Using Overlay Cloud Network to Accelerate Global Communications

Zhaogui Xu, Ran Ju, Liang Gu, Weiguang Wang, Jin Li
Feng Li and Lei Han
Huawei Technologies
{*zhaogui.xu, juran, guliang3, weiguang.wang, mark.lijin*}*@huawei.com*
{*frank.lifeng, phoebe.han*}*@huawei.com*

***Abstract*—The global communication puts a strong demand on the "one global network". However, the public Internet faces great challenges to achieve this goal due to its inherent mechanisms. In this paper, we propose the global Overlay Cloud Network (OCN), a generic high-quality network built on the global public clouds, to accelerate global communications. We implement the OCN by three key techniques. First, we develop an automated topology planning technique to select the best candidate cloud nodes. Second, we design a novel time-varying routing technique to compute the best routes of OCN. Third, we implement a new dynamic transport layer to provide high-quality data transmission through OCN. We evaluate our prototype of OCN on real public cloud platforms. The results show that OCN substantially outperforms the existing underlay network with the RTT latency decreased by 15.2%, the packet loss rate reduced by 1.9% and throughput improved by 10X.**

I. Introduction

The trend of economic globalization and international trade liberation puts forward many new demands on the globalization of communications. Lots of applications, such as multinational corporate communications, live streaming, online games, and shopping, etc., have a strong demand for the "one global network", a network with the ability to provide high-quality end-to-end services globally. However, under the new demands of global communications, the existing public Internet cannot provide such high-quality services due to the following challenges. First, the public Internet is made up of thousands of sub-area networks. These sub-area networks are supported by different network providers with different communication capabilities. It is extremely challenging, if possible, to federate them to generate a high-quality global network. Second, the Internet protocol only offers an indiscriminate best-effort service without any guarantee of quality. Although this inherent mechanism is simple and effective in local areas, it faces great challenges to provide high-quality services at worldwide. Third, expensive dedicated lines with reserved resources have to be constructed manually to provide an SLA for different classes of business requirements. The construction procedure often takes a long time, which involves the cooperations of multiple departments and regions.

In fact, a lot of efforts have been made to improve the Internet capability and reliability, including dedicated networks, software-designed networks (SDN) [8], content-delivery networks (CDN) [11] and so on. While effective, each of these existing solutions has limitations. Deploying dedicated networks often need long-term construction with high costs and commercial risks. Route planning via SDNs mostly requires new hardware support and large scale changes of existing network devices. Caching or replication through CDNs is expensive and commonly limited to high-end resources.

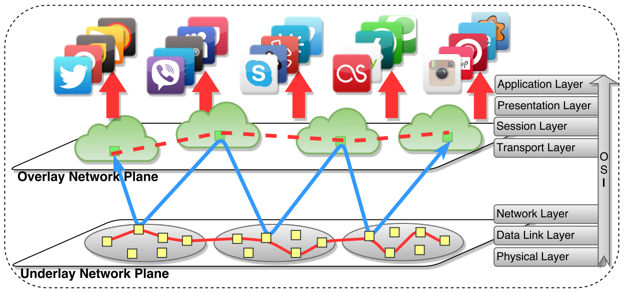


Figure 1: The network architecture.

In this paper, we propose the global Overlay Cloud Network (OCN), which is defined as a generic high-quality network built on the global public clouds. Figure 1 presents our new network architecture. We unify the existing Internet (also called the underlay network) to provide the fundamentals of data transmission (layer 1-3 of OSI) only, and move the complicated transport protocols and services to the OCN (layer 4-7 of OSI), which is called the second plane of the Internet. We consider the OCN has the ability to construct the second plane of the Internet because of the following highlights. First, a wealth of overlay cloud nodes can be chosen from various cloud vendors, and the APIs provided by cloud vendors enable us to automatically create/destroy these cloud nodes in seconds. That means OCN can be a kind of demand-driven network whose capacity can be rapidly expanded/reduced automatically. Second, the overlay network nodes are powerful and programmable. That means OCN enables us to implement a set of complicated mechanisms that traditional network devices do not, such as network state caching, time-varying routing, hop-by-hop retransmission, etc., and provide customized and differentiated services for different users and business. Third, overlay cloud nodes have strong reachability because of the cloud backbone network. That means a bunch of routing paths can be selected to avoid congestion. Combining with a set of other transport techniques, OCN can shield the differences of underlay networks and achieve global, timely and secure arrival of messages.

