Clairvoyant Networks

Cheng Jin*,Cristian Lumezanu†,Zhi-Li Zhang*,Haifeng Chen†
* *University of Minnesota,* †*NEC Laboratories America*
{*cheng,zhzhang*}*@cs.umn.edu,* {*lume,haifeng*}*@nec-labs.com*

***Abstract*—We use the term *clairvoyant* to refer to networks which provide *on-demand* visibility for any flow at any time. Traditionally, network visibility is achieved by instrumenting and passively monitoring all flows in a network. SDN networks, by design endowed with full visibility, offer another alternative to network-wide flow monitoring. Both approaches incur significant capital and operational costs to make networks clairvoyant.**

In this paper, we argue that we can make any existing network clairvoyant by installing one or more SDN-enabled switches and a specialized controller to support on-demand visibility. We analyze the benefits and costs of such clairvoyant networks and provide a basic design by integrating two existing mechanisms for updating paths through legacy switches with SDN, telekinesis and magnet MACs. Our evaluation on a lab testbed and through extensive simulations show that, even with a single SDN-enabled switch, operators can make *any* flow visible for monitoring within milliseconds, albeit at 38% average increase in path length. With as many as 2% strategically chosen legacy switches replaced with SDN switches, clairvoyant networks achieve on-demand flow visibility with negligible overhead.

I. Introduction

We use the term *clairvoyant* to refer to networks which provide *on-demand* visibility[1] for any flow at any time, *i.e.*, the ability to monitor *any* flow at *any* time *on-demand*. Traditional monitoring techniques, which deploy monitoring tools or devices [1], [2], [3] on the data plane, provide only static visibility; the placement of these tools and devices determines the flow visibility coverage. SDN networks are by design endowed with full visibility. Both approaches incur significant capital and operational costs to make (existing) networks clairvoyant.

In this paper, we present an alternative and cost-effective approach to make any existing network "clairvoyant" with *on-demand flow visibility*. A clairvoyant network consists of at least one SDN-enabled[2] switch and a specialized network controller. We adopt a *routing-and-monitoring* approach—by modifying the path of flows *on demand* to route them through monitoring devices and make them visible.

Clairvoyant networks aim to make flow monitoring more flexible by diffusing part of the cost required to set up and run a monitoring infrastructure to the myriad of flows being monitored. To keep the cost low, we require a handful (at least one) strategically placed SDN switches (e.g., by replacing one or a few legacy switches with SDN-enabled switches or simply installing new ones) within an existing network. SDN switches allow operators to program the data plane remotely to meet both routing and measurement goals [4], [5] and double as monitoring devices by supporting counting [6] and inspecting [7] packets or through custom monitoring scripts [8]. Our specialized controller can redirect flows through SDN switches, by incorporating two mechanisms, telekinesis and magnet MACs introduced in our previous work [9].

To show that clairvoyant networks can provide significant advantages to monitoring, we perform a measurement study on their benefits and costs. We first study the visibility that clairvoyant networks offer (Section III). Using real-world and synthetic topologies, we show that even one OpenFlow switch enables monitoring of any flow with many possible paths to choose from. Second, we study the cost of enabling network-wide visibility (Section IV) by answering the question of how much the flow and network performance degrades in exchange for visibility. Clairvoyant networks give operators a trade-off between the upfront cost to enable SDN-based monitoring and the performance penalty incurred by enabling such monitoring.

In the second part of the paper, inspired by our measurement results, we present a design for clairvoyant networks. We show how to integrate the existing mechanisms of telekinesis and magnet MACs [9] with the visibility tasks to design the clairvoyant controller (Section V). For the purpose of informing network architects and operators of the trade-offs in making their networks clairvoyant, we identify several key performance and cost indicators. We then provide a customized design, including a balanced SDN deployment strategy and a flow scheduling mechanism, that reduces both the upfront deployment cost and the flow and network overheads to offer a practical solution for deploying multiple visibility tasks at the same time (Section VI).

Clearly, modifying the path of a flow to make it visible is an intrusive mechanism which may change the properties (e.g., flow completion time) of the flow being monitored. While we show (Section IV) that the overhead required to enable visibility is small, it is ultimately at the latitude of network operators and users to decide if the cost is worth paying. For example, enterprise networks, where a few extra milliseconds of latency may be acceptable, are a better candidate for clairvoyant networks than data centers where application traffic must be delivered on time to maintain user satisfaction. In addition, privacy concerns may arise when flows are routed through forbidden parts of the network. As we describe in our design (Section V), operators can specify constraints on the path of flows.

[1]In this paper *visibility* focuses on "who is talking to whom and on what ports," namely, host-level communication patterns. We do not consider performance visibility issues here.

[2]We interchangeably use the terms SDN(-enabled), OpenFlow(-enabled), or programmable to refer to devices whose forwarding tables can be configured remotely from a centralized controller.

978-3-903176-17-1 / © 2019 IFIP

Clairvoyant networks provide a low-cost flexible monitoring substrate for enterprises where changing a flow's path is acceptable. They open up new directions in flow monitoring by allowing *hybrid monitoring applications* that take advantage of the monitoring capabilities of both SDN and legacy devices to build accurate, flexible and efficient monitoring.

II. Clairvoyant networks

SDN-based monitoring. A network flow is *visible* when its path traverses a monitoring device, such as an NetFlow/sFlow-enabled switch, a polling-enabled SDN switch, or any dedicated monitoring or packet capture appliance. Network-wide visibility of all flows is critical for network management applications such as traffic engineering, access control, anomaly detection, or heavy hitter detection [10], [11], [8], [12].

