2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems

Optimizing Secure SDN-enabled Inter-Data Centre Overlay Networks through Cognitive Routing

Frederic Francois and Erol Gelenbe
Intelligent Systems and Networks
Department of Electrical and Electronic Engineering
Imperial College, London SW7 2BT, UK
Email: {f.francois and e.gelenbe}@imperial.ac.uk

***Abstract*—More and more businesses are deploying their application(s) with different cloud providers which are close to their customers in order to provide better Quality of Service (QoS) to their end customers. In this work, an optimized and secure software-defined overlay network is proposed as an efficient mechanism to interconnect these geographically-dispersed applications compared to using only plain IP routing. A logically centralized *Cognitive Routing Engine* (CRE), based on Random Neural Networks with Reinforcement Learning, was developed to find with minimal monitoring overhead the optimal overlay paths when the public Internet is used as the communication means between the overlay nodes. CRE was evaluated by using an overlay network composed of hosts from 5 different public clouds where it was shown that the latency of CRE paths is most of the time within 5% of the latency of the optimal IP paths. Furthermore, it was also demonstrated that CRE is able to do asymmetric path optimization where the forward path is different from the reverse path for a given data centre pair in order to further improve QoS.**

I. Introduction

Nowadays, many cloud tenants are looking at using a combination of different public and private cloud infrastructure providers in order to host their distributed applications. The motivations behind this choice are that a single cloud provider is often unable to provide the required global footprint in order to provide the best Quality of Service (QoS) to the end-users of the cloud tenants [1] and also protection against natural disasters and infrastructure failures. The amount of cloud resources required may also vary with time and therefore dynamic scaling is required to optimize resources. Due to all these aforementioned reasons, there is a need to create flexible network overlays to interconnect the different locations where the applications are running. A network overlay is a virtual network which is deployed on top of one or more physical networks where the network overlay provider has full control over the virtual network and therefore, a network overlay is ideally suited to provide the illusion to distributed applications that they are operating on a single network.

In order for network overlays to be truly beneficial, they need to provide a minimum Quality of Service (QoS) and security to applications running on them. While the security of network overlays can be partly achieved through the use of encrypted links over the physical networks, it is much more challenging to provide a minimum QoS since the network overlays may be running on public networks with either minimal or non-existent Service Level Agreement (SLA). Fortunately, Software Defined Networks (SDNs) constructs can improve the QoS performance of overlay networks through traffic engineering that is customized for the QoS requirements of each application running on the overlay. Specifically, SDN allows an overlay flow to take the best path in the overlay by continuously monitoring the state of the overlay and installing the appropriate forwarding rules in the overlay gateways.

The two main objectives of this paper are to develop mechanisms to secure both the control and data plane of the overlay network. In effect, it is not desired that external parties are able to understand the different traffic that are being sent between the overlay gateways located in different data centres. In addition, the overlay gateways and their controller(s) should only accept commands and/or messages from each other. The second objective is to develop a traffic engineering scheme which is capable of providing the necessary QoS needed by the different applications running on the overlay network to function properly. Extensive experiments on the public Internet with major data centre providers will be used to verify the correct behavior of the whole architecture.

II. Related Work

While many prominent cloud providers operate and optimize their own private Wide Area Network (WAN) which interconnects their data centres [2] so as to have more control over the QoS experienced compared to using the public Internet, it is sometimes necessary to use different cloud providers in order to minimize costs as well as to improve the QoS experienced by the end users of the merged/hybrid cloud since different public clouds have different set of locations for their data centres [3], [4]. Since as far as we know there is no dedicated WAN between the data centres of different cloud providers, it is necessary to use the public Internet to interconnect the different data centres. Moreover, a public cloud tenant can have its own private data centre and use the public cloud to have elastic compute and storage capacity as well as reduced latency to its customers [5]. Hence, it is highly likely that the tenant will have to connect his private cloud to the public cloud through the public Internet so as to reduce provisioning time and costs. One of the objectives of this work

2375-0227/16 $31.00 © 2016 IEEE
DOI 10.1109/MASCOTS.2016.26

is to optimize the public Internet paths between these different data centres in order to achieve better QoS.

