End-to-end Network Throughput Optimization Through Last-mile Diversity

Ning Wu, Ao Tang
School of Electrical and Computer Engineering
Cornell University
Ithaca, USA
nw276@cornell.edu, at422@cornell.edu

***Abstract*—In this paper we propose a platform that optimizes the available end-to-end throughput in real time through overlay networks. With the knowledge the network topology and conditions, it strives to achieve the optimal end-to-end throughput by exploring the last-mile diversity. It allows the flexible and responsive per-end-user selection of the edge node for the overlay networks, and thus can fast recover from network failures or performance degradation. We present our design of the end-to-end throughput optimization system with detailed discussion of each component including dynamic routing engine, performance monitor and information exchange. Our experimental results from a real-world deployment show that it not only brings up to 5 times throughput gains in the presence of 0.05% loss, but also improves 20% throughput when facing delay increase in the original path.**

***Index Terms*—End-to-end Throughput; Routing; Overlay Networks**

I. Introduction

Internet is known to be unreliable. In spite of various levels of Service Level Agreement (SLA) offered by Internet Service Providers (ISP), most individual users and enterprises still suffer from performance degradation from time to time such as low speed, large delay and high packet loss rate. To improve the reliability of user's network experience, the very first question is how to realize a global end-to-end control without which any effort to seriously boost network performance is bound to fail. This however has been a highly complex issue in the Internet, for two reasons. First, the current Internet is an aggregation of a large number of networks owned by many ISPs with different economic interests [2]. The standard way to obtain end-to-end SLAs is to create private networks through business contracts among them. This obviously is costly and takes lots of time to realize. Second, the management and configuration complexity of Internet hardware routers produced by major vendors is substantial. The flexibility, availability and cost efficiency is increasingly limited by those hardware routers. The prevailing routing protocols being used in today's Internet such as the Border Gateway Protocol (BGP) [1] are known to have limitations in realizing any performance-aware dynamic routing solution such as [4]–[7]. BGP routing neither incorporates the information of path performance or link capacity, nor allows fine-grained overriding of BGP-specified forwarding behavior [3]. With these two severe constrains, network providers and operators face great difficulties in realizing reliable, high-performance routing control.

In face of the above mentioned challenges in the Internet infrastructure, overlay is an enabler of achieving flexible control of routing, requiring no changes in the underlay Internet. Overlay network architecture places a virtual network over the physical infrastructure, leveraging the dynamic control of network resources on an abstraction layer. Although there exist various overlay-based techniques of performance-aware routing on fast timescale [8], [9], they focus on routing inside the overlay network whereas the performance of the upstream and downstream path in the last mile for the end users is beyond their control. In the case where the communication between the end user and the edge node is through the public Internet, the end-to-end performance can still be largely affected by any unpredicted delay and congestion over the Internet. In spite of lacking control with the last-mile route, more reliable end-to-end throughput is possible if there exists diversity and flexibility in selecting the edge node.

In this paper, we propose the design of a platform that allows flexible and responsive routing control of the end-to-end path. We present that exploring the geographical and physical diversity in the last-mile can effectively avoid the failure and congestion in the last-mile once being detected, and thus to a large extent provide end-to-end reliability and high performance. Furthermore, the throughput performance can be jointly optimized if we combine the real-time traffic conditions of both the core network and the last-miles to make the end-to-end routing decisions. The remainder of the paper is organized as follows: Section II discusses the background of overlay network and the technical challenges in achieving our design goals. Section III describes the overall system architecture together with the design of each component. Section IV presents experimentally the throughput improvement over an overlay network with the system deployment we proposed.

II. Background

A. Overlay Network

Various overlay networks have been designed and developed for different purposes since a few decades ago [10], [11], [13]. Figure 1 shows an example of the end-to-end path through an overlay network. As the interface point between the overlay network and the end users, the Point of Presence (PoP) in each

location is responsible for assisting the application traffic to enter or leave the overlay core network. Traffic from end users are directed into the ingress PoP, forwarded through the virtual overlay networks to the egress PoP, and ultimately delivered to the destination. Naturally, we break down the end-to-end path into three parts: the virtual overlay network and the two last-miles referring to the connections between the edge PoPs and the end users on both sides.

