Using SDN and Reinforcement Learning for Traffic Engineering in UbuntuNet Alliance

Josiah Chavula, Melissa Densmore, Hussein Suleman
Computer Science Department
University of Cape Town
South Africa
Email: jchavula@cs.uct.ac.za

***Abstract*—Software Defined Networking (SDN) provides opportunities for dynamic and flexible traffic engineering. This paper discusses how UbuntuNet Alliance National Research and Education Networks (NRENs) can improve bandwidth utilization and reduce inter-NREN latencies through implementation of SDN-based traffic engineering and applying network metrics in selection of inter-NREN paths. Additionally, the paper looks at the utility of applying Reinforcement Learning to path selection, using network data obtained through an SDN controller. Results from simulations using the UbuntuNet topology show an increase in total throughputs when multipath is employed. Furthermore, simulation results show that where latency is the key metric for computing rewards, lower latencies are achieved.**

***Index Terms*—Software Defined Networking, Reinforcement Learning, Multipath Traffic Engineering, NRENs, UbuntuNet Alliance.**

I. Introduction

The UbuntuNet Alliance - the regional internetwork for National Research and Education Networks (NRENs) in southern and eastern Africa - recently embarked on the Africa Connect Project to build high-speed NREN interconnections through cross border terrestrial fiber optic networks. The UbuntuNet topology now comprises eight Network Operating Centers (NOCs) acting as interconnection points, as well as multiple intra-continental and transcontinental links. A detailed description of the current UbuntuNet topology is given in Section III-B. However, despite the improved physical interconnection, a recent investigation into traffic routing between African NRENs revealed that a larger proportion of the inter-NREN traffic still traverses exchange points in Europe, resulting in high latencies [1]. More generally, the level of peering and interconnectivity among Africa's Internet Service Providers (ISPs) remains low [2]. This suggests that optimal end-to-end path selection for Africa's inter-NREN communication remains a problem.

Optimal path selection requires that the quality of links in the topology is continually evaluated to ensure that paths with better performance have a higher probability of being utilized [3]. However, for large scale networks, the use of end-to-end active measurements for dynamic path ranking is neither efficient nor scalable [4]. Given that SDN controllers maintain a global view of the topology and have, at their disposal, a large volume and variety of network data, it is worthwhile to investigate a data driven approach. Some studies [5], [6] have shown that correlations learned from network controller data can be utilized to improve resource allocation and network performance. It is worthwhile therefore to investigate a data driven [7] approach where controllers data is used to inform path selection.

Software Defined Networking (SDN) provides new opportunities for flexible management of Internet routing and packet forwarding [8]. SDN separates switches' control plane from the forwarding plane, and this separation enables remote and dynamic configuration of forwarding tables. As a result, SDN achieves at least three important things that are useful for interdomain traffic engineering [9]: forwarding packets based on multiple header fields, remote configuration of forwarding rules, and dynamic/programmatic configuration of the forwarding rules.

This paper evaluates how the ability to discover alternate paths and dynamically configure routes based on path characteristics, could help improve optimal utilization and network performance of the fiber optic cable system between Africa's NRENs. More specifically, this paper discusses how the UbuntuNet Alliance can improve bandwidth utilization and reduce inter-NREN latencies by implementing SDN-based traffic engineering and applying network metrics to achieve dynamic selection of paths. Further, the paper looks at the utility of applying Reinforcement Learning (RL) to path selection by using network data obtained through an SDN controller and through inter-switch probing. For evaluation, an SDN-based network emulation is implemented in Mininet, applying Reinforcement Learning algorithm Q-learning to distribute traffic through multiple forwarding links, with the aim of maximizing throughput and reducing latency.

II. Related Work

Standard approaches for influencing selection of paths across multiple domains have relied on manipulating the Border Gateway Protocol (BGP), but these approaches have been unreliable and inefficient [10]. A key problem is that, as an inherently single path system, BGP does not disseminate alternate routes. Each BGP router selects and advertises only the best path [11] to its neighbors. By propagating only a single path (default route), the multipath diversity available in an internetwork is diminished. Yet, it is not always the case that the default BGP routes offer the best performance. One study [12] has shown that in multipath environments, better alternative paths with lower loss rate and delay are available between 30% and 80% of the time.