One of the first overlay routing scheme deployed in WAN is *Resilient Overlay Network* (RON) [6] which showed that QoS-aware overlay routing can have higher QoS performance compared to using underlay routing directly. RON probes all the links in the overlay network in order to find the best paths but this is known to be sub-optimal in terms of monitoring since in [7] they show that *constant* global monitoring is not necessary to obtain near-optimal paths. [7] is also the closest related work since the authors also use cognitive routing but they implement it in a distributed manner where each overlay node is doing its own measurements of the overlay links and paths. In this work, the cognitive routing is run in a *logically* centralized controller and the network measurements are pooled into a single database which is accessible by all cognitive routing instances, which are created to cater for different types of traffic (including different source and destination). Furthermore, a cognitive routing instance can update the network representation inside the controller once it has allocated a path so that other cognitive routing instances know about the decision *immediately*. This avoid oscillations in routing since a cognitive routing instance could make decisions using stale network measurements if cognitive routing instances do not share their decisions.

III. Overall Overlay Architecture

In this work, it is assumed that a cloud tenant/customer has deployed its applications in a combination of private and/or public clouds operated by different cloud providers with the data centres being geographically dispersed. In order to transfer data between different cloud providers, it is often required to use the public Internet since in most cases cloud providers have a dedicated Wide Area Network (WAN) only between their own data centres and also to key Internet Exchange Points (IXP). Fig. 1 shows how the different data centres, where the cloud tenant has deployed its applications, are connected together through a SDN overlay network which is composed of an overlay gateway in each data centre. An overlay gateway is connected to other overlay gateways through encrypted Layer-2 Virtual Private Network (VPN) point-to-point links which allow the communication between the data centres to be secure. The topology of the overlay can be either full-mesh or more sparse depending on the requirements of the tenant's applications. The tenant's applications in each data centre use the overlay gateway in order to communicate with other applications located in other data centres. Each overlay gateway is made up of a SDN OpenFlow switch which is controlled by a logically centralized controller. The controller is responsible for choosing which overlay path a traffic flow should take in order to maximize the objective function set by the tenant's traffic engineering policy.

IV. Secure Control and Data Plane

Since the public Internet is used to communicate both control and data plane traffic between the different data centres,

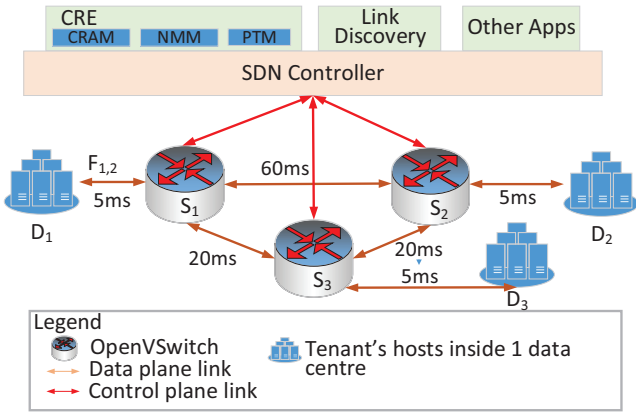

Fig. 1: Network architecture showing where the Cognitive Routing Engine is located and how it interacts with other components of the architecture.

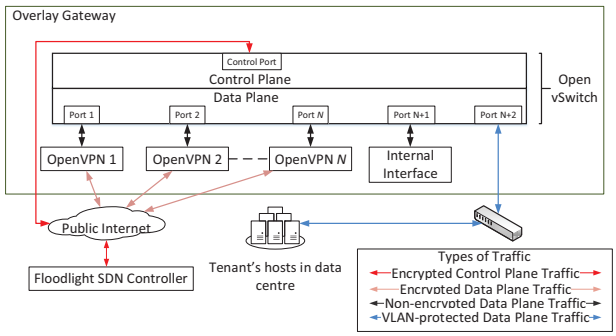

Fig. 2: The different components of an overlay gateway.

there is a requirement to encrypt the overlay links which form part of the data plane of the network and also the control plane communication between the SDN OpenFlow switches and the logically centralized SDN controller. For the data plane, we use the open source software *OpenVPN* [8] to provide the encrypted Layer-2 point-to-point links between the overlay gateways situated in different data centres. *OpenVPN* works in a server-client paradigm where for one particular overlay link, one gateway will act as the server while the other as the client. When an *OpenVPN* instance is active, it creates a virtual network interface which can be connected to the *OpenFlow* switch. For the control plane of the overlay, we make use of the open source software switch *Open vSwitch* [9] in *OpenFlow* [10] mode along with the open source *OpenFlow* controller *Floodlight* [11] which runs our own traffic engineering scheme. In-depth details are given about our traffic engineering scheme in the next section. Fig. 2 shows how the SDN switch, controller and *OpenVPN* instances act together to form an effective overlay gateway which can be connected with other gateways in order to provide secure and optimized path(s) between the tenant's applications which are located in different data centres.

