2011 IEEE 4th International Conference on Cloud Computing

Evaluation of Network Topology Inference in Opaque Compute Clouds Through End-to-End Measurements

Dominic Battré*, Natalia Frejnik*, Siddhant Goel†, Odej Kao* and Daniel Warneke*

*Technische Universität Berlin, Berlin, Germany
Email: {firstname}.{lastname}@tu-berlin.de

†Technische Universität München, Munich, Germany
Email: goel@in.tum.de

***Abstract*—Modern Infrastructure-as-a-Service (IaaS) clouds offer an unprecedented flexibility and elasticity in terms of resource provisioning through the use of hardware virtualization. However, for the cloud customer, this virtualization also introduces an opaqueness which imposes serious obstacles for data-intensive distributed applications. In particular, the lack of network topology information, i.e. information on how the rented virtual machines are physically interconnected, can easily cause network bottlenecks as common techniques to exploit data locality cannot be applied.**

In this paper we study to what extent the underlying network topology of virtual machines inside an IaaS cloud can be inferred based on end-to-end measurements. Therefore, we experimentally evaluate the impact of hardware virtualization on the measurable link characteristics packet loss and delay using the popular open source hypervisors KVM and XEN. Afterwards, we compare the accuracy of different topology inference approaches and propose an extension to improve the inference accuracy for typical network structures in data centers. We found that common assumptions for end-to-end measurements do not hold in presence of virtualization and that RTT-based measurements in paravirtualized environments lead to the most accurate inference results.

***Keywords*-cloud computing; network topology inference; network tomography; virtualization**

I. Introduction

In the recent years, cloud computing has gained tremendous popularity as a new promising platform for distributed, large-scale applications. At the forefront of the cloud movement are commercial products like Amazon EC2 [1] or the Rackspace Cloud [2]. Following the idea of Infrastructure-as-a-Service (IaaS), these products allow their customers to rent large sets of compute resources on a short-term pay-per-usage basis. The rented compute resources are usually delivered in form of virtual machines (VMs) which are hosted inside the companies' data centers. The customers thereby need not (and typically are not supposed to) know about the physical location of their VM. They can rely on their VMs running "somewhere in the cloud" without knowing which VMs are hosted on the same physical server, which servers share the same rack, etc.

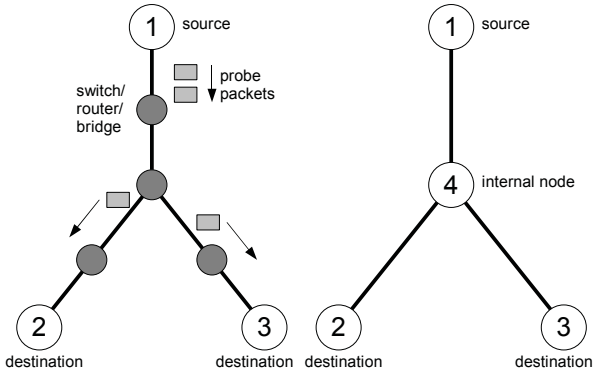
While this opaqueness contributes much to the cloud's flexibility and simplifies management tasks for the data center operator, it hinders the efficient execution of distributed applications which require to exchange large amounts of data among the rented VMs. In particular, the lack of topology information, i.e. knowledge about the VMs' physical interconnections, makes it impossible to exploit data locality and increases the risk of network bottlenecks.

A prominent example of an application class which highly benefits from the cloud's elasticity on the one hand but suffers from the lack of topology information on the other hand is massively-parallel data processing [3]. Since network bandwidth is typically a scarce resource for these kinds of applications, processing frameworks like Google's MapReduce [4] or its open-source implementation Hadoop [5] attempt to run processing tasks either right on the node which stores the task's input data or on a node in close proximity. The proximity between the compute nodes is defined by the number of network switches a data packet must traverse from one node to the other.

While this type of proximity is easy to determine in a static cluster, it imposes some serious obstacles for today's IaaS clouds. Diagnostic tools like *traceroute* can potentially provide a course-grained network topology including routers. However, these tools rely on the cooperation of the internal network components [6] and are unable to identify link-layer network components like switches or bridges, which also play an important role for exploiting data locality.

As a result of the current limitations, this paper focuses on network topology inference based on end-to-end measurements (also called network tomography) in IaaS clouds. Figure 1 illustrates the overall idea. A set of network end nodes (for example VMs) are connected through a physical routing tree (Figure 1a). The physical routing tree's structure, in particular the internal nodes (e.g. network switches, routers, or bridges), is unknown to the end nodes. One or more source nodes from the set of end nodes then send a series of probe packets to a set of destination nodes.