[13] proposed the Resilient Overlay Network (RON) that could fast recover from failures and performance issues of the Internet taking advantage of underlying Internet redundancy. Although the overlay network is well developed, the existing performance-aware routing techniques via the overlay network are mostly dedicated to the core network without jointly considering the last mile [8], [9], [13]. As overlay network is widely used particularly in Content Delivery Networks (CDN), end-to-end reliable performance is a major design goal that most CDNs focus on. However, their techniques are not applicable to the end-to-end performance problem with a pair of end users, because the servers under this context belong to a part of the CDNs which are controlled within the core networks rather than end hosts outside. Furthermore, their solutions only apply to web-based applications traffic.

B. Last Mile

Routing within the core network is managed by the routing protocols implemented in the overlay networks, whereas the routes for the last-mile are beyond the control of the overlay network. Therefore, it is nontrivial for the overlay network to react to the congestion or failure occurred in the last-mile and guarantee the end-to-end performance. For example, if the end user in Figure 1 experiences much lower throughput and the congestion is detected to be in the last-mile connection between the ingress PoP and the end user, then changing the last-mile path by switching over to a different ingress PoP may possibly improve the end-to-end performance. To achieve the design goals, the main technical challenge lies in how to flexibly control the route for the last-mile.

The state-of-art approaches to attracting end-user traffic to the core network include DNS [12], [14] and Anycast [16], [17]. Most DNS services do not consider the real-time performance when deciding the ingress PoP. Even for those who consider performance metrics such as latency, the measurement is done on coarse scale in both temporal and spatial dimension (at most once per day, only measure latency between regions). Furthermore, the DNS service is not aware of the end-user rather than the local DNS resolver on behalf of the end user which could be geographically far away or experience completely different performance.The server will make decision on the ingress POP for each resolver rather than the actual clients, so the performance information for the client is not possible to be mapped to the resolver. One solution is the EDNS client-subnet-prefix standard (ECS) [18] that allows the resolver to specify a prefix of the client's IP when requesting domain name translations on behalf of a client. This enables the end-user mapping on the level of client subnet, but it does not consider the specific end-to-end performance for each end user [15]. Moreover, the proposed protocol is far from being adopted by all ISPs. Anycast approach makes the end user's IP address to be transparent. However, it lacks the flexibility of performance-aware routing control because the performance metrics used in BGP protocol do not represent the real-time performance information.

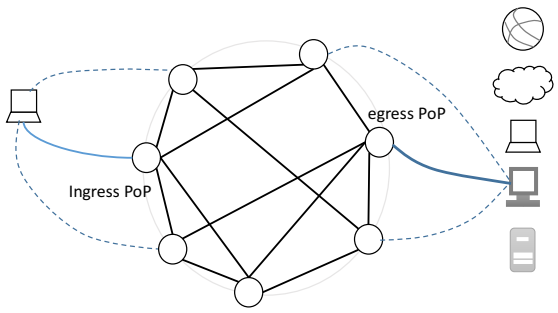


Fig. 1. End-to-end path through an overlay network.

III. Design

The goal of our work is to optimize the end-to-end throughput per end user among the last-mile diversity. In particular, when the available throughput of the current path becomes consistently lower than some other path, the system is expected to quickly detect it and switch to the better path no matter the issue is caused by the core network or the last mile. This requires the system to have the ability of fine-grained performance monitoring, flexible routing control, and highly responsiveness to performance degradation. We make the following assumptions in our design. First of all, we focus our attention on the throughput performance for a single TCP flow. Multiple flows can always achieve high aggregated throughput at the price of packet loss due to bandwidth competition. Secondly, we assume there are multiple end-to-end paths that satisfy the customer's performance requirement. Without this assumption, it is not possible to explore the last-mile diversity to guarantee reliable performance when the initial path fails. Furthermore, the routing for the last mile between the end user and a given ingress/egress PoP is via the Internet which is beyond our control. Last but not least, we assume the connection of any application is through the domain name. The application of the traffic is not limited by web content, but arbitrary application such as file transfer, remote login, and real time streaming.