A. Algorithm to Generate Overlay Configuration Files

In order to generate the necessary cryptographic keys required to authenticate and encrypt the control and data plane,

Algorithm 1: Generate PKI and *OpenVPN* Configuration Files

```
Let N be the set of {gateway_name, public_ip} pairs ;
Let L be the set of bidirectional links with a link being
  {head_gateway_name, end_ gateway_name};
Create 1 Switch and 1 Controller Certificate Authority (CA);
for gateway n in N do
    Generate cryptographic key pair for n;
    Use Switch CA to sign certificate for n;
end
for each controller c do
    Generate cryptographic key pair for c;
    Use Controller CA to signed certificate for c;
end
for gateway n in N do
    for link l in L do
        if n is a head gateway in l then
            Generate 1024-bit Diffie-Hellman key for n;
            Generate a OpenVPN server configuration file for n;
            Generate a OpenVPN server configuration file for end
            gateway of l;
            Remove l from L;
        end
    end
    Generate script to first set-up all OpenVPN instances for n and
    set-up Open vSwitch on n to manage the virtual interfaces created
    and connect to controller;
end
```

we deploy a Public Key Infrastructure (PKI) on a secure machine. Since deploying and managing a PKI for a large number of Overlay Gateways may be challenging, we designed an algorithm which will take as input the target topology of the overlay network together with the public IP address of the overlay gateways and output the necessary cryptographic keys and *OpenVPN* and *Open vSwitch* configuration files/script for each individual overlay gateway.

The algorithm in Algo. 1 first start by generating 1 Certificate Authority (CA) for the switches and another the controller(s). Then for each switch in the overlay network, a cryptographic key pair is generated along with a certificate which is signed by the switch CA. The same is done for the controller(s) but the controller CA is used to sign the certificate for the controller(s). In the second stage, the algorithm creates a customized pair of *OpenVPN* server and client configuration files—based on a server and client configuration template respectively—for each overlay link. Since each overlay link needs its own *OpenVPN* server and an overlay gateway may more than one overlay link, it is necessary for each *OpenVPN* server instances on a single gateway to communicate on different network ports and the associated *OpenVPN* clients must be configured accordingly to know which port they need to connect to. In addition, each *OpenVPN* server needs a different Diffie-Hellman key for added security.

B. Deploying Overlay

The last step of the Algo. 1 is to create an individual script for each gateway server which upon running will first initiate all *OpenVPN* clients and/or server instances and then create the *Open vSwitch* instance which will have be configured to use the overlay SDN controller. The switch is then connected to all the interfaces created by the *OpenVPN* instances and also to the network interface through which the tenant's hosts inside the data centre can be reached.

V. Control Plane

Our traffic engineering solution is implemented in the form of a module application called *Cognitive Routing Engine* (CRE) in the SDN controller. Fig. 1 shows where CRE is located in the overall control plane architecture and how it interacts with the various other components of that architecture. In Fig. 1, the *OpenVSwitches* (OVSs) are SDN-enabled packet switches and use OpenFlow v1.3 [10] to communicate with the SDN controller. The *SDN Controller*, a.k.a Network Operating System, is responsible for sending and receiving OF messages from the OVSs. In addition, the SDN controller typically orchestrates the different applications running on it and act as an OF protocol translator for the applications. The *applications* are pieces of software running on top of the controller which provides specialized network functions such as routing. In this particular instance, there are 2 applications of particular interest: *Link Discovery* and *CRE*. The Link Discovery application discovers the data plane topology of the network while the CRE application efficiently finds and installs new suitable paths in the network as requested by other SDN applications. CRE is made of 3 main modules [12]: the *Cognitive Routing Algorithm Module* (CRAM) implements the Cognitive Packet Network (CPN) algorithm using Random Neural Networks (RNNs) with Reinforcement Learning (RL) [13] to find network paths which maximize a customizable objective function and therefore, meet the QoS requirements of host applications, the *Network Monitoring Module* (NMM) efficiently either uses past network measurements and/or probes and/or poll the network to get the necessary network state information to update the RNNs in the CRAM, and the *Path-to-OF Translator Module* (PTM) is able to convert the paths found by the CRAM into the appropriate set of OF messages so that paths are either created or updated with minimum inconsistency in the network.