Another multipath challenge is that at the data-link layer, most networks deal with loops by implementing Spanning Tree Protocol (STP) to determine the loop-free paths (spanning tree) between every pair of switches in the network. The use of a spanning tree path between pairs of the SDN switches results in redundant links in the network being disabled. As a result, end-to-end communication gets restricted to single paths when, in fact, redundant and possibly better paths are available. STP eliminates the multipath capability that should otherwise be available in the topology.

A number of SDN-based multipath traffic engineering approaches have been proposed with the aim of improving network resilience and performance [13]. For example, HiQoS [14] makes use of multiple paths between source and destination and applies a queuing mechanism to guarantee QoS for different types of traffic. The approach calculates path costs using a weighted combination of the estimated price, stability, physical distance and bandwidth of the links. HiQoS controller periodically measures the bandwidth utilization of each queue along the path, and the path with the minimal bandwidth utilization of a queue is selected as the optimal path for a new flow. Similarly, M2SDN [15] considers link utilization to dynamically schedule flows towards multiple less loaded paths. M2SDN calculates link costs based on utilization and packet drop rate, and attempts to split traffic on multiple paths, applying a path dependency parameter so as a to minimize usage of paths with intersections. [16] attempt to select multipath routes dynamically based on available network resources. The approach forwards flow data into multiple routes from the source to the destination based on utilization rate of the network for every route from the source to the destination.

Another multipath SDN implementation is Google's B4, an SDN WAN [17] that uses OpenFlow to centrally control WAN switches. The system is built with three key characteristics: balancing competing application demands at the network edge during resource constraint; using multipath forwarding/tunnelling to leverage available network capacity in accordance with application priorities; and dynamically reallocating bandwidth in the face of link/switch failures or shifting application demands. The architecture aggregates source to destination flows based on QoS requirements to form forwarding groups (FGs), and a universal controller installs FG-based rules in multiple site switches to form end-to-end tunnels.

The multipath traffic engineering proposed in this papers makes use of Reinforcement Learning algorithm Q-learning to enable forwarding devices to adapt and improve network performance by learning from experience. The design consists of an SDN controller, as well as a RL engine and Q-learning agents, working together to dynamically configure optimal forwarding rules.

III. Design and Implementation

This section provides a background and motivation for RL/Q-learning, describes the UbuntuNet topology, and presents a set of experiments designed to test the utility of applying Q-learning in a multipath SDN traffic engineering model.

A. Reinforcement Learning

The problems solved by reinforcement learning generally involve sequential decisions that can be modeled as Markov Decision Processes(MDPs) [18]. An MDP agent acts in an environment modeled as a tuple (S, A, P, R); where S is a set of states, A is a set of actions, and $P(s'|s, a)$ is a transition model for the probability of entering state s' after executing action a at state s. It must be assumed that there is at least one corresponding action a such that $P(s'|s, a) = 1$, i.e executing action a at s implies sending a data packet from router s towards s' results in the packet subsequently being at s'. Further, $R(s, a, s')$ represents the reward given to the learning agent for executing action a at s that caused transition into state s'. The rewards act as reinforcement signals to adjust forwarding-link priorities so as to enhance or diminish the probability that a specific next-hop is selected for the traffic [18]. A routing agent learns to adjust path selection policies based on experience and rewards and, through continuous modification of action selection policies, attempts to maximize some cumulative pay-off [18].

The state-action pairs utility value is called the Q-value, and is calculated by the Q-function. A Q-learning agent finds an optimal control policy by iteratively approximating its Q-values using prior Q-value estimates, a short-term reward $r = \rho(s, a) \in R$, and a discounted future reward. Thus, the goal of maximizing the cumulative reward is represented by an action-value function $Q(s, a)$:

$$Q^*(s, a) = (1 - \alpha)Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a')] \quad (1)$$

where the learning rate $\alpha \in (0, 1]$ models the rate of updating the Q-values, i.e how fast new information overrides previously learned information, and $\gamma \in (0, 1]$ represents a discount factor that scales the importance of the immediate reward (obtained for the action at s) versus rewards obtainable for actions at the resultant state s'.

B. Modelling the UbuntuNet Topology

The UbuntuNet core topology forms a ring through the alliance's Network Operation Centers (NOCs) in Cape Town, Mtunzini, Maputo, Dar-es-salam, Nairobi, Amsterdam, London, and back to Cape Town. NRENs in landlocked countries are connected via terrestrial fiber optic cables to the coastal NOCs: from Lusaka to Cape Town and Dar-es-salam; Lilongwe to Lusaka; Luanda to Cape Town; Kigali and Kampala to Nairobi.