Traditional flow monitoring achieves visibility by defining static monitoring tasks that require switch support [1], [13] or dedicated monitoring appliances [14], [3]. For example, to identify large flows, NetFlow/sFlow-enabled switches sample packets and build flow-level packet counters. Monitoring tools must be strategically deployed across the data plane to enable network-wide visibility, and carefully tuned to avoid overloading the data plane [15]. Offloading the monitoring tasks to specialized off-path appliances by mirroring packets, *e.g.*, using SPAN [2] or TAP [14], may relieve the load on the data plane, but requires careful coordination to avoid oversubscribing the mirroring ports or paths and may not be amenable to real-time traffic analysis.

SDN disrupts traditional monitoring practices by providing better control and visibility over the network. First, SDN allows operators to remotely update switch forwarding entries on demand, enabling more flexible and dynamic monitoring tasks [8], [16], [17]. Second, SDN-enabled switches double as monitoring devices. They support flow-based counters to monitor utilization [18], [19], [20] or help inspect traffic to detect unauthorized access [4], [5] or security threats [7].

An important impediment to SDN-based monitoring has been the significant upfront investment cost it requires. Upgrading the network to SDN is prohibitive for most enterprises as it requires replacing most, if not all, legacy switches with SDN-enabled switches [21]. Recent work proposes hybrid SDN and legacy (or partially programmable) networks to lower the deployment cost of SDN while providing most of its benefits. However, with hybrid networks, operators have visibility only over the flows that traverse the SDN switches and cannot monitor the traffic in the legacy part [22], [21].

Proposed idea. In line with previous research [18], [19], [20], [7], we consider a flow to be visible[3] when it traverses an SDN switch. We propose to make *all* flows visible *on-demand* in a hybrid network by redirecting them (temporarily or permanently) through an SDN switch. In this way, operators could apply existing SDN-based monitoring mechanisms to monitor all flows, including those whose default path does not traverse an SDN switch. When monitoring is finished, the flows could be reverted to their original path. This would dramatically decrease the cost of deploying and using SDN-based monitoring, as a wholesale upgrade to SDN is not necessary to enable network-wide visibility.

[3]Throughout the paper, a flow is "visible" when it traverses an SDN switch and "invisible" otherwise. In Section III, we discuss how to make a flow visible to legacy monitoring devices rather than SDN switches.

Towards this goal, we introduce *clairvoyant networks*: partially programmable networks that offer operators the ability to monitor any flow any time on demand. Any enterprise network can become clairvoyant by deploying as few as one SDN-enabled switch and a specialized network controller, which we call the clairvoyant controller.

Clairvoyant networks are made possible by a path update mechanism initiated by SDN switches to control routing through legacy devices [9]. As described in Section V, we can change the path of any flow traversing the legacy network using OpenFlow-based mechanisms. The granularity of visible flows ranges from per source-destination to a specified tuple of packet header fields. While the path update mechanism works at the source-destination granularity, SDN switches can slice and monitor flows at a finer granularity (*e.g.*, port numbers) once flows' paths have been updated.

Clairvoyant networks raise several questions about the feasibility and cost of flow monitoring by changing the paths of flows. First, how many flows can we make visible by updating their paths compared to a simple hybrid network? While clairvoyant networks focus on SDN-based monitoring (*i.e.*, a flow is visible when it traverses an SDN-enabled switch), is it possible to redirect flows through traditional monitoring devices (*e.g.*, NetFlow/sFlow-enabled switches). Finally, what are the cost and performance trade-offs involved in changing the path of a flow to make it visible? We explore these questions through data-driven simulations and real-world deployments in Sections III and IV. We then present a basic design for clairvoyant networks in Section V.

III. Flow visibility

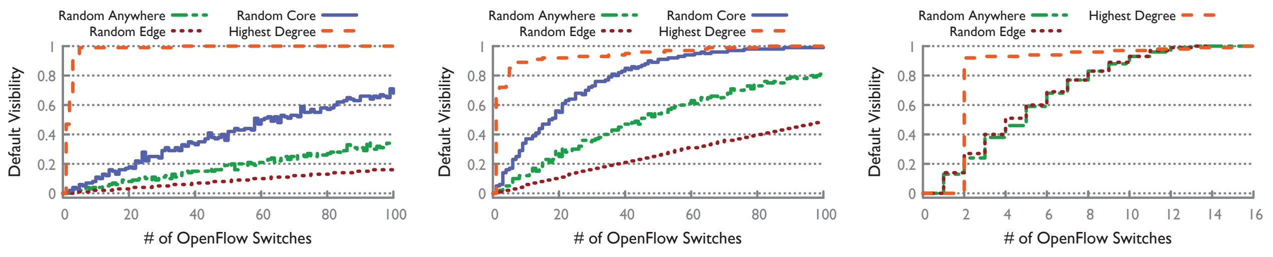
Do clairvoyant networks make more flows visible than simple hybrid SDN networks that have no flexibility to update legacy paths? In this section, we investigate the extent to which clairvoyant networks provide visibility both through SDN switches and using legacy monitoring devices.

