presents the results of simulating the expansion of the topologies with regard to key performance metrics: throughput, latency, lost rate and jitter.

III. NETWORK PERFORMANCE SIMULATIONS

The following simulations try to provide a comprehensive interpretation of the highly interconnectable networks assessed. In this scope, when setting up our simulation models, we are considering topology characteristics, communication pattern and the amount of data injected in the network as being the most relevant components for our scenarios.

In order to simulate the proposed scenarios, we used NS3, an open source discreet event network simulator capable of supporting network performance measurements related to throughput, delay, number of lost packets, jitter etc.

As in our previous evaluation [9], we have selected a shortest path based routing algorithm as opposed to a Spanning Tree Protocol implementation in order to evaluate the real potential of the topologies without blocking redundant ports for communication. The networks are subject to a uniformly distributed random traffic model that is widely accepted and is unbiased towards various

 ISBN: 978-1-61208-591-3

topologies (e.g., fat tree behaves better under localized traffic patterns between neighbors in the same pod/cluster).

Besides the traffic pattern, the amount of traffic that will be injected in the interconnection network also plays an important role. Above a certain threshold the network begins to saturate, the ratio between throughput and injection rate starts to drop and the efficiency decreases. The saturation limit for hypercube and torus is 60% and 40%, respectively [15]. Therefore, we configure the input traffic level to be at 30% the maximum link capacity, below the saturation limit. Due to the random traffic pattern selection the results were averaged from a runtime of 30 seconds during which application based connection flows would be established between randomly paired hosts in the network.

As expected from the initial abstract metric analysis, we observe from Figure 4 that, even though the throughput per flow is declining when the networks scale from 8 nodes up to 512, the hypercube topology presents the most consistent behavior. A decrease of approximatively 5% in hypercube compared to 15% corresponding with the torus or a drastic 70% decrease in fat tree performance is measured.

Similar behavior occurs in the investigation of the delay (Figure 5) where even though the torus and hypercube demonstrate a similar delay, the 3-layer fat tree does not indicate well performance on scaling with an average of up to 40 ms per flow in 512 switched network. Same trend is observed when analyzing the rate of lost packets (Figure 6) and jitter (Figure 7) where hypercube and torus outperform the fat tree based network.

We have simulated and tested the performance of a full mesh topology (all switches have direct connections to every other switch in the network) under the same conditions as the other interconnects in order to perform a sanity check for the setup. First, this serves as a verification for all the configuration parameters employed, data rates, individual link delays, simulation time etc. Secondly, the test demonstrates the confidence in the results, which were not affected by hardware processing power when the topologies scaled from 8 to 512 nodes.

We can conclude from this section that, even though the cost indications and performance characteristics are similar for all studied topologies with nodes under a 64 count, clear superior operations are displayed by the hypercube followed by the torus.

IV. SDN For Data Center Interconects

Applying high radix topology designs into large data centers by employing current routing or switching technologies encounters numerous obstacles. A Layer 2 topology builds upon high density port switches that can process packets at line speed therefore, this implies less configuration and administration overhead. However, such a solution does not scale well due to the limitations of a flat topology and restrictions to a broadcast domain.

A routing based deployment can segment the broadcast domains, use existing performant routing protocols and present better scaling properties. Nevertheless, this comes at the expense of additional delays incurred by additional packet processing times in routers, which are not only slower but are also more expensive and require more complex administration [16]. In data centers, generally we see a compromise based on a combination of the two but also replacing network elements with expensive multilayer switches is an available option. However, SDN can truly get the best of both scenarios by giving routing capabilities to lower cost, white-box switches and automate administration operations with a topology manager module in the controller.

SDN adoption has raised many concerns about its impact on performance and scalability mainly due to the fundamental aspect of decoupling the control plane from the data plane. Ideas that a centralized controller is unlikely to scale as the network grows has led to some reluctance and certain expectations that some failure would occur when the number of incoming requests increases over supported limits. These assumptions can generally be sourced to the misconception that SDN implies the use of one physically centralized controller. Architectures involving a distributed control plane are, however, a valid way to construct a Software Defined Network and address the scalability issue, while also providing control plane resilience. Such solutions have already been demonstrated in projects like Onix and HyperFlow [17][18]. Yeganeh argues in [19], that there is no underlying bottleneck to SDN scalability, such a concern is not fundamentally unique to SDN. Even though a distributed SDN architecture would incur similar manageability problems as in a non-SDN approach, it would still be significantly easier to manage compared to having multiple heterogeneous switches running autonomous, vendor-specific applications.