We implement the prototype of OCN by three key modules,

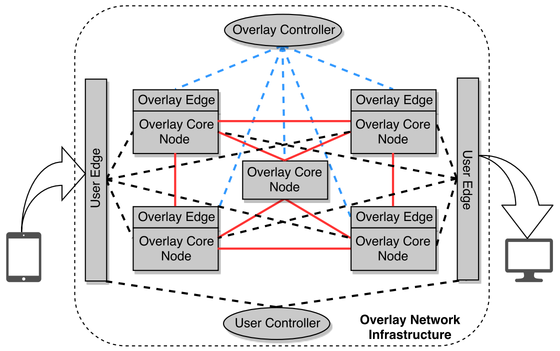


Figure 2: The architecture of OCN.

namely, topology planning, time-varying routing and dynamic transport. For the topology planning, we first automatically detect the available cloud nodes and then measure their network conditions, and finally make a determination if adding the detected nodes into the OCN. For the time-varying routing, we first monitor the network conditions of each overlay node to discover the time-varying rules and then make predictions for their near future network conditions, and finally compute the best route paths to bypass those nodes with bad network conditions. For dynamic transport, we implement a set of new techniques, including hop-by-hop retransmission when datagrams get lost, redundant transmission and detouring before the link condition becomes worse, and automated expansion when the resources are running out.

We deployed the OCN on live cloud platforms with 229 candidate cloud nodes all over the world from various vendors, and compared the performance of OCN with existing public Internet by real data transmissions. The statistics on 3.5 billion datagram transmissions shows that OCN significantly outperforms the public Internet with The RTT latency and packet loss rate improved by 15.2% and 1.9%, respectively. We set a typical aim of 200ms delay and 1% packet loss of the long distance communication. The results show that the satisfaction rate is substantially improved from 46.61% to 73.59%. We also choose several cross-border links to evaluate the practical performance of OCN via downloading and video streaming. The results show that the average downloading throughput and video resolution of OCN reaches 33Mbps and 2160p, while the public Internet only has 2.5Mbps and 480p, respectively.

The rest content of this paper is structured as the following. Session 2 discusses the network architecture of OCN. Session 3 illustrates the detailed design, including topology planning, time-varying routing and dynamic transport. Session 4 presents the evaluation settings and results. Session 5 compares OCN with existing related work. The last session gives a brief conclusion.

II. Architecture

Figure 2 presents the network architecture of OCN, which consists of five kinds of nodes, namely, the overlay network controller, overlay network core nodes, overlay edge nodes, the user controller and user edge nodes.

The overlay controller. The overlay controller is the brain of the entire overlay network, which mainly has three function points. First, it maintains the overlay network topology and senses the changes in the network topology timely. Second, it detects the time-varying rules of the network conditions according to the measurement data reported by the overlay core nodes and makes predictions of the network conditions for the near future. Third, it calculates the best route for each overlay core node according to the predictions and then delivers the route to each overlay core node.

The overlay core nodes. The overlay core nodes also have three function points. First, each core node will continuously measure the network conditions with other core nodes, and report the measurement data to the overlay controller. Second, each core node will be responsible for forwarding the datagrams and ensuring their reliability. Third, when the network is becoming bad, it will make a decision to detour.

The overlay edge nodes. The overlay edge nodes are the bridges between the overlay core nodes and user edge nodes. Each overlay edge node will be responsible for finding the best route in the overlay network according to the user service packets and features introduced by the user edge nodes and forward the overlay packets to the destination user edge nodes.

