

Measuring VLAN-Induced Dependencies on a Campus Network

Ahmed Mansy, Mukarram Bin Tariq, Nick Feamster, Mostafa Ammar
{amansy,mtariq,feamster,ammar}@cc.gatech.edu

School of Computer Science, College of Computing.
Georgia Institute of Technology. Atlanta, GA 30332

Abstract. Many enterprise, campus, and data-center networks have complex layer-2 virtual LANs (“VLANs”) underlying the layer-3 network. The interaction between layer-2 and layer-3 topologies can introduce dependencies that are not visible when analyzed solely from layer 3; this paper studies the extent and impact of these dependencies. We first present EtherTrace, a tool that infers the layer-2 topology using data passively collected from Ethernet switches. Using this tool, we infer the layer-2 topology for a large campus network and compare the two topologies. Our comparison yields some striking initial findings: almost 70% of layer-2 edges are shared by 10 or more IP edges, and a single layer-2 edge may be shared by as many as 34 different IP edges. This sharing has significant implications for both robustness and for network tomography. Applying network tomography to the IP topology to diagnose failures caused by layer-2 devices results in only 54% accuracy, compared to 100% accuracy when tomography is performed at layer 2.

1 Introduction

Network virtualization allows many logically distinct networks to share common physical hardware. Virtual networks can amortize costs, since multiple networks can share physical hardware. Second, they provide flexibility, since logically distinct networks can be created without expensive (and disruptive) changes to the physical infrastructure itself. Recent work has also shown that virtual networks can even simplify certain types of network management tasks [19]. Technologies for building virtual networks are proliferating; they exist both in practice (e.g., Virtual LAN, or “VLANs”, [10]) and in research environments, e.g., VINI [2,3,9], OpenVZ [16], and OpenFlow switches [15].

Virtual networks typically provide some isolation between the networks that run on top of them. For example, Ethernet VLANs restrict broadcast domains and contain hosts in one or separate networks based on their communication needs, which can simplify administration and security tasks. Similarly, VINI provides resource isolation among the slices to prevent experiments from interfering with each other. Isolation offers many benefits, but it also hides underlying dependencies; a user or operator who is not aware of these dependencies may end up making poor design decisions or incorrectly diagnose problems. Networks that appear independent may in fact have common failure modes at lower layers; for example, congestion or failures in the underlying network can simultaneously affect many seemingly independent networks. Understanding

the dependencies that are induced by virtualization can assist operators with planning and make diagnosis more tractable.

This paper focuses on how VLANs may create significant dependencies that are not visible from the IP layer alone. VLANs are a type of virtual network where many distinct LANs can coexist on a fixed set of physical switches and links. The goal of this study is to determine the dependencies that exist among IP subnets that run atop the campus VLANs and its implications for network fault diagnosis and localization. This paper presents three contributions:

- ***EtherTrace*, a passive layer-2 topology discovery tool.** *EtherTrace* infers the VLAN (layer-2 network) topology using mostly passive measurements of bridge and ARP tables from the switches in the network.
- **An analysis of VLAN-induced dependencies.** We analyze VLAN-induced dependencies in the layer-3 network segments for the campus-wide network at Georgia Tech (a network with over 1,000 distinct VLANs) and find that as many as 85% of layer-2 edges shared by at least 6 distinct layer-3 edges; some layer-2 edges are shared by as many as 30 distinct layer-3 edges.
- **An analysis of the effect of hidden dependencies on diagnosis.** We demonstrate how we can improve the accuracy and specificity of fault localization by using the information about the hidden dependencies. We observe nearly twice as better accuracy and nearly four times improvement in specificity in our results.

The rest of this paper is organized as follows. In Section 2, we present the *EtherTrace* algorithm for inferring the dependencies. In Section 3, apply this algorithm to the layer-3 network on the Georgia Tech. campus network. In Section 4, we analyze the extent and nature of the shared dependencies among layer-3 segments. In Section 5, we describe the cross-layer fault localization method to improve the accuracy and specificity of the inference. We review the related work in Section 6, and conclude in Section 7.