A. Overall Architecture

Figure 2 shows the overall architecture of our end-to-end throughput optimization design. It consists of two critical components: the end-user agent and the edge server. The edge server distributed in each PoP monitors the performance of the core and shares the statistics with the end-user agent. It also performs data forwarding on the overlay network. The end-user agent is deployed in the end user's device,

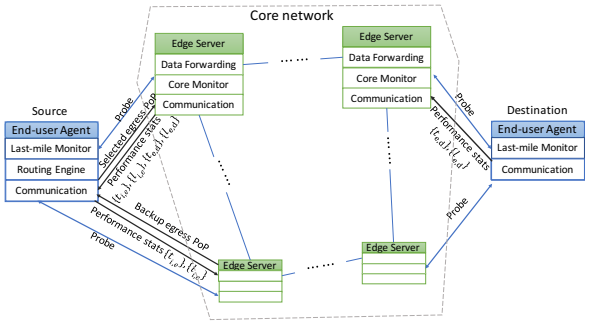


Fig. 2. End-to-end throughput optimization system design.

responsible for controlling the route, monitoring the last-mile performance, and communicating with the core network. The last-mile performance statistics monitored at the destination agent is propagated backward to the source end-user agent. As the end-user agent communicates with the edge server, decisions on which ingress and egress PoP to select can be made by the source agent based on the real-time understanding of the network.

Our design has the following advantages: it takes less time to respond to outage and performance failure occurred in between the end user and the ingress PoP; it requires less overhead for performance monitor in the core network because the edge server only need to measure the performance per overlay link but not per flow in the core; it does not impose any limitations on the routing in the core, so routing in the core network can be flexibly controlled and customized by the overlay network.

B. End-to-end Dynamic Routing

Consider a core network with a set $\mathcal{P}$ of PoPs worldwide. For a given user with a pair of source s and destination d, we define a set of ingress PoP candidates $\mathcal{I}_s \in \mathcal{P}$ for the source s and a set of egress PoP candidates $\mathcal{E}_d \in \mathcal{P}$ for the destination d. The end-to-end path denoted as $p_{s,d}$ is a composition of three subpaths $p_{s,i}$, $p_{i,e}$ and $p_{e,d}$. We assume that inside the overlay network the route between any ingress and egress PoP is controlled by specific performance-aware routing algorithm deployed. Note that the route of the last mile is beyond the control of the overlay network, so controlling the end-to-end routing turns to be determining the ingress PoP and egress PoP for a given source-destination pair.

We aim to ensure reliable performance of available TCP throughput and dynamically update the routing to react to any performance failures in the currently selected path. It is known that TCP exhibits complex behavior under window control and congestion control. The available bandwidth is affected by multiple factors including buffer limits, physical capacity and link characteristics. In path selection, only link conditions of latency and loss rate vary among multiple paths holding all the physical conditions to be identical. As a result, the performance metrics we use in path selection is the relative evaluation of the available bandwidth given the current delay and loss conditions in each path. The performance metrics are based on the following well-known Mathis model [19]

$$Throughput = \frac{MSS \times C}{RTT \times \sqrt{loss}},$$

where C is a constant that incorporates the loss model and the acknowledgment strategy. The throughput is linearly proportional to $1/RTT \times \sqrt{loss}$ in the presence of packet loss. Here we only evaluate the TCP relative performance to compare among candidate paths, rather than computing the precise value. To find the optimal path that achieves highest TCP throughput, we consider $RTT \times \sqrt{loss}$ as the cost estimation for each end-to-end path. The performance cost from s to d is dependent of the choice of the ingress PoP i and egress PoP e, denoted as $w_{s,d}^{i,e}$. The optimal path is selected by finding the optimal ingress and egress PoP, denoted as i^* and e^*, that result in the least path cost:

$$w_{s,d}^{i^*,e^*} = \min_{i \in \mathcal{I}_s, e \in \mathcal{E}_d} w_{s,d}^{i,e}.$$