The user edge nodes. The user edge nodes are a kind of logic nodes, residing as applications/processes in the user devices. A user edge node has three kinds of function points. First, it introduces the user's data traffic into the overlay network and eventually pulls it out of the overlay network. Second, it senses the service features (e.g., RTT, packet loss and throughput sensitivity) from the user device and selects an appropriate overlay edge node to land the overlay network. Third, it caches the user data packets and prepares to resend the data when some exceptions happened in the overlay network.

The user controller. The user controller is the control entry for users to get access to OCN. It will decide which overlay edge node should be selected for the specific user edge node.

The high-level work flow of OCN. We now overview the high-level work flow through a sample scenario. Suppose the user A wants to communicate with another user B by a mobile application. The user edged node in A's phone will sense the service feature (e.g., RTT sensitivity) and take over the data traffic. It first asks the user controller to get access to OCN. The user controller will select two best corresponding overlay edge nodes to get on (off) board OCN according to the source (destination) IP. The starting overlay edge node will forward the datagrams to the overlay core nodes (hops) according to its source route. Datagrams will be transmitted hop-by-hop until reaching the ending overlay edge node. During this period, a series of measures (e.g., hop-by-hop retransmission, multi-path redundant transmission, etc.) will be taken to ensure the quality of service. The ending overlay edge node will forward the data to the destination user edge node which will eventually pull out the traffic from OCN to user B's application.

III. Design

In this section, we illustrate the design of OCN, which consists of three parts, namely, topology planning, time-variant routing and dynamic transport.

A. Topology Planning

To ensure the high quality of service, we should carefully choose overlay nodes from a number of global cloud nodes. More specifically, we need to consider the following two points. First, the overlay nodes should satisfy the performance demands, so that the geographical distribution and network conditions of these nodes should be measured carefully. Second, the candidate overlay nodes should be easily maintained with low costs as different cloud platforms have different pricing strategies. Completing topology planning needs a heavy workload, so we automatize the planning procedure. We have three steps to complete the topology planning: (1) detecting candidate cloud nodes; (2) measuring network conditions and (3) selecting the overlay nodes.

To detect candidate cloud nodes, we use automated web crawlers to analyze the web sites of mainstream cloud vendors and query the geographical information of their available public cloud nodes. With the information of these candidates, we then leverage the APIs provided by cloud vendors to automatically create VM instances. We measure their network conditions (e.g., RTT delay, packet loss probability, etc.) with each other by using the ping protocol and automated tools like *Tking* [7]. Finally, we select the overlay nodes from the detected candidates according to their network condition metrics. We model the overlay nodes selection as an 0-1 planning problem by minimizing the transport cost between any geographical region at worldwide. The optimization is formulated as the following.

$$\begin{aligned}\underset{\boldsymbol{X}}{\operatorname{argmin}} \sum_{m\in R}\sum_{n\in R, n\neq m} \Big(& \sum_{c_p\in C_m}\sum_{o_q\in O_m} x_{o_q}\cdot cost_{up}(c_p, o_q) \\ & + \sum_{o_i\in O_m}\sum_{o_j\in O_n} x_{o_i}\cdot x_{o_j}\cdot cost_{ol}(o_i, o_j) \\ & + \sum_{c_{p'}\in C_n}\sum_{o_{q'}\in O_n} x_{o_{q'}}\cdot cost_{down}(c_{p'}, o_{q'})\Big) \\ s.t.: \sum_{o\in O_m} & \geq 1, m\in R, \boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_{|R|}], \\ & x_o \in \boldsymbol{x}_i, i = 1, 2, ..., |R|, x_o = \{0, 1\}\end{aligned} \tag{1}$$

where R represents the region; $cost_{up}(c_p, o_q)$ denotes the traffic cost from the client c_p to overlay node o_q; $cost_{ol}(o_i, o_j)$ denotes the traffic cost between overlay node o_i and o_j; $cost_{down}(c_{p'}, o_{q'})$ denotes the cost from overlay node $o_{q'}$ to client $c_{p'}$. The target is to compute the node vector $\boldsymbol{X}$ by minimizing the overall data traffic costs. In our experimental network, we only consider the latency as the traffic cost for simplicity. We use the world population data to model the demands of clients, which is similar to [12].