2 *EtherTrace*: A Passive Layer-2 Topology Discovery Tool

This section describes *EtherTrace*, a tool that infers the set of layer-2 elements (i.e., switches and switch ports) for each layer-3 path. We begin with a brief overview of VLANs; we then describe the *EtherTrace* algorithm.

2.1 Background

Ethernet-based Local Area Networks (LANs) use learning bridges to connect the network segments. These bridges use a spanning-tree algorithm to achieve a loop-free topology. The bridges learn the location of hosts on the network by “listening” for the frames sent by the hosts; if a bridge hears a frame sent by host *A* on a port *x*, then the bridge forwards all the future frames for *A* to port *x*. Bridges maintain this information in bridge tables, where entry in the table comprises the MAC address of the host and port on which the frames from the host were heard. Bridge-tables are dynamic: table entries timeout if not refreshed by new frames from the hosts.

On *Virtual LANs (VLANs)*, a single bridge may be simultaneously assigned to one or more VLANs. The bridges on each VLAN form separate spanning-trees, thus allowing multiple simultaneous virtual topologies on the network. To support VLANs, the entries in bridge tables also include a VLAN identifier, which is copied from the frames. Bridges forward traffic within each VLAN using the bridge tables, but do not forward traffic across VLANs. A router connects VLANs to allow inter-VLAN communication. Most modern routers and switch-router devices support VLAN interfaces which allows these devices to have their interfaces appear as multiple virtual-interfaces, each on a separate VLAN.

2.2 EtherTrace Algorithm

EtherTrace uses a simple observation to discover layer-2 devices corresponding to each layer-3 path: Because the bridges form a tree, only one layer-2 path between any pair of hosts on the network exists at any time. Thus, the bridges along the path between two hosts on the same VLAN must always hear frames from those two hosts on two separate ports. On the other hand, bridges that are not along the path between the hosts will always hear the frames from the two hosts on the same port. Although *EtherTrace* can also determine the order of switches and ports along a path, the order is usually not important for dependency analysis. Thus, we present only the method for determining the set of layer-2 switches and ports on a path.

Notation. We refer to a snapshot of bridge-table entries from all the switches in the network for a particular host x as T_x . Each entry, $e \in T_x$, is a 3-tuple $(e.b, e.p, e.v)$, where the three elements refer to the bridge, the port, and the VLAN-identifier fields, respectively. We refer to the MAC-address of host x as m_x . For hosts x and y , $T_{x\Delta y}$ refers to the symmetric difference¹ of sets T_x and T_y . $B(T_x)$ and $V(T_x)$ refer to the set of bridges and set of VLANs that the entries in T_x refer to, respectively.

Determining the Path Elements. To determine the set of switches on the IP path between two hosts, *EtherTrace* first uses the ARP tables to obtain the MAC addresses of the two hosts. To determine the layer-2 path, *EtherTrace* considers two cases.

Case 1: Hosts on the same VLAN and IP subnet. From the bridge tables, *EtherTrace* determines the bridge-table entries involving the two hosts, and among these entries determines whether for some VLAN identifier(s), there exist a set of switches which hear the MAC addresses of the two hosts on two different ports on that VLAN. Specifically, for hosts x and y , the elements on the path are given as a set of 3-tuples as follows:

$$\hat{S}(x, y) = \{(e.b, e.p, e.v) : e \in T_{x\Delta y}, \\ e.b \in B(T_x) \cap B(T_y), e.v \in V(T_x) \cap V(T_y)\} \quad (1)$$

The first constraint in Equation 1 prunes all the bridge table entries that are common for the two hosts. The second and third constraints ensure that *EtherTrace* includes only the bridges that hear from both hosts on same VLANs.