For any $i \in \mathcal{I}_s$ and $e \in \mathcal{E}_d$, the end-to-end path cost from s to d is given by

$$w_{s,d}^{i,e} = T_{s,d}^{i,e} \sqrt{L_{s,d}^{i,e}},$$

where $T_{s,d}^{i,e}$ and $L_{s,d}^{i,e}$ is the latency and loss metrics of the end-to-end path given the ingress PoP i and egress PoP e. Let $t_{x,y}$ and $l_{x,y}$ denote the latency and loss rate that are actually measured between any arbitrary node x and y. The latency metrics $T_{s,d}^{i,e}$ are additive on the end-to-end path in the following form:

$$T_{s,d}^{i,e} = t_{s,i} + t_{i,e} + t_{e,d}.$$

The delay and loss rate in the last miles are monitored by the end-user agents in the source and destination end hosts. They use active probing to measure the loss and latency in real time for each ingress/egress PoP candidate. Note that a limit exists in the granularity of loss rate being measured. If the interval of each probing is τ, then the loss rate that can be detected during probing time period Δ_p is no smaller than τ/Δ_p, which we denote as σ. As a result, any loss rate below σ is reported as 0. Since the throughput is linearly proportional to $1/RTT$ in the case of zero packet loss, we set a lower bound for the end-to-end aggregated loss rate to characterize the cost evaluation for zero loss:

$$L_{s,d}^{i,e} = \max\{1 - (1 - l_{s,i})(1 - l_{i,e})(1 - l_{e,d}), \sigma\}.$$

In the core network, edge servers do not measure the performance metrics for a particular user pair or application, but rather the performance of the overlay link between two PoPs which is determined by the current aggregated traffic. Routing in the core relies on the specific implementation of the overlay networks which is independent of the selection of ingress and egress PoPs for end-to-end path routing. Being

unaware of how traffic is routed in the core network, the end-user agent at the source determines the ingress and egress PoP based on the three sets of statistics collected.

C. End-user Agent

End-user agent is the key component in order to realize flexible fine-grained routing control and real-time monitoring in the last mile of the end-to-end path. The agent is deployed in the end user's device, performing mainly three functions: end-to-end routing control, last-mile performance monitor, and communication with edge servers. We here describe each of the functions in detail.

Routing control: The agent in the source end user controls the route for the end-to-end path, excluding the sub-route between the ingress PoP and egress PoP which is controlled within the core network independently. The routing control on the end-user agent involves three steps: determining the ingress and egress PoP, directing traffic to the correct ingress PoP and notifying the core network with the correct egress PoP. The ingress and egress PoP are determined by consulting the solutions for the optimal path described in Section III-B. The source agent stores the delay and loss rate metrics collected for each subpath, as are required in the routing computation. Table I shows an example of the routing table maintained by the source agent with an update interval of Δ_r. It contains the *Current Route* being used with respect to the ingress and egress PoP, as well as the *Proposed Route* that has the least end-to-end path cost. In order to avoid path flipping, the new routing decision is made in a conservative and cautious way such that the *Proposed Route* becomes the *Selected Route* only if either of the two conditions holds: it has a cost below 80% of the *Current Route*'s cost; or it is consistently the *Proposed Route* for the last three update cycles. Otherwise the *Selected Route* remains to be the *Current Route*. Furthermore, we insert the *Backup Route* into the routing table which represents the best egress PoP for each given ingress PoP other than the one associated with the *Selected Route*.

If an update of *Selected Route* involves the change of the ingress PoP, the end-user agent in the source will modify the local DNS hosts file with the corresponding ingress PoP IP address so that any traffic destined to the given destination are redirected to the ingress PoP provided. Besides attracting traffic to the correct ingress PoP, the agent need to notify the selected ingress PoP about the selected egress PoP, and notify each alternate ingress PoP about its corresponding *Backup Route* to have a default route installed. This notification messages are transmitted through the communication channels between the agent and edge servers.

Performance Monitor: End-user agent uses active probing to measure the RTT and loss rate between the source and ingress PoP, and between the destination and egress PoP. Originating the probing packets at the end-user agent for the last-mile measurement prevents the probing from being blocked by the firewall in the end user. The edge server deployed in ingress and egress PoP is responsible for replying to the received probing packets. The probing packet is sent once every τ time interval. By checking the sequence numbers, the end-user agent measures the RTT and the round-trip loss rate based on the reply packets every Δ_p time period. $t_{s,i}$ and $t_{e,d}$ are computed by averaging the RTTs among all probing packets. $l_{s,i}$ and $l_{e,d}$ are viewed as the fraction of lost probing packets over all sent packets. Since we set symmetric routing for the forward and backward paths, it is reasonable to estimate the loss rate for the round trip.