V. Performance Evaluation of SDN DC Topologies

With the scope of evaluating and comparing SDN versus STP implementations on Hypercube, Torus and Jellyfish topologies, we used Mininet, a network emulation software that uses process-based virtualization to run multiple virtual OpenFlow switches and hosts on a same physical machine.

The SDN network controller of choice was Floodlight. The integrated topology and forwarding module enable the efficient use of the high number of redundant paths for networks like torus and hypercube and calculates connections based on shortest path between each pair of source and destination. However, link contention still occurs, influencing performance as seen in the results.

We have employed Iperf, a linux networking tool in order to measure network performance characteristics. We have focused on the same performance metrics identified as the most common attributes for network characterization in academic research, similar to our previous simulation scenarios: throughput, packet delay, jitter and loss rate. For the same reason as stated in Section 2, random data traffic patterns were configured, as well as a shortest path routing algorithm in the Floodlight SDN controller. To benchmark the fidelity of our test setup, a full mesh interconnection was assessed and subjected to the same tests.

 ISBN: 978-1-61208-591-3

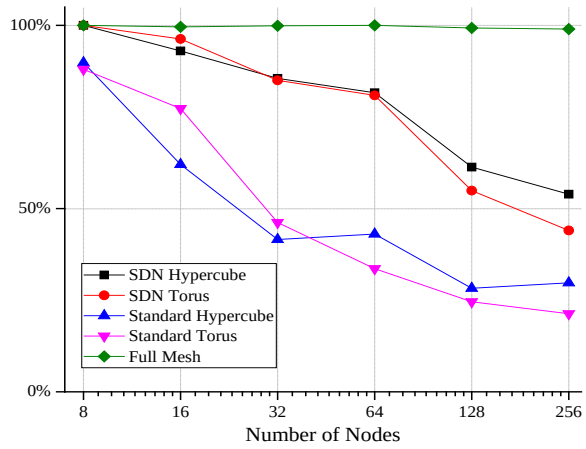

Fig. 8. Average throughput per node

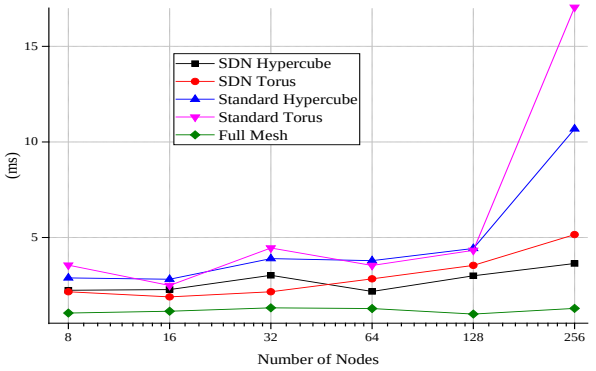

Fig. 9. Average latency per flow

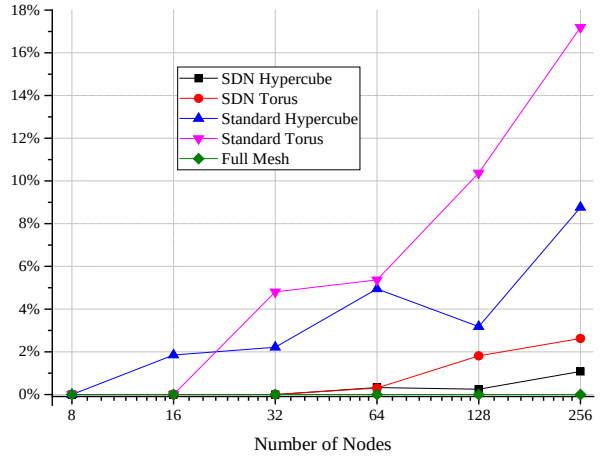

Fig. 10. Average loss rate per flow

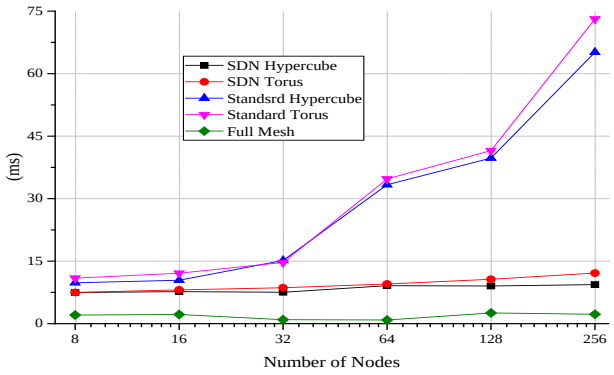

Fig. 11. Average jitter per flow

Normal traffic operation is presumed, as well as a constant injection rate (simple TCP transport) per application running on each server. Since network contention grows, the average throughput for each connection decrease when the networks expand in size including more switches and hosts.

