Measuring the Perforamnce of RON-Style Networks in Today's Internet

Asad Ranavaya
New York University
ar7182@nyu.edu

Weiqi Hu
New York University
wh2168@nyu.edu

Abstract

This paper evaluates the performance of the RON-style overlay network in the context of modern internet infrastructure. Through extensive experimentation and analysis, we determine that RON-style architecture is less effective at optimizing throughput in current network conditions than it was two decades ago. However, our study also highlights the resilience of modern internet infrastructure, with no downtime in ping availability observed during our experiments. We conclude that while RON-style architecture may not be as effective as it once was in the area of performance boosting, further research should be directed towards developing cloud-aware overlay technologies that are better suited to modern cloud computing environments and goals.

1 Introduction

Modern day internet is known for its reliability and fast speeds, however, this has not always been the case. For over two decades, the use of Resilient Overlay Networks has provided many benefits when compared to the bare-bones internet infrastructure of routing protocols like BGP(1). The use of a RON was originally studied to provide fast rerouting in the face of down-time while also providing a framework to find optimal routing paths that focus on latency and throughput. In this paper we re-examine a network that aims to emulate the original RON setup and log its performance gains relative to both the current internet and that of the internet in 2001. In this paper we take advantage of the superior cloud infrastructure that allows us to instantly deploy nodes to all regions of the world for a more in-depth and complete study. [1]

1.1 Importance

We have chosen to study this design primarily for two reasons, the need for continuing re-examination of previous architecture, and to study the performance of the internet at large. It is crucial to re-determine the efficacy of influential papers years after they have been released to determine what valuable insights about modern internet infrastructure remain.

1.2 Measuring the Internet

Measuring the performance of the internet has been a subject of interest for researchers for many years. Various studies have been conducted to evaluate different aspects of the internet, including latency, throughput, and packet loss. For example, in this early 2005 study (10), the authors analyzed the latency and throughput characteristics of the internet using active measurements. They found that the latency and throughput of the internet vary significantly based on the geographical location of the hosts.

In another early study (8), the authors examined the performance of the internet's end-to-end path quality due to selection. They measured the routing paths, latency, and throughput of different paths in the internet and found that the paths had better alternatives in 30-80% of cases. While this is a much older study, it is important to note that characteristics of the study and goals are similar to our own purposes today.

More recent studies have focused on measuring the performance of the internet in the context of cloud computing. For example, in this study (9), the authors evaluated the performance of the internet for cloud applications for the purposes of scientific computing. In doing so they focused on properties prevalent in the cloud setting and examined whether concepts such as hyper-threading, containerization, and virtualization etc. had a measurable impact on efficiency and throughput. They found that higher allocations of vCPU and proper hyperthreading can improve Cloud VM perforamnce as well as throughput.

These studies demonstrate the importance of measuring the performance of the internet and the need for accurate measurement tools and techniques to evaluate the Internet's performance in different contexts.

[1]All codes used and experimental data found for this work are available at https://github.com/asadranavaya/nyu-cs2620-001-spr23

2 Architecture

Cloud Provider	Region	Instance Type
AWS	US West 1	T2.Micro
AWS	US East 2	T2.Micro
AWS	West 2	T2.Micro
AWS	EU Central 1	T2. Micro
AWS	EU North 1	T3.Micro
AWS	SA East 1	T2.Micro
Azure	US East	B1
Azure	France Central	B1
Azure	Japan East	B1
Azure	Brazil South	B1
Azure	South Africa North	B1
GCP	US Central	E2
GCP	US South	E2
GCP	Europe North	E2
GCP	Australia South East	E2
GCP	Asia East 1	E2

Table 1: Virtual machine's used in experiment

Due to the ability to have nodes in any region of the world, we chose to deploy virtual machine instances in every continent where services are provided. We leveraged three cloud providers: Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure.

2.1 Server

An important thing to note is that the VM's hosted by GCP and AWS maintain ephemeral public IPv4 addresses, therefore a server node is required to start the experiment application. The server node is hosted on an AWS server that maintains a static ip address that all of the nodes have knowledge of. The server application is started and is given the number of nodes to expect and how long the experiment should run. After all nodes connect to the server, the server disseminates the ip address of all the other nodes as well as how long each node should run. An input log is also created from the server in which to later run the map-reduce program to analyze optimization opportunities.