978-0-7695-4460-1/11 $26.00 © 2011 IEEE
DOI 10.1109/CLOUD.2011.30



(a) physical routing tree (b) logical routing tree

Figure 1: The logical routing tree (b) as inferred from the physical routing (a) based on end-to-end measurements

Based on these probe packets the end nodes can measure characteristics (like latency or loss) of the individual network path and correlate these to infer a likely logical routing tree (Figure 1b). The logical tree will not include all the internal nodes of the physical routing tree, but those at which the network path to two end nodes diverges.

Although there has been vivid research in the field of network topology inference recently (e.g. [7], [8], [9]), the applicability of this work in IaaS clouds has not been studied so far. Most previous approaches share a strong focus on large-scale networks, like the Internet, which are characterized by a large number of nodes, limited throughput, and considerable packet loss as well as latency. In contrast to that, the networks in today's cloud data centers are characterized by high throughput links and transfer latencies that are orders of magnitude smaller than the ones in wide-area networks. Moreover, to our knowledge, this paper is the first to study the inference process in presence of hardware virtualization, which has already been found to have a significant impact on the network link characteristics [10]. The contributions of this paper can be summarized as follows:

- We analyze the network path characteristics delay and loss rate and carefully study their suitability as proximity metrics for topology inference in virtualized environments. In particular, we will evaluate how susceptible the individual characteristics are to varying I/O load inside KVM and XEN-based VMs.
- Based on this analysis, we compare the accuracy of the inferred topologies for two different latency-based measurement approaches.
- Finally, we propose an extension to existing topology inference algorithms which improves the inference accuracy for typical network structures in data centers.

The rest of this paper is structured as follows: In Section II we discuss recent developments in the field of network topology inference and summarize the most prevalent approaches. To assess the impact of hardware virtualization we provide an initial evaluation on the path characteristics loss rate and delay in Section III. Section IV contrasts the accuracy of the inferred topologies for two different latency-based measurement techniques and introduces our extension. Section V concludes the paper and briefly discusses open issues and future work.

II. Related Work

The first work in terms of network topology inference based on end-to-end measurements has been conducted in the context of multicast networks ([11], [12]). However, due to the poor availability of multicast in real-world networks, several projects also studied topology inference based on unicast end-to-end measurements. Coates et al. [7] presented a method to capture path delay in unicast routing tree topologies called sandwich probing. Based on this novel probing scheme, the authors devised a Markov Chain Monte Carlo (MCMC) procedure to infer the most likely network topology. In [13] Castro et al. demonstrated how to express the inference problem as a hierarchical clustering problem and proposed their agglomerative likelihood tree (ALT) algorithm. Shih and Hero later extended their work by finite mixture models and MML model order penalties [8]. Ni et al. addressed the complexity of the proposed clustering algorithms and incorporated node joins and departures in the inference process [9]. Shirai et al. [14] as well as Tsang et al. [15] considered topology inference based on round-trip time (RTT) measurements. However, neither work examined the effects of hardware virtualization.

In the context of cloud computing, topology awareness has been considered by the following papers:

In [16] Kozuch et al. presented Tashi, a location aware cluster management system. Tashi features a so-called resource telemetry service which is capable of reporting the distance between a pair of VMs according to some user-defined metric. However, the way the resource telemetry service obtains the location data of the VMs is not addressed in the paper. Gutpa et al. [17] introduced a hosting framework for VMs. Their framework is able to deduce the traffic pattern of distributed applications running inside these VMs. Based on the observed traffic patterns, VMs are migrated inside the cluster such that the locality of a data transmission is improved. Ristenpart et al. [18] discussed the location of VMs inside Amazon EC2 from a security point of view. Based on observations like common routing paths, common IP address prefixes or packet RTTs, the authors examine the possibility to detect colocated VMs and exploit the colocation for attacks.

III. Network Path Characteristics in Clouds

To infer likely network topologies based on end-to-end measurements, it is important to understand the characteris-

tics of the paths which the probe packets travel. Bestavros et al. [19] established the theoretical foundation for topology inference in unicast networks. They showed that a broad class of link characteristics like throughput, packet loss rate, or packet delay can be used as a basis for a proximity metric given that these characteristics obey certain properties.

As an initial step towards topology inference in clouds, we will analyze the impact of different types of hardware virtualization on link *loss* and *delay*. Unlike in case of specialized network hardware, packets routed between VMs and their respective host system may experience unexpected delays or congestions as a result of high system load or scheduling strategies of the host's kernel. This initial analysis will highlight whether the common assumptions [19], [9] for end-to-end-measurements in unicast networks still hold.

