Large-scale Network Protocol Emulation on Commodity Cloud

Anirup Dutta
University of Houston
adutta2@uh.edu

Omprakash Gnawali
University of Houston
gnawali@cs.uh.edu

Abstract

Network emulation allows us to evaluate network protocol implementations, typically in higher fidelity than simulations. This advantage comes at a cost. Emulation often requires much larger IO or computational resources than simulations. As a result, it is common to see some research projects doing simulations with up to hundred thousand nodes while emulations typically scale up to a few hundred nodes. In this paper, we present CloudNet, a network protocol emulation platform that leverages the commodity cloud computing service to scale emulations to thousands of nodes. CloudNet uses a light-weight virtualization technique called LXC containers to emulate a single node. The network protocol code and the protocol state for each node is maintained in its respective container. CloudNet then uses properties of the network topology to determine where to place these containers among many physical machines researchers might rent on the cloud service. CloudNet's careful mapping of nodes to the containers makes network performance more predictable and suitable for emulation even on a shared commodity cloud, which were previously thought to be unsuitable for serious network emulation. Through extensive experiments, we establish that CloudNet is scalable to thousand-node networks while providing accurate emulation results.

I. Introduction

The networking research community has a long history of building tools and methodologies to evaluate network protocols. Theoretical analysis allows us to understand the protocols at the algorithmic level. We can implement the key ideas of the protocol in a simulator and study the protocol injecting a bit of realism, for example, wireless link model and protocol state machines [1]–[3].

Emulation takes this study of protocols a step further. We can evaluate the actual implementation of the protocol, often in the same runtime environment as the deployment target. Emulation often is resource-intensive. An emulator must evaluate the entire implementation, not just the core idea behind the protocol. Despite this disadvantage, emulations are widely used because they produce results that are similar to the results on the testbeds, and the target deployments.

Recent advances in node virtualization makes it possible to emulate a large number of nodes in a single physical machine. We can run the protocol code for each node in its own virtual machine and create a network of emulated virtual nodes. We can thus emulate several nodes in a single machine thereby enabling emulation of a large network on a small network of physical machines. This technique however does not scale because a physical machine cannot host more than a few tens of virtual machines while providing acceptable performance.

Light-weight virtualization can help scale these emulations to very large networks. In light-weight virtualization, *containers* are the unit of virtualization. Each physical machine can host several hundred *containers*. Mininet [4], [5] is a recent effort that uses this approach and shows how to emulate a network of several hundred nodes in a single machine. However, this solution does not scale beyond the number of containers that can be run in a single machine.

In this paper, we present CloudNet, a network emulator that can emulate networks with thousands of nodes. Our work leverages two technology trends to make this possible. First, CloudNet uses machines provided by the commodity cloud service. The researchers can perform experiments on a large number of machines for the small expense of renting machines from cloud service providers such as Amazon. Second, CloudNet leverages light-weight virtualization technique to further scale the size of the emulation.

The lack of consistent performance in the shared cloud environment and the challenge it presents to network experiments have been identified as the key challenge in building an emulator in the Cloud. In a shared cloud environment such as Amazon EC2, it is possible for one of our emulation instances to share the same physical host with an instance running a webserver of a popular website. When the other instances in the same physical machine demand more resources, naturally fewer resources are available to our emulation instances. To address this challenge, CloudNet only uses Cluster instances, which have ample computation and communication resources and come with loose guarantees about resource allocation. This is in contrast to earlier work that used micro or small instances, which make large emulation affordable but come with the inconsistent and unpredictable performance.

While using Cluster instances with ample resources addresses the performance consistency to some extent, it becomes prohibitively expensive to emulate large networks. CloudNet uses light-weight virtualization in each Cluster instance to economically scale the emulation while providing more consistent networking performance. Thus, CloudNet becomes as economical as emulation with micro or small instances but with more consistent performance.

In this paper, we make these contributions:

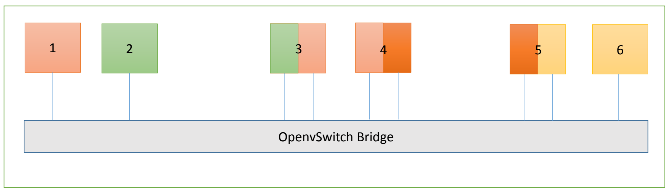


Fig. 1: CloudNet emulation of a 6-node network. The different colors represent different networks. If we wish to send packets from 2 to 6 we need to do it via 3,4 and 5.