In Figure 8, we observe the measured throughput is higher by roughly 45% for the SDN test cases with 256 nodes torus and hypercube. A 13ms decrease of packet delay, for SDN torus of the same size is noticeable in Figure 11. Loss rate is also reduced considerably for both topologies with at least 7 percentage points, as shown in Figure 9. Connection stability is improved with SDN technology by a reduction of jitter with 50ms, measurement that applies to networks of 256 switches; see Figure 10.

We plot in Figure 12 the comparison of the average throughput normalized by the theoretical link bandwidth capacity in the Jellyfish scenario. A two-fold increase in performance is observed for the SDN setup with 120 switches, with the difference slowly decreasing as the network scaled. The average packet delay, as seen in Figure 13, was considerably lower in the SDN scenario, with a small dependence on the number of switches employed and presenting less than 1/6th of the delay measured with STP for networks with more than 150 switches. Network jitter and loss rate were also more favorable in the SDN scenario: in Figure 14, we observed a reduction in jitter varying from 7% to 33%, and Figure 15 shows that the packet loss between the two systems remain within a 2% difference range independently of the number of switches.

VI. CONCLUSION AND FUTURE WORK

The first part of our paper focused on the rich and diversified design space of Data Center topologies and the differences between them. We can infer from our simulation results and the mathematical evaluation that, even though the cost indications and performance characteristics are similar for all studied topologies with under 64 nodes, clear superior operations are displayed by the hypercube followed by the torus on the downside of wiring complexity and scalability cost.

Our SDN versus STP emulation results demonstrate that the SDN deployments based on the studied topologies torus, hypercube and jellyfish, considerably outperform the STP implementation in throughput, latency, jitter and loss rate. No larger networks were tested due to the limitations of our emulation environment indicated by the degradation of the full mesh performance. Such results are representative for

 ISBN: 978-1-61208-591-3

small to medium sized Data Centers however, much larger scales can be achieved with a distributed control plane solution [18][19], whereas the network setups discussed could represent individual clusters among many.

In addition to our previously presented work [11][12], concerning the performance gains achieved with SDN in highly interconnected network topologies, these results strengthen the arguments referring to how SDN can be used to leverage the multiple redundant paths in a network. Furthermore, we believe that a combination of SDN with MultiPath Transmission Control Protocol (MPTCP) can lead to an even more efficient network infrastructure utilization.

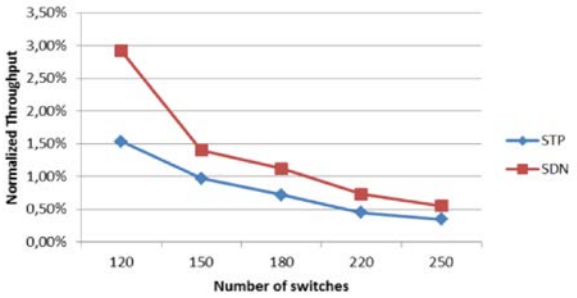

Fig. 12. Average throughput per node (Jellyfish)

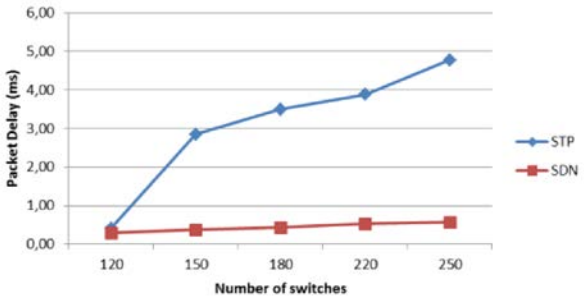

Fig. 13. Average delay per flow (Jellyfish)

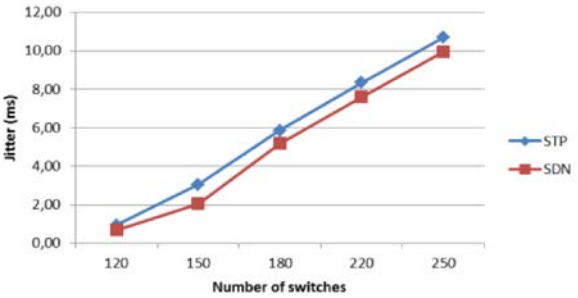

Fig. 14. Average jitter per flow (Jellyfish)