All experiments presented in the following were conducted on our local cloud testbed with 64 VMs hosted on eight physical servers. We compare the characteristics of the network links for KVM [20] and XEN [21]-based VMs. For KVM we also considered VMs with unmodified device drivers (full virt.) and modified ones (paravirt.). A detailed description of the testbed can be found in the appendix. The confidence intervals, if shown in the plots, represent a confidence level of 95%. If not shown, confidence intervals have been small and omitted to improve legibility.

A. Inference based on Packet Loss

Packet loss is often considered as a link characteristic for topology inference [12], [22]. The idea is that a source node sends probing packets to at least two receiver nodes. The receivers measure their individual packet loss rate. From correlations in the loss rates it is then possible to deduce common subpaths between the source and the receivers.

Ideally, loss measurements are conducted in a multicast network, so that a dropped packet affects the loss rate of all of its receivers in the same way. In unicast networks, the effect of a multicast transmission on the loss rate can be mimicked by a series of back-to-back unicast transmissions. However, a crucial prerequisite is that the loss rate of all unicast packets within a probe is positively correlated on a common subpath between the source and the receivers [9].

To verify whether this prerequisite is fulfilled in a virtualized environment we let each VM of our cloud setup consecutively send probes consisting of two unicast packets to all other VMs within the same setup. The interval between two consecutive probes has been 100 ms. Within one probe, the two unicast packets were sent back-to-back, i.e. with no intentional delay between the packets. Since both unicast packets are destined for the same receiver (i.e. their common subpath equals the entire path both packets travel through the network) we expect to see either both packets of the probe arrive at the receiver or none at all.

Figure 2 depicts the loss rates we observed depending on the generated background traffic. Overall loss rate denotes the percentage of probes in which either one or both unicast packets did not reach the destination. 1-packet loss refers to the percentage of probes where only one of the two unicast packets arrived at the receiver. Details on the generation of the background traffic can be found in the appendix.

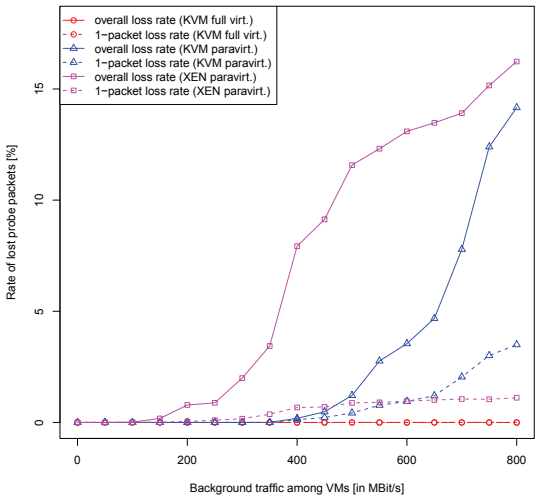


Figure 2: Observed loss rates against background traffic for different types of virtualization

As one fundamental obstacle we found that with full virtualization (KVM full virt.) we were unable to create any significant packet loss. At a background traffic of approximately 200 MBit/s per VM the overhead of the I/O virtualization became so high that the VMs fully utilized their assigned CPU core. Consequently, we hit a CPU bottleneck before we were able to cause any network overflow or congestion which may have resulted in packet loss.

The same experiment with paravirtualization showed different results. For both KVM and XEN-based virtualization we were able to observe packet loss of up to 15 or 16%, respectively. However, again it required considerable amounts of background traffic to be generated. Problematic about the experiments with KVM and paravirtualization is also that the correlation of packet losses within one probe is poor. Depending on the rate of background traffic, for 25% (800 MBit/s) to 65% (400 MBit/s) of all lost probes, one of the included unicast packets still reached its destination.

B. Inference based on Packet Delay

Topology inference based on packet delay follows a similar idea as the previous loss approach. Within one probe a pair of unicast packets is sent back-to-back to two receivers. As long as all unicast packets travel along the same subpath, they are expected to experience similar delays. Receiver pairs with a long shared subpath from the source are likely

to have highly correlated delays while the delay of receiver pairs with a short common subpath is expected to diverge.

To analyze the impact of different virtualization techniques on the correlation of path delays we let each VM of cloud setup consecutively send probes consisting of two unicast packets to all other VMs. Again, the two unicast packets were sent back-to-back to the respective receiver. Like in the loss experiment, the common subpath of both unicast packets was identical to their overall path through the network. Hence, if packet delay on the common subpath was correlated, the second unicast packet would have to arrive at the receiver without any significant delay after the first one.

