Network Telemetry: Towards A Top-Down Approach

Minlan Yu
Harvard University
minlanyu@seas.harvard.edu

ABSTRACT

Network telemetry is about understanding what is happening in the current network. It serves as the basis for making a variety of management decisions for improving the performance, availability, security, and efficiency of networks. However, it is challenging to build real-time and fine-grained network telemetry systems because of the need to support a variety of measurement queries, handle a large amount of traffic for large networks, while staying within the resource constraints at hosts and switches. Today, most operators take a bottom-up approach by passively collecting data from individual devices and infer the network-wide information they need. They are often limited by the monitoring tools device vendors provide and find it hard to extract useful information. In this paper, we argue for a top-down approach: We should provide a high-level declarative abstraction for operators to specify measurement queries, programmable measurement primitives at switches and hosts, and a runtime that translates the high-level queries into low-level API calls. We discuss a few recent works taking this top-down approach and call for more research in this direction.

CCS CONCEPTS

• **Networks** → **Network measurement**;

KEYWORDS

Network telemetry, network management

1 INTRODUCTION

Today, it is increasingly challenging to build and manage networks for data centers, enterprises, and ISPs, which have increasing requirements for high performance, high utilization, high availability, and security. To meet these requirements, the first step is to understand what is going on in the current networks. Often understanding network state is the basis for and much harder than making the right management decisions afterward. For example, to ensure good and predictable performance, operators need to monitor the delay, packet loss, and throughput for individual flows to quickly diagnose performance problems whenever they happen. To increase the network utilization (e.g., to near 100% [21]), operators need to first continuously collect per-flow statistics and capture their changes in real-time and then based on these statistics to adapt routing. To ensure that the cloud has 99.9% or 99.99% of uptime, operators need to quickly identify failure signatures and diagnose their root causes. Finally, to improve cloud security, it is critical to detect attacks in a timely fashion, especially from those attackers with a programmable infrastructure (e.g., botnets) who can dynamically change their attack characteristics. In summary, to meet all the key requirements of network management, we need a *network telemetry* system to report the state of the network in a timely and fine-grained fashion.

There have also been growing interests in network telemetry in industry. MarketsandMarkets predicts that the network analytics market is worth 2.32 Billion USD by 2020 [1]. Major cloud providers [10, 12] keep enhancing their network monitoring tools. There is also a growing number of network analytics startups [2–4, 11], which focus on providing better analysis and visualization of the network state. These network analytics solutions heavily rely on a scalable and efficient network telemetry system that provides fine-grained and real-time network state information.

To build such a network telemetry system, there are several goals: First, the system should support a diverse set of queries on fine-grained and real-time network information for a variety of management tasks. Second, the system should scale to a large amount of traffic and a large number of switches and hosts, especially with growing traffic and network sizes. Third, we should achieve the above two goals while using limited resources at switches and hosts and processing packets with limited time. We will elaborate on these challenges in Section 2.

Today's networks often use a *bottom-up* approach for network telemetry. Operators have to configure individual network devices based on specific monitoring tools available at these devices (e.g., NetFlow, sFlow, tcpdump), passively collect a massive amount of data from these tools, and infer network-wide state from the collected device-level data. To be able to understand network state using such a bottom-up approach, operators have to collect a lot of data from devices with high storage, bandwidth, and processing overhead. Operators then aggregate such data and translate it into a network-wide view that they need, which sometimes is a highly manual and tedious process that requires operators' expertise.

Instead, we argue for a *top-down* approach that redesigns the network infrastructure to make network visibility as a first-class citizen. This requires changing the measurement practice in three aspects: (1) high-level declarative abstractions that decouple network-wide measurement queries with underlying measurement mechanisms at devices; (2) an efficient runtime system that translates high-level queries to low-level configurations, and manages resources across management tasks; (3) appropriate APIs and measurement primitives in the underlying hosts and switches.

For the rest of the paper, we first discuss the importance of network telemetry, its challenges, and the limitations of the bottom-up approach in Section 2. We describe the new trend of the top-down approach in Section 3. We survey recent work that uses the top-down approach for network telemetry and discuss future work in this area in Section 4. Section 5 concludes the paper.

2 THE IMPORTANCE AND CHALLENGES OF NETWORK TELEMETRY

In this section, we highlight the importance of network telemetry in modern networks in two aspects: First, what are the key questions operators expect to get from the telemetry system; Second, what are the new trends in modern networks that increase the need for network telemetry.