TABLE I
ROUTING TABLE MAINTAINED IN THE END-USER AGENT.

Ingress PoP	Egress PoP	Cost	Current Route	Proposed Route	Selected Route	Backup Route
i_1	e_1	w^{i_1,e_1}	×	×	×	×
i_1	e_2	w^{i_1,e_2}	✓	×	×	✓
i_2	e_1	w^{i_2,e_1}	×	✓	✓	
i_2	e_2	w^{i_2,e_2}	×	×	×	

Communication: The communication between the end-user agent and the edge servers include three types of control messages: handshake messages, measurement statistics and notification of egress PoP selection. At the startup stage of the end-user agent, the source agent discovers the alive edge servers via ping packets and selects the K closest PoP into the ingress PoP set $\mathcal{I}_s$ based on the RTTs of the ping packets. Similarly, the destination agent selects the set of egress PoP $\mathcal{E}_d$. The relationship of source and ingress PoP is confirmed by exchanging handshake messages between the end-user agent and the edge server of the selected PoP. Once the control channel is established in the last miles, the source agent periodically pulls the measurement statistics of the core network from the edge servers of the ingress PoPs, and pushes notification of egress PoP selection to the corresponding edge servers. The last-mile performance statistics measured by the destination end user are propagated back to the source end user via the current returning path.

D. Edge Server

The edge server is distributed in each edge PoP of the core network. It monitors the latency and loss rate between itself and each other PoP using active probing. Because the underlying physical infrastructure is invisible for the overlay network, it is even more important to be responsive to failures or changes in the network conditions. The performance of the overlay link between any two PoPs is expected to be measured on fast timescale. Once receiving pull request from the source ingress PoP, the edge servers immediately reply back with the latest statistics. The other job of the edge servers is to perform data forwarding. Various techniques exist in realizing data forwarding through overlay networks. One approach being widely adopted is to encapsulate the IP packets with an overlay header that contains the address information of the overlay node. Another way is to manipulate the IP addresses in the original packet without inserting additional header. By changing the destination IP address of the original packet to the proper edge nodes, the traffic from end users are directed to

the overlay network. Since the original destination is missing in the packet header, this approach requires a mapping between the source and the edge nodes to be stored in the edge nodes before user's traffic arrive. We use the latter mechanism in our experiments to avoid the additional overhead caused by encapsulation and decapsulation.

IV. Evaluation

In this section, we evaluate our design of end-to-end throughput optimization. We describe the testbed setup and testing scenarios in Section IV-A, and present the experimental results in Section IV-B that show the throughput improvements achieved.

A. Testbed Setup

We set up an overlay network testbed worldwide shown in Fig on Microsoft's Azure cloud platform. The overlay network topology consists of 8 PoPs in different geographical locations, being connected via vxlan tunnel. Software switch is running in each of the PoP, and Open Shortest Path First (OSPF) is implemented as the routing protocol within the core network. As we mentioned above, routing inside the core network is invisible to the end-user agent. The end-user agent is deployed in the end users located in Ithaca and San Francisco, and the edge server is running in each PoP. We set probing interval τ to be 3ms, and both the measurement duration Δ_p and the agent's routing update interval Δ_r to be 3 minute. Therefore the minimum detectable loss rate σ is $1/60000$.

We measure the throughput for a single TCP flow in the form of the file transfer speed via scp from the end user in Ithaca to the end user in San Francisco The file of size 2Gbits is transferred every 30 minutes. We further choose the size of ingress PoP set to be 3, and egress POP set to be 2, to reasonably bound the last-mile delay. At the startup stage, the source agent discovers its ingress PoP set $\mathcal{I}_s$ to be {Richmond, Toronto and Chicago}. The destination agent discovers its egress PoP set $\mathcal{E}_d$ to be {San Francisco, Seattle}. Our evaluation of the end-to-end throughput optimization considers network condition changes in either the last mile or the core network. We compare the throughput of our routing solutions against the throughput using the geographically closest ingress and egress PoP which is Richmond and San Francisco respectively.

B. Experimental Results

Loss in the last mile: We evaluate the effectiveness of our system when facing loss in the current last mile. Internet is possible to suffer from loss at any point of the last mile, but here we simplify the testing scenarios by simulating the loss at the interface of the edge PoP to present the proof of concept. The loss is simulated using the netem [21] functionality provided by Linux tc tool. The measurement of TCP throughput for the current selected path is sampled every 30 minutes by performing the file transfer.