Figure 3 illustrates the average interarrival times between the first and the second unicast packet of a probe measured at the receiver. We distinguish between the interarrival times of those probes which have been exchanged between VMs on the same physical host (intra-host interarrival time) and those between VMs on different hosts (inter-host interarrival time). As a baseline, we also plotted the packet interarrival time we measured for probes between the unvirtualized hosts.

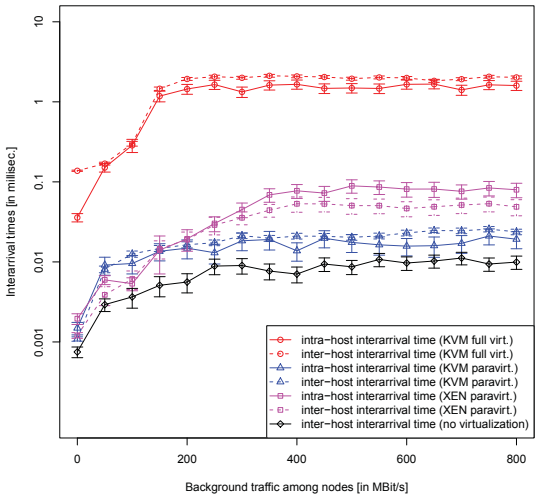


Figure 3: Observed packet interarrival times against background traffic for different types of virtualization

The results indicate that paravirtualization increases the average interarrival time approximately by factor 10 compared to the unvirtualized baseline case. However, even with large amounts of background traffic the average temporal gap between the arrival of the first and the second unicast packet is still considerably smaller than 0.1 milliseconds.

For full virtualization (KVM full virt.) the situation is different. Even at a relatively modest background traffic of 150 MBit/s per VM, the interarrival times grow to more than one millisecond. Given that the average packet RTT in today's local area networks is typically below one millisecond, it is unreasonable to assume any kind of correlation with respect to packet delay for this kind of virtualization.

After having examined the impact of virtualization on the possible correlation of packet delay, our second experiment focused on the effects of virtualization on the observable packet delay itself. Due to the lack of a global clock, measuring delay on a timescale of microseconds in a distributed system is a cumbersome task. As a remedy, we measured the packet RTTs instead. Figure 4 shows the results.

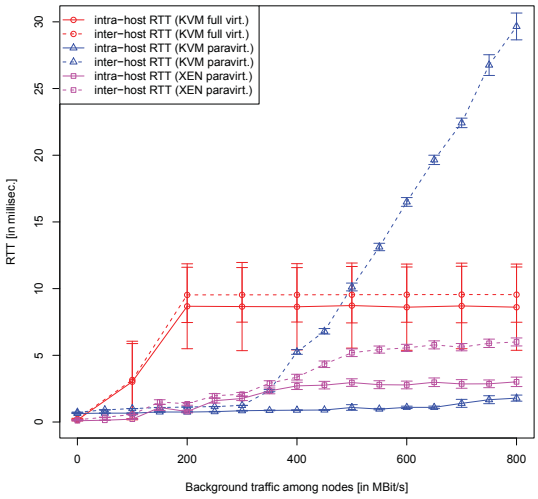


Figure 4: Average RTTs against background traffic for different types of virtualization

The results distinguish between RTTs which have been measured between VMs running on the same physical host (intra-host RTT) and those running on different hosts (inter-host RTT). In general, the RTTs are highly influenced by the level of background traffic. With full virtualization (KVM full virt.), the average RTT rises up to approximately 9 milliseconds at a background traffic of 200 MBit/s per VM. More importantly, the measured values show large variations for this type of virtualization, such that intra-host and inter-host RTTs appear essentially the same.

In contrast to that, the variance of the RTTs from the experiments with paravirtualization is much smaller. Moreover, we observed a distinct gap between the intra-host and inter-host RTT for both KVM as well as XEN-based VMs which continued to grow as the level of background traffic increased. Especially, for the KVM-based VMs the average inter-host RTT eventually rose up to almost 30 milliseconds.

C. Discussion

As an intermediate result of our effort to infer network topologies based on end-to-end measurements in IaaS clouds we can state that virtualization can have a significant impact on the observable characteristics of a network link.

For topology inference based on packet loss, both the KVM and the XEN virtualization layer destroyed the correlation of unicast packet loss on common subpaths, which is assumed by existing inference approaches [9]. In terms of packet delay, we experienced a similar problem for fully virtualized environments. Here the virtualization layer introduced significant gaps in the packet interarrival times which also renders any assumption about correlated packet delays on common subpaths unreasonable.