2.2 Overlay Nodes

The overlay nodes are most crucial portion of any overlay network as they operate as the routing points in-between destinations. Each node operated a python pinging and wget operation at intervals of 1 second and 30 seconds respectively to more truthfully re-implement the original RON experiment. Our setup consisted of 5 AWS T2.Micro VM's and a T3.Micro instance in Stockholm, 5 B1 instances in Azure, and 5 E2 instances in GCP. Each vm had special inbound rules that allowed all ICMP requests as well as http requests. A more nuanced approach would only have inbound protocols accepted for a set of prefixes known ahead of time to the RON.

To gather information about the underlying paths and optimization opportunities each node collected information on every other node via a multi-threaded python process. If a node could not reach another node within 1 second for pinging it was recorded as 1 full second of latency, which would later be marked as unreachable for that time frame.

For completeness it should be of note that each machine on the VM's were only given 1 core to multi-thread. We allowed the output buffer to fill and let the OS manage when to fsync to the file, while it is negligible for the throughput measure file, it may have cost some paths latency on the ping measure. This should not be an issue as we analyze the path stability as function of times optimized throughout the 16 hours.

To ensure that the network was accurately measured and that local factors such as CPU processing time did not affect our results, we implemented a 60-minute ping program to run sequentially and ensured that it was the only program running during the experiment. We compared the results of the sequential program with those of a parallel one and found no significant differences. Therefore, we determined that it was safe to run the ping program in a multi-threaded fashion. As for throughput, we considered the data processing time of the operating system when transferring files, but this should not be a significant issue as the file size is small at only 1 MB. The only major concern we had is that the throughput results may be adversly affected due to the nginx server vending to each other n nodes. However, the free tier VM's we used only offered 1 vCPU.

3 Methodologies

We measured the latency at every second and from each node to all nodes. This resulted in a distance matrix of size 16^2 for each time frame. To analyze the paths with any potential optimization opportunities in latency we used a modified version of the any pairs shortest path Floyd-Warshall algorithm. To recover the actual paths we maintained a separate matrix per time frame to rebuild the shortest paths between nodes.

3.1 Internet Measurement as Function of time

The internet is an ephemeral system where observations made today may not be the same as observations made tomorrow. This is simply a function of both constantly changing data planes as well as the continuing advancements in technologies that drive the internet as a whole. This is also a driving force in our experiment, to determine if fundamentals in internet architecture have substantially changed from 2001.

To overcome this fleeting nature we ran experiments in time intervals of 30 minutes, 1 hour, and 16 hours over the course of 1 month. Doing so allowed us to determine stability between routes as well as stability between the number of optimal hops found. We also chose to do this to mirror the original RON experiment in which multiple trials were ran to accurately gauge internet speeds and weed out any fluctuations and outliers.

3.2 Analyzing Results

To create the analysis results we used a couple of map-reduce programs to combine each nodes' multi-threaded output. The ability to use map-reduce paradigms were a crucial development in our longer experiment as combining the output from each node at that scale would not have been feasible on our local machines.

3.3 Optimizing Latency

The overarching idea for our map-reduce design is similar to most simple map-reduce programs with the exception that we return a list of lists in the first reduce phase:

$$M_1 : (K_1, V_1) \rightarrow list(K_2, list(V_2))$$
$$R_1 : (K_2, list(V_2)) \rightarrow (K_3, list(list(V_3)))$$
$$R_2 : (K_3, list(list(V_3)) \rightarrow list(K_4, V_4)$$

To gather the latency information between the nodes, each node created as output for unit of measurement the time frame, destination ip, start send time, latency measure from python-ping, and finally the total send time. Because each node is reachable by and from every other node, $n-1$ output files are created by a particular node, and $n(n-1)$ output files are created in total. The first map-reduce program maps each node's $n-1$ output files to a single file with each line being the combined reach-ability to every node at all time frames. It also puts each node in the same order on the line in which the nodes originally contacted the server so that a distance matrix can created.