2.1 What do we need from network telemetry?

The first question to answer when we talk about any network telemetry system is always what data operators need to collect from the network. This is a hard question to answer even for operators. We informally surveyed operators from multiple cloud providers and ISP networks, their answers are often highly related to the monitoring tools available in their network devices. For example, operators often mention that they need NetFlow data [5], syslogs [6], and TCP statistics [7]. This means the network state operators need today is highly constrained by the information they can get from network devices.

Thus our goal is to reverse such the bottom-up thinking. We argue that if we can provide abstractions for operators to specify their queries for network states independent of the low-level device capabilities, they can more freely think about what they need to see in the network.

Another way to answer the question of what kind of information we need from the network is to look at the applications. The ultimate goal of the network telemetry system to serve network management tasks and the main goals of managing a network are to achieve high performance for the applications using the network, to provide high network utilization to reduce the cost, and to improve the network availability and security. Here are a few examples of management tasks:

Profiling resource usage: Profiling individual tenants and applications is critical for billing, provisioning network resources, and planning future network design. To ensure accurate and efficient billing, provisioning, and planning, profiling need to be fine-grained and periodically performed to capture changes in resource usage. To support fine-grained profiling, operators need to collect fine-grained and accurate statistics about individual flows at various locations (e.g., every hop in the network and at hosts) on a fine time scale.

Identifying traffic pattern changes: When traffic patterns deviate from normal cases, operators need to quickly detect the changes, identify the reasons, and react to them. For example, if a tenant suddenly receives a burst of traffic or two VMs who do not normally communicate start to communicate a lot, these may be signals indicating network events such as malicious behaviors, software bugs, and failures. Operators need to analyze the traffic pattern changes and identify the root cause. Even when the pattern changes are legitimate, operators still need to accommodate such changes with better traffic engineering, load balancing, or job scheduling. This means in addition to the normal counters such as the number of packets and bytes kept in NetFlow, operators are also interested in other statistics such as standard deviations, quantiles, traffic entropy in different flow granularities (e.g., 5 tuple flows, sources, subnets, tenants) and time granularities.

Diagnose failures and performance problems: Failures and performance problems can have a variety of root causes such as link/device failures, congestion, misconfigurations, slow routing convergence, application software bugs, etc. Each of these root causes requires different types of information to diagnose. And it is hard to know ahead of time what type of information is useful. Therefore, ideally, we would like to get all the fine-grained (e.g., per packet) information on a fine time scale. We should also get the information in a timely fashion because operators need to quickly identify, isolate, and fix these problems before applications or tenants notice, to minimize the impact on tenants and on cloud revenue.

2.2 New needs for network telemetry

There are a few trends in network management that make a network telemetry system even more important.

First, with the trend of software-defined networking, network management solutions have become more automated than before. With networks growing to larger scales, higher speed, and higher link utilization, human operators can no longer afford to look at a screen of data to understand the network state. Instead, we have to rely on automated solutions which dynamically drill down based on the current conditions and even automatically react to network events. Such an automated reaction requires the network telemetry system to provide fine-grained events with high accuracy on a fine time scale (e.g., transient traffic bursts). We need high accuracy because a wrong signal can easily lead to a chain of wrong reactions in a fully automated system. We need information on a fine time scale to enable fast reactions to emergencies (e.g., failures).

Second, given the increasingly complex interactions between applications and the network, it is challenging to diagnose performance problems for these applications [40]. For example, to identify TCP incast [38], we not only need to know the information of individual flows that suffer from incast, but also the times when they are generated at hosts and when they arrive at switches, and all the other packets that collide with the flows at switches. Therefore, to diagnose performance problems for applications, we need fine-grained information for individual flows over time and across hosts and switches.

Third, with the growth of cloud computing and IoT devices, the number and types of attacks also keep growing [13]. Attackers often have a programmable infrastructure (e.g., botnets) to dynamically change the types and characteristics of the attacks to make the attacks harder to detect. To keep up with the attack dynamics, it is important to enable the flexible measurement support that can quickly capture attacks as they happen, rather than waiting for vendors to add a new attack detection feature at switches.