- We show how to achieve a partially consistent networking performance for network emulation in a shared cloud environment like Amazon EC2.
- We present the design and implementation of an emulator that economically scales to thousands of nodes.
- We present experiments to evaluate the correctness of the such large scale emulators. We use those experiments to evaluate CloudNet.
- We present, to our knowledge, the first emulation of DTN protocol on a 1000-node network as a case study that exercises the capabilities of CloudNet.

II. CloudNet Design

In this section, we present the design of CloudNet, which meets these design goals: scaling to thousand-node networks and beyond; low cost platform; and correctness of emulation results.

A. Network Nodes

CloudNet uses light-weight virtualization mechanism called Linux Containers where each container represents a single node. The network protocol code for a node runs in its respective container. We used LXC for launching the containers in the machine. LXC uses Linux kernel mechanisms to provide lightweight virtual system with full resource isolation and resource control for running application or a system [6], [7]. Due to the shared kernel between the containers, it is possible to run hundreds of LXC containers, i.e., emulated nodes, in a single machine while ensuring CPU cycle and memory isolation across the containers. Thus, because of their light-weight nature with sufficient resource isolation, LXC containers are a good choice in node representation in our goal of designing a scalable emulator.

B. Network Connectivity

Each LXC container represents a node in the network we wish to emulate. The network connectivity between the containers represent the link between the nodes.

We configure LXC container with at least two network interfaces: one connected to the default LXC bridge and the other connected to the Open vSwitch bridge. Open vSwitch [8] is a software switch which can be used to connect virtual machines. Open vSwitch supports many standard protocols and management interfaces. It allows QoS for each virtual machine interface. It supports openflow protocol as well as many tunneling protocols like GRE, IPSEC, etc. The default LXC bridge and the Open vSwitch bridge are not connected. We NAT out of the box for the LXC containers through the LXC bridge and we create the logical network for emulation experiments using Open vSwitch. CloudNet provides two ways to define the topology of the emulated network.

1) Creating Topology Using OpenFlow: This approach to defining emulation topology using OpenFlow [9] is borrowed from Mininet. When there is a link between the nodes in the topology we wish to emulate, the controller allows forwarding between those nodes thereby *creating* an emulated link. We attach an OpenFlow controller to Open vSwitch with custom rules on how Open vSwitch must forward the packets.

2) Creating Topology Using Subnets: We can use subnets to provide or limit connectivity between different nodes in the emulated networks. We use the LXC bridge as the default gateway in our containers. We setup routing rules such that the packets meant for destination addresses belonging to the same subnets as the network interfaces of the containers will go through the Open vSwitch bridge. Any packet meant for destinations other than those belonging to the same subnets as container's interfaces are dropped. Thus, if we need to provide a link between two nodes, together with our routing rules, we assign them ip addresses in the same subnet.

The problem of determining the optimal number of subnets required to emulate a large and complex network and the membership of each subnet is equivalent to finding maximal cliques in a graph. Every maximal clique constitutes a subnet. The nodes belonging to the same clique are members of the same subnet. If a node is a member of multiple cliques, then it will be member of multiple subnets, which also means that it will have multiple virtual interfaces connected to Open vSwitch. We find the maximal cliques in a graph using the Bron-Kerbosch algorithm [10] and we used a software library called ipgraph [11] which implements the Bron-Kerbosch algorithm.

C. Link Attributes

In many network emulations, the topology may have links with different loss rates or delays. For example, we may want to emulate links with propagation delay of several seconds in a DTN emulation. We use Netem to constrain the bandwidth and delay of a link.

D. Distributing Nodes Across Instances

Emulating a network with thousands of nodes will require thousands of containers, which will not fit in a single machine. Now, we describe how CloudNet distributes the containers across multiple machines while providing consistent network performance between the nodes.

We run Open vSwitch in each of the instances. All the LXC containers in each instance is connected to Open vSwitch bridges on that instance. We used GRE (Generic Routing Encapsulation) tunnels to connect the Open vSwitch bridges on the different instances. Connecting the bridges using GRE tunnels in Amazon EC2 requires using public IP addresses for the instances.[1] CloudNet requires only a small number of Cluster Instances to emulate even a large network, and only the Instance (not individual nodes) needs public IP address. We use NTP to synchronize the time across the instances.

