Path Inference in Data Center Networks

Kyriaki Levanti*, Vijay Gopalakrishnan†, Hyong S. Kim*,
Seungjoon Lee†, Emmanuil Mavrogiorgis†, Aman Shaikh†

*Carnegie Mellon University †AT&T Labs – Research

I. Introduction

Path inference is the ability to determine the sequence of devices traversed by a packet within an IP network. This is a fundamental building block for several network and service management applications such as troubleshooting, planning, and verification. For example, if a user has trouble accessing a service, one of the first things a network operator would like to determine is the path from the user to the device hosting the service. Only after knowing the path, can the operator start looking for problems along the path.

In this paper we focus on path inference in data center networks. Data center networks play vital role in today's Internet. Popular web-based services and critical enterprise applications are hosted in large data centers. Recent advances such as cloud computing and cellular-based data usage have also increased the importance of data centers. With the increasing importance, however, also comes increasing complexity. Supporting a wide array of applications and traffic types while meeting all their performance and security requirements results in complex network designs such as routing of traffic using a combination of layer-2 and layer-3 mechanisms and extensive use of overlays. The increasing complexity of data centers has made it difficult to manage these networks. Imagine a customer experiencing degraded performance. To troubleshoot the issue, network operators need to identify all the intermediate devices underpinning the path to the service. This requires operators to track and correlate information across layers – often a hard and time-consuming task for humans.

While there exist various tools and approaches to determine paths in a network, none of them can reliably provide information about all the devices in the path from a user to a service. For example, traceroute [1] is a widely used tool to identify the path taken by packets. While traceroute is useful during the occurrence of an event, if is not effective when analyzing past events. There are other sources of path information such as NetFlow, packet forwarding tables, etc. They, however, do not provide accurate path information either. As a result, there is a critical need for a generic tool that performs accurate path inference for data centers in a scalable manner.

In this paper, we present a system called Chartis (*map* in Greek) that is designed to satisfy this need. Given information about a packet's header (e.g., source/destination IP addresses), Chartis infers the physical path of the packet by using: (i) information about the physical connectivity of devices, and (ii) routing information from device configurations or control-plane messages. Specifically, Chartis emulates how network devices route packets on a hop-by-hop basis. Thus, Chartis not only provides the path, but also the decision logic used by each device to arrive at its routing decision (i.e., the next-hop). The path and the decision logic information provided by Chartis are extremely useful in network troubleshooting. Chartis also enables what-if analysis, by allowing operators to determine how paths change in the event of failure, maintenance or design changes in the network. The main contributions of this paper are:

- We present Chartis, a cross-layer path inference system for data center networks. Chartis finds paths consisting of layer-3 routers, layer-2 switches connecting these routers and other middle-boxes.
- We verify accuracy of Chartis by comparing its results with other data sources within a set of data centers operated by a major U.S.-based cellular service provider. The evaluation shows that Chartis is able to infer the entire end-to-end paths accurately within these data centers.
- Chartis is deployed and used as part of a system for visualizing end-to-end paths. We describe some real-world use cases and examples where Chartis played a key role in troubleshooting.

II. Related Work

Path inference can be performed through active probing (traceroutes), traffic monitoring (NetFlow records), and control plane monitoring (FIB dumps or routing protocol messages).

Traceroute [1] is the most common tool for inferring the layer-3 path of a probe packet to a specified destination IP. However, it is not always feasible to represent all possible types of data packets as traceroute probes. Moreover, it cannot be used to troubleshoot past problems, unless traceroute is run on a continuous basis between all source-destination pairs, which is prohibitively expensive for even moderate-sized networks. Finally, traceroute is often disabled on routers for security considerations.

NetFlow-based path inference is limited by its sampling ratio. In addition, paths can only be determined for the traffic that actually flows through the network. Address translation (e.g., NAT) also complicates path inference using this approach. Trajectory sampling [2] suffers from the same drawbacks.