VI. Cognitive Routing

CRAM is a routing algorithm which uses CPN based on the Random Neural Network (RNN) [13], [14] with Reinforcement Learning (RL) [15]–[17]. The approach is inspired from the idea that search in a random environment can be successful even when the searcher does not initially know which direction to pursue [18] even in highly chaotic environments [19].

For a given flow, a new RNN is created for each OVS along the path initially found through shortest path routing based on hop count. A RNN is constructed, where each neuron represents one active port of the OVS. The decision on which port to use as the next hop can be done in either an exploratory or an exploitation way. When CPN is in exploratory mode, it chooses the output port randomly, but when CPN is in exploitation mode, it chooses the neuron which has the highest probability of being excited as described in the next paragraph. Exploratory mode is chosen with a probability 5% or 10% in many implementations of CPN [20], [21], with the exploitation

mode chosen otherwise. If the next hop OVS does not have a RNN for this particular flow, a new RNN is created for it.

In the RNN each neuron's potential is a non-negative integer, and we denote by q_i the probability that the i^{th} neuron's potential is positive, or equivalently that the neuron is excited and can "fire". Thus the neuron with the highest q_i is selected as the output when the RNN is in exploitation mode and the most excited neuron determines the next hop OVS of the path. The $q_i,\ 1 \leq i \leq |P|$ where P is the set of neurons of the RNN, are calculated as:

$$q_i = \frac{\lambda_i^+}{r_i + \lambda_i^-} \tag{1}$$

where λ_i^+ and λ_i^- are respectively the total arrival rates to neuron i of excitatory or inhibitory spikes:

$$\lambda_i^+ = \sum_{j \in P} q_j w_{j,i}^+ + \Lambda_i^+,\ \lambda_i^- = \sum_{j \in P} q_j w_{j,i}^- + \Lambda_i^-,\ \text{where } j \neq i. \tag{2}$$

where $w_{j,i}^+$ and $w_{j,i}^-$ are the excitatory and inhibitory weights from neuron j to neuron i, and the total firing rate of neuron i is given by $r_i = \sum_{j \in P} [w_{i,j}^+ + w_{i,j}^-],\ j \neq i$.

In the CPN algorithm, the quantities $w_{i,j}^+$ and $w_{i,j}^-$ are determined by using reinforcement learning. The value of the objective function, such as the delay O_{sd}^t of a path with source s and destination d is measured at successive times t and reward to be maximized is denoted $R_{sd}^t = (O_{sd}^t)^{-1}$. Then γ_{sd}^t the historical value of the reward at time t is updated

$$\gamma_{sd}^t = \beta \gamma_{sd}^{t-1} + (1-\beta) R_{sd}^t, \tag{3}$$

where $0 \leq \beta \leq 1$ is an exponential averaging parameter.

If $R_{sd}[t] \geq \gamma_{sd}^t$ then the previous decision to select output i is considered valid and the corresponding weights are updated:

$$w_{j,i}^{+|t} = w_{j,i}^{+|t-1} + R_{sd}^t,\ j \neq i,\ w_{j,k}^{-|t} = w_{j,k}^{-|t-1} + \frac{R_{sd}^t}{|P|-2}, \tag{4}$$

for $k \in P$ and $k \neq i, j$, while all the other $w_{i,l}^+,\ w_{i,l}^-$, remain unchanged for $l \neq i$. On the other hand, if $R_{sd}[t] \leq \gamma_{sd}[t]$, i.e. the reward of the new path is less than the threshold and therefore the RNNs have made the wrong decision, the weights are updated to reflect that other paths should be tried:

$$w_{j,k}^{+|t} = w_{j,k}^{+|t-1} + \frac{R_{sd}^t}{|P|-2},\ k \neq i,\ w_{j,i}^{-|t} = w_{j,i}^{-|t-1} + R_{sd}^t, \tag{5}$$

for $j \neq i$. In order to prevent the link weights from increasing indefinitely, they are re-normalized at each step t by first calculating the new r_i^* by using the updated W values and then updating the link weights as follows: $W = W \frac{r_i}{r_i^*}$. CRAM also keeps a record of the best Z recent paths found for each path request and the best path out of the Z paths is used for the actual routing of the traffic flow if the existing path has been active for longer than a minimum time period to avoid frequent path fluctuations.