The second map-reduce program is a reduce heavy program in which the mapping phase simply operates as a pass through to shuffle and sort the data. The reducer receives as input key each time frame and a list of n node reach-ability lists. Using the same order from the first map-reduce program, the reducer combines each time frame into an analyzable fully connected Di-graph of nodes. We tested both 1 reducer and n-reducers to see if there was a difference in order of output. This had no bearing on the final analysis and dynamic chosen number of reducers seems to offer the best analyzing time of 16.43 seconds with 46 reducers compared to a near linear upscale of around 600 seconds with only 1 reducer.

3.4 Optimizing Throughput

Throughput is a much more difficult measurement to try and optimize for. Unlike latency, it's measurement cannot be easily found by chaining machines with lower latency's. Instead, the best throughput is measured by the bottleneck link in which throughput is the lowest. When measuring throughput we log the transmission times and rates it takes for a node to transfer a 1 MB file from each node. Due to VM resource constraints, we decided to transfer 1MB files every 30 seconds. To achieve this, we deployed an nginx HTTP web server on each node, and each node made an HTTP request to the other nodes' servers to obtain the file resources. When the throughput data was logged, we gathered the time information by using a map-reduce program, just like we did in section 3.3.

4 Experimental results

To analyze our setup we ran four experiments of length 30 minutes, 60 minutes, and two 16 hour long experiments. The two experiments of note are the short 60 minutes and second longer 16 hour experiment[2]. A primary motivating factor in running the shorter experiments were for gathering a small proof of concept within Cloud Provider free tier limits as well as being able to determine if our results scaled with time.

4.1 Dataset

The datasets we collected were of two types, latency from each node to every other node in fully connected fashion, as well as throughput. Each dataset has the corresponding time frames in seconds or every 30 seconds.

4.2 60 Minute Experiment

Next Hop	Number	% of Total
0	587848	72.57
1	191046	23.58
2	29584	3.65
3	1522	0.18

Table 2: Observed Optimized Paths for 60 Minutes.

We collected 3600 time samples for the latency measurement and 120 time samples for the throughput measurement. When analyzing how this network holds up against previous overlays, we took a look at how many network patterns stayed the

[2]The first 16 hour long experiment we ran was not able to gather throughput information due to a storage issue from the free tier setup on our GCP and Azure nodes. The analyzation of latency, however, the difference in latency measures from either 16 hour experiment was not statistically significant and we conclude 1 is enough for our purposes.

Start	Intermediate Hops	Destination
Azure Brazil	AWS Sao Paulo → AWS Ohio → AWS Oregon	Azure Japan

Table 3: Interesting Route with Multiple Hops

same and how many deviated. One primary point of contention from (1), was that the most often optimal paths taken were of length 1 hop. We see a similar result in our dataset.

Looking at table 2, we see that roughly 27.4% of paths had optimzation opportunites, however, the majority are grouped around 1 hop. The next thing to look at is the path stability within these found paths, and whether a certain path holds for a substantially large given period of time. In the original paper we are trying to study, normally path optimization would be done in real time using mixed integer linear programming (MILP). For our purposes, we analyzed data after the fact. Therefore, to caclulate route stability we used an additive metric in our modified Floyd-Warhsall Algorithm that counted each time a path was chosen over the most direct route offered by BGP. Doing so allows us to determine which optimizations were the result of small fluctuations and those which were more consistent.

When looking at the optimization path scheme, we can see that almost always, the optimized path is from a region far away, and that it seeks to find the closest entry point into the cloud provider. It then uses the cloud provider's own optimized routes over the default BGP selected black-box route. This behavior is similar to that seen in (4) where the most optimal route is often an egress point in the cloud provider network that is closest to the destination Autonomous System (AS).

The next point of analysis is the actual routing paths themselves. One route that stood out to us in this smaller experiment was the following path in table 3.

This path is abnormal at 3 hops for a route and was chosen 17% of the time, which indicated that it is not a trivial path for this time segment. It also seems to suggest that going through AWS's cloud infrastructure, even for 3 hops, is more preferable than Azure infrastructure. When looking at this we realized that a potential flaw in our dataset could arise from our lack of US Azure nodes to choose from, therefore we corrected this in our two longer experiments. Even so, one would think the direct route vended between Azure's Brazil and Japan nodes would internally choose itself over other outside nodes.

A final interesting path to note from this dataset is found in table 4. This route seems to suggest that the most preferable route into African countries is from a central European provider (7).