Most routers allow users to extract the forwarding table itself through command line interfaces. This data can naturally

This work was supported in part by ICTI.

978-3-901882-48-7 © 2012 IFIP

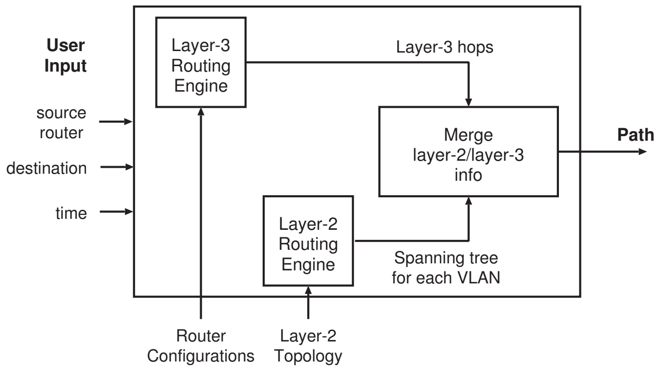

Fig. 1. Chartis System Architecture

be used for path inference. However, FIB extraction often places substantial load on the router, and hence, is avoided by network operators. Collecting messages exchanged by routing protocols is a more practical and non-intrusive approach. However, this data alone is not sufficient for path inference especially in data centers where routing is often governed by configuration (e.g., static routes).

From the discussion above, it is clear that performing accurate path inference at scale requires both routing protocol messages and device configurations. However, for the data centers we considered, routing protocol messages are not available. As a result, we primarily infer paths using configuration data, which shares some similarity with other works on configuration analysis [3]–[5]. We note though that Chartis can be easily extended to work with routing protocol messages if they are available.

III. The Chartis System

A. Chartis Overview

The goal of Chartis is to determine the path taken by a packet in the data center given information such as the source router/interface, select fields from IP packet header (e.g., destination address), and time. While the goal (and potential approach) is conceptually simple, realizing it in practice is complex since data centers typically use several routing protocols in an inter-dependent manner (e.g., VRFs, intra-domain routing protocols, static routing, route redistribution, VLANs etc.). They also employ a combination of IP layer (layer-3) and link layer (layer-2) routing and forwarding (refer to [6] for background on layer-3 and layer-2 routing). Moreover, data centers have middleboxes such as NAT and gateways, which may perform packet transformations (e.g., private to public IP address translation by NAT boxes).

Figure 1 presents architecture of Chartis, where it takes both layer-3 and layer-2 information, and emulates the routing and forwarding decisions performed at each hop to determine the path taken by a packet. Given a user's input, Chartis first determines the layer-3 path from the source to the destination, and then determines layer-2 path between every pair of hops in the layer-3 path. It then presents the combined layer-2/layer-3 path as its output. In the event that Chartis is unable to reach the destination (e.g., the destination is external to the data center), it returns as much of the path as possible.

```
input: dstIP, current_node, in_interface
while (current_node)
do
  # Get VRF for the incoming interface.
  vrf  = get_vrf (current_node, in_interface)

  # Get available routes to dstIP.
  i_route = connected_interfaces(vrf, dstIP)
  s_route = static_routes(vrf, dstIP)
  o_routes = ospf_routes(vrf, dstIP)
  route_set = {i_route, s_route, o_routes}

  # Select most preferred route(s).
  most_specific_routes =
                longest_prefix_match(route_set)
  selected_route = compare_ADs(most_specific_routes)

  # Find next-hop node and incoming interface.
  # The next-hop IP is given by selected_route.
  (in_interface, next_node) =
             find_nexthop(current_node, vrf,
                          selected_route.nexthop_IP)
  current_node = next_node
done
```

Fig. 2. Logic followed by the layer-3 routing engine.