There were two key metrics for the experiment; latency and link capacity. To estimate latency between NRENs, link delays were calculated from estimated fiber optic cable lengths.

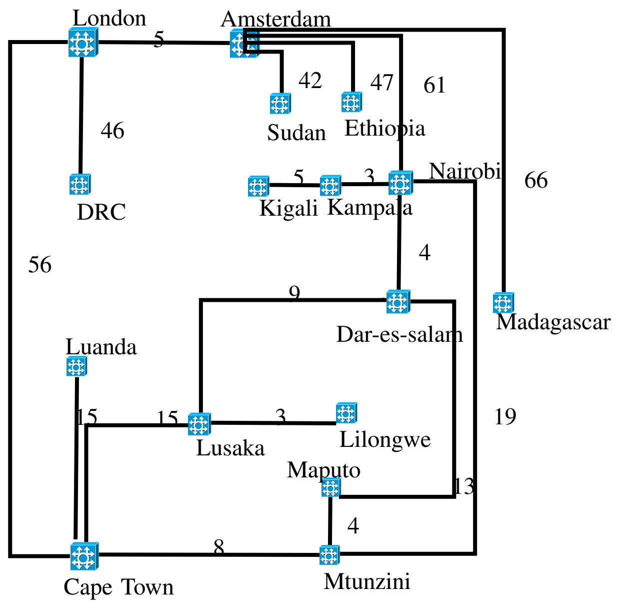

Fig. 1. UbuntuNet Alliance Topology

Terrestrial distances between inland cities were obtained using Google Maps roads, whereas distances between sea port NOCs were obtained using PortDistance [1]. These distances were used to estimate link delays between NOCs, translating every 200km to 1 ms latency. In Figure 1, the link weights represent the link delays used in the experiments.

In terms of link capacities (bandwidth), the UbuntuNet Alliance has, as of June 2016, a total link capacity of 2.18 Gbps linking the alliance's region to Europe [19]. This capacity includes 2 STM-4 links (2 X 622 Mbps) on the east cost of Africa, from Mtunzini to Amsterdam, with landing points in Maputo, Dar-es-Salam and Nairobi. On the west African cost, the capacity comprises of a single STM-4 (622 Mbps) and 2 STM-1 links (2 X 155 Mbps), from Cape Town to London. The backbone between the UbuntuNet countries is made up of STM-4 links (622 Mbps): between Dar-es-Salam and Cape Town via Lusaka; between Mtunzini and Nairobi; and between Mtunzini/Maputo and Dar-es-Salam. There are also 2 STM-4 links (2 X 622 Mbps): between Nairobi and Kampala; and a single STM-4 between Kampala and Kigali.

Figure 1 shows the topology used in this study. Multiple physical links between a pair of NOCs were aggregated into single links. The overall link capacities were scaled down by a factor of 10 to cope with limitations of the network emulator. This was done to cope with link speed limitations within the Mininet network simulation environment.

C. SDN Topology

In this experiment, the UbuntuNet Alliance core topology (shown in Figure 1) was emulated as an SDN network, with each of the NOCs in the alliance represented with an SDN switch. The topology was built in a Mininet SDN emulator [20], which provides for a network controller, Openflow switches, linux hosts and network links. Furthermore, Mininet's hosts run a standard Linux kernel and network stack, and can therefore run real network applications. An Ryu Openflow controller was connected to the Nairobi NOC - chosen to host the SDN controller because it is the most central NOC in the topology. NRENs were also modeled as switches connected to the core topology switches. The UbuntuNet topology map indicates that four of the alliance's members - Sudan, Ethiopia, Madagascar and the Democratic Republic of Congo (DRC) - are connected to UbuntuNet through either the London or Amsterdam NOCs. Traceroute measurements suggest DRC is directly connected to the London IXP, while the other 3 are directly connected to the Amsterdam IXP.

As Figure 1 shows, the UbuntuNet has loops in its topology. If the NRENs were to implement a mechanism for dynamic multipath selection, it would be possible for KENET to dynamically exchange traffic with TENET through either Amsterdam or Mtunzini. TENET would also be reachable from KENET via either London or Mtunzini gateways.