¹ The symmetric difference of sets A and B is $A \cup B - A \cap B$

There are two sub-cases for Case 1. Although usually network administrators configure each VLAN to correspond to one IP-subnet, it is possible to *merge multiple VLANs* by connecting two non-trunk ports with a loop-cable. If the two hosts are on such VLANs, the above algorithm will still work correctly, because the merged VLANs will appear in $V(T_x) \cap V(T_y)$. The second sub-case occurs when a pair of bridges along a path are connected through a passive component, such as a hub or a repeater element. *EtherTrace* cannot recognize such passive elements. If the two hosts are on such a segment, *EtherTrace* declares an empty path between the hosts. In most modern network deployments, each host connects directly to a switch-port and there is little, if any, communication that takes place over hubs or buses. As a result, the cases where *EtherTrace* fails, seldom arise.

Case 2: Different VLANs and IP-subnets, layer-3 traceroute is available. If an layer-3 traceroute is available between the two hosts, then we infer the elements in the layer-2 path by obtaining the layer-2 path elements for each IP-path segment. If the layer-3 traceroute between the hosts x and y is $\{h_1 \cdots h_k\}$, $h_1 = x$ and $h_k = y$ then the layer-2 elements on the path is:

$$S(x, y) = \bigcup_{i=1:k-1} \hat{S}(h_i, h_{i+1}) \quad (2)$$

2.3 Implementation

We have implemented *EtherTrace* using Python and a SQL backend. We continually update the database with ARP and bridge table snapshots from the routers and switches in the network, as well as the traceroutes among the CPR nodes. The Python front end implements the *EtherTrace* algorithm for finding layer-2 path between any pair of nodes on the Georgia Tech. campus network that share a VLAN, or if we have a layer-3 traceroute between those nodes. *EtherTrace* helps network administrators infer the dependencies of the path between many pair of nodes on the network without requiring access to those nodes themselves.

3 Dataset and Topology Inference

This section describes the data that we obtained from the Georgia Tech campus network the process of inferring the layer-2 topology by applying *EtherTrace* to this data.

3.1 Dataset

Types of Data. We rely on three sources of data, all from the Georgia Tech campus network. The first is the bridge table entries obtained from all the switches; we poll these switches every four hours using SNMP. The second source is the ARP tables; we poll these routers hourly. These tables provide us with IP-address to MAC-address mappings. The third source of data is the layer-3 traceroutes between 79 CPR nodes [6]

end hosts that are deployed in mostly distinct subnets on the campus network. These nodes perform traceroutes to each other at 5-minute intervals.

Completeness and Consistency. Because the ARP and bridge table entries usually expire at an interval much shorter than our respective polling intervals, a single snapshot of the network state may not contain all the MAC addresses or bridge-table entries. To overcome this problem, we retain entries from the previous snapshots that are not overwritten in the current snapshot.

Unfortunately, retaining older entries increases the risk of inconsistent information in our dataset, for two reasons. First, the topology might change between successive snapshots due to failures or reconfigurations or hosts moving to different parts of the network. Similarly, the IP addresses might change between successive snapshot due to DHCP lease expirations. To mitigate these effects, we restrict ourselves to snapshots obtained on March 25, 2008, for the comparison presented in this paper. There are no reported network failures or outages on this day for the network. To mitigate the inconsistencies arising from IP-address changes, we restrict our analysis to MAC addresses that map to only a single IP address across the snapshots on March 25, 2008. To mitigate the inconsistencies arising from mobile hosts on the campus wireless network, we only consider the hosts on the wired network.

Size of the Dataset. Overall, the dataset contains bridge table entries from 114 switches, ARP tables from 94 routers, containing 88,932 unique MAC addresses, and traceroutes between 79 CPR nodes.

3.2 Topology Inference

We first measure the layer-3 topology using the layer-3 traceroutes between the CPR nodes. We determine that a layer-3 edge exists between two IP routers if those routers appear as adjacent hops in any of the traceroutes between the CPR nodes. To map distinct IP addresses to routers, we use the DNS name associated with the routers to cluster the aliases that share the same domain name under a single node; other tools such as RocketFuel [18] also use this technique.