E. Choice of Instance Type

Previous work has shown that virtualization in Amazon EC2 causes highly unreliable network performance [12]. However in such work, researchers used small instances. Such inconsistent performance would not meet CloudNet's requirement of *correct* emulation. CloudNet uses a smaller number of instances with more resources in Amazon EC2, then uses LXC containers to decrease the number of instances required to emulate a large network. For example, cluster Instances have 60.5 GB of RAM with 88 ECUs (equivalent to 16 cores). These instances can be launched together in a single placement group to assure high and consistent network performance between the instances. To emulate a 1000 node network, with 200 containers per node, we only need five instances. Because we control the execution environment of the containers in each instance, we can provide more consistent network performance across the emulated nodes compared to running the nodes directly on top of Amazon EC2 and leaving it to Amazon to provision the network. Furthermore, there is high and consistent network connectivity between Cluster instances. Thus, the emulated nodes within these instances achieve consistent network performance.

F. Difference from Mininet

CloudNet borrows the technique of using lightweight virtualization to emulate a node from Mininet. However, Mininet can run on a single machine limiting its scalability to the number of containers one machine can host. CloudNet, on the other hand, carefully places the containers on different instances using the maximal clique organization, and connecting them with the topology defined by the user, thereby scaling beyond what a single machine can run.

III. Evaluation

In this section, we study how CloudNet satisfies its design goals. We use tools such as iperf to validate the correctness of CloudNet and DTN [13] as a case study to evaluate the usefulness of CloudNet.

A. Network Performance in Amazon EC2

Correctness is the most important requirement of any network emulator. To evaluate CloudNet's correctness, we create a network topology using CloudNet and run iperf over those topologies to measure bandwidth and jitter. If CloudNet emulation is correct and consistent, we expect the results of measurements in the experiment to match the specified properties of the topology. Otherwise, CloudNet results are not correct. In addition, we compare the correctness of emulation done on CloudNet vs directly on top of Amazon EC2. A network emulation directly on Amazon EC2 uses an instance to represent a node and uses virtual interfaces to connect the

[1]One possible reason is that the visibility of the private IP addresses is limited to a certain portion of the EC2 cloud.

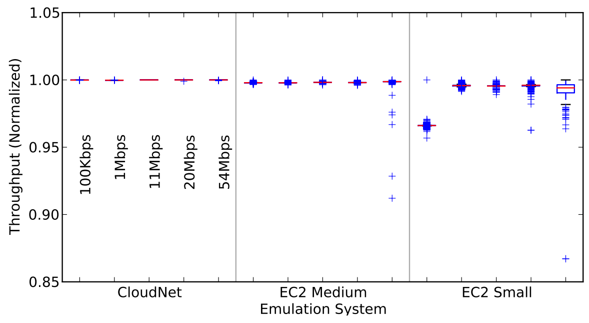


Fig. 2: Boxplot of iperf throughput normalized by the nominal throughput for the setup varying from 100 Kbps to 54 Mbps.

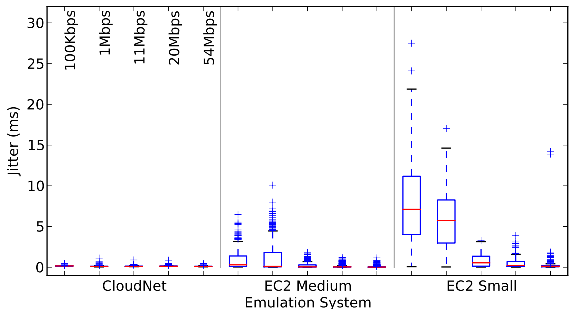


Fig. 3: Boxplot of jitter in path latency measured during experiments for setups varying from 100 Kbps to 54 Mbps.

instances to create the user specified network topology. This experiment will help us understand if CloudNet provides any advantage over running emulation directly on top of Amazon EC2. In each experiment, we created a small network of 10 nodes over 9 hops and made 50 bandwidth and jitter measurements using iperf. The experiments were performed multiple times a day over five days (to understand if performance consistency holds at different times of the day and different days of the week).