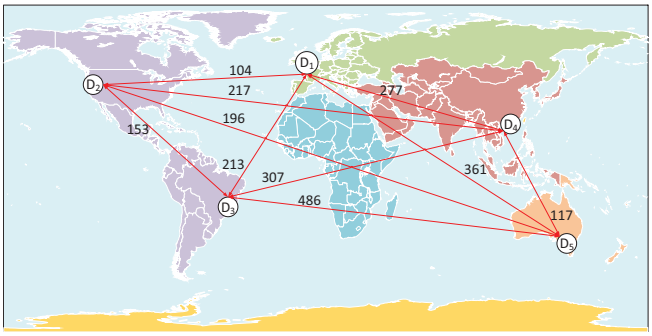

Fig. 3: Approximate location of the 5 data centres selected for the overlay network with the round-trip time in ms between each data centre pair.

TABLE I: Optimal Latency Paths for Non-optimal IP Paths

Source Overlay, X	Destination Overlay, Y	Optimal Overlay Path, $G_{X \to Y}^{\text{opt\_ip}}$	Decrease in 1-way latency (%)
1	5	$D_1 \to D_2 \to D_5$	16.8
3	5	$D_3 \to D_2 \to D_5$	38.2
5	1	$D_5 \to D_2 \to D_1$	16.8
5	3	$D_5 \to D_2 \to D_3$	38.2

VII. Evaluation

A. Topology and Nomenclature

The secure overlay network was deployed in 5 geographically-dispersed data centres which are each owned by a *different* prominent cloud provider. Fig. 3 shows the approximate location of the 5 data centres used: D_1 is the South West of United Kingdom, D_2 is in the central West of USA, D_3 is in Sao Paulo, Brazil, D_4 is in Hong Kong and D_5 is Melbourne, Australia. A path between D_X and D_Y may be either unidirectional, in which case we use $G_{X \to Y}$ or bidirectional and use $G_{X \leftrightarrow Y}$. CRE works on a unidirectional basis. It is not necessarily true that the default unidirectional IP path $G_{X \to Y}$ from a data centre D_X to another data centre D_Y is the same as the default unidirectional IP path $G_{Y \to X}$ from D_Y to D_X. This is a well-known fact [22]. Therefore, taking the Round Trip Time (RTT) divided by 2 does not always give the correct one-way latency between two gateways. That's why the term "optimal IP" is used throughout the evaluation section rather than simply "optimal". All latencies/RTTs in this paper are data plane latencies/RTTs.

B. Path Optimization using Shortest Path First

After the overlay network was created, the RTT in ms between the overlay gateways using only the default IP routing was measured by pinging between the different overlay gateways and the results are shown on the links in Fig. 3. These results are only a snapshot and can vary over time.

If it is assumed that the latency of the unidirectional IP path between any 2 overlay gateways is half the RTT between them and a Shortest Path Algorithm (SPF) is ran with weight equal to half the RTT of the paths, it was found that default IP routing does not give the best latency for 4 out of the 20 unidirectional paths, i.e. 20% of IP paths are non-optimal in terms of latency. This result is expected since it

is well-known that BGP is non-optimal in terms of QoS only since peering policies depend also on the peering agreements between different networks and these agreements are partially driven by commercial interests [23]. The 4 non-optimal IP paths based on latency is given in Table I along with the optimal path through the overlay. It can be observed that data centre providers: D_3 and D_5 do not have optimal peering policy in terms of latency and that data centre provider D_2 has excellent peering policies since all two aforementioned data centre providers can use it as an intermediate node and reduce their latencies to other data centres. Another observation is the latency reductions are significant.