Start	Intermediate Hops	Destination
GCP Asia East	Azure France	Azure South Africa

Table 4: Interesting route through Africa

Next Hop	Number	% of Total
0	10378709	70.3
1	3870327	26.24
2	493591	3.34
3	2970	0.02
4	3	-

Table 5: Observed Optimized Paths for 16 hours.

4.3 16 Hour Experiment

In this experiment we collected 115,200 time frames of latency information and 1920 time frames for throughput over two 16 hour runs. We also added a US Azure node in the US East 1 region to compensate for the previous lack of US Azure nodes[3].

A primary benefit of running this longer trial is the ability to see if our previous results scale, and when looking at the optimal route hops in table 5, they do. We see both the default optimal path and 1 hop paths vary by just 2-3% and the rest are pretty much the same. Therefore, we can easily say that the pattern originally observed of 1 intermediary routing hop remains true. We also see that a similar trend arises when looking at the stability of the routes chosen. We get a couple of charts when graphing the resulting data:

To look at the most optimal gains we have organized the data by number of times the path has been optimized in figure 1. Essentially, we can separate out fluctuation in the internet at large with the definite stable optimizations. With that being said, the most stable paths realistically only see an average latency optimization of around 9 to 12 percent with the actual speed being about 2 to 3 ms in gain.

Looking at the actual paths in table 6, we see some continuing patterns. Often the most preferrable optimization seeks to stay either within the same cloud provider as long as possible or egress into the destination cloud provider as soon as pos-

[3]As an administrative note, all of our AWS nodes were migrated to a different account at this point on a paid tier for higher throughput limits

Path	Percent of Time Optimized	Avg. Latency (ms)
AWS Oregon → AWS Ohio → GCP Europe North	99.2	1.92
GCP Europe North→ AWS Ohio → AWS Oregon	99.2	1.95
Azure France Central → AWS Ohio → AWS Oregon	97.2	2.18
AWS Oregon → AWS Ohio → Azure France Central	96.1	2.19
Azure Japan → AWS Oregon → AWS Ohio	91.1	2.55
AWS Ohio → AWS Oregon → Azure Japan	90.9	2.56
AWS Oregon → AWS Ohio → Azure Brazil South	86.6	1.84
Azure Brazil South → AWS Ohio → AWS Oregon	85.8	1.84

Table 6: Routes with greater than 10% average optimized latency and greater than 75% Stability

sible. This results in a west-bound path originating in a region like Ohio travelling within the same AWS cloud inside the US until offloading into a region outside, say Japan. This indicates that the common sense of Cloud infrastructure being faster and superior to regular BGP routing. However, the fact that our nodes are being presented with optimization opportunities inside the cloud leads to a preliminary conclusion, the cloud providers are allowing BGP to handle egress traffic with no optimization within.

We noted an interesting path from the first experiment, Route 1. This path is interesting because it chose a path from Azure to AWS and then back to Azure, which is counter to the previous posit that each cloud infrastructure is superior to BGP as they have more modular control. In the shorter trial runs, Route 1 had an optimization of around 17 percent, however, in this much longer trial it was only optimized 1412 times which translates to around 2.45 percent of the time. It also only optimized latency on average by about 0.665 percent. However, when we take into account the the new Azure US node we see this route appear in table 7.

Start	Intermediate Hops	Destination
Azure Brazil	Azure US East	Azure Japan

Table 7: Interesting Route with multiple hops.

This route was chosen a much more significant amount at 37,374 times, or about 65 percent. This shows that cloud infrastructure does offer faster routes, but suggests that due to the minimal performance gain may let BGP handle route selection. A final intersting thing to note about the latency optimization portion of our study is that of the stable paths, some of the routes may only have had a 2ms latency gain, however, that gain accounted for nearly 25 percent improvement of the paths. A popular path chosen nearly 100 percent of the time was from Azure's US east node through AWS Ohio and into AWS's Oregon node. It was optimized on average by about 24.7 percent or about 2 ms in latency. With regards to return paths, we found