In the first scenario we imposed loss onto ingress PoP. Figure 4a shows the throughput results of each measurement

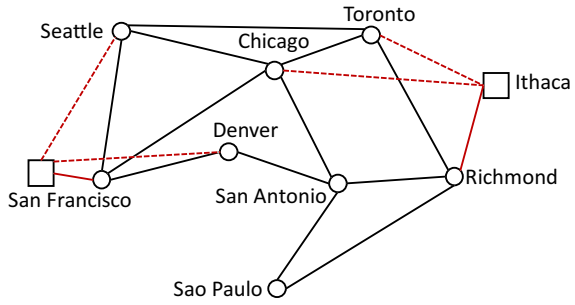
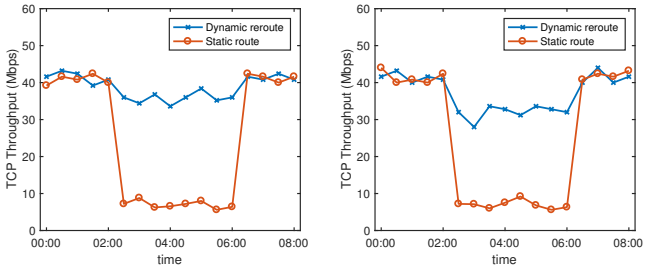


Fig. 3. Network topology.



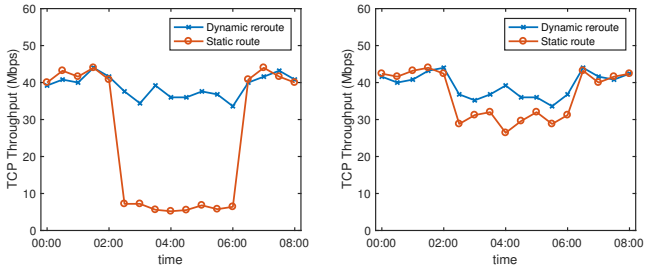
(a) (b)

Fig. 4. End-to-end throughput in the presence of loss in the last mile: (a) between the source and the ingress PoP, (b) between the egress PoP and the destination.

sampling. At the beginning, the selected ingress PoP was consistently Richmond. 0.05% loss was added to Richmond at time 2:15. In the presence of loss, the throughput of the original path significantly dropped down to less than 10 mbps. With the ability of monitoring the real-time performance and exploring the route diversity, our system changed the ingress PoP from Richmond to Toronto. Around 15% performance degradation after reroute was mainly caused by the delay increase in the new path. We removed the simulated loss rate at time 6:15. Fig 4a further demonstrates that our system is also able to improve the performance by finding better path if existing.

In the second scenario, we considered adding 0.05% loss on the current egress PoP (San Francisco) between time 2:15 and 6:15. As illustrated in Fig 4b, the above 80% throughput drop for the static path is similar to that in the first scenario. The destination agent detected the loss in the last mile within 3 minutes. With the knowledge of the loss increase in the current path, the source agent selected an alternate path with Toronto as the ingress PoP and Seattle as the egress PoP, resulting in around 20% throughput decrease on average due to the delay growth.

Loss and failure in the core: We next evaluate how our system reacts to network condition changes in the core network. Although the routing inside the core network is uncontrollable and invisible for the end-user agent, it strives to select the best pair of ingress and egress PoP among all



(a) (b)

Fig. 5. End-to-end throughput in the case of: (a) loss increase in Denver PoP, (b) failure in San Antonio PoP.

candidates, with the real-time information of latency and loss rate between each pair collected from the edge servers. Our evaluation in this part involves the cases of unexpected loss increase and link failure in the core network. Fig 5a shows the throughput results when we simulated 0.05% loss in Denver PoP from time 2:15 to 6:15. Our end-user agent in the source learned about the loss increase in the subpath and selected an alternate path with Toronto as the ingress PoP and San Francisco as the egress PoP.