A. Methodology

Network topologies. We evaluate the feasibility of clairvoyant networks on three network topologies, described in Table I. The "Large" and "Small" are the real topologies of a large-scale campus network [23] and of a smaller campus backbone network [24]. We generate the "Medium" topology

Name	Source	# Switches/Edge/Core	Max/Avg/Min Degree
Large	[23]	1577 / 1160 / 417	65 / 2.15 / 1
Medium	this paper	493 / 355 / 138	19 / 3.11 / 1
Small	[24]	16 / 14 / 2	15 / 4.5 / 3

Table I: We use two real-world ("Large" and "Small") and one synthetic ("Medium") network topologies to demonstrate the feasibility of clairvoyant networks.



(a) "Large" topology (b) "Medium" topology (c) "Small" topology

Figure 1: Default visibility, as we vary the number and placement of OpenFlow switches.

to model a medium-size enterprise network. In doing so, we try to preserve the features observed in the real "Large" topology: more edge switches than core switches, and multiple components connected through high-degree core switches.

Deployment. We consider four placement strategies for SDN-enabled switches: random anywhere, random edge, random core, and highest-degree. Random strategies replace a legacy switch with an OpenFlow switch at random. Random anywhere and random core provide base cases for comparison, while random edge is intended to model a scenario where operators deploy software switches on edge hypervisors or servers. The highest-degree strategy replaces legacy switches in decreasing order of their degree and reflects a best case scenario where upgrading the most "influential" switches first.

Network flows. We consider all flows that could be installed in the network, *i.e.*, between all pairs of edge switches. We do not take into account the popularity of a pair of switches (*e.g.*, some edge switches connect to more hosts) because it does not affect the visibility of a flow. We assume a flow is between two different IP addresses, without taking into account port numbers, to match the granularity provided by the path update mechanism [9]. Unless otherwise noted, every experiment provides aggregated values over 100 runs, resetting the switch placement after each run.

Visibility. The *(flow) visibility* of a network is the probability that a random flow is visible, *i.e.*, traverses a monitoring device. Visibility takes values between 0 and 1. All flows in a network with visibility 1 can be monitored. For example, a network where all switches and routers support NetFlow or where all switches are SDN-enabled has visibility 1. We further classify visibility according to the type of device that provides it. *Natural visibility* represents the visibility achieved from monitoring flows at SDN-enabled switches, while *supervisibility* characterizes a network where flows are monitored at legacy monitoring devices such as NetFlow-enabled routers or IDSes. We measure both the natural and supervisibility that a clairvoyant network provides while varying both the number of OpenFlow switches and their placement strategy.

B. Natural visibility

Natural visibility describes the ability of a clairvoyant network to make any flow visible by routing it through an SDN-enabled switch. As the controller can set up any path through an OpenFlow switch, *the natural visibility of any clairvoyant network is 1.*

However, part of the natural visibility may not even require setup from the controller: if the flow's default path traverses an OpenFlow switch, then it is not necessary to use the clairvoyant controller to make it visible. To understand the benefit that clairvoyant networks provide, we must evaluate how much of their natural visibility is achieved using the clairvoyant controller. For this, we compute the *default visibility*: the probability that any flow is visible initially on its default path.

Figure 1 and Table II show the default visibility of each network, as we vary the number and placement of OpenFlow switches. When replacing more switches, more flows are likely to be visible initially. The highest-degree placement performs best, since high-degree nodes partition the network in many separate connected components. Most flows are likely to be between components, so they must traverse a high-degree node. This result implies most flows are visible by default when upgrading the top highest degree legacy switches to SDN. However, upgrading those switches is also costlier as they would support more flows and higher throughput.

Although the ability to set up a flow's path through an OpenFlow switch is important, the number of possible paths is equally critical. Path diversity offers operators more flexibility in reaching both monitoring and routing goals in path setup. Figure 2a shows the average number of paths that enable visibility for the flows whose default paths are not visible, *i.e.*, do not traverse an SDN-enabled switch, in the "Large" topology (Table II shows results for all topologies). Replacing the high-degree switches increases path diversity and enables more flexible monitoring. Path diversity is significant, regardless of the switch placement strategy.

C. Supervisibility

Although OpenFlow switches provide monitoring capabilities, being able to use traditional monitoring devices in a clairvoyant network, such as NetFlow-enabled legacy switches, may alleviate some of the monitoring load on OpenFlow switches. While all flows can be set up through a specific OpenFlow switch, not all flows can be set up through a particular legacy device. In fact, flow paths can be set up through a legacy device only if the device is on a path between an OpenFlow switch and the source or destination of a flow. The supervisibility reflects the ability of a clairvoyant network to set up paths through legacy devices.

We compute the minimum number of legacy monitoring switches necessary to achieve network-wide supervisibility,

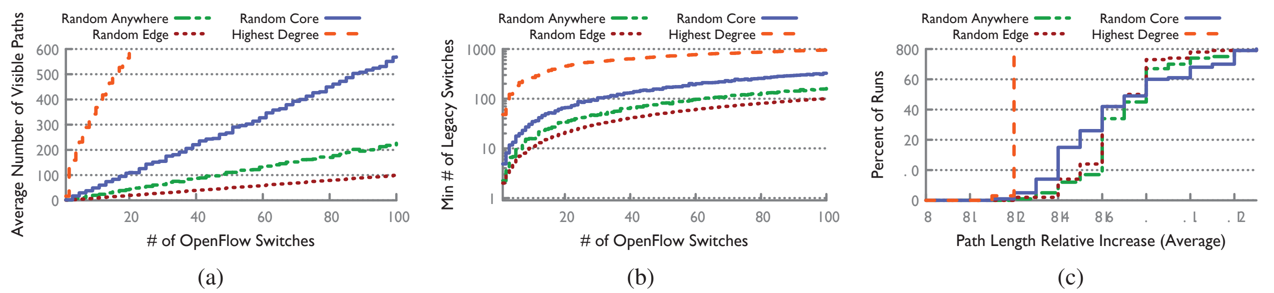


Figure 2: In the "Large" topology, (a) the average number of possible visible paths for flows whose default paths are not naturally visible, (b) the minimum number of legacy monitoring devices needed to achieve full supervisibility, and (c) the average flow stretch increase for the top five shortest visible paths when we have one OpenFlow switch.