As an illustration, consider the topology in Figure 1, and the traffic between the Kenyan NREN, KENET, and the South African NREN, TENET. KENET has its NOC in Nairobi, whereas TENE has two NOCs; in Cape Town and in Mtunzini. Thus, if the topology were to have a mechanism for dynamic multipath selection, traffic between KENET and TENET could flow through the four paths:
(1) $CapeTown \leftrightarrows London \leftrightarrows Amsterdam \leftrightarrows Nairobi$;
(2) $Mtunzini \leftrightarrows Nairobi$;
(3) $Mtunzini \leftrightarrows Maputo \leftrightarrows DarSalam \leftrightarrows Nairobi$; and
(4) $CapeTown \leftrightarrows Lusaka \leftrightarrows DarSalam \leftrightarrows Nairobi$.

In the emulated topology, the Link Layer Discovery Protocol (LLDP) [15] was used to obtain link and switch states in the topology. All forwarding paths were stored and used as alternate routes for each source-destination switch pair. The use of LLDP helped maintain a global view of network topology and retain a multipath environment.

D. Implementing Q-learning in the SDN

To implement Q-learning in the experimental topology, each node (SDN switch) is modeled as a state s, and a next-hop switch as s'. Performance rewards are calculated based on distance (packet delay) between s and s', available capacity on the $s \leftrightarrow s'$ link, and the resultant load (number of flows) at the next hop s'.

Each of the core topology's SDN switches conduct active and passive measurements to monitor its egress links. A network controller collects performance and utilization data from all the core switches and links, and employs the Q-learning algorithm to calculate rewards and adjust forwarding rules at each node.

In this study, the Q-learning implementation consisted of a local Q-values table at each network node, as well as a global aggregation table managed by the network controller. Each switch in the topology performs QoS measurements to each of its next neighbors, and the results from such a measurement

[1]https://www.searates.com/reference/portdistance/

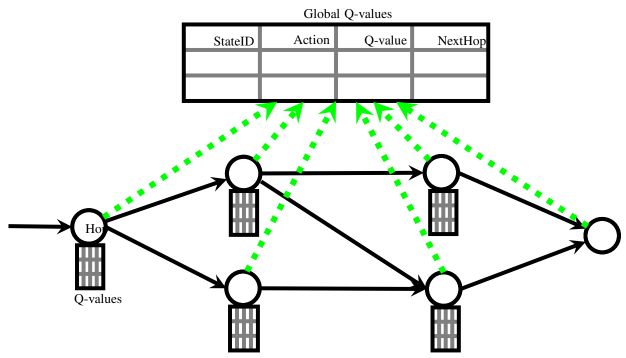

Fig. 2. Local and global Q-value tables

(latency, jitter, packet loss) are passed on to the controller. The controller uses the active measurement data, as well as interface-level statistics (number of flows, packet count) to calculate a reward value that it uses to update the Q-values.

Once the path metrics to the next hop have been collected, an aggregation function is used to calculate the reward as a composite value of path metrics. Each metric value is scaled to decimal of a maximum possible value. For example, path delay is scaled as a fraction of the maximum possible delay on a link (1000 ms considered in the experiments). Link load is a fraction of the available link bandwidth versus the capacity of the link.

In this implementation, a weighting variable Λ is used to aggregate a set of path metrics K, into a reward value $r(s,a) = \sum_{i=1}^{n} \Lambda_i K_i$; where $\sum_{i=1}^{n} \Lambda_i = 1$, and K = {latency, available-bandwidth}.

The Q-values records consist of tuples with a state identifier, action, a pointer to the next state for each action, and reward value for the action. A state represents a hop in the network topology. Since each hop handles traffic going to different destinations, a state in this work is defined by the node name and a destination's IP prefix. This means each hop may have several states associated with it, one for each destination IP prefix for traffic going through it.

After taking a state-action and the reward having been calculated, a record $< s, a, s', Q(s,a) >$ is written into the Q-values table.

On SDN switches, Q-values are transformed into interface priorities that determine the next hops for each destination. The process of selecting the optimal end-to-end path for traffic flow is achieved by probabilistically selecting the forwarding link based on Q-values at each switch.

E. Q-learning procedure and Path Selection

As the traffic flow commences between the source and destination hosts, a network controller commences learning episodes in which the controller performs active and passive measurements between all adjacent SDN switches. Active measurements are used to measure latency between switches, whereas passive measurements are used to obtain the load and residual capacity in each link. Capacity in this sense is measured in terms of available link bandwidth relative to the number of flows and packets coming through each switch interface.