Chartis primarily relies on two data sources to determine paths: router configurations and topology/connectivity information. Router configurations contain sufficient information to determine layer-3 paths. However, there are challenges associated with the parsing of router configuration files — especially those stemming from differences in configuration language across vendors or hardware/software platforms of the same vendor. Instead of attempting to parse and extract the details in individual configuration files, we leverage existing tools [7] to translate these files into a vendor-neutral format. Chartis then uses this pre-processed data to extract relevant information for route computation. While router configurations also contain layer-2 information (e.g., VLANs), this alone is not sufficient. Hence, Chartis relies on the layer-2 topology gathered using link layer discovery protocol (LLDP) or Cisco discovery protocol (CDP) to determine connectivity.

B. Layer-3 Routing Engine

This component emulates routing mechanisms configured on a router and determines layer-3 path in a hop-by-hop fashion. Given a destination address and an incoming interface on the router, the layer-3 engine determines the outgoing interface, the next-hop router(s), and the incoming interface on each next-hop router. Figure 2 shows the steps followed by the engine.

1. Identifying VRF from Incoming Interface

Modern routers allow multiple routing and forwarding tables, known as VRFs to co-exist. When a packet arrives on a router, it is forwarded by looking its destination address up in a VRF, which in turn is determined based on the incoming interface of the packet. Note that routers often allow virtualization of physical interfaces by combining multiple interfaces into a single unit or slicing a single interface into multiple units, which can be in different VRFs. From the point-of-view of Chartis though, an incoming interface allows unambiguous identification of the VRF of the incoming packet. In the case where a router is not using any VRFs, Chartis assumes there is a single VRF on the router, and assigns all the interfaces (including the incoming one) to this VRF.

2. Determining Next-Hop IP

Once the engine identifies the VRF, it considers only those routes in the specific VRF and tries to find prefixes that match the destination address. For this, it considers static routes and OSPF routes. Static routes are determined from the configuration, while determining OSPF routes requires Chartis to interface with an OSPF path calculation engine [3].

Among all the identified routes to prefixes matching the destination IP, Chartis extracts the route to the longest matching prefix and then finally determines the most preferred route based on the *Administrative Distance* associated with each route. In case there are multiple equal-cost paths to the destination (i.e., ECMP routing), a router selects the next hop using a router-specific hashing algorithm using packet header information. Since this hashing algorithm is proprietary, Chartis does not emulate ECMP and instead outputs all possible paths that a packet could take to the destination. While in theory, this could lead to an exponentially large set of paths, in practice we see that the number of possible paths is modest. Moreover, we see that despite the multiple paths, they eventually merge at a common aggregation point.

3. Determining Next-Hop Router and Interface

The next-hop provided by the most preferred route is an IP address, not the actual router and its incoming interface. While identifying the next-hop router and interface would be trivial if an IP address is unique across the network, we observe that some (private) IPs are assigned to multiple interfaces. The use of virtual IPs for fault tolerance can also make it challenging to map an IP to the router and interface. Chartis uses configuration information about VLANs, VRFs, and routing resilience mechanisms to identify correct next-hop interface; details can be found in [6].

C. Layer-2 Switching Engine

The layer-2 engine deals with path inference at layer-2 for Chartis. The engine starts with the layer-2 topology determined by LLDP or CDP, and the VLAN information specified in the configurations in order to generate the per-VLAN topology. Specifically, for each VLAN, the layer-2 switching engine searches through the configurations for interfaces that are explicitly assigned to this VLAN. Then, it retrieves the links that interconnect these interfaces. By combining the interfaces with the links, the engine arrives at the topology of the VLAN.

Next, the engine calculates the spanning tree for each VLAN in two steps: (1) Selection of the root bridge, where the switch with the lowest "bridge ID" becomes the root bridge. The bridge ID is computed based on the priority specified in the configuration, and the MAC address of the switch. (2) Calculation of the least cost path from every switch to the root bridge in terms of costs configured for layer-2 links. Once the spanning tree is known, determining the path between any two nodes in the tree is trivial.