To infer the layer-2 topology, we use the *EtherTrace* algorithm described in Section 2. For the CPR nodes that are in the same VLAN, i.e., the ones that do not have an IP router between them, we use apply Case 1 of the *EtherTrace* algorithm. There are fewer than 5% of CPR node pairs that share a subnet. For the rest of the paths between the CPR node pairs, we apply Case 2. Specifically, to obtain the layer-2 path between such nodes, we obtain the layer-2 path corresponding to each of the layer-3 hops along the layer-3 path and concatenate these to obtain the overall path. We use the IP address of the router as it appears in the traceroute, not the one that we obtain after de-aliasing.

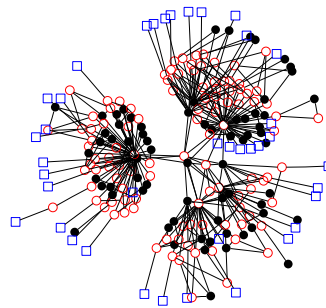
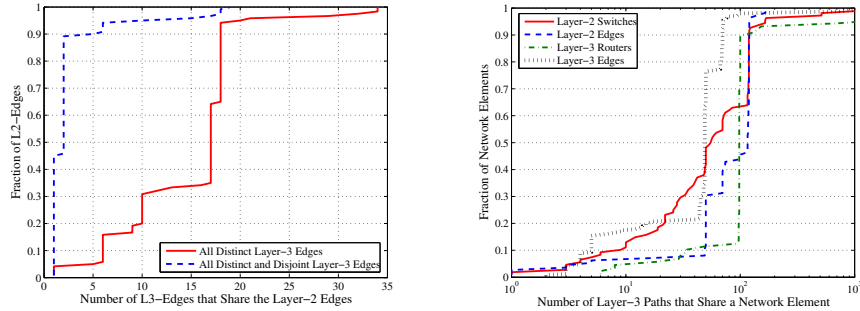


Fig. 1. Layer-2 Topology on the Georgia Tech Campus Network. Legend: Circles (Switches), Solid Circles (Routers), Squares (CPR Nodes).



(a) Distribution of sharing of layer-2 edges among distinct layer-3 edges. (b) Distribution of sharing of layer-2 and layer-3 network elements among layer-3 paths.

Fig. 2. Distribution of sharing of network elements among layer-3 paths and edges.

There is one caveat here: for *EtherTrace* algorithm requires that the MAC addresses of the two end-points of the layer-2 are on the same VLAN. Unfortunately, traceroute’s ICMP TTL time-exceeded messages use the IP address of the return interface of the router as the source address. As a result, if the layer-3 paths are asymmetric, then the adjacent IP addresses that appear in a traceroute may belong on different subnets and therefore different VLANs, making the IP traceroute unsuitable for computing the layer-2 path. In this work, we assume that layer-3 paths are symmetric and the above problem does not arise. In cases where this assumption does not hold, we can obtain the IP addresses of the forward interface on the router from router configuration. This approach is feasible because enterprises have control over their routers.

Figure 1 shows the layer-2 topology discovered for the Georgia Tech campus network using the above methods. The topology shows the layer-2 switches (circles), as well as the IP routers (solid circles) and the CPR nodes (squares). The network is divided in three large clusters of nodes that are connected through a core network in the middle. The IP level network structure agrees with the ground-truth network configuration that OIT at Georgia Tech made available to us.

4 Analysis of VLAN-Induced Dependencies

In VLAN environments, the layer-2 network infrastructure may be shared among many VLANs and IP subnets; specifically, several layer-3 subnets may be overlaid on the same physical layer-2 infrastructure. As a result, we expect that the layer-3 network segments that may appear disjoint or independent are actually not independent. To quantify the extent of these dependencies, we analyze the extent of infrastructure sharing among layer-3 edges and paths.

Infrastructure Sharing Among Layer-3 Edges. Figure 2(a) shows the distribution of number of distinct layer-3 edges that traverse any given layer-2 edge. The solid line in Figure 2(a) 85% of layer-2 edges are common to at least 6 distinct layer-3 edges,

50% of layer-2 edges are shared by 17 or more distinct layer-3 edges and we found one layer-2 edge that was shared between 34 distinct layer-3 edges!