For the implementation, the grid search will give us the optimal solution, but its time complexity is extremely high. We hence use a heuristic-based solution, called binary particle swarming optimization (BPSO) [6], instead. Although the solution of BPSO maybe not optimal, its high efficiency makes the overlay node selection much more feasible. Our evaluation in Section IV also demonstrates that the selected overlay nodes are able to improve the transmission performance substantially.

B. Time-varying Routing

The OCN links often have obvious trends and periodic fluctuations due to a set of reasons such as living habits, festivals and etc. As such, we can make forecasts based on these rules and compute the best routes to bypass nodes with bad network conditions. The computation of time-varying routes consists of three steps. We first monitor the network conditions of each OCN link and then analyze the rules to make forecasts. Finally, we compute the best routes of each overlay node to others.

Frequently monitoring the conditions of OCN will take lots of network resources and make the network unscalable. To address this issue, we leverage an existing lightweight measurement called *pingmesh* [3]. We measure a set of network conditions every half an hour, including RTT latency, packet loss probability and bandwidth, and use a machine learning based approach to make predictions of the near future according to the collected data metrics. More specifically, for each OCN link, we use the following weighed function to represent the time-varying rule.

$$f(t) = \sum_{i=1}^{k} w_i f(t - T_i) \tag{2}$$

where T_i denotes the strong periods (e.g., $6h, 12h$ and $24h$) and w_i is the period weight. The whole procedure is a kind of time-series prediction. Let X be the collected network condition data of an arbitrary OCN link and $x_t \in X$ be the data (e.g., RTT) at time step t. We represent the feature vector of x_t as $X_t = [x_{t-T_1}, x_{t-T_2}, ..., x_{t-T_k}]$. Since the prediction is an online procedure intertwined with the route computation, it has a high requirement for efficiency. In our experimental network, we choose a linear regression model and the backend optimization algorithm is SGD which supports incremental training. The training target is to figure out $W = [w_1, w_2, ..., w_k]$ that minimizes $|x_t - \sum_{i=1}^{k} w_i x_{t-T_i}|$.

Once getting the predicted network condition of each OCN link, we then compute the best route for each overlay node. The computation is modeled as a classical shortest path problem and can be solved by many existing algorithms (e.g., Dijkstra, A*). The general shortest path algorithms (SPAs) tend to provide a single path, but in OCN, we expect to obtain multiple paths for redundant links. Therefore, our implementation leverages the MPS algorithm based on SPA. Moreover, when computing multiple paths, we also consider the exclusion of links among these paths for the sake of detouring to bypass congested links (see Section III-C).

C. Dynamic Transport

The data transmission of OCN needs a new dynamic transport layer that enables to provide a high quality of service. In the following, we first illustrate the design of transport protocols and then present our key techniques to improve the transport quality based on the protocols.

Transport protocols. As illustrated in Section II, the overlay edge node will introduce the data traffic onto OCN. During

Outer IP Header	Outer UDP Header	Overlay Header	Inner IP Header	Payload

Overlay Header			
0	1	2	3
version + TTL (8 bits)	header len (16 bits)		msg type (8 bits)
packet number (32 bits)			
timestamp (32 bits)			
flow property (12 bits for latency, 12 bits for bandwidth, 8 bits for packet loss)			
flow id (16 bits)		path id (8 bits)	hop counts (8 bits)
hop list (each hop has 16 bits)			
payload (raw application data + raw IP header)			

Figure 3: The design of the overlay header.

this period, the overlay edge node will encapsulate the original header into the overlay header. The overlay header will then be placed into an outer UDP header and further into an outer IP header. Figure 3 presents the details of the overlay header. The 1st-3rd 4 bytes are standard. The 4th 4 bytes represent the flow property, including 12 bits of latency, 12 bits of bandwidth and 8 bits of packet loss probability. Note that the flow property represents the service of quality and it decides how to forward the datagrams. The next 4 bytes dedicate the flow identifier (16 bits), the routing path identifier (8 bits) and overlay hop counts (8 bits). The following bytes will hold the hop list with each having 16 bits. The ending bytes of the overlay header will place the payload, including the raw application data and IP header. The encapsulated datagram will be forwarded hop by hop until reaching the destination overlay edge node. For each overlay node, when an overlay datagram comes, it first locates its index in the hop list and then accordingly finds its next hop. The overlay header will be encapsulated into another UDP header with its destination IP pointed to the one of next hop. If it is the last hop (i.e., the ending overlay edge), the overlay header will be decapsulated and the original IP datagram will be eventually handed over to the destination.