i.e., any flow's path would traverse at least one of these legacy switches. Figure 2b presents the results for the "Large" topology. Interestingly, the highest-degree strategy performs poorly compared to the other strategies: more monitoring-enabled legacy switches are needed to cover all flows and achieve a supervisibility of 1. This is because there are more paths through highest degree switches and we need more legacy monitoring devices to cover all of them.

One interesting finding is that when we place OpenFlow switches at edge, the minimum number of legacy switches needed to cover all flows is lowest and the same as the number of OpenFlow switches. The reason is any SDN switch can redirect all the flows to go through itself and then through one of its adjacent legacy switches. Of course, placing few switches at the edge may increase the path length unnecessarily.

To maximize the number of visible flows it sees, a monitoring-enabled legacy switch should be located as close to an OpenFlow switch as possible. We confirm that all legacy switches in the experiments from Figures 2b are indeed adjacent to OpenFlow switches. This observation also defines an upper bound on how many monitoring-enabled legacy switches we need to cover all flows: the total number of active interfaces on all OpenFlow switches.

IV. The Cost of Visibility

Setting up flow paths through monitoring devices may introduce performance penalties to flows and overhead in the network. While monitoring applications may have their own overhead, here we focus on several key cost indicators related the effect of updating the path of a flow and whose value depends little, or not at all, on how flows are monitored.

A. *Overhead on flows*

How does visibility affect the performance of a flow? We evaluate two flow performance metrics. The *flow stretch* represents the relative increase of the new flow path's length compared to the default path. It reflects the penalty in end-to-end latency that a flow would pay for becoming visible. The *flow stress* captures the maximum number of other distinct flows with which a flow shares any link. Flow stress models the change in throughput that a flow may see when it becomes visible, and captures the ability of clairvoyant networks to offer monitoring paths that are lightly loaded.

We compute the average flow stretch of the top five shortest visible paths for each flow for all runs. Figure 2c shows the detailed results for when we have a single OpenFlow switch; Table II shows statistics for more switches. As expected, placing OpenFlow switches at the edge has the largest performance penalty, since a visible path may need to stretch to the other side of the network. The results show that with only 2% of switches upgraded to OpenFlow, the average visible path is only 1.3 times greater than the default path. This means that, even given the choice between several paths, a monitoring application would still likely select a fairly short visible path for a flow that is not visible by default.

B. *Overhead on the network*

How does visibility affect the network links? We define the *network stress* as the maximum number of flows that traverse any link in the network. Table II shows the relative increase in network stress across various placement strategies. High-degree strategies do not add much to the network stress when making flows visible, while the other strategies require more OpenFlow switches to keep the network stress low.

How does visibility affect the network switches? The OpenFlow switches may see overhead increase in clairvoyant networks, when compared to simple SDN networks, as they are queried more frequently or mirror traffic for further analysis. We consider three metrics for the cost imposed on switches in clairvoyant networks—memory usage, CPU utilization, and the number of forwarding entries—and study each metric as we increase the number of flows made visible.

First, we measure the CPU utilization and memory usage on an iwNetworks OpenFlow switch in two scenarios: when the clairvoyant controller polls the flow statistics every second and when the switch mirrors traffic (*e.g.*, to the clairvoyant controller or a dedicate server). Previous research [25] shows that the performance of OpenFlow switches decreases as the controller polls for statistics. Mirroring packets to the controller, on the other hand, packs the captured packets as the payload of PacketIn messages [26], which is done by the switch's CPU. Table III shows the results as we increase the number of concurrent flows. Clairvoyant networks add little overhead to the SDN switches even with many flows being monitored at the same time.