Next, we study disjoint layer-3 edges, i.e., those that do not share a router on either vertex. The dashed line in Figure 2(a) represents the distribution of sharing among such layer-3 edges. We find that only about 45% of the layer-2 edges are not shared between multiple layer-3 disjoint edges. Another 45% of the layer-2 edges are shared between two distinct and disjoint layer-3 edges. We found that 4% of the layer-2 edges were shared between 17 or 18 distinct and disjoint layer-3 edges!

The above results imply that layer-3 paths and segments that might appear distinct and, in some cases, completely disjoint may actually not be independent because they share common underlying physical infrastructure. As a result, these path segments will experience correlated performance and reliability; congestion or failures in the underlying network will simultaneously affect many layer-3 paths, even as these layer-3 paths appear disjoint or independent.

Infrastructure Sharing Among Layer-3 Paths. Figure 2 shows the extent of sharing of network elements among the layer-3 path and layer-3 edges. Figure 2(b) presents the distribution of the number of layer-3 paths among the CPR nodes that traverse a particular layer-2 switch, a layer-2 edge, a layer-3 router or a layer-3 edge. Layer-3 paths are somewhat more likely to traverse a common layer-3 router than a common layer-2 switch. The least-used 50% of the switches appear on only 50 or fewer layer-3 paths, whereas the least used 50% of routers appear on up to 120 layer-3 paths. This result suggests that for the less frequently occurring network elements there is less sharing of layer-2 elements among layer-3 paths than there is sharing of layer-3 elements. The more commonly occurring switches and routers have similar numbers of layer-3 paths traversing them. Figure 2(b) shows a similar trend for sharing of layer-2 and layer-3 edges among layer-3 paths.

There are two causes for these characteristics. First, there are significantly more switches in the network than there are routers. While a typical layer-3 path traverses more switches than routers, the large number of switches ultimately induces less sharing. Second, we observed that more frequently used routers and switches occur in the network core, and their centrality naturally induces more sharing. There is little difference in the extent of layer-2 and layer-3 sharing among paths in the core is that the switches in the core because the core has a single IP subnet.

5 The Effect of Hidden Dependencies on Network Diagnosis

Hidden cross-layer dependencies in networks that are overlaid on virtual networks have implications for network planning, tuning, and diagnosis. In this section, we present preliminary results concerning the implications of hidden dependencies on for tomography-based network fault diagnosis.

Emulation Setup. We emulate layer-2 link failures on the Georgia Tech campus topology shown in Figure 1, and use the simple binary tomography techniques [8] to try to localize this fault using the set of failed layer-3 paths as input. Binary tomography localizes a fault by finding the smallest common failure set that explains the set of failed paths. Given a failed edge (or edges), the method explains the failure by finding the

smallest hitting set that explains the set of failed paths. Because finding smallest hitting set is NP-complete, we use a randomized approximation algorithm, where we repeatedly find a random greedy solution and pick the smallest solution among the repetitions.

The emulation proceeds as follows. For each edge on the network, we determine the layer-3 paths between the CPR nodes that traverse the failed layer-2 edge. To measure the effects of hidden dependencies in virtualized networks, we perform fault localization using the following two distinct methods:

1. **Conventional approach.** As in conventional layer-3 binary tomography, we use the layer-3 edges on the affected layer-3 paths that do not appear in any of the non-affected layer-3 paths, as the set of dependencies for each affected layer-3 path.
2. **Cross-layer approach.** We apply *EtherTrace* to determine the set of layer-2 edges corresponding to the affected layer-3 paths that do not appear on any of non-affected layer-3 paths as the set of dependencies for the affected layer-3 paths.

In each case, we use the hitting set algorithm to find the smallest common set of edges that intersects with each dependency set. We use the set of edges in the hitting set as the location of the fault. For the first method, the hitting set comprises a set of layer-3 edges, and for the second, it comprises a set of layer-2 edges. For comparison, we convert the former to the corresponding layer-2 edges as well, and then compare the two solutions for accuracy and specificity. We compare these two approaches for every layer-2 edge in the network.