D. Dealing with IP Address Transformation

In addition to routers and switches, data centers sometimes include devices that also modify destination and/or source address in IP headers of the packets. Example of such devices are Network Address Translators (NATs) and tunnel anchors. Examples of the latter include devices used as start and end-points of tunnels in cellular data networks [8], as described in Section IV-A. NATs usually change source or destination IP address of a packet, while tunnel anchors perform encapsulation/decapsulation of packets. Chartis handles such devices provided relevant information is available in the configuration files. For example, configuration of NATs usually specifies how IP addresses translation is done; Chartis uses this to change source or destination address of the packet. Configuration of tunnel anchors usually specify information about the tunnel and how packet should be encapsulated and decapsulated. While a packet is traversing a tunnel, it ends up containing multiple IP headers; in such cases, Chartis requires a user to specify fields from these multiple headers as input.

E. Limitations of Chartis

The current implementation of Chartis for the cellular data center is based on snapshots of router configurations and topology information for path calculation. However, these are static pieces of information. Data centers, on the other hand, may experience dynamics that cannot be captured using this information. For example, the current system may return an inaccurate path in the case of a failure (e.g., interface failure). Since Chartis abstracts the functionality of routing protocols assuming that they have reached steady-state, it cannot account for transient states during routing re-convergence after an event (e.g., device failure). This limitation, however, is not unique to Chartis and is applicable to other existing path inference techniques (e.g., [9]). In addition, data centers may have devices that make use of proprietary information, or decisions that are not captured in configuration. Chartis will not be able to capture such instances accurately. For example, in the presence of multiple equal-cost paths, Chartis returns all possible paths, but cannot predict the exact physical path of a particular packet. This is because the forwarding decision in case of equal-cost paths depends on a vendor-specific hash algorithm computed over the packet's header.

IV. Chartis Verification

In this section we present evaluation of our Chartis implementation in terms of its accuracy. We focus on presenting our results for data centers used by a cellular provider and briefly mention other results later in this section.

A. Dataset

Our primary data comes from data centers operated by a major U.S.-based cellular service provider. The four data centers used in our study are a subset of all the data centers operated by this provider to serve data traffic generated by millions of smartphones and laptops operating all over the country. The data centers include a wide array of network devices, ranging from general purpose routers and switches to specialized devices, some of which are specific to cellular data services (e.g., cellular gateways [8]). These devices come from multiple vendors and perform various roles to support different services and traffic types. We collect configurations for all the devices as well as use LLDP/CDP to collect link-layer connectivity between devices.

Each data center has a packet capture device deployed at the edge. For our evaluation, we used the header information (e.g., destination IP) of 4,000 packets captured by these devices in February 2011, and inferred path for each of these packets. These packets belong to multiple services (i.e., Access Point Names [8]), and include traffic coming from mobile devices to the data center (*forward* direction), or going back from the data center to mobile devices (*reverse* direction). Finally, for verification purposes, we collected routing table snapshots. For reasons beyond our control, we could get routing information from only a subset of devices. These devices, however, carry majority of the packets ($>$ 90%). We ensured that snapshots were collected around same time as the captured packets.

B. Verification Results

We ran Chartis using the information from 4,000 packets captured on two different dates. For each packet, we also know the direction (i.e., forward vs. reverse) and the collection location. The path inference in Chartis starts at the first node at which the packet entered the data center. For packets in the forward direction, this is the location at which the packets were captured. Determining entry point for reverse direction was not so trivial since the capture point itself is not at the entry, and some packets go through NAT devices before the capture point. Due to space constraint, we do not elaborate on this further.

We compared each path inferred by Chartis with the snapshots of layer-3 forwarding tables and verified that the Chartis paths match with the routing decisions made by individual routers on a hop-by-hop basis. We find from LLDP/CDP output that in these cellular data centers, all layer-3 hops use direct layer-2 connectivity. We also verified a few sample paths returned by Chartis with the network operators.