IV. Experimental Evaluation

This section describes a set of experiments that were conducted to evaluate the simulated SDN topology, with reinforcement learning being used to adjust forwarding rules.

1) **Single lowest latency path forwarding:** This experiment was setup with the aim of evaluating performance when a single path is selected for each flow. The controller determined and configured the single lowest latency path between each pair of hosts in the network. This is implemented by using only delay between adjacent switches to calculate the rewards and Q-values. Each switch then forwarded all the traffic towards the egress link that had the highest Q-value.
2) **Single highest capacity forwarding:** This experiment was setup with the aim of evaluating performance when a single path is selected for each flow. The controller determines and configures the single highest capacity path between each pair of hosts in the network. This was implemented by considering only the residual bandwidth on the links to calculate the rewards and Q-values. The switch forwarded all the traffic on the egress link that has the highest Q-value.
3) **Multipath forwarding based on latency and capacity:** This experiment was setup to evaluate the performance when the switches forward traffic through multiple paths towards the destination host. The rewards and Q-values are calculated based on delay between switches, as well as the residual capacities in the link. The switches then use the Q-values to split the flow packets probabilistically, in fixed size blocks, to the egress links' Q-values.
4) **Multipath forwarding based on latency:** In this experiment, multiple links paths were used for each flow, but the rewards and Q-values were influenced only by the path delay. The egress link that was part of the shortest delay path to the destination was awarded higher rewards, and thus carried more traffic to the destination.
5) **Multipath forwarding based on capacity:** In this setup, multiple forwarding paths were used, with the reward and Q-values being influenced solely by links' residual capacity. The egress link that is part of the path with the highest capacity receives higher rewards and Q-values, and therefore carries more traffic.

V. Test Traffic

Each experiment primarily aimed to measure three aspects of network performance: latency, throughput, and jitter, aggregated at flow level and network level. In all the experiments, Iperf [21] was used for measuring performance characteristics. More specifically, the TCP version of Iperf was used to measure throughput between network hosts, while the UDP version of Iperf was used to measure latency and jitter.

To measure throughput, each end host randomly selected another end host and initiated a TCP Iperf throughput test for 60 seconds. All end hosts looped through this process for at least 20 mins, thereby measuring throughput to at least 20 other end hosts.

To measure latency, jitter and packet loss, all the end hosts again looped through the process of randomly selecting another end host and initiating and Iperf transmission. The traffic characteristics used in this case were based on [22] as follows:

1) **Protocol flow:** UDP to TCP ratio: 3:1
2) **Flow Duration:**
   - 0 - 2 sec : 45% of all the traffic
   - 2 sec - 5 mins : 55% of all the traffic
3) **Flow rate:**
   - Short flows (0 - 60 sec) : 1 Bps - 10 kBps
   - Medium flows (1 min - 5 mins) : 100 Bps - 50 kBps

VI. Results and Discussion

This section describes the results for the experiments. The metrics that are presented in this paper are throughput, latency and jitter. The first part describes the aggregate results for the whole topology. A more detailed discussion on network performance is given in the Section VI-B, focusing on a single source-destination pair - Cape Town and Nairobi. These two nodes have multiple routes between them and are used in this for illustration.

A. Network wide performance

Network-wide results indicates that the multipath configuration achieved higher throughput, but with higher jitter and latency. One possible cause for high jitter is due to packets of the same flow traveling on different paths. With contiguous packets of the same flow taking different paths, they have a higher possibility of arriving at the destination with variable delays, and possibly out of order. Out of order packets need to be buffered and reassembled by the receiving TCP device, and this could potentially result in increased packet loss and retransmissions.

B. Performance between two end nodes - Cape Town and Nairobi

This section focuses specifically on Cape Town and Nairobi, two of NOCs in the UbuntuNet topology between which multiple paths exist.

1) Throughput: Results from the network emulation experiments indicate a substantial differences with regards to the range of achieved throughput between single path communication and multipath communication. Multipath configurations achieved throughput levels that are generally higher single path configuration. However, it is noted the total achieved bandwidth is still less than the aggregate capacities of the multipath links. The only single path mechanism that had higher throughput that multipath is the one in Experiment 5, where the single highest capacity path was used for each source-destination pair. As can be observed from Figure 3 and Figure 4, Experiments 2 and 5 (both use available bandwidth for path selection) achieved the highest throughput, with Inter-Quartile Ranges (IQR) of 40 - 65% and 45 - 70 Mbps respectively. This is higher than the IQR of 30-55 Mbps achieved for Experiments 1, 3 and 4.