Hop-by-hop retransmission and detouring. In the original TCP/IP protocol, if a datagram gets lost in one hop, it will have to be retransmitted again from the original sender. However, such retransmission is not necessary. We only need to resend the lost datagram again from the last hop before it getting lost. Figure 4 (A) presents the procedure. By using hop-by-hop retransmission, the packet loss probability of the original sender can be maximally reduced to 0 and the latency can be dramatically decreased as well. The hop-by-hop retransmission is especially effective for those packet loss based congestion control algorithms. It can avoid falsely reducing the sending rate due to random packet loss or short-term spurs of some intermediate links. To bypass the real congestion, we also develop a detouring technique (shown in Figure 4 (B)). We monitor the packet loss probability (plp) of each intermediate link in an in-band manner and if the plp of a link (e.g., link $(b) \to (c)$ in Figure 4 (B)) reaches a threshold, the sending hop (b) will discard the original routing path and directly retransmit the lost packet to the target overlay edge (d) instead. Meanwhile, it will also send a detouring signal to the source overlay edge (a), which will cause it to choose another path to transmit packets to (d). Note that our time-varying routing module will recognize the congestion and update the route table soon, so the detouring transmission will not last too long.

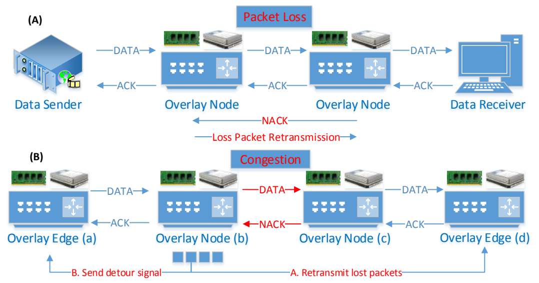


Figure 4: Hop-by-hop retransmission and detouring.

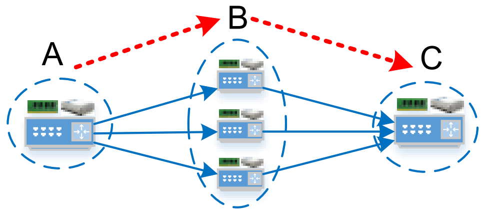


Figure 5: Expansion of OCN.

Multi-path redundant transmission. A single-path transmission often cannot provide a reliable service due to the instability of the underlay network. The unreliability will affect the customer experience, especially for those latency-sensitive and packet loss-sensitive applications. Therefore, we leverage the redundant transmission to ensure the quality of service in probability. When computing the routes, we produce multiple paths for each overlay link with each one labelled by a score representing its network quality. We carefully produce some redundant datagrams according to the scores and send them from multiple paths to the receiver. As a result, either the original packets or the redundant packets being received will ensure the quality and integrity of the transmission. Note that although the redundant transmission will bring some extra cost, it can effectively upgrade the network quality.

Dynamic expansion of OCN. The cloud vendors provide lots of APIs for us to automatically create/destroy VM instances. Therefore, the capacity of OCN can be expanded on demand. We developed an automated dynamic expansion mechanism to help load balance overlay nodes with heavy traffics. The monitor in each overlay node will inspect its real-time throughput and report it to the overlay controller every few minutes. The controller will make forecasts for the future throughput of each group and make a decision if the nodes in the group should be expanded or reduced. Note that we cluster all the nodes in the same geographical region of a vendor as a group. The forecast algorithm details is specified in Section III-B. Figure 5 shows the routing and transport strategy. The routing path distributed by the controller is a group-level path. For example, Figure 5 presents a group-level path $A \to B \to C$. When relaying packets, each hop will first locate the next group according to the routing path. Then, it will find the active nodes in the group from the node table. Finally, it will distribute the packets evenly to different nodes for load balancing.