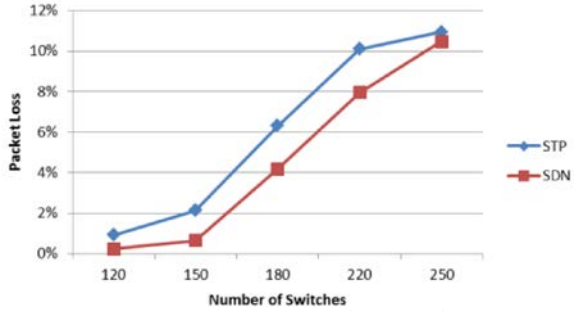

Fig. 15. Average loss rate per flow (Jellyfish)

ACKNOWLEDGEMENTS

This work has been funded by the EU FP7 Marie Curie project ABACUS and by CNPq, *Conselho Nacional de Desenvolvimento Científico e Tecnológico* – Brazil.

REFERENCES

[1] Cisco Global Cloud Index: Forecast and Methodology, 2015–2020, White Paper, 2016.

[2] M. Al-Fares, et al., “A Scalable, Commodity Data Center Network Architecture,” in Proceedings of the ACM SIGCOMM conference on Data communication (ACM), pp. 63-74, New York, 2008.

[3] G. L. Vassoler, et al. “Twin Datacenter Interconnection Topology, " in IEEE Micro, Vol.34, Issue 5, pp.8-17, 2014.

[4] H. Wu, et al. "MDCube: a high performance network structure for modular data center interconnection, " In Proc. of ACM CoNEXT, 2009.

[5] A. Greenberg et al. ,“VL2: A scalable and flexible data center network, ” in Proc. of the ACM SIGCOMM conference on Datacommunication (ACM), pp.51-62, New York, 2009.

[6] J. Chen, P. Gillard, C. Li, “Performance Evaluation of Three Network-on-Chip Architectures (Invited),” in First IEEE International Conference on Communications in China: Communications QoS and Reliability (CQR), pp. 91-96, 2012.

[7] J. K. Dennis Abts, "High Performance Datacenter Networks", by Morgan & Claypool, 2011.

[8] A. Singla et al., "Jellyfish: networking data centers randomly" in Proc. Of the 9th USENIX conference on Networked Systems Design and Implementation, pp. 17-17, California, 2012.

[9] B. Andrus, J. J. V. Olmos and I. T. Monroy, ”Performance Evaluation of Two Highly Interconnected Data Center Networks”, in Proc. International Conference on Transparent Optical Networks, Budapest, 2015.

[10] Network Simulator 3, found at: https://www.nsnam.org/.

[11] B. Andrus, J. J. V. Olmos, V. Mehmeri and I. T. Monroy „SDN data center performance evaluation of torus and hypercube interconnecting schemes, “ in Advances in Wireless and Optical Communications (RTUWO), 2015.

[12] V. Mehmeri, J. J. V. Olmos, and I. T. Monroy “Capacity Extension of Software Defined Data Center Networks With Jellyfish Topology, ” in Asia Communications and Photonics Conference (ACP), 2015.

[13] A. Singla et al.,, "Jellyfish: networking data centers randomly" in Proceedings Of the 9th USENIX conference on Networked Systems Design and Implementation, California, pp. 17-17, 2012.

[14] W. J. Dally, “Performance Analysis Of K-Ary N-Cube Interconnection Networks,” in IEEE Transactions On Computers, vol. 39, no. 6, pp. 775-785, 1990.

[15] A. S. Muhammed Mudawwar, "The k-ary n-cube Network and its Dual: a Comparative Study," in IASTED International Conference on Parallel and Distributed Computing and Systems, 2001.

[16] K. Jayaswal, “Administering Data Centers”, Wiley Publishing, 2005.

[17] T. Koponen et al., “Onix: A Distributed Control Platform for Large-scale Production Networks”, in Proc. of the 9th USENIX conference on Operating systems design and implementation, California, Article No. 1-6, 2010.

[18] A. Tootoonchian and Y. Ganjali, “HyperFlow: a Distributed Control Plane for OpenFlow”, in Proc. of the Internet Network Management Conference on Research on Enterprise Networking, California, pp.3-3, 2010.

[19] S.H Yeganeh, et al., "On scalability of software-defined networking", in IEEE Communications Magazine (Institute of Electrical and Electronics Engineers, New York), Vol. 51, Issue 2, 2013.

 ISBN: 978-1-61208-591-3