Among all conducted experiments, observing link latency in paravirtualized clouds appears to be the most promising way to successfully deduce topology information among the involved VMs. Although we partly observed large increases in the RTTs under high load with both KVM and XEN, the individual values showed only little fluctuations at a particular level of background traffic. Moreover, we were able to measure statistically reliable differences between intra-host and inter-host RTTs.

IV. Topology Inference

Having examined the impact of virtualization on the observable link characteristics loss and delay, we will now deal with the actual topology inference process. As a reaction to our findings from the previous section we will focus on delay-based approaches and only consider paravirtualization.

Like most existing approaches (e.g. [11], [13], [12]) we attempt to reconstruct the logical routing tree through agglomerative hierarchical clustering. Starting with each VM as an individual cluster, this clustering technique progressively merges clusters based on a similarity metric γ until only one cluster is left. The approach requires an initial set of similarity values $\gamma_{i,j}$ for each pair of VMs i and j.

In the following we will discuss two different delay-based measurement approaches which can be used to construct $\gamma_{i,j}$ and then contrast their performance in the inference process.

A. Obtaining Initial Similarity Values for the VMs

To obtain the required pairwise similarity metric $\gamma_{i,j}$ Coates et al. proposed a delay-based measurement technique called *sandwich probing* [7]. Compared to classic packet delay measurements, sandwich probing eliminates the need for synchronized clocks on the sender and the receiver node because it only measures delay differences.

As illustrated in Figure 5 a sender node s sends out a sequence of so-called sandwich probes. Each sandwich probe consists of three packets, two small packets destined for receiver j separated by a larger packet destined for receiver i. The second small packet is expected to queue behind the large one at every inner node of the routing tree (e.g. bridge, switch, etc.). This induces an additional delay Δd between the small packets on the shared links. Δd can be used as a similarity value $\gamma_{i,j}$ because the larger $\gamma_{i,j}$ becomes the longer the common subpath from s to i and j must be. The longer the common subpath from s, the closer i and j must be in the logical routing tree. Since all sandwich probes originate from a single sender s, all inferred logical routing trees will have s as their root node.

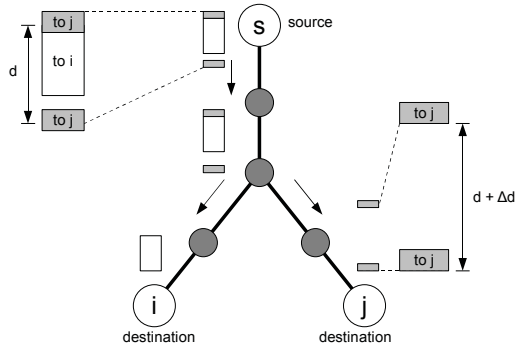


Figure 5: The general idea of sandwich probing

Another way to overcome the necessity for synchronized clocks is to measure path delay based on the *round-trip time (RTT)* between all pairs of leaf nodes i and j. As opposed to sandwich probing the RTT measurements are not conducted from a single sender node s. Instead, each leaf node $l \in L$ (L is the set of leaf/end nodes) conducts its on measurements. However, the overall measurement complexity (i.e. the number of messages that must be transfered to obtain $\gamma_{i,j}$ for all pairs of $i, j \in L$) is still $\mathcal{O}(|L|^2)$. As a small RTT $r_{i,j}$ between two leaf nodes i and j indicates a close proximity in the inferred logical routing tree, the similarity value $\gamma_{i,j}$ must be defined as $\frac{1}{r_{i,j}}$.

B. Accuracy of the Inferred Topologies

To assess the accuracy of the inferred topologies that can be achieved based on end-to-end latency measurements in paravirtualized environments we collected samples for $\gamma_{i,j}$ using both sandwich as well as RTT probing. For the sandwich probing, we set the delay between the two small packets to $d = 10$ milliseconds. We also experimented with other values for d, however, found this one to provide the best overall results. In total, we collected approximately 50 probes for each pair of leaf nodes, each measurement technique, each level of background traffic, and each virtualization technology. For the actual clustering we used the ALT algorithm as proposed by Castro et al. [13].

Figure 6 shows the accuracy of the inferred topologies for KVM as well as XEN-based VMs and sandwich as well as RTT-based probing against different levels of background traffic. The accuracy of the inferred network topology is expressed as the distance between the inferred and the real topology in the Robinson-Foulds metric [23]. The distance between two labeled trees in the Robinson-Foulds metric is essentially the number of edges that must be inserted/deleted to transfer one tree into the other.