IV. Evaluation

We deploy OCN on real public cloud platforms and evaluate its performance by running real applications. In the following,

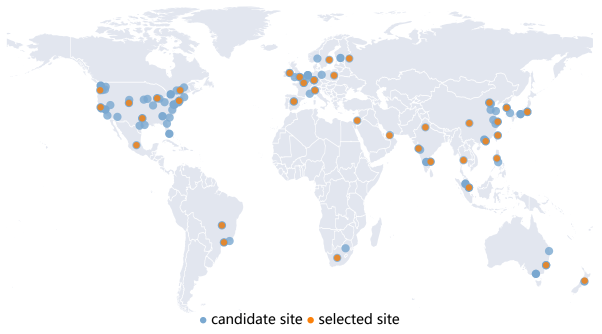


Figure 6: OCN topology planning.

we first illustrate how we deploy the OCN topology and then present the main experimental results, including transport performance and routing performance.

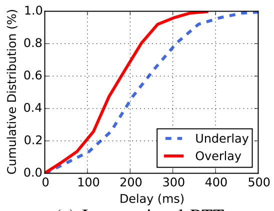
A. OCN Deployment

We developed a crawler which automatically found 229 candidate nodes from the world top 50 public cloud vendors such as AWS, Azure, Aliyun and so on. We then employed the binary particle swarming optimization [6] and selected a few optimized locations that minimize the transport cost between any pair of users from anywhere. The world population data [9] was used to model the user demands to OCN. The 229 candidate nodes and the 37 optimal selection are given in Figure 6. Note that some candidate nodes from different clouds are overlapped. In fact, we find a single cloud vendor can hardly have well global network coverage. To construct a better base topology, we choose a diversity of clouds from different vendors. The optimization for overlay node selection only takes less than a minute on a 3GHz E5-2690 CPU, which makes it possible to capture user requirements and update node selection flexibly per hour. For example, OCN could generate more virtual nodes around the time zones in peaking hours and release some resources in idle time periods.

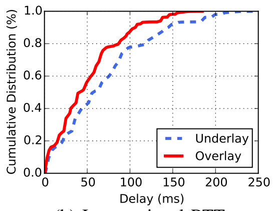
B. Experimental Results

We mainly evaluate the performance of OCN from three dimensions: (1) how OCN transport performs compared to the underlay network; (2) how OCN routing paths perform compared to directed connections and (3) how OCN performs on real applications compared to the underlay network.

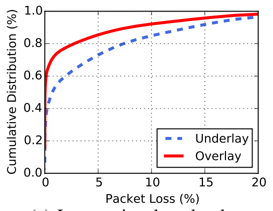
Transport performance. To comprehensively evaluate the performance, we classify the transmission links into inter-regional links and intra-regional links according to the geographical distances. The intra-regional links are usually in the same country or neighboring countries. If a link is inter-regional, its geographical distance is very long and usually transcontinental. We test the network in a full mess manner for two weeks (15∼28 April 2018). Figure 7 summarizes the observed RTT and packet loss rate results of 3.5 billion datagrams as CDFs. Observe that all of the OCN lines are above the dotted underlay line, indicating OCN outperforms the underlay network. Especially, the improvement of inter-regional links is significant, with the RTT reduced by 30ms and the packet loss rate reduced by 1.91% on average. We



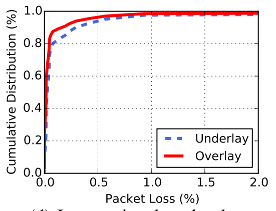
(a) Inter-regional RTT



(b) Intra-regional RTT



(c) Inter-regional packet loss



(d) Intra-regional packet loss

Figure 7: Overlay transport evaluation.

Table I: Overlay routing evaluation. (DC denotes direct connection. SP denotes the shortest path.)