Figure 2 shows throughput achieved by iperf when doing emulation with CloudNet versus using Amazon EC2 to do those emulations. The first observation is that iperf achieved constant normalized throughput across multiple experiments with 50 iterations each for all throughput settings. Thus, the measured throughput matches the ground truth. The second observation is that emulation directly on top of Amazon EC2 medium instances are less consistent (despite much higher instance rental cost as we show later). The results are worse (and unacceptable) with Amazon EC2 small instances. Thus, we conclude that throughput available to emulation on CloudNet is consistent and correct.

Figure 3 shows the jitter in path latency measured during the experiments. The topologies used in these experiments were specified to have no jitter. Thus any considerable jitter could lead to inaccurate emulation results. We observe that the jitter with CloudNet is smaller than emulation that uses Amazon EC2 Medium instances, and significantly smaller than the emulation that uses Amazon EC2 Small instances. Thus,

	Normal	Uniform	Pareto	Pareto Normal
1 Hop - EMD	0.6	0.08	0.36	0.04
Error (%)	0.6	0.08	0.36	0.04
9 Hops - EMD	7.29	6.90	7.62	7.8
Error (%)	0.81	0.76	0.84	0.86

TABLE I: Earth movers distance between theoretical and observed values.

Distribution	1 Hop Delay (sec)	9 Hop Delay (sec)	9 Hop Delay / 1 Hop Delay	Error (%)
Normal	0.0998	0.9045	9.06	0.66
Uniform	0.1002	0.9044	9.02	0.22
Pareto	0.0994	0.8984	9.03	0.33
Pareto Normal	0.0992	0.9004	9.07	0.77

TABLE II: Latency analysis for the different distributions.

we find that latency measurements on CloudNet emulations are close to the ground truth.

B. Correctness of Network Impairments

CloudNet can configure links to have certain delays as required by emulations. For example, create a network where certain links have a latency of 1s. In this section, we describe experiments to test if these impairments are correctly emulated in CloudNet.

In these experiments, we configure links with 100ms latency and 20ms jitter that have normal, pareto, and uniform distributions. We use normal and pareto distributions provided in Netem but inject our own uniform distribution table to Netem kernel module.

a) One-hop Experiments: We sent 2000 UDP packets between 2 nodes and measured the packet latency between the nodes. If CloudNet correctly emulates the delays, we expect the observed latency to be the same as the intended (i.e., generated) latency. Figure 4 shows the results from this experiment which compares observed, generated and theoretical latencies. We find that the plots of the observed delay values confirm to the type of distribution used for introducing delay with less than 0.6 percent error as shown in Table I.

b) Multi-hop Experiments: We use CloudNet to setup a 10 node, i.e., 9-hop network. We sent UDP packets between the nodes at the two ends of the topology. If CloudNet works correctly, we expect the packets to take an average of 900ms to travel between the nodes. Figure 5 shows that that the observed latency closely matches the generated latency. Table I shows that the error is 0.76%-0.86%.

c) Correctness of Latency Distributions and Averages: Next we evaluate if the distribution of latency generated by CloudNet is correct. We configure the nodes to generate latency with normal distribution in the 9-hop topology. The sum of independent normal distributions is also a normal distribution: we expect the 9-hop latency to be normally distributed. We performed QQ tests on the observed delay values for the 1 hop and 9 hop experiments to determine if the delays were distributed normally. We performed the normality test on the delay values. We found the P value for 1 hop is 0.22 and for 9 hops it is 0.23. The mean in case of the 9 hop network should be equal to 9 times the mean for 1 hop network. We found that the measured one-hop to 9-hop latency confirm to this ratio with less than 0.77% error as shown in Table II.

Through these experiments, we find that CloudNet can provide controlled and correct link properties to network emulation.

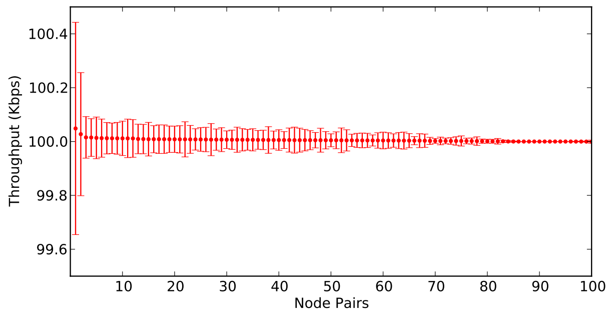


Fig. 6: Iperf throughput results with maximum bandwidth limited to 100kbps on the 1000 node network. The node pairs have been sorted by the mean of the throughput values.