Other experiments: We also verified Chartis in a controlled environment by using active probing, traceroute data, and NetFlow data in a campus network. We refer readers to our technical report [6] for detail.

V. Case Studies

Chartis is deployed as a sub-component of a system used for visualizing end-to-end paths (from a cell tower to the Internet) for the cellular service provider mentioned in Section IV. The visualization system, which is heavily used by network operations, engineering and planning teams, relies on Chartis to provide path information within data centers. On an average business day in March 2012, the system served about 150 views to about fifteen unique users. The total number of unique users during the entire month was 115 across various organizations. Below we briefly describe two instances where the system enabled rapid isolation of service performance issues. In both cases, detailed path information provided by Chartis was critical in quick determination of the root cause and the network element responsible for the problem.

A. Service Latency Increase

On December 5th, 2011, monitoring infrastructure in the service provider detected latency increase for traffic going through a particular SGSN (Serving GPSR Support Node). After verifying that the SGSN continued to use the same GGSN (Gateway GSN), the operations team used the visualization system to examine paths within offices where the SGSN and the GGSN resided. Based on this information, the team identified that a route had changed between the last router in the SGSN data center and the first router in the GGSN data center. This detour was caused by a shutdown interface on a router along the path, which resulted in the latency increase.

B. Service Throughput Degradation

On February 25th, 2012, the monitoring infrastructure detected degradation in throughput for traffic going through a particular data center during peak hours. Multiple operations team members used the system to analyze paths within the office, and determined that a link on a router was operating at reduced capacity, leading to throughput degradation.

VI. Conclusions

This paper presented Chartis, a path inference system for data center networks. Chartis takes as input router configurations and topology data to emulate hop-by-hop path calculation and overcomes the limitations of existing path tracing tools such as traceroute. We verified that the inferred paths by Chartis match the set of devices actual packets traverse. We also demonstrated that how Chartis was used to perform various network management tasks such as service troubleshooting.

References

[1] V. Jacobson, “Traceroute,” ftp://ftp.ee.lbl.gov/traceroute.tar.gz, February 1989.

[2] N. Duffield and M. Grossglauser, “Trajectory sampling for direct traffic observation,” *IEEE/ACM Transactions on Networking*, vol. 9, no. 3, pp. 280–292, June 2001.

[3] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford, “Netscope: Traffic engineering for IP networks,” *IEEE Network*, vol. 14, no. 2, pp. 11–19, March/April 2000.

[4] D. Maltz, G. Xie, J. Zhan, H. Zhang, G. Hjalmtysson, and A. Greenberg, “Routing design in operational networks: A look from the inside,” in *Proceedings of ACM SIGCOMM*, 2004.

[5] G. Xie, J. Zhan, D. Maltz, H. Zhang, A. Greenberg, G. Hjalmtysson, and J. Rexford, “On static reachability analysis of IP networks,” in *Proceedings of IEEE INFOCOM*, 2005.

[6] K. Levanti, V. Gopalakrishnan, H. S. Kim, S. Lee, E. Mavrogiorgis, and A. Shaikh, “Path inference in data center networks,” http://www.research.att.com/~ashaikh/papers/chartis-tech-report12.pdf, AT&T, Tech. Rep., 2012.

[7] D. Caldwell, A. Gilbert, J. Gottlieb, A. Greenberg, G. Hjalmtysson, and J. Rexford, “The cutting EDGE of IP router configuration,” in *SIGCOMM CCR*, 2004.

[8] H. Kaaranen, A. Ahtiainen, L. Laitinen, S. Naghian, and D. V. Niemi, *UMTS Networks: Architecture, Mobility and Services*. Wiley, 2nd edition, 2005.

[9] A. Shaikh and A. Greenberg, “OSPF monitoring: architecture, design and deployment experience,” in *Proceedings of USENIX NSDI*, 2004.