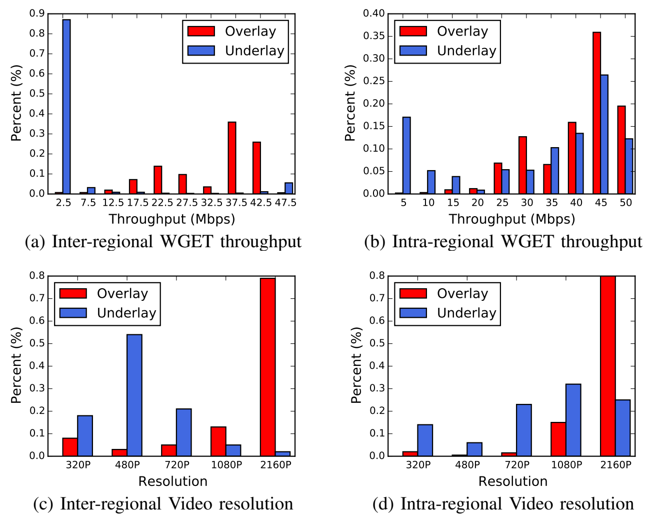
Link type	RTT of DC (ms)	RTT of SP (ms)	#Hops
Inter-regional	214.07	184.10	3.32
Intra-regional	68.38	52.54	2.37
Average	167.99	142.48	3.02
Speedup	15.2%		-

also set a typical aim of 200ms delay with 1% packet loss for inter-regional links, and 100ms delay with 1% packet loss for intra-regional links. The data shows that delay satisfaction rate of the underlay network is only 46.61% for inter-regional links and 77.51% for intra-regional links, respectively. Promisingly, the rate of OCN reaches 73.59% and 87.36%, respectively.

Routing performance. According to our experiment, about 11.2% underlay links show obvious periodicity and 7.1% links bear lots of glitches. Most of these links are transnational or inter-operator links. The rest 81.7% are relatively stable, which are mostly local, domestic or intra-operator links. The prediction model stated in Section III-B detects powerful responses at the period of hours, days and weeks for those periodic links. The 30 minutes overall prediction error (MSE) of delay is 2.83ms and 0.19% for packet loss, showing that our prediction model fits the real network metrics well.

An offline overlay routing test gives some interesting results as shown in Table I. The shortest paths are 15.2% faster than direct connections in average. Specifically, with a probability of 70%, the shortest path passes by an additional node besides source and target, which shows the necessity of overlay routing. The average number of hops of a shortest path is 3.02, implying mostly one mediate hop would be better than direct connections.

Application performance. We use two typical applications, namely, wget downloading and adaptive video streaming, to evaluate the throughput of OCN compared to the underlay network. For both of the two applications, we choose 5 intra-regional links and 5 inter-regional links from our experimental OCN and keep running them for one week, and collect throughput (resolution) at every 30 seconds. Figure 8 presents the comparison results. Observe that OCN outperforms the underlay network on both inter-regional and intra-



(a) Inter-regional WGET throughput (b) Intra-regional WGET throughput

(c) Inter-regional Video resolution (d) Intra-regional Video resolution

Figure 8: Application performance.

regional links. The improvement of inter-regional links is especially significant. The downloading throughput of 88% inter-regional link samples through the underlay network is only around 2.5Mbps while the average throughput of OCN reaches 33Mbps. The resolution of most inter-regional and intra-regional link samples of OCN reach 2160p while underlay only has 480p for inter-regional links and 720∼1080p for intra-regional links. To present better, we also upload a video demonstration of the performance onto Youtube and the url link is https://youtu.be/ocp5mqb38rk.