		Random anywhere			Random edge			Random core			High-degree	Every-edge
Large	**# OF switches**	1	5	20	1	5	20	1	5	20	1	25
	Default visibility	0.0	0.02	0.08	0.0	0.01	0.03	0.01	0.06	0.18	0.48	0
	Possible paths	1.4	10.9	44.4	1.0	5.0	20.0	2.9	29.6	110.4	16.6	2320+
	Min switches (for supervisibility)	2.24	9.47	33.77	1.0	5.0	20.0	4.86	17.72	66.33	48.0	25
	Flow stretch	1.9	1.4	1.3	1.8	1.5	1.4	1.7	1.4	1.2	1.4	1.2
	Flow stress increase	21.2	7.6	3.7	21.3	7.5	3.2	20.9	7.7	3.2	7.7	1.0
	Network stress increase	4.4	2.3	1.5	4.4	2.1	1.5	4.3	2.2	1.3	2.1	1.0
Medium	**# OF switches**	1	5	20	1	5	20	1	5	20	1	8
	Default visibility	0.01	0.06	0.25	0.01	0.03	0.11	0.05	0.17	0.55	0.71	0
	Possible paths	3.2	16.3	60.6	1.0	5.0	20.0	8.2	42.6	173.2	19.7	710+
	Min switches (for supervisibility)	2.54	8.18	29.37	2.0	6.0	21.0	3.72	13.62	56.19	6.0	8
	Flow stretch	1.9	1.4	1.2	2.0	1.5	1.3	1.7	1.3	1.1	1.5	1.2
	Flow stress increase	10.1	3.6	1.6	10.5	4.0	1.6	9.3	2.7	1.4	2.7	1.0
	Network stress increase	3.6	1.8	1.2	3.7	1.9	1.1	3.3	1.5	1.0	1.1	1.0
Small	**# OF switches**	1	5	10	1	5	10	1	-	-	1	1
	Default visibility	0.13	0.59	0.93	0.14	0.6	0.93	0.0	-	-	0.0	0
	Possible paths	6.7	31.8	41.7	3.3	15.9	30.5	34.1	-	-	34.1	28+
	Min switches (for supervisibility)	1.89	5.9	11.2	2.0	6.0	11.0	1.0	-	-	1.0	1
	Flow stretch	1.5	1.1	1.0	1.9	1.3	1.3	1.2	-	-	1.2	1.5
	Flow stress increase	5.8	1.5	2.2	6.7	2.0	1.5	0.9	-	-	0.9	1.1
	Network stress increase	3.2	1.0	1.0	3.5	1.4	1.2	0.5	-	-	0.5	1.0

Table II: Results for visibility and cost metrics for the three topologies.

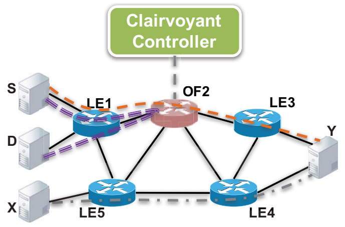


Figure 3: Clairvoyant networks require as few as one SDN-enabled switch. The clairvoyant controller can make the flow (S, D) and the flow (S, Y) visible to switch $OF2$ by setting up the paths. (X, Y) is an invisible flow.

V. Design

Any enterprise network can become clairvoyant by deploying at least one SDN-enabled switch and a specialized controller—which we call the clairvoyant controller.

The controller consists of two layers: path update and visibility enabler. It receives visibility tasks from operators specifying what flows to monitor and, if necessary, updates the paths of the flows to make them visible. The visibility enabling layer reads and schedules visibility tasks. How to monitor a flow, *i.e.*, polling specific counters, sampling packets, checking header field values is a separate process, at the latitude of the operator, and outside the design of the clairvoyant controller.

Changing paths. The clairvoyant controller can change the path of any flow (*i.e.*, per source-destination pair), even when the flow does not traverse any SDN switches. This allows us to gain visibility over any flow without the need to modify existing legacy hardware devices or software components. As shown in Figure 3, our clairvoyant controller can make the flow (S, D) visible to the SDN-enabled switch $OF2$. We summarize the design and properties of the path update framework below.

Number of flows	CPU (Query)	CPU (Mirror)	Mem (Query)	Mem (Mirror)
1	0.05 %	2.26 %	0.22 KB	0.33 KB
10	0.15 %	2.37 %	0.22 KB	0.51 KB
100	1.05 %	5.31 %	0.22 KB	2.12 KB

Table III: CPU and memory load increase on an OpenFlow switch: (i) when the controller polls the switch for statistics per second; (ii) when the switch mirrors packets to the controller.

Updating the path of any flow incorporates two mechanisms, *telekinesis* and *magnet addresses*, originally described by Jin *et al.* [9] and summarized in this section. With telekinesis, OpenFlow switches send special *seed* packets to the legacy switches on the new path to be installed. This relies on the ability of an SDN controller to use PacketOut control messages to instruct OpenFlow switches to send custom-made packets into the network. The seed packets take advantage of MAC learning to *manipulate* legacy switches into updating a single forwarding entry in their routing tables.

The path update framework routes using fictitious MACs (called *magnet* MACs) associated with hosts. Magnet MACs are fictitious MACs that do not correspond to any real host on the network, but are created by the controller for the purpose of controlling routing & forwarding behaviors of hosts and legacy switches. When sending seed packets, the source MAC is set as a magnet MAC associated with the path destination. The seed packet triggers the installation of a forwarding entry for the magnet MAC. The seed packets are also required to be ARP packets and can reach the source host of the path. Thus, the source learns to associate the destination with its new magnet MAC. The path update framework uses different magnet MACs to set up different paths for delivering traffic from other source hosts to the same destination. The last OpenFlow switch on each path rewrites the magnet MAC to the native MAC based on the destination IP address.

To make the flow between S and D visible to the SDN-enabled switch $OF2$, we update its path from $S - LE1 - D$ to $S - LE1 - OF2 - LE1 - D$. To install this new path, the controller crafts a seed packet with source MAC as a magnet MAC (*e.g.*, *MAGNET*) and destination MAC as S's

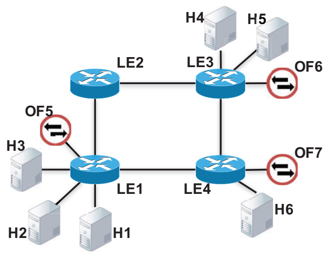


Figure 4: In a clairvoyant network, we can place SDN-enabled switches in every-edge—connecting each edge legacy switch (LE1, LE3, or LE4) to one SDN switch (OF5, OF6, or OF7).