Note that the inferred tree produced by the ALT algorithm as always binary [13]. Although there exist algorithms that

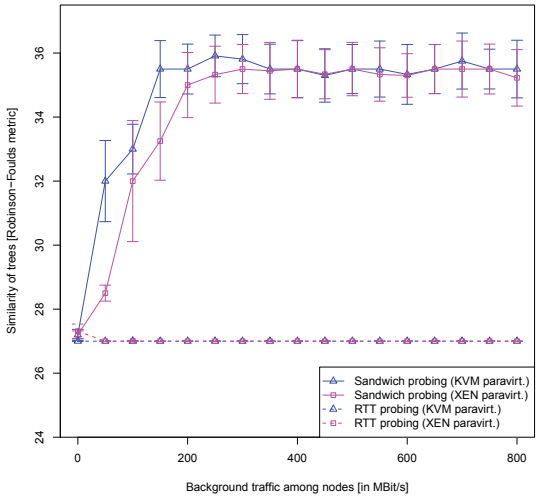


Figure 6: Accuracy of the inferred topologies against background traffic using the ALT algorithm

can also infer network topologies based on general trees, these algorithms also typically start by constructing a binary tree first and then apply stochastic methods to transform the binary tree into a likely general one [7], [22]. Hence, the constructed binary tree can be regarded as a robust starting point for the topology inference process. If an inference algorithm is unable to estimate a network topology accurately based on a binary tree, it will not be able to estimate it accurately based on a general tree either. We will present an approach to transfer the binary tree into a general tree in the next subsection.

In the absence of background traffic the average distance of the inferred network topology tree to the real one centers around 27 for both virtualization and probing techniques. However, with increasing background traffic the accuracy of those topologies that were inferred based on the sandwich probes starts to diminish. The reason for this can be explained by the increasing delay for intra-host VM communication under high background traffic.

As highlighted in Section III all packets which are passed to the physical network experience a significant additional delay even for moderate levels of background traffic. Although we found those additional delays to be relatively stable on a timescale of milliseconds, their variance is large enough to blur the subtle delay differences of intra-host VM communication. As a result, the delay differences for all receiver VMs which do not run on the sender VM's host suffer from a large variance and therefore impede the inference process in presence of background traffic. In contrast to that, the topologies inferred based on the RTT probes, where each VM issued its own probe packets, are more stable towards increasing background traffic.

C. Transferring Binary Trees into General Trees

As pointed out in the previous subsection the topologies inferred by the ALT algorithm always follow the structure of a binary tree. Binary trees provide the largest number of degrees of freedom and thus are able to fit the measured data most closely [7]. Therefore they are a reasonable starting point to unbiasedly contrast the impact of different probing techniques on the inference accuracy.

However, the network topology in most data centers can be rather described by a general tree. The inferred binary tree can be considered an "overfitted" version of the physical network tree which includes more internal nodes than actually exist and therefore automatically decreases the inference accuracy in most network setups.

To overcome the problem of overfitting, several methods have been proposed (e.g. [7], [22]). Most of them apply computationally demanding heuristics to reconstruct likely general trees based on the initially inferred binary tree. Since most inference approaches were designed for large-scale networks like the Internet, which do not allow any assumptions about the structure of the routing tree, these heuristics are a reasonable choice. However, in contrast to the Internet, the network structure in data centers is much more regular. Typical network architectures of today's data centers consist of either two- or three-level trees of switches or routers [24]. Hosting multiple VMs on one server might add another level of depth to the tree but, for example, a network topology tree with a depth of more than three in a single IP subnet is very unlikely to occur in practice.

The extension we propose in the following exploits this regularity. It is based on the assumptions that network topology trees with a depth greater than d are unlikely to occur. Moreover, it assumes that all leaf nodes are likely to have a similar depth in a data center network topology tree.

Our extension is subdivided into two operations: The first operation `reroot` takes the initial binary tree as input and chooses that inner node as the tree's new root node which minimizes the difference between the leaf nodes with the highest and the lowest depth. The operation accounts for the fact that the root node created by the clustering algorithm is not necessarily correct. For example, when using sandwich probing, the root node of the inferred binary tree is always the source of probe packets s. The rerooted tree is then passed to the second operation `limitDepth(d)`. This operation continuously identifies the leaf node with the highest depth, cuts it out, and appends it to its former parent's parent node as long as the depth of the tree is greater than d. Figure 7 illustrates the impact of our extension on the inference accuracy for different values of d. Compared to the initial binary tree (Figure 6) our extension reduces the Robinson-Foulds distance to the actual tree which represents our testbed's network topology by 16 to 21 (depending on d) on an average. To improve legibility we only show the

results for KVM (paravirt.) and omit confidence intervals.