Finally, there is an increasing number of heterogeneous and programmable network devices and components in networks. These components include programmable ASICs (e.g., P4 [8]), NetFPGA, programmable NICs, software switches, virtualized network functions, and RDMA, etc. These devices also bring new challenges and opportunities for measurement because they provide different performance and programming APIs for applications. A network telemetry system should collect detailed and customized information for all these devices in the network to fully profile the end-to-end performance and locate performance problems.

In general, the more information operators can get about the network (from the static configuration of topology and routing, the traffic from each tenant, to individual packets traversing at each device), the more operators can make the right control decisions (e.g., routing, traffic engineering, load balancing, and congestion control). Therefore, it is important to support a wide variety of measurement queries that are not limited to what network devices provide today.

2.3 Key challenges

There are three key challenges in building a network telemetry system:

Diverse, fine-grained, and real-time queries: All the parties involved in networks – operators, application developers, and end users – have different measurement requirements to understand the network. Each of them may initiate different measurement queries at different times such as understanding traffic for traffic engineering, detecting attacks when a sudden burst happen, or diagnosing performance problems for individual applications. Most of these measurement queries require fine-grained flow-level or packet-level information to understand anomalies that only happen for a small set of flows/packets or to fully understand the root cause of a networking event. These queries should also be answered in real-time to support fast reactions to failures and attacks before they affect network availability and security.

Scalability: We face scalability challenges when we collect fine-grained information from large-scale networks. Networks today, especially in data centers, can have millions of virtual machines and hundreds of thousands of switches. IoT networks can also have millions of IoT devices. At each device (e.g., VMs, switches, IoT devices), there is a large number of events to measure. For example, at a switch with multi-Tbps traffic, there can be as many as a million active flows [31]. At a host with a 10Gbps NIC, we can see up to 14.8 million packets per second. These numbers mean that we need to inspect millions of packets at line rates and store millions of entries at each device. With increasing line rates, there is an increasing amount of data to collect at each device and yet less time to collect the data.

Limited resources and processing time: It is hard to measure a large number of flows and packets with just limited resources at switches and hosts. At switches, vendors often devote their limited memory resources to more important control functions such as packet forwarding and firewalls. At hosts, most of the CPU resources are used for revenue-generating applications. Only with the leftover resources, we can support network telemetry. Yet, network telemetry sometimes requires even more resources than control functions. This is because network telemetry not only processes all the packets but also store information for these packets.

In addition to the limited resources, there is also a limited time for collecting information. For example, for a 40 Gbps port at a switch, we only have 12 ns to process each packet. In such a short time, switches have to complete many packet forwarding functions such as packet header parsing and table lookups, in addition to network telemetry. Similarly, for a 10 Gbps port at a host, we only have 70 ns to process a packet using the host CPU.

2.4 Today's Bottom-up Measurement Practice

Today, most networks support network telemetry using a *bottom-up* approach: Operators collect information from hosts and switches using standard measurement tools, aggregate the per-device information into a centralized collector, and analyze the data to extract the information they want. There are three key problems of such an approach:

Network devices capture too little information: Today, we are no short of network monitoring tools at network devices. For example, we have flow-level counters (e.g., NetFlow [5] and sFlow [39]), logs at switches and hosts (e.g., SNMP [9], Syslog [6]), and packet traces(e.g., tcpdump). However, due to the limited resources and processing time at switches and hosts, most of these tools only collect aggregated or sampled information. For example, today's data centers often apply a sampling rate of 1 in 1K for collecting NetFlow counters every few minutes. As a result, we may miss a lot of important information related to transient events (e.g., microbursts) or a small number of flows (e.g., anomaly flows).

Operators have too much information to process: On the other hand, when operators aggregate the information collected from all the hundreds of thousands of switches and hosts, there is often too much information, making it hard to find the needles in the haystack. For example, to detect a superspreader (i.e., the source VM that suddenly starts to talk to many other VMs), operators have to inspect all the VMs, extract their flow-level records, and aggregate them by source, and compare the number with history numbers. Instead, if we can provide operators with a clear abstraction of specifying the type of flows they are interested in, and directly provide them the related information, it would their jobs much easier and less error-prone.

Operators lack network-wide view: The third problem is that today operators have to configure individual devices and get measurement data from each of them, take great efforts to manually integrate diverse data across devices, and correlate them with network configuration information to understand the network-wide events they care about (e.g., performance problems, failures). This is often hard to be done correctly and efficiently, given the timestamp differences across devices, different granularities of aggregation and sampling at devices, and routing changes. Moreover, as monitoring tools at individual devices get more configurable (e.g., flexible NetFlow [17]), it becomes more challenging for operators to learn these knobs and identify the best ways to configure these knobs.