It was expected that Experiment 5 would be have the highest throughput given that it employs multipath forwarding and favors higher capacity links. However, it is noted that Experiment 5 achieved slightly lower throughput than Experiment 2, in which a single highest capacity link was used. The reduction in throughput can be attributed to packet loss and retransmissions owing to packets arriving out of order. Since packet multiplexing in Experiment 5 does not consider path latencies, there is an increased likelihood of contiguous packets being forwarded towards paths that have significant differences in delays, resulting in higher packet loss. In Experiment 2, on the other hand, since only a single highest capacity link was used, higher throughput was achieved without the negative effects of the out of packets. Although Experiments 2 and 5 achieved higher throughputs, their disregard for path delay resulted in the worst latencies.

Experiment 1 achieved the lowest throughput. This should be expected as the setup uses only a single path for forwarding, and calculates rewards and Q-values based on link delays without any regard to link capacity.

2) Latency: Figure 5 shows the recorded latencies between the Cape Town and Nairobi nodes in the topology, with the highest latencies being for Experiments 2 and 5. This was to be expected, considering that both Experiments 2 and 5 don't consider path delays in rewarding forwarding paths, instead, they only consider the link capacities.

Experiment 1 achieves the lowest and least dispersed latencies. This should be expected, considering that this configuration chooses a single lowest latency path. As a result, all the flow's packets flow on the same path, thereby having very small deviations in the packets' end-to-end latencies.

3) Jitter: Jitter is caused by deviations in packet delay, and this can easily be the case when packets of the same flow follow different paths. As can be observed from Figure 6, the single path setups in Experiments 1 and 2 experienced the least amount of jitter. On the other hand, all the three multipath approaches had significant levels of jitter. Experiment 5 had the highest jitter, as expected, due to its non-consideration for delay when allocating forwarding rewards/Q-values.

Dealing with jitter in packet-level forwarding requires careful balance for the size of the packet blocks. Ideally, if the latency difference between the multiple paths is significant, then it is helpful to use bigger blocks of packets (number of packets forwarded to each link based on the Q-value). On the other hand, if the latency difference between the paths is insignificant, then it might be helpful to use smaller buckets of packets so as to maximize utilization of all the links.

VII. Conclusion and Future work

This paper has shown how different types of QoS can be achieved by the use of SDN's dynamic path configurations.