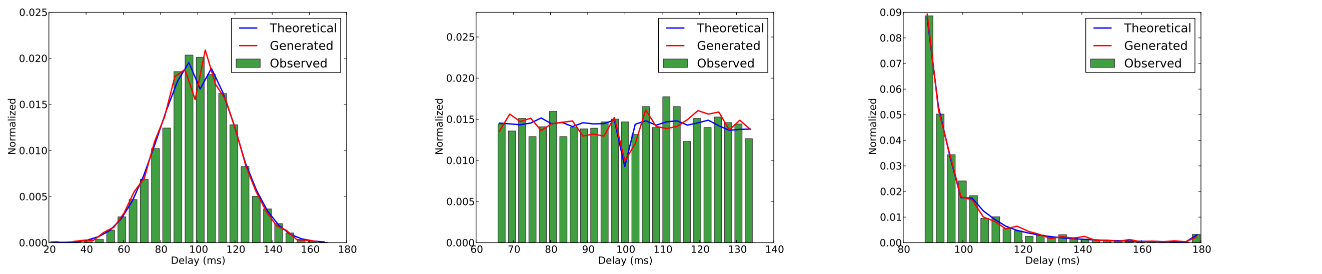
C. Scalability

To test the scalability of CloudNet, we generate a 1000 node topology with minimum, average, and maximum node degree of 1, 6.4, and 13 respectively. We setup this topology using the technique described in Section II. Every instance contained 200 containers. So we used 5 cluster instances. We randomly picked 100 pair of random nodes resulting in maximum and minimum path lengths of 22-hop and 1-hop respectively. We then used iperf to send UDP packets between all the pair of nodes simultaneously with each flow with a cap of 100kpps. The results from the tests are shown in the Figure 6. The net throughput achieved per flow is close to the expected value of 100kbps.

To test if the latencies stay close to the ground truth even when the user topologies add delays, we emulate a 1000 node network with a link latency of 1 second on each link. Table III shows the mean RTT and standard deviation in delay between random pairs of nodes at 1-4 hops. There is a small base latency between the node-pairs: this is a system overhead of CloudNet. We observe that the overhead increases as expected with the number of hops but stays small.

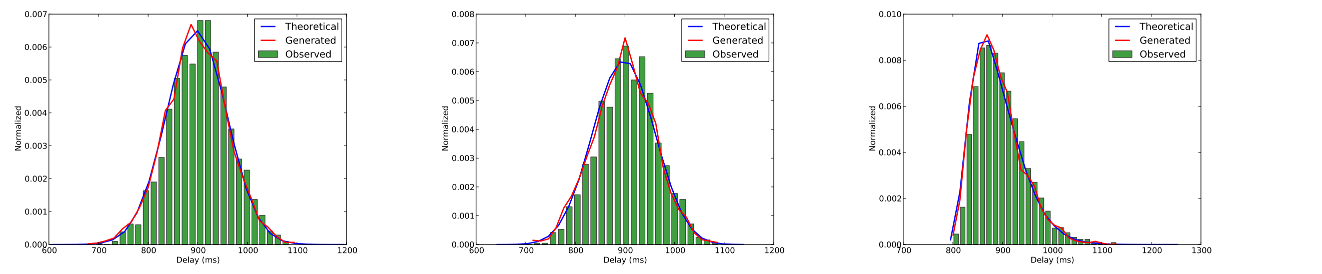
To compare Cloudnet's scalability with Mininet, we perform a 1000 node experiment on Mininet on a 8 core server with 48 GB of RAM. This experiment stressed Mininet in two ways. First, the experiment setup took more than an hour and the machine became unresponsive. Second, we were not able to use ping to send data between the nodes: the system dropped the packets because this experiment was overloading the server. This result should not come as a surprise: Mininet uses a single machine architecture and is designed to work with a few hundred nodes.

Through these experiments, we found that CloudNet can scale to 1000-node topologies with predictable and expected performance.



Link Latency - Normal Distribution Link Latency - Uniform Distribution Link Latency - Pareto Distribution

Fig. 4: Theoretical vs Generated vs Observed delays in a 1 hop network with link latency based on given distributions.



Link Latency - Normal Distribution Link Latency - Uniform Distribution Link Latency - Pareto Distribution

Fig. 5: Theoretical vs Generated vs Observed delays in a 9-hop network with link latency based on given distributions.