3 A TOP-DOWN APPROACH FOR NETWORK TELEMETRY

3.1 The top-down approach

To address the problems of today's bottom-up approach, we promote a top-down approach for network measurement. Figure 1 shows the key components in our top-down approach:

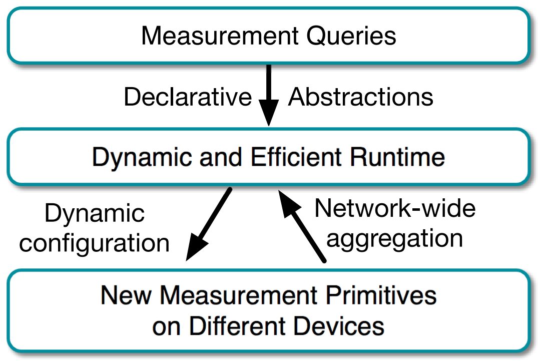

Figure 1: **Top-down network telemetry**

Declarative measurement abstraction: We introduce *declarative measurement abstractions* for operators to clearly express their measurement requirements without worrying about when, where, and how to answer the queries. In this way, they can spend more time focusing on defining the right set of information they need to diagnose network performance problems, to help serve applications better, etc.

The declarative measurement abstractions should include the following principles: *(1) Intent based:* Operators should be able to freely express their measurement intent independent of the underlying measurement system. *(2) Named principles:* Operators should be able to describe the traffic they want to measure not based on packet header fields, but based on high-level names such as DNS names, tenants, applications, or even dynamic flow properties (e.g., delay, loss rate, etc.). *(3) Network-wide:* Operators should be able to specify their queries about the network-wide state without worrying about where in the network the measurement should be performed. *(4) Many concurrent queries:* Different management tasks, tenants, or applications should be able to specify their queries at the same time, without knowing the existence of other queries.

Efficient network-wide runtime system: Our telemetry system should achieve the four above goals subject to minimizing the measurement overhead. Thus, we need *a runtime system* that automatically matches the measurement abstractions with primitives at devices, dynamically allocates resources across queries and handles network dynamics such as mobile hosts and routing changes. We also improve the interactions between measurement and control by optimizing measurement queries for specific control functions.

To answer the queries operators specify, the runtime system must provide information with the following properties: *(1) in real-time:* To quickly react to network events (e.g., traffic changes) and make management decisions (e.g., traffic scheduling, performance isolation), we need to provide real-time reports for some queries with low overhead. *(2) in different granularities:* We may provide aggregated information for some queries (e.g., to understand the overall traffic volume), but provide fine-grained information for other queries (e.g., to identify anomalies). *(3) with different levels of accuracy:* Our system should meet the accuracy requirements of measurement queries while staying within the resource constraints.

To support these properties, we need to design new data structures and algorithms that minimize the memory we use to keep the data, especially when the amount of data grows with the increasing high link speed (e.g., 40Gbps and higher in data centers) and the larger scale networks (with more traffic). We also need to reduce the bandwidth usage between devices and the network-wide data analyzer, while ensuring timely report of network states. Moreover, it is important to multiplex resource usage across queries to best improve the performance of all the queries.

New measurement primitives at devices: Given the measurement abstractions, we can now design novel data structures and system optimizations at different devices (VMs, hypervisors, NICs, and switches). Because we can now focus on the information that operators care about based on their specification, we can now design better measurement algorithms towards these targeted data.

However, we cannot simply design the most efficient data structures for each individual measurement queries because there are various types of queries and new queries appear as operators gain a deeper understanding of their network behaviors. Therefore, we need to carefully design the *measurement primitives* at different devices to make them both *generic* in supporting diverse measurement requirements, and *efficient* in packet processing performance with limited resources and capabilities.

The measurement primitive design is specifically customized for each type of devices, including hosts, switches, or other programmable devices (e.g., NetFPGA, smart NICs). Different devices have different capabilities and views as summarized in Table 1.

Switches often have limited programmability (e.g., P4 [8]) while hosts have more flexible programmability. Switches are limited in on-chip memory size and the number of packet processing stages, while hosts are often limited in CPU resources. However, switches have much faster packet processing speed (in Tbps) than hosts. Reconfigurable devices such as NetFPGA and smart NICs are often in between switches and hosts in terms of both programmability and packet processing speed.