Lastly, we failed San Antonio PoP and assumed the routing strategy implemented in the core network was able to detect failures and reroute accordingly. In Fig 5b, the line of static failover represents the case that the ingress and egress PoP remained statically to be Richmond and San Francisco, while the core network rerouted traffic from Richmond PoP to San Francisco PoP through Toronto to avoid the failed node. With dynamic reroute, the source agent updated the ingress PoP to be Toronto once finding this path had a smaller end-to-end path cost. The new path selected by the agent outperforms the static failover solution by 20% higher throughput.

V. Conclusion

We propose a platform for last-mile route control in overlay network architecture. It aims at optimizing the end-to-end throughput by exploring the last-mile diversity based on the network topology and conditions. In this paper we present our design of each component in the system. We further carry out experiments in an overlay networks with the deployment of our end-to-end throughput optimization solutions. The experimental results demonstrate that the end-to-end throughput can be significantly improved in face of network condition changes in both loss and latency.

References

[1] Rekhter, Yakov and Li, Tony and Hares, Susan, “RFC 4271: A Border Gateway Protocol 4 (BGP-4),” RFC 4271, Internet Engineering Task Force, 2006.
[2] Clark, David, “The Design Philosophy of the DARPA Internet Protocols,” ACM SIGCOMM Computer Communication Review, 1988.
[3] Caesar, Matthew and Rexford, Jennifer, “BGP Routing Policies in ISP Networks,” IEEE Network, 2005.
[4] Elwalid, Anwar, Cheng Jin, Steven Low, and Indra Widjaja, “MATE: MPLS adaptive traffic engineering,” INFOCOM 2001.
[5] Kandula, Srikanth, Dina Katabi, Bruce Davie, and Anna Charny, “Walking the tightrope: Responsive yet stable traffic engineering,” ACM SIGCOMM Computer Communication Review, 2005.
[6] Alizadeh, Mohammad, et al., “CONGA: Distributed congestion-aware load balancing for datacenters,” ACM SIGCOMM Computer Communication Review, 2014.
[7] Michael, Nithin, and Ao, Tang, “Halo: Hop-by-hop adaptive link-state optimal routing,” IEEE/ACM Transactions on Networking (TON) 23.6 (2015): 1862-1875.
[8] Yap, Kok-Kiong et al., “Taking the Edge off with Espresso: Scale, Reliability and Programmability for Global Internet Peering,” Proceedings of ACM SIGCOMM, 2017.
[9] Schlinker, et al., “Engineering Egress with Edge Fabric: Steering Oceans of Content to the World,” Proceedings of ACM SIGCOMM, 2017.
[10] Hagens, R. A., M. T. Rose, and N. E. Hall. “Use of the Internet as a Subnetwork for Experimentation with the OSI Network Layer,” Internet Engineering Task Force, Feb 1989. RFC 1070.
[11] Eriksson, Hans. “Mbone: The multicast backbone,” Communications of the ACM 37.8 (1994): 54-61.
[12] Guardini, Ivano, Paolo Fasano, and Guglielmo Girardi. “IPv6 Operational Experience within the 6bone,” Proc. Internet Society (INET) Conf, 2000.
[13] Andersen D, Balakrishnan H, Kaashoek F, Morris R. “Resilient overlay networks,” ACM, 2001.
[14] Nygren E, Sitaraman RK, Sun J. “The akamai network: a platform for high-performance internet applications,” ACM SIGOPS Operating Systems Review, 2010.
[15] Chen F, Sitaraman RK, Torres M. “End-user mapping: Next generation request routing for content delivery,” ACM SIGCOMM Computer Communication Review, 2015.
[16] Ballani, Hitesh, and Paul Francis. ”Towards a global IP anycast service.” ACM SIGCOMM Computer Communication Review, 2005.
[17] Katabi, Dina, and John Wroclawski. ”A framework for scalable global IP-anycast (GIA).” ACM SIGCOMM Computer Communication Review, 2000.
[18] C. Contavalli, W. van der Gaast, D. Lawrence, and W. Kumari. “Client Subnet in DNS Requests,” IETF Draft draft-vandergaast-edns-client-subnet-02, July 2015.
[19] Mathis, Matthew, et al. “The macroscopic behavior of the TCP congestion avoidance algorithm.” ACM SIGCOMM Computer Communication Review, 1997.
[20] https://azure.microsoft.com/.
[21] https://wiki.linuxfoundation.org/networking/netem.