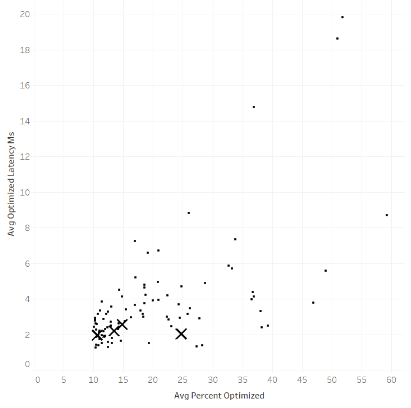

Figure 1: Top 7 Paths with >10% Optimization

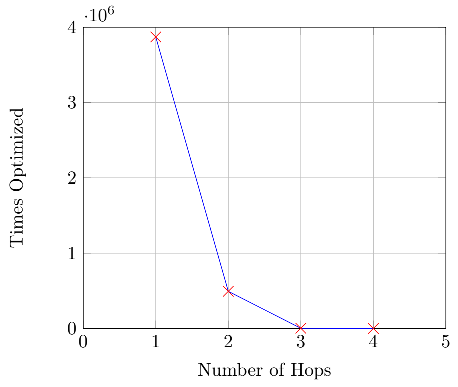

Figure 2: Number of route hops vs times optimized

that on average, greater than 90% of paths that found a better route also found the same return route with minor variations in the latency metrics. This helps us better understand the relationship with stable paths.

To greater understand route stability, we graphed the times a path was optimized by the number of hops it had. It is quite clear from figure 2 that the most stable and often chosen routes are very greatly shifted towards 1 hop. We also graphed the Average latency by times optimized in the histogram on figure 3, which clearly shows that high stability route's had low gains, and only the extremely fleeting routes had large margins of gain.

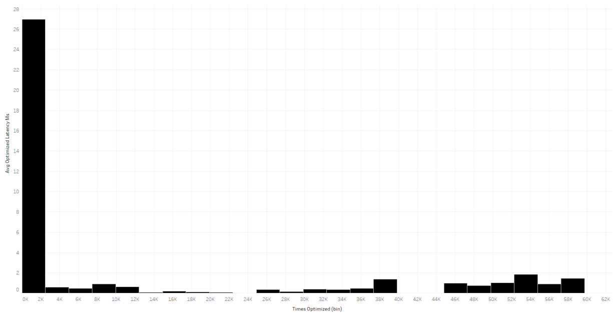

Figure 3: Histogram of Optimization Time to Avg Latency

Remark

A note on outliers: We had some paths with over 2000 percent optimization gains. However, these gains were only ever seen once. In a specific instance the original latency was 24 ms and the optimized latency was 0.9ms. The original and optimal paths were chosen from between Asian servers. It is reasonable to conclude that these paths were a result from momentary fluctuations in the internet and that there is no feasible way to get a reliable gain from these fleeting moments of arbitrage.

4.4 Throughput Analysis

From the original RON paper, due to the time-varying and unpredictable nature of available bandwidth on Internet paths, optimizing throughput is not a higher priority than optimizing latency and loss rate. Instead, they strive to find a path that avoids an insufficient throughput than optimizing to eliminate minor throughput differences.

In the 16-hour experiment, we are committed to discovering the current state of the throughput under VMs transferring data in today's cloud era compared to two decades ago. We collected 1920 time frames for throughput and found that it has a similar trend as the data with latency, e.g., there are 63% paths that are already optimized, the path that takes one hop has around 87% of the optimized paths, and the remaining has 12.5%. We can see that more paths are optimized with more than one hop in the topology compared to the latency part. A reasonable explanation is that the underlying Internet throughput is more likely affected by the unstable network environment. At first glance, the RON-style architecture seems very helpful when optimizing the throughput. However, there is only around 3% of paths that at least doubled the throughput from our dataset. In other words, most optimized paths are relatively minor improvements over the original paths, often in unstable network conditions. Moreover, about 5% of the paths get double the throughput optimization in the original RON network evaluation. Therefore we conclude that RON-style architecture is less effective at optimizing throughput in current network conditions than two decades ago.

Looking at throughput analysis from the perspective of cloud providers also reveals some interesting phenomena. When measuring throughput between VMs from the same cloud provider, we found that the number of optimized paths between VMs from AWS was much higher than between GCP and Azure. In our 16-hour experiment, around 35000 AWS paths get optimized, while both GCP and Azure have about 4000 paths that get optimized. According to the discovery in the Skyplane paper, the egress intra-cloud throughput is consistently higher than inter-cloud in both GCP and Azure but keeping the same in AWS(4). This phenomenon indicates that a cloud-aware overlay network is necessary and may perform better than the naive RON architecture.