Different devices offer different views of the network state. For example, since hosts are closest to applications, they can often directly report application-related or network stack problems. In contrast, switches only see aggregated traffic information through them, but they have unique access to in-network problems (e.g., link failures, network congestion).

Due to the device differences, it is critical to automatically correlate different sources of information from diverse devices, and integrate them with network configuration information (e.g., topology, routing) to provide a unified view of the entire network to operators.

3.2 Differences with other approaches

We now compare the top-down approach with a few alternative measurement solutions: universal measurement design and passive logging of historical measurement data.

Differences with universal measurement design: Recent work [23, 33] identifies a single measurement solution (flow sampling [33] or Universal Sketch [23]) that works for a variety of measurement queries. While this approach is useful for a common set of measurement queries today, it may neither work for new measurement

Devices	Switch	Host	Programmable devices
Programmability	fixed primitives (e.g., P4)	Flexible	medium flexibility
Resources for measurement	Limited on-chip memory, #stages	Limited CPU	Limited on-chip memory
Packet processing performance	High	Low	medium
View	aggregated traffic, in-network state	application info	location dependent

Table 1: Different capabilities and views on heterogeneous devices

queries that the proposed primitives do not support, nor work for new algorithms and data structures that improve today's solutions. Essentially, the two approaches differ in the level of primitives devices can support: Should they support specific measurement solutions that work for a variety of measurement queries or use measurement building blocks that allow programming different solutions? The answer depends on how expensive it is to implement the two approaches in switches and hosts (in terms of chip resources, power usage, etc.) and what the resource-performance tradeoffs are.

Differences with logging historical measurement data: Today, many networks still rely on the bottom-up approach to collect measurement data [31, 34] and keep the data in a database for future queries. To reduce the storage overhead, they periodically summarize the data into coarse-grained flows and time intervals. For example, operators may keep per minute flow counters for the past hour, per hour information for the past day, and per day information for the past month, etc. The limitation of this approach is the lack of fine-grained information at devices, too much information for operators to process in the database, and the lack of network-wide view as discussed in Section 2.4. With the growth of measurement data on larger networks and links with higher link speed, it becomes fundamentally challenging to track every detail of the network at a large scale, at a fine timescale, and with high accuracy and low resource usage.

While our top-down approach addresses these limitations, it also has its own limitations: We may miss the opportunities to look into history data that operators have not thought about recording in their queries; We may also miss a transient event if it takes too long to narrow down to the right query.

Ultimately, we believe a combination of the two approaches would work: We rely on the top-down approach to capture data for real-time queries while also collect coarse-grained data in the database for future queries.

4 RECENT ADVANCES IN NETWORK MEASUREMENT AND FUTURE WORK

In this section, we describe a few recent works that share the same vision of our top-down approach and discuss open questions. This section is by no means complete in covering all the related works but hopefully provides a taxonomy of the works in this space.

4.1 Declarative queries

Queries on high-level names: Similar to how software-defined networks allow operators to specify their control functions based on high-level names [15], we should allow operators to freely describe the traffic they want to measure using *high-level names* such as hosts, applications, and tenants. The paper [16] proposes *intentional network monitoring*, which allows operators to collect traffic based on *intents* such as people, applications, or devices. It then automatically maps these intents with low-level packet header fields by leveraging the information learned from other systems such as Domain Name System (DNS) and Border Gateway Protocol (BGP). Sonata [19] provides a declarative interface using expressive dataflow operators on extensible tuple-based abstractions. Sonata can express queries such as detecting attacks and newly opened TCP connections and then compiles these queries down to switch rules with limited memory.

Path-level and network-wide queries: In addition to monitoring traffic at a selected location, it is also important to check a flow along network paths. For example, traffic engineering requires measuring the ingress-egress traffic matrix; debugging a congested link requires determining the set of sources sending traffic through that link; locating a faulty device might involve detecting how far along a path the traffic makes progress. Instead of collecting traffic information at all the devices along the path and integrate them afterward, PathQuery [29] proposed a declarative query language for efficient path-based traffic monitoring. Path queries are specified as regular expressions over predicates on packet locations and header values, with SQL-like "groupby" constructs for aggregating results anywhere along a path. Only when packets satisfy a query are the packets counted, sampled, or sent to collectors for further analysis, which significantly reduces data collection overhead.