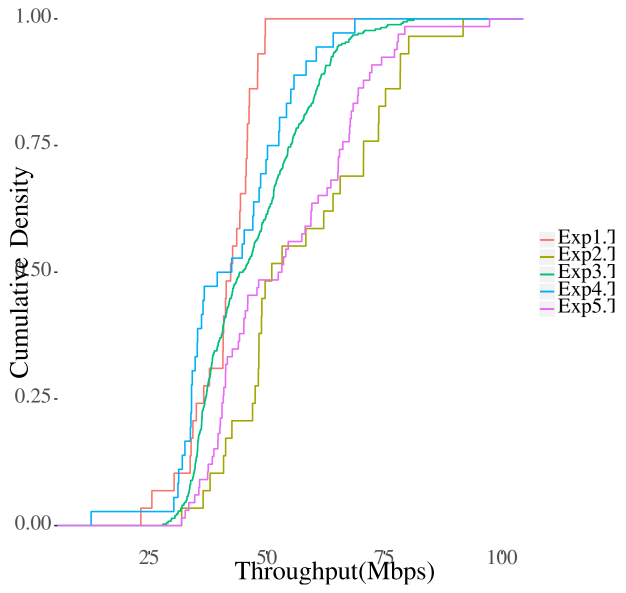

Fig. 3. Throughput distribution between Cape Town and Nairobi

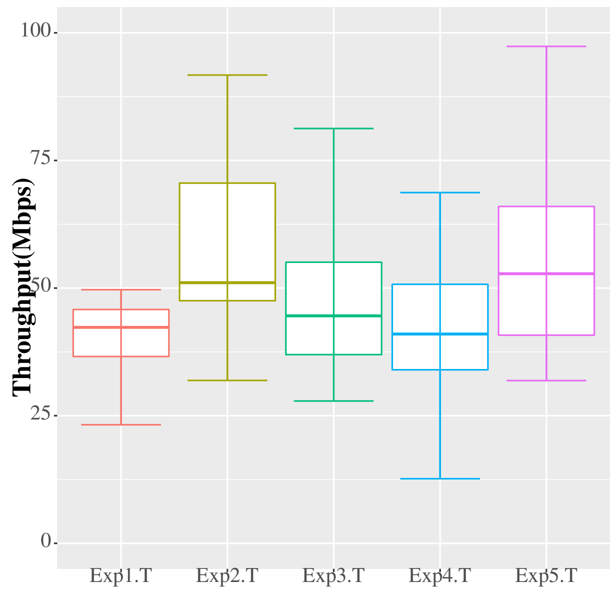

Fig. 4. Throughput between Cape Town and Nairobi

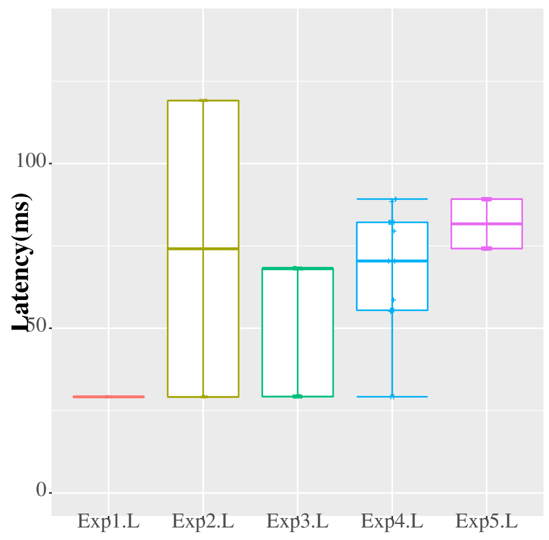

Fig. 5. Latency between measurements between Cape Town and Nairobi

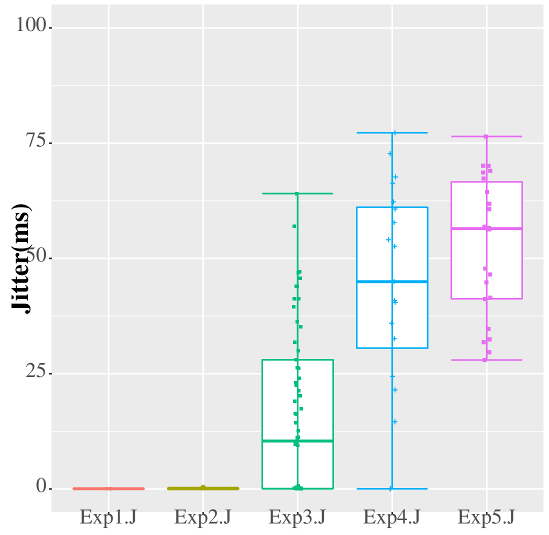

Fig. 6. Jitter between Cape Town and Nairobi

The results indicate that packet level multipath forwarding is able to increase throughput, but also introduces significant levels of jitter. The best throughput in multipath setting was achieved when the primary determinant of reinforcement rewards was the links' available bandwidth (Experiment 5). However, this configuration gave the worst performance in terms of jitter and latency. Of the multipath configurations, the best performance in terms of latency and jitter was obtained when the rewards were given on the basis of both the available link capacity and latency (Experiment 3). On the other hand, single path forwarding is seen to provide the lowest jitter. In terms of latency, the best performance is obtained with single path forwarding, where the rewards are based on the link delays.

A key challenge observed was high level of jitter when multipath is used. To deal with this problem, the configuration must aim to minimize the usage of different paths for contiguous data frames. This can be achieved by having larger packet buckets sizes, so as to minimize the number of contiguous packets following the different paths. However, if the bucket sizes are going to be too large, then for prolonged periods, only one of the links will be heavily utilized, while the other links are idle. This will be almost similar to single path forwarding. One way to solving this problem is by dynamically setting the optimal packet bucket sizes, by considering the delay imbalance between the multipaths. Future work will evaluate mechanisms for setting the optimal size of the packet bucket sizes so as to maximize usage of the multipaths while minimizing jitter.

Acknowledgments

The authors acknowledge and appreciate the financial support received for this research from the Hasso-Plattner-Institute (HPI), Potsdam University.