C. CRE vs default IP vs optimal IP: 1^{st} hour

In this experiment, the RTT of the CRE and default IP path between any 2 overlay gateway was measured every 5 minutes by averaging 10 pings at each measurement instance. The experiment was carried for 24 continuous hours but results for the 1^{st} hour are analyzed here and the whole 24 hours are analyzed in the next subsection. Plot (a), (b), (c) and (d) of Fig. 4 show the RTT performance of the default IP, CRE and optimal IP paths for the path $G_{1\leftrightarrow5}$ and $G_{3\leftrightarrow5}$. It can be seen that the CRE RTTs rapidly converge to the optimal IP RTT for both paths and that the RTT optimality gap between the CRE and optimal IP path reduces to less than 5%. The optimality gap may not be 0 even if CRE uses the same paths as for optimal IP because of 2 main potential reasons: (i) CRE paths uses encrypted links so all communication between the overlay gateways need to be both encrypted and decrypted which cause additional packet delays and (ii) ping measurements are not perfectly accurate due to varying operating system delays.

Plot (e) and (f) of Fig. 4 show the RTT performance of the default IP, CRE and optimal IP paths for the path $G_{3\leftrightarrow5}$. The plots shows an *unexpected* event since CRE it converges away from the default IP and optimal IP path to give a smaller RTT of around $230ms$ compared to around $260ms$ for optimal and default IP as shown in the plot (e) of Fig. 4. As a result, the optimality gap between the CRE and optimal IP RTTs becomes largely negative at around -10%. When the logs were checked, it was seen that rather than using the default IP path of $G^{\text{def\_ip}}_{1\rightarrow3} = D_1 \rightarrow D_3$, CRE was using $G^{\text{CRE}}_{1\rightarrow3} = D_1 \rightarrow D_2 \rightarrow D_3$ and keeping the reverse path to $G^{\text{CRE}}_{3\rightarrow1} = D_3 \rightarrow D_1$. Hence, the better bidirectional path between D_1 and D_3 during this period of time is asymmetric as commonly observed in the public Internet [22].

D. Analysis of Whole 24 Hours

Here, the whole 24 hours of the experiment is analyzed, i.e. 289 measurements for each overlay pairs. Fig. 5 (a) shows the cumulative distribution function plot for the RTT optimality gap between the CRE and optimal IP paths over 24 continuous hours with a 5-minute measurement interval. It can be observed that for most paths, the CRE paths gives a RTT that is within $\pm5\%$ of the optimal IP RTT. A negative optimality gap within -5% can be obtained due to slightly inaccurate ping measurements. For paths $G_{1\leftrightarrow5}$ and $G_{3\leftrightarrow5}$, the optimality gap is very high at first because CRE uses the shortest path first based on *hop count* for the first path and this is equivalent to doing default IP routing but CRE converges quickly to the optimal IP path and successfully find the optimal path throughout the experiment. The most unexpected result from Fig. 5 (a) is the RTT optimality gap performance for paths $G_{1\leftrightarrow3}$ is at around -10% most of the time, this is due to asymmetric routing as mentioned previously. These impressive results are due to CRE ability to tract the true optimal.

In the 24-hour experiment, there were 108 unidirectional paths inserted by CRE; out-of-which 20 unidirectional *initial* paths were initially inserted in order to route the traffic between any 2 gateways in the overlay network. Hence, 88 unidirectional paths were changed and Fig. 5 (b) shows which paths change the most. Unidirectional paths $G^{\text{CRE}}_{3\rightarrow4}$ was the most unstable with the path fluctuating back and forth between $D_3 \rightarrow D_4$ and $D_3 \rightarrow D_2 \rightarrow D_4$. One observation is that most of the paths in Fig. 5 (b) either originates or terminates at D_3. Fig. 5 (c) shows when the path changes happen. It can be observed that most path changes happen between 9 to 18 hours into the experiment. Since there are several potential reasons why this is the case, it is difficult to know the exact reasons why traffic to/from D_3 is behaving this way without further knowledge about the operations of D_3.

VIII. Conclusions

In this work, it was demonstrated that it is possible to deploy a secure overlay network over the public Internet in order to interconnect geographically-dispersed data centres and still provide better round-trip time between different data centres compared to using the plain public Internet. This is thanks to the *Cognitive Routing Engine* which is able to efficiently monitor and find near-optimal QoS paths.

References

[1] P. Mulinka and L. Kencl, "Learning from cloud latency measurements," in *2015 IEEE Int. Conf. on Commun. Workshop (ICCW)*, June 2015, pp. 1895–1901.

[2] 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.