NetSight [20] introduced a *network-wide* query framework to support applications that require extracting the full journey of selected packets. Such a network-wide query framework can be useful for a variety of management tasks such as interactive network debugging, live invariant monitoring, and network profiling and logging. However, these queries still rely on first capturing packet histories from all the switches in the entire network. Based on the collected data, each application can analyze the collected data to answer the network-wide queries. It is an open question on how to reduce the overhead by performing more query-specific data collection.

Managing states across queries Statesman [35] provides a network state management service that captures various aspects of the network such as which links are alive and how switches are forwarding traffic. Statesman then shares such information across management applications and allows applications to change the states (e.g., routing) back to the network. It would be great to follow Statesman to manage more states than topology and routing.

Future work: A broader set of networks Most of the above work focuses on queries for data center networks. Other networks such as enterprise networks, ISP networks, cellular networks, and IoTs introduce new challenges for measurement queries: *(1) New types* *of queries:* Some networks may have new types of devices (e.g., many NFVs in ISP and cellular networks) and new types of queries (e.g., for security, for in-network processing functions) that require us to introduce new high-level names and query abstractions. StroboScope [37] provides a good first step for defining query languages for traffic mirroring in ISP networks. *(2) New measurement constraints:* Some networks may have policies on which devices to collect measurement data and constraints on which types of measurement data to collect due to privacy or other access constraints. It is important to introduce ways to specify these constraints and introduce new measurement solutions that accommodate these constraints. *(3) New scalability challenges:* IoT networks, in particular, can easily be millions of devices. It is important to continuously monitor each IoT device because it has frequently changing events and reactions (e.g., cameras on moving vehicles) while the communications among the devices and between the devices and the central controller are limited.

4.2 Efficient runtime to compile queries

Given the declarative queries, the next question is how to build a good runtime that maps queries to low-level measurement primitives. Although there have been many runtime supports for control functions in SDN, there is limited research on the runtime for measurement queries.

Resource management for concurrent queries: DREAM [25, 26] proposes a resource management framework for multiple concurrent measurement queries while ensuring a user-specified level of accuracy. The key observation is that there is a tradeoff between resource usage and accuracy for measurement queries. The tradeoff depends on the type of queries, their parameters, and traffic characteristics. DREAM does not assume an a priori characterization of this tradeoff, but instead dynamically searches for a resource allocation that is sufficient to achieve a desired level of accuracy. DREAM can support more concurrent queries with higher accuracy than several other alternatives.

Composing control applications: Pyretic [24] introduces new abstractions for building applications out of multiple, independent modules such as measurement, routing, and load balancing. Pyretic then introduces two operators: the parallel composition operator that allows multiple policies to operate on the same set of packets, and the sequential composition operator that allows one policy to process packets after another. Pyretic also enables each policy to operate on an abstract topology that implicitly constrains what each module can see and do.

We should extend Pyretic to support the composition of measurement queries. For example, we need to investigate ways to allow the parallel composition of measurement queries that share a subset of measurement data. We can also study sequential composition that allows a control application to get feeds from a measurement query.

Future work: There are still several challenges on building the runtime: *(1) real-time reports at large scale:* In data centers with thousands of switches and hundreds of thousands of hosts, it is an open question on how to provide real-time reports from all these devices for many concurrent queries. *(2) Handle dynamics:* Another question is how to handle the cases when the mappings between the high-level queries and the underlying measurement dynamically change with query changes, traffic changes, and topology changes. *(3) Provide the right level of information:* When a tenant sends queries about its own traffic, the runtime needs to provide the right level of information that is related to the tenant's traffic. For example, when a tenant experiences performance problems, it is not helpful to expose all the topology changes and traffic changes to him/her. Instead, the runtime should filter the right set of changes that are relevant to the performance problem and provide them in a meaningful way to the tenant.

4.3 New data-plane primitives

Commodity and Programmable switches: DREAM [25] leverages the flow-based counters at commodity switches. Several proposals leverage PISA switches to implement measurement using match-action tables [19] or hash-based data structures [22, 23, 41]. Recent work [30, 37, 42] leverages "match and mirror" functionality on commodity switches to track traffic. Other work [14, 18, 32] relies on pings and traceroutes to diagnose network failures.