Fault Localization Accuracy and Specificity. We define localization to be accurate if the hitting set that we obtain contains the original failed layer-2 edge. We define specificity as the multiplicative inverse of size of the hitting-set (i.e., , a smaller hitting set is reflected with higher specificity).

Accuracy. For the layer-2 edge failures that have a unique hitting set, the cross-layer approach yields 100% accuracy; the conventional approach yields only 54% accuracy.

Specificity. The 95th percentile of specificity using the cross-layer approach is exactly 1: the hitting set is completely specific and does not contain any extraneous edges. The average hitting set size is 1.48. On the other hand, the 95th percentile of specificity using the conventional approach is 0.11, and the average is 0.27.

Thus overall, using layer-3 dependencies to localize the failed links is only half as accurate and one-fourth as specific as using the layer-2 dependencies.

6 Related Work

We discuss two areas of related work. We first discuss previous work that attempts to infer the layer-2 LAN topology. We then present previous work that addresses cross-layer dependencies and its effects on diagnosis.

There are several existing tools for discovering Ethernet network topology. *EtherTrace* relates most closely to techniques that infer layer-2 topology passively from bridge-table entries [4, 14]. The topology discovery algorithm in [14] is efficient as it can discover a large number of Ethernet interconnections using few probing nodes that spoof their MAC addresses to inject bridge-table entries in the switches and then

observe the disruptions in connectivity. Bejerano *et al.* [4] use the bridge-tables in a manner similar to *EtherTrace*, but *EtherTrace* extends this previous work because (1) it works for VLANs and (2) it uses IP traceroutes to construct topologies spanning multiple subnets. Sebos *et al.* [17] use location correlation to find shared-risk link groups of network components. Our work differs in the sense that we determine the shared-risk link group based on whether layer-3 links share the underlying layer-2 infrastructure. Cisco’s Discovery Protocol (CDP) is a proprietary protocol for performing layer-2 traceroutes on a network, but it requires all switches in the network to run CDP [5].

Kompella *et al.* [12] also argue for cross-layer visibility for better planning, maintenance and failure diagnosis. Our work highlights the need for such visibility in the case of virtualized networks. Several previous work have used binary [7, 8, 13] and probabilistic [1, 11] shared risk groups for network diagnosis; this work is the first to study VLAN-induced dependencies and their implications for fault diagnosis.

7 Conclusion and Future Work

This paper studied VLAN-induced dependencies on a campus network. We developed *EtherTrace* a passive layer-2 topology discovery tool that operates in VLAN environments, to discover the layer-2 topology for corresponding set of layer-3 hosts. Using *EtherTrace*, we inferred the layer-2 topology for the Georgia Tech campus network; our analysis included data from 94 routers, 114 switches, and almost 90,000 unique MAC addresses over the course of one day.

We also determined shared layer-2 infrastructure (and, hence, shared dependencies) along otherwise seemingly disjoint layer-3 paths. Our results show that seemingly independent layer-3 path segments can actually be dependent: some layer-2 edges were shared by more than 30 distinct IP links, and some layer-2 edges were shared by as many as 18 node-disjoint layer-3 edges. We also studied how such hidden shared dependencies among layer-3 edges can result in significant inaccuracies and imprecision in fault localization; our results show that a cross-layer tomography approach that incorporates layer-2 and layer-3 topology information can improve accuracy by a factor of two and specificity by a factor of four.

Although this paper has exposed the extent and some consequences of VLAN-induced dependencies in a single campus network, much work remains to fully understand the nature of interactions between VLANs and layer-3 topologies in campus and enterprise networks. One area that deserves further study is how knowledge about these dependencies could help influence network provisioning and planning. For example, network operator that is aware that many IP-layer paths in fact share common physical elements may decide to trunk VLANs differently or route traffic differently at the IP layer. Similarly, an operator could use knowledge of these dependencies to help determine the impact of VLAN trunking decisions (e.g., sizes of bridge tables and ARP tables, traffic volumes) before those configurations are deployed on the network itself. Along the lines of planning for resilience, a major drawback of *EtherTrace* is that it only exposes the current spanning tree topology, *not* any nodes or links that are not part of the spanning tree but might be used in the case of failure. Designing tools—and perhaps also the protocols themselves—to help operators discover the IP-to-layer-2 mappings that exist in failure scenarios could also greatly improve network planning and operations.