Hops	RTT	
	Mean (ms)	σ
1	2000.98	2.28
2	4002.16	5.89
3	6005.70	9.20
4	8006.28	10.64

TABLE III: RTT for paths of different hops in a 1000 node network. Each link was configured with a delay of 1s.

D. Case Study

As a case study of a realistic use of CloudNet, we perform large scale experiments with DTN2. We randomly chose 100 pair of nodes from a multi-hop 1000 node topology. We use dtnperf and dtping to send data between the node pairs. The one hop goodput obtained is similar to what was reported in prior work [14] when the authors did experiments with physical machines. The multi-hop goodputs, as expected, decrease with an increase in path length (Figure 7). Figure 8 shows the dtnping packet latency between the node pairs. The high variation in latency is due to bundle queuing in the intermediate hops as different flows cross each other. Thus, the throughput and latencies we obtained in our experiment match the expected output and also what has been previously reported.

Table IV compares the cost of performing a similar experiment using small or medium instances separately for each node. The cost analysis affirms that CloudNet provides a cheaper alternative to performing emulation directly on top of EC2 instances, while providing correct results and scaling beyond state-of-the art solutions such as Mininet.

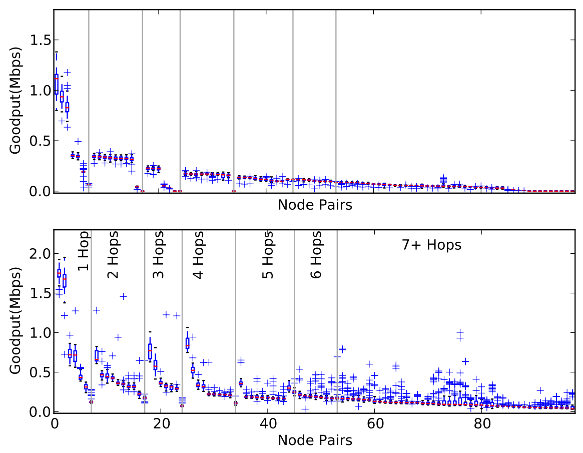


Fig. 7: Throughput achieved between node pairs using DTN when each link has 1 s latency (top) and without any network impairment (bottom). Payload size is 100KB.

IV. Related Work

In this section, we overview the types of tools the networking community has built to evaluate network protocols.

Link Emulation: Single link emulation can be done on hardware (using channel emulators) or on software (using tools such as Netem). Prior work has shown that when correctly configured, Netem provides a realistic estimation of impaired network conditions and is sufficient for most networking experiments [15].

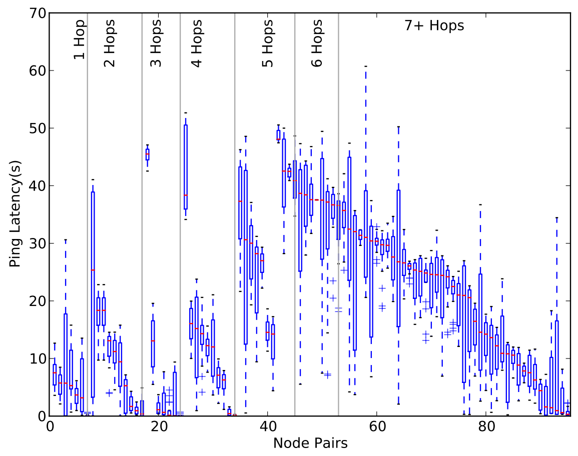


Fig. 8: End-to-End latency for each pair of node using dtnping.

Network Emulation: Mininet [4] [5] uses light-weight virtualization by isolating certain OS resources, thus allowing emulation of large networks in a single machine. However, scalability becomes an issue when we want to emulate larger networks than can be tested in a single physical machine. Emulab [16] light-weight virtualization technique, FreeBSD jails, to setup multiple virtual interfaces per process group, similar to Mininet and CloudNet. CloudNet provides better resource isolation across the emulated nodes than Emulab and shows how we can use it on the commodity clouds. There is some prior work in data centers to optimize VM placement and routing [17]. CloudNet uses the concept of placement groups in Amazon EC2 where the virtual machines are placed as close to each other so that we can efficiently use the resources.