Marple [28] introduces a new programmable key-value store primitive on switch hardware. The key-value store performs flexible aggregations at line rate (e.g., a moving average of queueing latencies per flow), and scales to millions of keys. Marple can support a set of new switch queries that could previously run only on end hosts while only occupying a modest fraction of a switch's hardware resources.

Hosts: At hosts, Trumpet [27] presents an event monitoring system that monitors every packet and reports network-wide events at millisecond timescales. Trumpet allows operators to describe new network events such as detecting correlated bursts and loss, identifying the root cause of transient congestion, and detecting short-term anomalies at the scale of a data center tenant. SNAP [40] collects network stack information with low computation and storage overhead, and use that to identify performance problems that happen at applications or the network (or both).

Coordinating switches and hosts: SwitchPointer [36] observes that monitoring at end-hosts often requires more resources but less visibility into the network while network switches often have more visibility into the network but limited resources. SwitchPointer coordinates switches and hosts by using switch memory as a "directory service" which stores pointers to the relevant telemetry data at end-hosts.

Future work: There need to be more works on improving measurement primitives at different devices in the following aspects: *(1) New measurement primitives on different platforms:* We need to introduce new designs that are well suited for NFVs (which need a different amount of CPU and memory for different types of traffic) and IoT devices (which also care about energy consumption). *(2) Transforming existing measurement primitives to the top-down approach:* There have been many measurement tools on switches and hosts such as SNMP counters, tcpdump, and NetFlow [17]. It would be interesting to transform these bottom-up solutions to *top-down* ones so that operators can focus on the actual network-wide performance or failure problem rather than configuring individual devices for these tools. *(3) Primitives that support emergent streaming algorithms:* There have been many recent advances in streaming algorithms for different measurement queries (e.g., DDoS detection, entropy estimation, microburst detection etc.). It is important to identify a general set of primitives in switches and hosts that support these algorithms better.

5 CONCLUSION

In this paper, we discuss the importance and challenges of network telemetry. We promote *the top-down approach* which enables operators to write measurement queries in a declarative way. To answer these queries, we need a runtime that maps the queries to the programmable primitives at switches and hosts. We discuss recent efforts in building such network telemetry systems and future research directions.

ACKNOWLEDGMENTS

I would like to thank Jennifer Rexford for discussions on the initial version of the paper.