Remark

We found an interesting observation when analyzing the latency and throughput; They follow the same trend in most cases, except for the relationship between average optimized values and the number of hops. In Figure 4, a lower latency can be found even with a more significant number of hops, while in the throughput part, the gains decrease with the number of hops. This observation implies that one intermediate node is sufficient for maximizing throughput gains and that the conventional wisdom of smaller gains with higher overhead remains true.

5 Discussion

5.1 Evaluation with existing RON study

The original goal of RON encompassed many different aspects such as overlay configurations, outage detection, routing around failures, policy routing, application level routing, and more (1). Our study was primarily concerned with determining the effectiveness in an overlay network to gain substantial routing speed boosts. With this goal, however, we were also able to determine if there was any significant downtime in modern internet infrastructure. Using a counter within our map-reduce code, we noted 0 downtime in ping availability (when our nodes were working) for every trial run. As one might expect, the outages of before are not nearly as constant and we would have to run our program for a substantially longer time to encapsulate the frequency of network outages today. Based on the tendency of network connectivity improving over time, the resilience portion of overlay networks should not be the primary concern when choosing to setup this style of virtual network.

However, the aspects of application-specific routing and policy routing remain important considerations. In recent years, many studies have focused on virtual networks that optimize application-specific

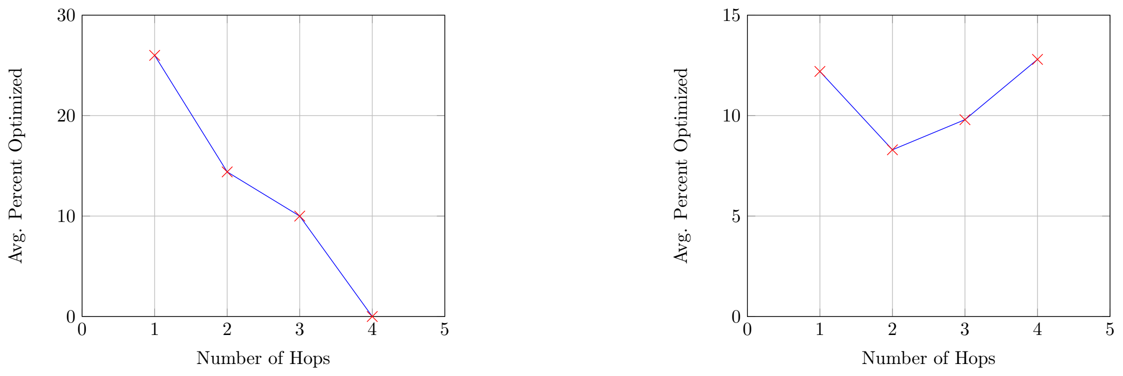

Figure 4: Throughput vs. Latency: The average value optimized with the number of hops

network performance and implement policy routing. For example, Sun et al. proposed a Quality of Service (QoS) guaranteed routing mechanism for software-defined networks (12). Integrating RON with such underlying routing networks could provide path selection backup and reliability.

Furthermore, RON can support decentralized applications that rely on peer-to-peer communication between nodes. Huang et al. proposed a new mechanism for blockchain propagation in edge-enabled smart cities using RON (3). The proposed mechanism combines RON with edge computing to enhance scalability, reliability, and security of blockchain propagation in smart cities.

In summary, while the original goal of RON encompassed multiple aspects of overlay networks, we focused primarily on the effectiveness of RON in improving routing speed. However, RON still has potential applications in virtual networks for application-specific routing and policy routing, as well as supporting decentralized applications such as blockchain propagation in edge-enabled smart cities.

6 Related Work

There is a wealth of literature exploring the optimization of network performance through the use of overlay networks, including the original RON paper(1). One notable work in this area is the ViNe (2) project, which proposed a new architecture for virtual network embeddings that takes into account network resources such as bandwidth and latency.

Another important area of related work dealing with overlay networks was performed by Akamai (11). In this analytical paper, the CDN responsible for a large portion of internet traffic discuss in great detail the various types of overlay networks including three key types: caching, routing, and a more novel security network. Their conclusion is that the power of overlay networks in comparison with the vanilla services is next evolution of the Internet.