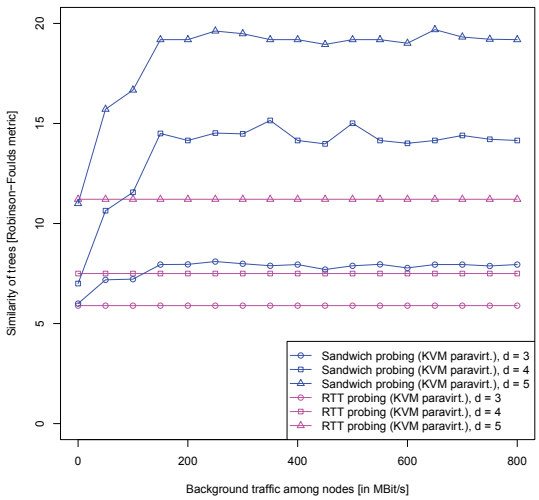


Figure 7: Accuracy of the inferred topologies using the ALT algorithm and the depth limitation

V. Conclusion

In this paper we examined to what extent the physical network topology of VMs inside an IaaS cloud can be inferred through end-to-end measurements. Therefore, we provided an initial analysis of the network path characteristics loss and delay and examined the impact of hardware virtualization using the open source hypervisors KVM and XEN. Based on this analysis, we contrasted the accuracy of the inferred topologies for two different latency-based measurement approaches. Finally, we proposed an extension to existing clustering-based inference algorithms. This extension allows to transform overfitted binary trees into general trees which are likely to describe the network topology of a cloud data center.

In sum, we can conclude that topology inference in IaaS clouds is a challenging subject. Even under moderate system load hardware virtualization has a significant effect on the measurable network path characteristics and destroys important correlation properties which are assumed by most inference approaches. Among all conducted experiments on our mid-size cloud testbed, RTT-based delay measurements led to the most accurate inference results with an average Robinson-Foulds distance to the actual topology of under 6. However, we also observed large delays introduced by the virtualization layers (especially KVM) under high background traffic. In our opinion, the variance of these delays currently leaves little potential to reliably infer passive network components like link layer switches.

In general, we think our work represents a valuable contribution to the current efforts of porting data-intensive distributed applications to the cloud. For future work we are curious to follow new developments in the field of virtualization. In particular, we think hardware-assisted I/O virtualization has a strong potential to reduce the current I/O overhead and can have a positive effect on the performance of topology inference.

Acknowledgment

We thank HP for supporting this work through an Open Collaboration Grant.

Appendix

All experiments presented in this paper have been conducted on our local IaaS cloud testbed. The testbed consisted of eight physical servers with the following configuration:

CPU	Two Intel Xeon 2.66 GHz CPUs (model E5430)
RAM	32 GB
Network	Intel Corporation 80003ES2LAN 1 GB/s Ethernet (connected to local 1 GB/s switch)
OS	Gentoo Linux
Kernel	2.6.34-xen-r4 for XEN-based experiments, 2.6.32-gentoo-r7 otherwise

The interconnect between the servers was a 1 GBit/s Ethernet network with an HP ProCurve 1800-24G switch. During the experiments we deployed 64 VMs, eight VMs per host, using the Eucalyptus cloud software [25].

We employed the two open-source virtualization techniques XEN and KVM during our experiments. For the KVM-based experiments we also distinguished between full virtualization (with unmodified device drivers) and paravirtualization (with the modified virtio device drivers). A detailed VM configuration is given in the following table:

CPU	1 CPU core
RAM	2 GB
Network	Bridged network, device driver `e1000` for KVM (full virt.), `virtio_net` for KVM (paravirt.), and `xennet` for XEN
OS	Ubuntu Linux 9.10 (Karmic Koala)
Kernel	2.6.31-22-server for XEN, 2.6.32-gentoo-r7 otherwise

To generate the background traffic during the experiments we devised a small auxiliary program which was executed on each VM during our experiments. The program was capable of generating UDP traffic at an adjustable data rate. Each generated UDP packet carried 8 KByte of payload. To achieve a fair mixture between inter-host and intra-host traffic, we setup four VMs on each physical host to exchange background traffic among each other while the other four VMs transmitted data to VMs on a different host. So, for example, at a background traffic level of 50 MBit/s, all eight VMs generated network traffic at 50 MBit/s. However, only four VMs actually utilized the physical network.

All the time, the testbed was dedicated to our experiments. Thus, side effects from other applications can be excluded.

REFERENCES

[1] Amazon Web Services LLC, "Amazon Elastic Compute Cloud (Amazon EC2)," http://aws.amazon.com/ec2/, 2010.