[3] Google's data center locations. Accessed: 20 May 2016. [Online]. Available: https://www.google.com/about/datacenters/inside/locations/index.html

[4] Aws global infrastructure. Accessed: 20 May 2016. [Online]. Available: https://aws.amazon.com/about-aws/global-infrastructure/

[5] Public cloud vs. private cloud vs. hybrid cloud. Accessed: 20 May 2016. [Online]. Available: http://www.intel.co.uk/content/www/uk/en/cloud-computing/cloud-101-video.html

[6] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," *SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, pp. 131–145, Oct. 2001.

[7] O. Brun, L. Wang, and E. Gelenbe, "Big data for autonomic intercontinental overlays," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 575–583, March 2016.

[8] OpenVPN. [Online]. Available: https://openvpn.net/

[9] Open vSwitch. [Online]. Available: http://openvswitch.org/

[10] (2015, March) Openflow switch specification, version 1.3.5. Open Networking Foundation. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.3.5.pdf

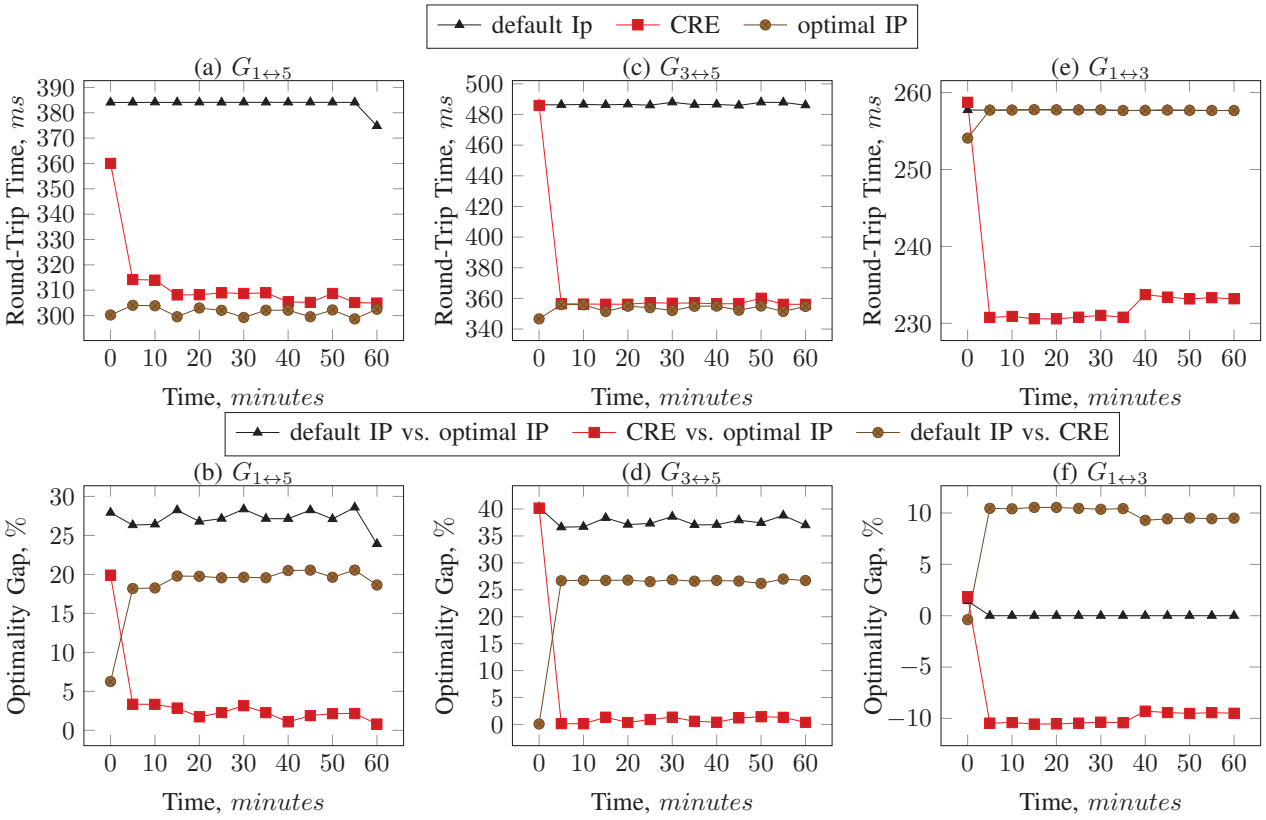

Fig. 4: Comparison of Round Trip Time when: *default IP*, *CRE* and *optimal IP* path are used.