MAC (in the Ethernet header), source hardware address as *MAGNET* and source protocol address as D's IP address (in the ARP header). Our controller uses PacketOut to send this seed packet from $OF2$ to S. This packet triggers the addition of a new forwarding entry in $LE1$ for the *MAGNET* MAC with corresponding incoming port and the update of the ARP table on S. Another seed packet with *MAGNET* MAC and S's IP address is sent from $OF2$ to D, and the ARP table on D is updated in a similar manner.

Enabling visibility. To make flows visible, we provide a simple language for network operators to create *visibility tasks* for the clairvoyant controller. With a visibility task, the operator simply sets up a flow to be monitored at a specific location in the network. A visibility task consists of an action, a monitoring target, a monitoring location, and optionally, a monitoring mirror. The action specifies whether the controller should add or delete the task. The monitoring target is a tuple of (source IP, destination IP) and represents the source and destination of the flow to be monitored. The monitoring location represents the SDN switch where the flow will be monitored. If operators do not have a preference for the location, the field is null. For example, the visibility task "(S, D) $OF2$" indicates that the flow (S, D) will be monitored at $OF2$, "$(*, Y)$ NULL" indicates any flow to Y can be monitored anywhere. Optionally, the operator can specify a monitoring mirror to have the flow mirrored to another device.

The clairvoyant controller takes visibility tasks as input and translates them into seed packets with magnet MACs that, in turn, generate forwarding rules that change the path of the flows. Setting up a path using the magnet MACs follows the description in Section V. Disabling a visibility task is similar and it requires the controller to send seed packets that route the flow back to the default path. In Section VI, we describe a more complex task scheduling mechanism, inspired by experimental results, and designed to reduce the cost of achieving visibility for multiple flows at the same time.

VI. Case study: edge visibility

Here, we consider a specific deployment scenario and associated design decisions that enable visibility for flows with negligible performance degradation.

SDN switch deployment. To reduce the path stretch of monitored flows, we propose to introduce a few (hardware or software) SDN switches to connect to all edge legacy switches (*i.e.*, all legacy switches that connect to end hosts) such that each edge switch connects to at least one SDN switch. In this way, changing the path of any flow adds at most two hops (from the edge legacy switch next to the source or destination to the connected SDN switch and backward). Figure 4 shows an example with three edge legacy switches connected to SDN switches. To make the flow $(H1, H5)$ visible, the controller redirects it through $OF5$ or $OF6$.

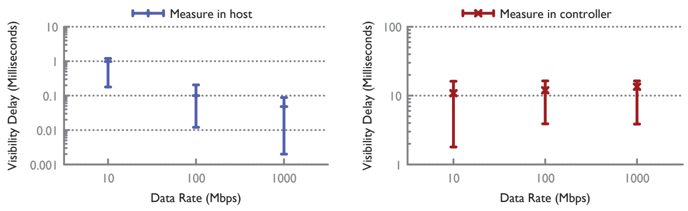


Figure 5: Visibility delay (the time to make a flow visible) remains low as we vary the data rate. We measure the visibility delay from both the host (left) and the controller (right).

By pushing visibility to the edge of the network, we guarantee that any flow has negligible performance degradation when made visible. In addition, non-target flows that traversing different edge switches do not get affected at all, since we scope the path change to only between an edge legacy switch and its adjacent SDN switch. However, when multiple flows are monitored by the same SDN switch on the same link, they may compete for the bandwidth. We discuss how to alleviate this problem later in this section.

The last column in Table II shows the cost of this deployment strategy. With 48-port hardware OpenFlow switches and each port connected to one edge legacy switch, we need only 2% more OpenFlow switches to cover every edge switch. An alternative, cost-effective deployment is to do away with hardware switches and deploy software switches on additional servers connected to the edge legacy switches. As expected, the average flow stretch and stress are smaller than other deployment strategies with the same number of SDN switches. Flow paths can extend on the average 1.5 times when made visible, while the competition for the same monitoring device is slightly higher than on the default path.

Visibility scheduling. When multiple flows are made visible through the same SDN switch, they will compete for the capacity of the link(s) connecting the SDN switch to its adjacent legacy switches. We propose time-based scheduling: one or more flows become visible in separate time slots such that the throughput of all flows in the same slot is lower than the capacity of the shared link.

First, the controller measures the throughput of each competing flow in a round-robin manner: it makes each flow visible for a small period (*e.g.*, 1s) and polls the counters associated with flow at the end of the visibility period. In Section VII, we show that making a flow visible and reverting it back to its original path is fast and consumes negligible resources.

Second, the controller combines all visibility tasks with the same monitoring locations in such a way that the sum of the throughputs of all flows from the same group of tasks does not exceed the capacity of the shared network link. We use a greedy heuristic to assign groups of tasks to each monitoring link at each monitoring interval. The visibility tasks in each group are enabled for each interval then disabled then enabled again until a task is deleted.

VII. Evaluation

In this section, we first show the clairvoyant controller can enable a flow's visibility fast while introducing negligible performance degradation. Second, we demonstrate the clairvoyant controller is scalable—can handle tens of thousands of simultaneous visibility tasks on one OpenFlow switch. We perform experiments on a real-world testbed in our lab. The testbed consists of six Dell servers (each with four 1 Gbps Ethernet interfaces), five Cisco Catalyst legacy switches [27], and two iwNetworks OpenFlow switches [28]. Each experiment is conducted for 100 times.