References

[1] J. Chavula, N. Feamster, A. Bagula, and H. Suleman, "Quantifying the effects of circuitous routes on the latency of intra-africa internet traffic: A study of research and education networks," vol. 147, pp. 64–73, 2015.

[2] A. Gupta, M. Calder, N. Feamster, M. Chetty, E. Calandro, and E. Katz-Bassett, "Peering at the internets frontier," in *Passive and Active Measurement Conference 2014*, 2014.

[3] R. Desai and B. Patil, "Cooperative reinforcement learning approach for routing in ad hoc networks," in *Pervasive Computing (ICPC), 2015 International Conference on*. IEEE, 2015, pp. 1–5.

[4] A. Jain and J. Pasquale, "Internet distance prediction using node-pair geography," in *Network Computing and Applications (NCA), 2012 11th IEEE International Symposium on*. IEEE, 2012, pp. 71–78.

[5] T. Wolf, J. Griffioen, K. L. Calvert, R. Dutta, G. N. Rouskas, I. Baldine, and A. Nagurney, "Choice as a principle in network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 105–106, 2012.

[6] G. N. Rouskas, I. Baldine, K. Calvert, R. Dutta, J. Griffioen, A. Nagurney, and T. Wolf, "Choicenet: Network innovation through choice," in *Optical Network Design and Modeling (ONDM), 2013 17th International Conference on*. IEEE, 2013, pp. 1–6.

[7] H. Yin, Y. Jiang, C. Lin, Y. Luo, and Y. Liu, "Big data: transforming the design philosophy of future internet," *IEEE network*, vol. 28, no. 4, pp. 14–19, 2014.

[8] C. E. Rothenberg, M. R. Nascimento, M. R. Salvador, C. N. A. Corrêa, S. Cunha de Lucena, and R. Raszuk, "Revisiting routing control platforms with the eyes and muscles of software-defined networking." ACM, 2012, pp. 13–18.

[9] A. Gupta, M. Shahbaz, L. Vanbever, H. Kim, R. Clark, N. Feamster, J. Rexford, and S. Shenker, "Sdx: A software defined internet exchange," 2013.

[10] D. Saucez, B. Donnet, L. Iannone, and O. Bonaventure, "Interdomain traffic engineering in a locator/identifier separation context," in *Internet Network Management Workshop, 2008. INM 2008. IEEE*, Oct 2008, pp. 1–6.

[11] W. Xu and J. Rexford, "Miro: Multi-path interdomain routing," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 171–182, Aug. 2006.

[12] J. He and J. Rexford, "Toward internet-wide multipath routing," *Network, IEEE*, vol. 22, no. 2, pp. 16–21, 2008.

[13] Z. Shu, J. Wan, J. Lin, S. Wang, D. Li, S. Rho, and C. Yang, "Traffic engineering in software-defined networking: Measurement and management," *IEEE Access*, vol. 4, pp. 3246–3256, 2016.

[14] J. Yan, H. Zhang, Q. Shuai, B. Liu, and X. Guo, "Hiqos: An sdn-based multipath qos solution," *China Communications*, vol. 12, no. 5, pp. 123–133, 2015.

[15] W. Wang, W. He, and J. Su, "M2sdn: Achieving multipath and multi-homing in data centers with software defined networking," in *2015 IEEE 23rd International Symposium on Quality of Service (IWQoS)*. IEEE, 2015, pp. 11–20.

[16] S. Izumi, A. Edo, T. Abe, and T. Suganuma, "An adaptive multipath routing scheme based on sdn for disaster-resistant storage systems," in *2015 10th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA)*, Nov 2015, pp. 478–483.

[17] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a globally-deployed software defined wan," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, Aug. 2013.

[18] X. Xu, L. Zuo, and Z. Huang, "Reinforcement learning algorithms with function approximation: Recent advances and applications," *Information Sciences*, vol. 261, pp. 1–31, 2014.

[19] UbuntuNetAlliance, "State of the art of the research networking infrastructure - eastern and southern africa," in *IST-Africa Conference Proceedings, 2016*. IEEE, 2016.

[20] B. Heller, N. Handigol, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible network experiments using container based emulation," in *Proc. ACM CoNEXT*, Dec. 2012.

[21] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs, "Iperf: The tcp/udp bandwidth measurement tool," *htt p://dast. nlanr. net/Projects*, 2005.

[22] L. Quan and J. Heidemann, "On the characteristics and reasons of long-lived internet flows," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010, pp. 444–450.