In recent years, there has been growing interest in using cloud-aware overlay technologies to address the specific challenges cloud computing environments pose. One example of this is the Skyplane project (4). In this recent work, the factor of transferring buck data between cloud resources is focused. One novel point of view is they manage the tradeoff between performance and cost when measuring throughput because of the egress fees associated with network bandwidth.

In addition to these works, there have been numerous studies focused on the optimization of network performance through the use of various techniques, including software-defined networking (SDN) (5) and network function virtualization (NFV) (6). These studies have demonstrated the potential of these technologies to improve network performance, and have highlighted the importance of continued research in this area.

7 Conclusion

In conclusion, our study aimed to evaluate the effectiveness of RON-style overlay networks in optimizing network latency and throughput in today's cloud era. Through our experiments, we found that RON-style architecture can still provide latency improvements in routing paths, however, their gain is minimal in comparison to the original RON paper. Throughput optimization was even less effective, with only a small percentage of paths showing significant gains. Furthermore, we observed that latency and throughput trends follow a similar pattern, except for the relationship between average optimized values and the number of hops. This observation confirms that one hop is sufficient in RON architecture. While our study focused on evaluating the performance of RON-style overlay networks, we recognize that cloud computing is becoming increasingly prevalent. As such, cloud-aware overlay technologies should be emphasized in future research. Such technologies should aim to optimize not only latency and throughput but also consider the unique challenges posed by cloud environments, such as increased network complexity, virtualization, and multi-tenancy. Overall, our findings demonstrate the potential benefits of RON-style overlay networks in optimizing network performance. However, as network architectures continue to evolve in response to the demands of cloud computing, further research is needed to develop overlay technologies that are better suited to this new environment.

8 References

[1] Andersen, D., Balakrishnan, H., Kaashoek, F., and Morris, R. Resilient overlay networks. In *Proceedings of the eighteenth ACM symposium on Operating systems principles* (2001), pp. 131–145.

[2] Fischer, A., Botero, J. F., Beck, M. T., de Meer, H., and Hesselbach, X. Virtual network embedding: A survey. *IEEE Communications Surveys Tutorials 15*, 4 (2013), 1888–1906.

[3] Huang, J., Tan, L., Li, W., and Yu, K. Ron-enhanced blockchain propagation mechanism for edge-enabled smart cities. *Journal of Information Security and Applications 61* (2021), 102936.

[4] Jain, P., Kumar, S., Wooders, S., Patil, S. G., Gonzalez, J. E., and Stoica, I. Skyplane: Optimizing transfer cost and throughput using cloud-aware overlays. *arXiv preprint arXiv:2210.07259* (2022).

[5] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. Openflow: enabling innovation in campus networks. In *ACM SIGCOMM Computer Communication Review* (2008), vol. 38, pp. 69–74.

[6] Mijumbi, R., Serrat, J., Gorricho, J.-L., Bouten, N., Turck, F. D., and Boutaba, R. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys &amp Tutorials 18*, 1 (2016), 236–262.

[7] Nsoh, J. Localization of internet traffic: A solution to africa's digital divide.

[8] Savage, S., Collins, A., Hoffman, E., Snell, J., and Anderson, T. The end-to-end effects of internet path selection. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (New York, NY, USA, 1999), SIGCOMM '99, Association for Computing Machinery, p. 289–299.

[9] Shah, S. A. R., Waqas, A., Kim, M.-H., Kim, T.-H., Yoon, H., and Noh, S.-Y. Benchmarking and performance evaluations on various configurations of virtual machine and containers for cloud-based scientific workloads. *Applied Sciences 11*, 3 (2021).

[10] Shavitt, Y., and Shir, E. Dimes: Let the internet measure itself. *SIGCOMM Comput. Commun. Rev. 35*, 5 (oct 2005), 71–74.

[11] Sitaraman, R., Kasbekar, M., Lichtenstein, W., and Jain, M. *Overlay Networks: An Akamai Perspective*. 10 2014, pp. 305–328.

[12] Sun, W., Wang, Z., and Zhang, G. A qos-guaranteed intelligent routing mechanism in software-defined networks. *Computer Networks 185* (2021), 107709.