A. Visibility delay

We define the visibility delay as the time it takes to make a flow visible, *i.e.*, to update its path to traverse an SDN switch. We measure the visibility delay from the controller and from one of the endpoints of the flow. The controller visibility delay is the time between when the controller sends the first seed packet and when it receives the first mirrored packet. The endpoint visibility delay is the time between when the host receives the first seed packet and when it sends the first data packet on the new path.

To measure the visibility delay, we connect two servers and an SDN switch to a Cisco legacy switch. We start a flow between the two servers and vary its data rate. Initially, the flow traverses only the legacy switch, but goes through the SDN switch once we submit a visibility task for it.

Figure 5 (left) shows the visibility delay measured from the end host. It remains low when we increase the data rate. That the visibility delay decreases as we increase the data rate is an artifact of our measurement: when the data rate is low, the time between two consecutive packets is higher therefore our measurement error is higher. Figure 5 (right) shows the visibility delay measured from our clairvoyant controller. It is higher than the delay measured from the end host, because it contains (1) the round-trip time from the controller to the OpenFlow switch where the seed packet is injected, and (2) the round-trip time from the OpenFlow switch to the host. The first round-trip time is dominant due to the overhead involved in forwarding a data packet on the control channel.

Next, we set the data rate at 100 Mbps and increase the number of hops between the OpenFlow switch and the edge legacy switch. We find that the providing visibility does not significantly increase the flow completion time, even when sending flows on a five-hop hairpin path. On the other hand, we can keep a flow visible for as little as 0.1ms—the minimum amount of time we achieved between sending two consecutive seed packets. However, ARP implementations often have protection against ARP trashing, which limits the time between consecutive updates to the same ARP entry to one second. As a result, in practice, the smallest amount of time to maintain a flow's visibility is one second. Even with such a small visibility window, repeatedly enabling and disabling the visibility of a flow does not reduce its completion time. We observed only a 0.38% increase for a 10 GB flow when we enable and disable its visibility every second for 89s.

B. Scalability

Switch load. We evaluate clairvoyant networks when the OpenFlow switches are heavily-loaded using the same setup. Saturating the control plane to 99% CPU usage increases the visibility time measured on the controller by about 100 ms and does not change that measured on the host. In terms of high data plane (DP) load, we introduce 10 Gbps more background traffic to the OpenFlow switch and observe that the visibility time from both the controller and the host is not affected. The flow completion time changes are negligible among the cases where there is no additional load, high CPU load, and high data plane load.

Many visibility tasks. How does the clairvoyant controller perform when operators submit many simultaneous visibility tasks? We vary the number of visibility tasks and measure the time it takes the controller to enable them. A visibility task triggers two seed packets, one to the source host(s) and the other to the destination host(s). There are no flows running for this experiments; we measure the time for the clairvoyant controller to inject seed packets and insert the forwarding rules. Our result shows that the clairvoyant controller is capable of serving 15,000 individual visibility tasks on one OpenFlow switch in one second.

Next, we want to understand what happens when each visibility task updates an existing flow. In this experiment, we generate 100 flows and submit a visibility task for each flow. We are unable to generate more than 100 flows due to the limited number of servers in our testbed. Results in Table III show that making 100 flows visible simultaneously increases CPU usage by 5.31% and memory usage by 2.12 KB.

VIII. Discussion

Who can use clairvoyant networks? Primarily enterprises that require on-demand monitoring of their applications while accepting little performance degradation. As they may increase the application latency by rerouting flows through monitoring devices, clairvoyant networks are not suited for enterprises that run latency-sensitive applications. For such specialized networks, hardware-based solutions installed on the data plane provide a better benefit/cost trade-off for flow monitoring [10].

Deployment of clairvoyant networks in any enterprise is straightforward. Operators need to add at least one OpenFlow switch and the clairvoyant controller. To enable monitoring, one could proactively set up routes among all hosts through monitoring devices (for network-wide monitoring) or set up paths when flows start (for selective on-demand monitoring).

Interoperability with VLANs works by setting the interfaces of OpenFlow switches as trunk ports and crafting the seed packets with specific VLAN IDs (*e.g.*, the same VLAN

IDs as the access ports where the source and destination hosts are connected). This ensures that OpenFlow switches send seed packets and receive data packets on individual VLANs.

L3 routers restrict broadcast domains and connect subnets in enterprise networks. To enable flow visibility in L3 networks, clairvoyant networks could apply the technique proposed in Mille-Feuille [29] to collect traffic slices: configure ERSPAN [30] and leverage Fibbing [31] to program IGP. Combining clairvoyant networks and Mille-Feuille could achieve on-demand fine-grained flow visibility across the entire network (*i.e.*, both L2 and L3).

Programmable monitoring platforms offer customizable and dynamic monitoring by relying on the visibility and control provided by SDN [8], [20]. Clairvoyant networks open new directions for programmable monitoring by allowing flexible monitoring tasks that capture and analyze data from both OpenFlow and legacy devices.

IX. Conclusions

We introduced *clairvoyant networks*, which provide *on-demand* visibility for any flow at any time. We studied the feasibility of clairvoyant networks using real-world and emulated network topologies and showed that, even with a single SDN-enabled switch, operators can make *any* flow visible, albeit by increasing the average path length by 38%. When clairvoyant networks contain more SDN-enabled switches (as little as 2% of all switches), most flows can also be monitored on the legacy data plane with little impact on network performance. We also provided a basic design for clairvoyant networks by integrating an existing path update mechanism with a novel approach to specify and compile visibility tasks. Inspired by the feasibility study, we proposed a specific deployment scenario for clairvoyant networks. By connecting all edge legacy switches to at least one OpenFlow (hardware or software) switch and implementing flow scheduling in the clairvoyant controller, we are able to significantly reduce the cost of making (simultaneous) flows visible.