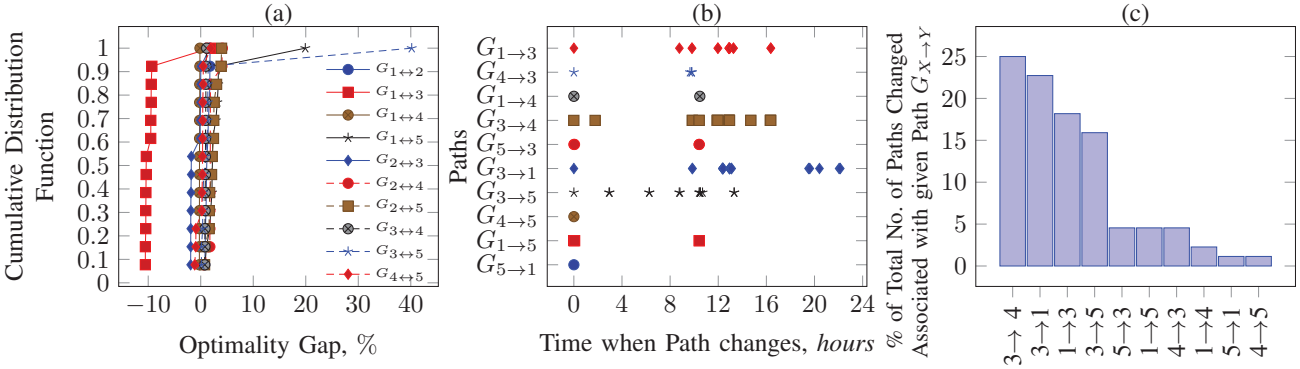

Fig. 5: Plot (a) shows the cumulative distribution function vs. the optimality gap between CRE and optimal IP paths, Plot (b) shows when the unidirectional CRE paths ($G_{X\to Y}$) changed between datacentre D_X and D_Y, Plot (c) shows which CRE paths changed the most over the 24-hour.

[11] Floodlight. [Online]. Available: http://www.projectfloodlight.org

[12] F. Francois and E. Gelenbe, "Towards a cognitive routing engine for software defined networks," in *Proc. of IEEE Int. Conf. on Commun.*, May 2016.

[13] E. Gelenbe, "Learning in the recurrent random neural network," *Neural Computation*, vol. 5, no. 1, pp. 154–164, 1993.

[14] ——, "The first decade of g-networks," *EUROPEAN JOURNAL OF OPERATIONAL RESEARCH*, vol. 126, no. 2, pp. 231–232, 2000.

[15] E. Gelenbe, Z. Xu, and E. Seref, "Cognitive packet networks," in *Tools with Artificial Intelligence, 1999. Proceedings. 11th IEEE International Conference on*, 1999, pp. 47–54.

[16] E. Gelenbe, R. Lent, and Z. Xu, "Design and performance of cognitive packet networks," *Perform. Eval.*, vol. 46, no. 2-3, pp. 155–176, Oct. 2001.

[17] E. Gelenbe, "Steps toward self-aware networks," *Commun. ACM*, vol. 52, no. 7, pp. 66–75, 2009.

[18] ——, "Search in unknown random environments," *Physical Review E*, vol. 82, no. 6, p. 061112, 2010.

[19] E. Gelenbe and F.-J. Wu, "Large scale simulation for human evacuation and rescue," *Computers & Mathematics with Applications*, vol. 64, no. 12, pp. 3869–3880, 2012.

[20] G. Sakellari, "The cognitive packet network: A survey," *The Computer Journal*, vol. 53, no. 3, pp. 268–279, 2010.

[21] E. Gelenbe and Z. Kazhmaganbetova, "Cognitive packet network for bilateral asymmetric connections," *IEEE Trans. on Industrial Informatics*, vol. 10, no. 3, pp. 1717–1725, 2014.

[22] A. Pathak, H. Pucha, Y. Zhang, Y. C. Hu, and Z. M. Mao, "A measurement study of internet delay asymmetry," in *Proc. of 9th Int. Conf. on Passive and Active Network Measurement*, 2008, pp. 182–191.

[23] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe, "The case for separating routing from routers," in *Proc. of the ACM SIGCOMM Workshop on Future Directions in Network Architecture*, 2004, pp. 5–12.