Network Emulation Timing: Time-Warp [18] explores the possibility of using time dilation in network emulation experiments. Future version of CloudNet may use this technique to offer added consistency in performance for emulations that requires very high-bandwidth. Slicetime is another effort to provide scalable and accurate network emulation [19]. Slicetime makes the simulations independent of real time constraint thus allowing simulation of complex and high performance networks when we have limited physical resources.

V. Discussion

The experiments in this paper showed emulation of homogeneous network. CloudNet design accommodates emulation of heterogeneous networks. The containers within an instance share the host operating system. However, emulated nodes of different platforms can be placed on different instances running different OS. Heterogeneity in CPU and memory resources can be implemented directly with the container mechanism.

A disadvantage of our setup is we cannot assure complete fidelity while using Amazon EC2 since we have no control over the infrastructure. However the containers that we launch within an Amazon EC2 instance are completely controllable. We can control the amount of memory, CPU and bandwidth available to them. Thus we turn an uncontrolled system into a system that provides predictable performance.

	CloudNet	Medium	Small	EC2 Region
Unit Price	$2.4	$0.12	$0.06	U.S. East(N. Virginia)
Number of Instances	5	1000	1000	U.S. East(N. Virginia)
Total Price	$12	$120	$60	U.S. East(N. Virginia)

TABLE IV: Cost of 1-hr emulation of 1000-node network

VI. Conclusions

In this paper, we presented CloudNet, which is an emulator that runs on cloud services such as Amazon EC2. CloudNet uses techniques to achieve consistent network performance despite having little control over the EC2 instrastructure. Through extensive experiments, we showed that the emulator replicates the results from experiments that used physical machines. We presented the results from emulation of DTN on a 1000-node network. Thus, we showed that CloudNet meets the design requirements of scalability and affordability. As future work, we will implement CloudNet on other cloud service providers.

References

[1] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: Accurate and scalable simulation of entire tinyos applications," in *ACM SenSys*. ACM, 2003, pp. 126–137.

[2] ns2, "The network simulator," 1989a.

[3] G. Pongor, "Omnet: Objective modular network testbed," in *MASCOTS*, 1993.

[4] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *HotNets*, 2010, p. 19.

[5] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible network experiments using container-based emulation," in *CoNext*, 2012.

[6] L. Organization, "lxc linux containers," http://lxc.sourceforge.net/.

[7] S. Graber, "lxc containers in ubuntu," https://www.stgraber.org/2012/05/04/lxc-in-ubuntu-12-04-lts/.

[8] B. Pfaff, J. Pettit, T. Koponen, K. Amidon, M. Casado, and S. Shenker, "Extending networking into the virtualization layer," *Proc. HotNets (October 2009)*, 2009.

[9] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM CCR*, vol. 38, no. 2, 2008.

[10] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," *CACM*, vol. 16, no. 9, pp. 575–577, 1973.

[11] G. Csardi and T. Nepusz, "The igraph software package for complex network research," *InterJournal, Complex Systems*, vol. 1695, 2006.

[12] G. Wang and T. E. Ng, "The impact of virtualization on network performance of amazon ec2 data center," in *INFOCOM*. IEEE, 2010.

[13] K. Fall, "A delay-tolerant network architecture for challenged internets," in *SIGCOMM*. ACM, 2003.

[14] W.-B. Pottner, "An empirical performance comparision of dtn bundle protocol implementations," http://www.ibr.cs.tu-bs.de/papers/poettner-tr201108.pdf.

[15] E. Kissel and M. Swany, "Validating linux network emulation," http://damsl.cs.indiana.edu/projects/phoebus/netem_validate.pdf.

[16] M. Hibler, R. Ricci, L. Stoller, J. Duerig, S. Guruprasad, T. Stack, K. Webb, and J. Lepreau, "Large-scale virtualization in the emulab network testbed," in *NSDI*. USENIX Association, 2008, pp. 113–128.

[17] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint vm placement and routing for data center traffic engineering," in *INFOCOM*. IEEE, 2012, pp. 2876–2880.

[18] D. Gupta, K. Yocum, M. McNett, A. C. Snoeren, A. Vahdat, and G. M. Voelker, "To infinity and beyond: time warped network emulation," in *SOSP*. ACM, 2005, pp. 1–2.

[19] E. Weingärtner, F. Schmidt, H. Vom Lehn, T. Heer, and K. Wehrle, "Slicetime: A platform for scalable and accurate network emulation," in *NSDI*. USENIX Association, 2011, pp. 19–19.