[2] Rackspace US, Inc., "The Rackspace Cloud," http://www.rackspacecloud.com/, 2010.

[3] D. Warneke and O. Kao, "Nephele: Efficient parallel data processing in the cloud," in *MTAGS '09: Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers*. New York, NY, USA: ACM, 2009, pp. 1–10.

[4] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[5] The Apache Software Foundation, "Welcome to Hadoop!" http://hadoop.apache.org/, 2010.

[6] B. Yao, R. Viswanathan, F. Chang, and D. Waddington, "Topology inference in the presence of anonymous routers," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 1, 30 2003, pp. 353 – 363 vol.1.

[7] M. Coates, R. Castro, R. Nowak, M. Gadhiok, R. King, and Y. Tsang, "Maximum likelihood network topology identification from edge-based unicast measurements," in *SIGMETRICS '02: Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*. New York, NY, USA: ACM, 2002, pp. 11–20.

[8] M.-F. Shih and A. Hero, "Hierarchical inference of unicast network topologies based on end-to-end measurements," *Signal Processing, IEEE Transactions on*, vol. 55, no. 5, pp. 1708 –1718, May 2007.

[9] J. Ni, H. Xie, S. Taitikonda, and Y. R. Yang, "Efficient and dynamic routing topology inference from end-to-end measurements," *IEEE Transactions on Networking*, vol. 18, no. 1, pp. 123–135, 2010.

[10] G. Wang and T. S. E. Ng, "The impact of virtualization on network performance of Amazon EC2 data center," in *INFOCOM 2010: Proceedings of the 29th Conference on Computer Communications. IEEE*. Washington, DC, USA: IEEE Computer Society, March 2010.

[11] S. Ratnasamy and S. McCanne, "Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements," in *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1, 21-25 1999, pp. 353 –360 vol.1.

[12] N. Duffield, J. Horowitz, F. Lo Presti, and D. Towsley, "Multicast topology inference from measured end-to-end loss," *Information Theory, IEEE Transactions on*, vol. 48, no. 1, pp. 26 –45, jan 2002.

[13] R. Castro, M. Coates, and R. Nowak, "Likelihood based hierarchical clustering," *IEEE Trans. on Signal Processing*, vol. 52, pp. 2308–2321, 2004.

[14] T. Shirai, H. Saito, and K. Taura, "A fast topology inference: a building block for network-aware parallel processing," in *Proceedings of the 16th international symposium on High performance distributed computing*, ser. HPDC '07. New York, NY, USA: ACM, 2007, pp. 11–22.

[15] Y. Tsang, M. Yildiz, P. Barford, and R. Nowak, "On the performance of round trip time network tomography," in *Communications, 2006. ICC '06. IEEE International Conference on*, vol. 2, June 2006, pp. 483 –488.

[16] M. A. Kozuch, M. P. Ryan, R. Gass, S. W. Schlosser, D. O'Hallaron, J. Cipar, E. Krevat, J. López, M. Stroucken, and G. R. Ganger, "Tashi: location-aware cluster management," in *Proceedings of the 1st workshop on Automated control for datacenters and clouds*, ser. ACDC '09. New York, NY, USA: ACM, 2009, pp. 43–48.

[17] A. Gupta, M. Zangrilli, A. Sundararaj, A. Huang, P. Dinda, and B. Lowekamp, "Free network measurement for adaptive virtualized distributed computing," in *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, April 2006, p. 10 pp.

[18] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," in *CCS '09: Proceedings of the 16th ACM conference on Computer and communications security*. New York, NY, USA: ACM, 2009, pp. 199–212.

[19] A. Bestavros, J. W. Byers, and K. A. Harfoush, "Inference and labeling of metric-induced network topologies," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 11, pp. 1053–1065, 2005.

[20] A. Kivity, "kvm: the Linux virtual machine monitor," in *OLS '07: The 2007 Ottawa Linux Symposium*, July 2007, pp. 225–230.

[21] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*. New York, NY, USA: ACM, 2003, pp. 164–177.

[22] N. Duffield, F. L. Presti, V. Paxson, and D. Towsley, "Network loss tomography using striped unicast probes," *IEEE/ACM Trans. Netw.*, vol. 14, no. 4, pp. 697–710, 2006.

[23] D. F. Robinson and L. R. Foulds, "Comparison of phylogenetic trees," *Mathematical Biosciences*, vol. 53, no. 1-2, pp. 131 – 147, 1981.

[24] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 63–74, August 2008.

[25] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus open-source cloud-computing system," in *CCGRID '09: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 124–131.