V. Related Work

The general idea of overlay network has been well studied in [1], [13], [2]. RON [1] improves the resilience and quality of the Internet by routing the traffic through overlay network nodes. Although both RON and OCN use the idea of overlay, their detailed techniques are very different. OCN leverages a set of techniques that RON does not, such as using a machine learning based method to predict the best routes of the future, conducting hop-by-hop retransmissions and detouring when some underlay links becomes bad and providing a better service by multi-path redundant transmission. Moreover, OCN is implemented through public clouds. Its capacity can be automatically expanded very fast. PlanetLab [13] is an architecture that supports the development of new network services. Applications in it are run in a slice acting as a network of virtual machines. However, we argue that they do not implement transport techniques like us either. Besides, it is managed for the benefit of the research community while OCN aims at commercial use comparable to dedicated networks. Some other virtual network infrastructures [2], [10], [5] are also developed for evaluating services and protocols. They are reaching low layers of the network stack, which is different from OCN.

Another related set of researches are SDN [8] and NFV [4]. SDN separates the control plane from the forwarding plane to provide fast transports. While effective, most techniques of SDN requires a large scale of hardware changes and deployments to current network devices. OCN is different. It is intended to build another network plane via the transport fundamentals of the underlay network instead of changing it. NFV aims to replace the dedicated network hardware with standard servers and realize the network functionalities via software. However, it also depends on the change in hardware. Besides, NFV mainly focuses on a single node rather than on the entire network as OCN.

VI. Conclusion And Future Work

In this paper, we propose a global Overlay Cloud Network, which is implemented by three key techniques, namely, automated topology planning, time-varying routing, and dynamic transport. We deploy OCN on real public cloud platforms and evaluate it comprehensively with the underlay network. The results show that OCN substantially improves the performance of the underlay network. In the future work, we will conduct a more thorough evaluation of OCN from different dimensions, including availability, scalability, stability and so on.

References

[1] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. *Resilient overlay networks*, volume 35. ACM, 2001.

[2] Andy Bavier, Nick Feamster, Mark Huang, Larry Peterson, and Jennifer Rexford. In vini veritas: Realistic and controlled network experimentation. In *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '06, pages 3–14. ACM, 2006.

[3] Chuanxiong Guo, Lihua Yuan, Dong Xiang, Yingnong Dang, Ray Huang, Dave Maltz, Zhaoyi Liu, Vin Wang, Bin Pang, Hua Chen, et al. Pingmesh: A large-scale system for data center network latency measurement and analysis. In *ACM SIGCOMM Computer Communication Review*, volume 45, pages 139–152. ACM, 2015.

[4] Hassan Hawilo, Abdallah Shami, Maysam Mirahmadi, and Rasool Asal. Nfv: state of the art, challenges, and implementation in next generation mobile networks (vepc). *IEEE Network*, 28(6):18–26, 2014.

[5] Ran Ju, Weiguang Wang, Jin Li, Feng Li, and Lei Han. On building a low latency network for future internet services. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pages 1–6. IEEE, 2017.

[6] James Kennedy and Russell C Eberhart. A discrete binary version of the particle swarm algorithm. In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, volume 5, pages 4104–4108. IEEE, 1997.

[7] Derek Leonard and Dmitri Loguinov. Turbo king: Framework for large-scale internet delay measurements. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 31–35. IEEE, 2008.

[8] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[9] U.S. Department of Commerce. Census. http://www.census.gov/, 2018.

[10] Rob Sherwood, Glen Gibb, Kok-Kiong Yap, Guido Appenzeller, Martin Casado, Nick McKeown, and Guru M Parulkar. Can the production network be the testbed? In *OSDI*, volume 10, pages 1–6, 2010.

[11] Ramesh K Sitaraman, Mangesh Kasbekar, Woody Lichtenstein, and Manish Jain. Overlay networks: An akamai perspective. *Advanced Content Delivery, Streaming, and Cloud Services*, 51(4):305–328, 2014.

[12] Guoming Tang, Kui Wu, and Richard Brunner. Rethinking cdn design with distributee time-varying traffic demands. In *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*, pages 1–9. IEEE, 2017.

[13] Jonathan S Turner, Patrick Crowley, John DeHart, Amy Freestone, Brandon Heller, Fred Kuhns, Sailesh Kumar, John Lockwood, Jing Lu, Michael Wilson, et al. Supercharging planetlab: a high performance, multi-application, overlay network platform. *ACM SIGCOMM Computer Communication Review*, 37(4):85–96, 2007.