REFERENCES

[1] www.marketsandmarkets.com/PressReleases/network-analytics.asp.
[2] netsil.com.
[3] www.pluribusnetworks.com.
[4] www.logicmonitor.com.
[5] www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html.
[6] datatracker.ietf.org/wg/syslog/charter/.
[7] www.web10g.org.
[8] p4.org.
[9] www.cisco.com/en/US/docs/internetworking/technology/handbook/SNMP.html.
[10] Azure network watcher. azure.microsoft.com/en-us/blog/announcing-azure-network-watcher-network-performance-monitoring-and-diagnostics-service-for-azure.
[11] Big monitoring fabric. www.bigswitch.com/products/big-monitoring-fabric.
[12] https://aws.amazon.com/blogs/security/tag/network-monitoring-tools/.
[13] Arbor Networks. Insight Into the Global Threat Landscape. goo.gl/15oOx3, February 2013.
[14] B. Arzani, S. Ciraci, L. Chamon, Y. Zhu, H. Liu, J. Padhye, B. T. Loo, and G. Outhred. 007: Democratically finding the cause of packet drops. In *NSDI*, 2018.
[15] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker. Ethane: Taking control of the enterprise. In *Proc. ACM SIGCOMM*, 2007.
[16] S. Donovan and N. Feamster. Intentional network monitoring: Finding the needle without capturing the haystack. In *ACM SIGCOMM Workshop on Hot Topics in Networking (HotNets)*, 2014.
[17] M. Grossglauser and J. Rexford. Passive traffic measurement for IP operations. In *The Internet as a Large-Scale Complex System*. Oxford University Press, 2005.
[18] C. Guo, L. Yuan, D. Xiang, Y. Dang, R. Huang, D. Maltz, Z. Liu, V. Wang, B. Pang, H. Chen, Z.-W. Lin, and V. Kurien. Pingmesh: A large-scale system for data center network latency measurement and analysis. In *SIGCOMM*, 2015.
[19] A. Gupta, R. Harrison, M. Canini, N. Feamster, J. Rexford, and W. Willinger. Sonata: Query-driven streaming network telemetry. In *ACM SIGCOMM*, 2018.
[20] N. Handigol, B. Heller, V. Jeyakumar, D. Mazières, and N. McKeown. I know what your packet did last hop: Using packet histories to troubleshoot networks. In *NSDI*, 2014.
[21] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat. B4: Experience with a globally deployed software defined wan. In *ACM SIGCOMM*, 2013.
[22] Y. Li, R. Miao, C. Kim, and M. Yu. Flowradar: A better NetFlow for data centers. In *NSDI*, 2016.
[23] Z. Liu, A. Manousis, G. Vorsanger, V. Sekar, and V. Braverman. One sketch to rule them all: Rethinking network flow monitoring with UnivMon. In *SIGCOMM*, 2016.
[24] C. Monsanto, J. Reich, N. Foster, J. Rexford, and D. Walker. Composing software-defined networks. In *NSDI*, 2013.
[25] M. Moshref, M. Yu, R. Govindan, and A. Vahdat. DREAM: Dynamic resource allocation for software-defined measurement. In *ACM SIGCOMM*, 2014.
[26] M. Moshref, M. Yu, R. Govindan, and A. Vahdat. Dynamic resource allocation for sketch-based software-defined measurement. In *Technical Report*, 2014.
[27] M. Moshref, M. Yu, R. Govindan, and A. Vahdat. Trumpet: Timely and precise triggers in data centers. In *ACM SIGCOMM*, 2016.
[28] S. Narayana, A. Sivaraman, V. Nathan, P. Goyal, V. Arun, M. Alizadeh, V. Jeyakumar, and C. Kim. Language-directed hardware design for network performance monitoring. In *ACM SIGCOMM*, 2017.
[29] S. Narayana, M. Tahmasbi, J. Rexford, and D. Walker. Compiling path queries. In *NSDI*, 2016.
[30] J. Rasley, B. Stephens, C. Dixon, E. Rozner, W. Felter, K. Agarwal, J. Carter, and R. Fonseca. Planck: Millisecond-scale monitoring and control for commodity networks. In *SIGCOMM*, 2014.
[31] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren. Inside the Social Network's (Datacenter) Network. In *SIGCOMM*, 2015.
[32] A. Roy, H. Zeng, J. Bagga, and A. C. Snoeren. Passive realtime datacenter fault detection and localization. In *NSDI*, 2017.
[33] V. Sekar, M. K. Reiter, and H. Zhang. Revisiting the case for a minimalist approach for network flow monitoring. In *IMC*, 2010.
[34] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, A. Kanagala, J. Provost, J. Simmons, E. Tanda, J. Wanderer, U. Hölzle, S. Stuart, and A. Vahdat. Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network. In *ACM SIGCOMM*, 2015.
[35] P. Sun, R. Mahajan, J. Rexford, L. Yuan, M. Zhang, and A. Arefin. A network-state management service. *SIGCOMM*, 2014.
[36] P. Tammana, R. Agarwal, and M. Lee. Distributed network monitoring and debugging with switchpointer. In *NSDI*, 2018.
[37] O. Tilmans, T. Bühler, I. Poese, S. Vissicchio, and L. Vanbever. Stroboscope: Declarative network monitoring on a budget. In *NSDI*, 2018.
[38] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. Andersen, G. Ganger, G. Gibson, and B. Mueller. Safe and effective fine-grained TCP retransmissions for datacenter communication. In *ACM SIGCOMM*, 2009.
[39] M. Wang, B. Li, and Z. Li. sflow: Towards resource-efficient and agile service federation in service overlay networks. *Distributed Computing Systems, International Conference on*, 0:628–635, 2004.
[40] M. Yu, A. Greenberg, D. Maltz, J. Rexford, L. Yuan, S. Kandula, and C. Kim. Profiling network performance for multi-tier data center applications. In *NSDI*, 2011.
[41] M. Yu, L. Jose, and R. Miao. Software defined traffic measurement with OpenSketch. In *NSDI*, 2013.
[42] Y. Zhu, N. Kang, J. Cao, A. Greenberg, G. Lu, R. Mahajan, D. Maltz, L. Yuan, M. Zhang, H. Zheng, and B. Y. Zhao. Packet-level telemetry in large datacenter networks. In *SIGCOMM*, 2015.