Acknowledgements

We would like to thank Russell Clark, Dan Forsythe, and John Merritt at Office of Information Technology at Georgia Tech for their help with various stages of this work, including making the network datasets available to us.

References

1. P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang. Towards highly reliable enterprise network services via inference of multi-level dependencies. In *Proc. ACM SIGCOMM*, Kyoto, Japan, Aug. 2007.
2. A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford. In VINI Veritas: Realistic and Controlled Network Experimentation. In *Proc. ACM SIGCOMM*, Pisa, Italy, Aug. 2006.
3. S. Bhatia, M. Motiwala, W. Mühlbauer, V. Valancius, A. Bavier, N. Feamster, J. Rexford, and L. Peterson. Hosting virtual networks on commodity hardware. Technical Report GT-CS-07-10, College of Computing, Georgia Tech, Oct. 2007.
4. Y. Breitbart, M. Garofalakis, B. Jai, C. Martin, R. Rastogi, and A. Silberschatz. Topology discovery in heterogeneous ip networks: the netinventory system. *IEEE/ACM Trans. Netw.*, 12(3):401–414, 2004.
5. Cisco Discovery Protocol. http://www.cisco.com/en/US/docs/ios/12_1/configfun/configuration/guide/fcd301c.html.
6. CPR: Campus-Wide Network Performance Monitoring and Recovery. <http://www.rnoc.gatech.edu/cpr/>, 2006.
7. A. Dhamdhere, R. Teixeira, C. Dovrolis, and C. Diot. Netdiagnoser: Troubleshooting network unreachabilities using end-to-end probes and routing data. In *Proc. CoNEXT*, Dec. 2007.
8. N. Duffield. Simple Network Performance tomography. In *Proc. ACM SIGCOMM Internet Measurement Conference*, Miami, FL, Oct. 2003.
9. N. Feamster, L. Gao, and J. Rexford. How to lease the Internet in your spare time. *ACM SIGCOMM Computer Communication Review*, 37(1):61–64, 2007.
10. IEEE 802.1Q - Virtual LANs. <http://www.ieee802.org/1/pages/802.1Q.html>, 2006.
11. S. Kandula, D. Katabi, and J.-P. Vasseur. Shrink: a tool for failure diagnosis in ip networks. In *MineNet '05: Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*, pages 173–178, New York, NY, USA, 2005. ACM.
12. R. R. Kompella, A. Greenberg, J. Rexford, A. C. Snoeren, and J. Yates. Cross-layer visibility as a service. In *Proc. 4th ACM Workshop on Hot Topics in Networks (Hotnets-IV)*, College Park, MD, Nov. 2005.
13. R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren. Ip fault localization via risk modeling. In *Proc. 2nd USENIX NSDI*, Boston, MA, May 2005.
14. B. Lowekamp, D. R. O'Hallaron, and T. Gross. Topology discovery for large ethernet networks. In *Proc. ACM SIGCOMM*, San Diego, CA, Aug. 2001.
15. OpenFlow Switch Consortium. <http://www.openflowswitch.org/>, 2008.
16. OpenVZ: Server Virtualization Open Source Project. <http://www.openvz.org>.
17. G. H. Panagiotis Sebos, Jennifer Yates and A. Greenberg. Auto-discovery of shared risk link groups. In *Optical Fiber Comm. Conference (OFC). Volume 3. Issue 2001*, pages 1–3, 2001.
18. N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with Rocketfuel. In *Proc. ACM SIGCOMM*, Pittsburgh, PA, Aug. 2002.
19. Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, and J. Rexford. Virtual routers on the move: live router migration as a network-management primitive. In *Proc. ACM SIGCOMM*, Seattle, WA, Aug. 2008.