X. Acknowledgments

We are grateful to TMA anonymous reviewers for their insightful comments. The research was supported in part by US DoD DTRA grant HDTRA1-14-1-0040, and NSF grants CNS 1618339, CNS 1617729, CNS 1814322 and CNS183677.

References

[1] "Introduction to cisco ios netflow - a technical overview." [Online]. Available: https://goo.gl/55rYF9

[2] "Configuring span and rspan." [Online]. Available: https://goo.gl/TDCAm8

[3] M. Roesch, "Snort - Lightweight Intrusion Detection for Networks," in *LISA*, 1999.

[4] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: Taking control of the enterprise," in *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4. ACM, 2007, pp. 1–12.

[5] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, and M. Martin, "Fresco: Modular composable security services for software-defined networks," in *Network and Distributed Security Symposium*, 2013.

[6] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM Sigcomm CCR*, vol. 38, pp. 69–74, 2008.

[7] C. Liu, A. Raghuramu, C.-N. Chuah, and B. Krishnamurthy, "Piggybacking network functions on sdn reactive routing: A feasibility study," in *SOSR*, 2017.

[8] M. Yu, L. Jose, and R. Miao, "Software defined traffic measurement with opensketch." in *NSDI*, vol. 13, 2013, pp. 29–42.

[9] C. Jin, C. Lumezanu, Q. Xu, H. Mekky, Z.-L. Zhang, and G. Jiang, "Magneto: Unified fine-grained path control in legacy and openflow hybrid networks," in *Proceedings of the Symposium on SDN Research*. ACM, 2017, pp. 75–87.

[10] M. Lee, N. Duffield, and R. R. Kompella, "Not All Microseconds are Equal: Fine-Grained Per-Flow Measurements with Reference Latency Interpolation," in *ACM Sigcomm*, 2010.

[11] S. Shin and G. Gu, "Cloudwatcher: Network security monitoring using openflow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?)," in *Network Protocols (ICNP), 2012 20th IEEE International Conference on*. IEEE, 2012, pp. 1–6.

[12] V. Sivaraman, S. Narayana, O. Rottenstreich, S. Muthukrishnan, and J. Rexford, "Heavy-hitter detection entirely in the data plane," in *Proceedings of the Symposium on SDN Research*. ACM, 2017, pp. 164–176.

[13] "sflow." [Online]. Available: http://sflow.org/

[14] "Understanding network taps." [Online]. Available: https://goo.gl/KfQ46H

[15] C. Estan, K. Keys, D. Moore, and G. Varghese, "Building a Better NetFlow," in *ACM Sigcomm*, 2004.

[16] M. Moshref, M. Yu, R. Govindan, and A. Vahdat, "Dream: dynamic resource allocation for software-defined measurement," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4. ACM, 2014, pp. 419–430.

[17] C. Kim, A. Sivaraman, N. Katta, A. Bas, A. Dixit, and W. Lawrence J, "In-band network telemetry via programmable dataplanes," in *SOSR Demos*, 2015.

[18] A. Tootoonchian, M. Ghobadi, and Y. Ganjali, "OpenTM: Traffic Matrix Estimator for OpenFlow Networks," in *PAM*, 2010.

[19] L. Jose, M. Yu, and J. Rexford, "Online measurement of large traffic aggregates on commodity switches," in *USENIX Hot-ICE*, 2011.

[20] C. Yu, C. Lumezanu, V. Singh, Y. Zhang, G. Jiang, and H. V. Madhyastha, "Monitoring network utilization with zero measurement cost," in *PAM*, 2013.

[21] D. Levin, M. Canini, S. Schmid, F. Schaffert, and A. Feldmann, "Panopticon: Reaping the Benefits of Incremental SDN Deployment in Enterprise Networks," in *USENIX Annual Technical Conference*, 2014.

[22] H. Lu, N. Arora, H. Zhang, C. Lumezanu, J. Rhee, and G. Jiang, "HybNET: Network Manager for a Hybrid Network Infrastructure," in *Middleware*, 2013.

[23] Y.-W. E. Sung, S. G. Rao, G. G. Xie, and D. A. Maltz, "Towards systematic design of enterprise networks," in *Proceedings of the 2008 ACM CoNEXT Conference*. ACM, 2008, p. 22.

[24] H. Zeng, P. Kazemian, G. Varghese, and N. McKeown, "Automatic test packet generation," in *Proceedings of the 2012 ACM CoNEXT Conference*. ACM, 2012, pp. 241–252.

[25] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "Devoflow: Scaling flow management for high-performance networks," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 254–265, 2011.

[26] "Openflow switch specification, 1.5.1," https://goo.gl/fAFg9R.

[27] "Cisco switches." [Online]. Available: https://goo.gl/9JDndy

[28] "iwnetworks switches." [Online]. Available: https://goo.gl/k17WD5

[29] O. Tilmans, T. Bühler, S. Vissicchio, and L. Vanbever, "Mille-feuille: Putting isp traffic under the scalpel," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*. ACM, 2016, pp. 113–119.

[30] "Configuring erspan, 2016." [Online]. Available: https://goo.gl/h3qaGL

[31] S. Vissicchio, O. Tilmans, L. Vanbever, and J. Rexford, "Central control over distributed routing," in *ACM SIGCOMM*, 2015.