NetVision: Towards Network Telemetry as a Service
Zhengzheng Liu† ‡, Jun Bi† ‡, Yu Zhou† ‡, Yangyang Wang† ‡, Yunsenxiao Lin∗ ‡
∗ Institute for Network Sciences and Cyberspace, Tsinghua University
† Department of Computer Science, Tsinghua University
‡ Beijing National Research Center for Information Science and Technology (BNRist)

Abstract—In-band Network Telemetry (INT) can provide fine-grained and accurate device-level telemetry metrics. Nonetheless, INT can track only a small ratio of devices and links and embedding telemetry data into normal packets brings high overhead and high operation complexity. Hence, we present NetVision, a powerful proactive network telemetry platform with high coverage and high scalability.

I. INTRODUCTION
With the rapid growth of devices and protocols, networks become rather sophisticated. A great number of network failures such as misconfigurations, hardware malfunction, and software bugs occur very frequently. Therefore, operators are eager for an efficient telemetry approach to promptly detect and locate the common and complex network issues (e.g., high latency, TCP incast, load imbalance, routing black hole).

Prior to the emergence of programmable data planes, operators have to indirectly detect network for delayed and imprecise telemetry data through terminals at network edges. However, In-band Network Telemetry (INT) [1] based on programmable data planes to a great extent relieves this dilemma. In INT, normal packets contain header fields interpreted as telemetry instructions. These instructions tell INT-capable devices which data to collect and write into normal packets as they traverse the network. By this means, INT can directly capture much more fine-grained and accurate device-level telemetry metrics (e.g., hop latency and queue length).

In spite of many benefits, INT has some inherent drawbacks as well. Firstly, INT detection scope is limited, which is hard to obtain the comprehensive network view. That is because telemetry paths and metrics have to be preassigned by operators and can not be altered at runtime. As a result, INT can only monitor certain telemetry data of specific paths. As a consequence, INT is very likely to miss some important network failures. In one word, INT can only track a small ratio of devices and links (low coverage). Moreover, due to monitoring network by normal packets, this brings high extra telemetry traffic overhead. Meanwhile, the payload ratio reduces a lot because of encapsulating telemetry instructions and data into each normal packet. In addition, each telemetry path requires two cooperated edge switches to communicate with an INT monitor. One encapsulates telemetry instructions and the other extracts telemetry data. The synchronization and coordination between the two edge switches are very complicated. That is to say, INT brings high telemetry overhead and high deployment and maintenance complexity (low scalability).

To solve the low coverage and low scalability problems of INT, we propose NetVision, a powerful proactive network telemetry platform with high coverage and high scalability.

Instead of passively capturing normal packets, NetVision actively sends suitable amount and format of probes to support on-demand analysis with low overhead. Owing to simplistic and flexible routing control offered by Segment Routing (SR), we can customize the probe path by changing SR labels at runtime. In that way, we can easily achieve the comprehensive network view with carefully designed probe paths. A single vantage point is enough for telemetry through cycled probe paths without much cooperation complexity. What’s more, designed telemetry instructions can tell the device which data is desired instead of all data. At last, device-level telemetry metrics can be offered by customizing packet processing logic based on programmable data planes.

Our contributions are as follows: (1) We propose an efficient proactive network telemetry platform with high coverage and high scalability. (2) We provision a suite of network telemetry primitives to introduce simplicity and convenience for operators. (3) We design the specified duplex-stack probe which comprises of forwarding stack and telemetry stack for flexibly forwarding and monitoring telemetry data respectively. (4) For sake of probe overhead and coverage compromise, we also offer a probe path algorithm.

II. DESIGN

A. Design Overview
As is shown in Figure 1, operators can specify telemetry requirements to NetVision and gain telemetry results without manipulating the underlying infrastructure. The NetVision telemetry platform consists of four main components: The telemetry antenna, the telemetry coordinator, the telemetry analyzer, and the telemetry service provider. General procedures to apply the platform are as follows: Telemetry applications (e.g., traffic engineering and network visualization) enforce
### Applications of the telemetry service API

<table>
<thead>
<tr>
<th>Applications</th>
<th>Telemetry Service API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>End-to-end Latency Measurement</td>
<td>0 = PathQuery(&quot;1:1&quot;, &quot;2:1&quot;) .Select(&quot;PathRTT&quot;) .Where(&quot;PathTrace=(1,1,1,1,1,1,1,1,1)&quot;)</td>
<td>Measure the path RTT between the port 1 of switch 1 and switch 2, which conforms to the limitation of the path tracer.</td>
</tr>
<tr>
<td>Link Black Hole Discovery</td>
<td>0 = PathQuery(&quot;i&quot;, &quot;j&quot;) .Select(&quot;i&quot;) .Where(&quot;PathTrace=(i,1,1,1,1,1,1,1,1)&quot;)</td>
<td>Discover the link black during the period of 5ms. We specify the path length equals 1 and determine whether the link is a black hole through the count of passed probes.</td>
</tr>
<tr>
<td>Real-time Packet Transmission Rate Calculation</td>
<td>0 = NodeQuery(&quot;1:1&quot;) .Select(&quot;PacketRxRate&quot;) .Period(&quot;5ms&quot;)</td>
<td>Calculate the average ingress packet transmission rate of port 1 of switch 1. Besides, we also support packet count, byte count, byte min, packet reception or drop rate, hop latency and port utilization.</td>
</tr>
</tbody>
</table>
| Node Black Hole Discovery         | 0 = NodeQuery("i", "j") .Select("SwitchId") .Where("OutPortCount==0") .Period("5ms") | Locate switch id of the node black hole by discovering the port not transmitting packets during the period of 5ms.

#### B. Telemetry Primitives

To simplify telemetry policy enforcement for operators, we provide a suite of convenient and expressive telemetry primitives, including telemetry metadata and query primitives. Telemetry metadata comprises ports, timestamps, latencies and so forth for switch nodes, paths. Besides, we design query primitives to adopt the metadata for RTT, forward loop, congestion, etc. As shown in Figure 2, we offer some typical applications of the telemetry service API which are end-to-end latency measurement, real-time packet transmission rate calculation and link or node packet black hole discovery.

#### C. Duplex-stack Probe

As Fig. 3 shows, the probe mainly contains two label stacks. SR stack comprises an outport label list and the list length for flexible probe forwarding. INT stack also comprises a label list and the list length for telemetry records. INT label is composed of switch ID, metadata bitmap determining metadata type followed by a telemetry metadata value list. During the probe transmission, SR labels are popped to forward probes while INT labels are pushed to record telemetry data.

#### D. Probe Generating Algorithm

To reduce probe overhead as far as possible, we utilize a simple probe path algorithm. Considering limited label capacity supported by realistic programmable devices and enormous network size, we are able to partition one big network into several smaller networks based on the label capacity and choose a vantage point for each one. We can collect and aggregate telemetry information from each vantage point. Then we can treat each link as two opposite direction edges in the graph. According to Euler Theorem of graph theory, such directed graph is supposed to have Euler Circuit. We can calculate the circuit by Hierholzer algorithm in linear time, O(E) [2], which can be our probe path.

#### III. Evaluation

We use Mininet to simulate a Fat-Tree Topology (Fig. 4). H22 continuously responds to periodic HTTP requests from H11. While H32 occasionally sends a traffic burst to H22. Using probe timestamps in T1, we can achieve latency without switch time synchronization. Meanwhile thanks to SR, we can specify back path same as income path without asymmetric routing path problem. According to above two goodness, more accurate and nearly instantaneous latency can be offered. As Fig. 5 shows, queuing latencies of A2 and T2 have same trend as HTTP request latency while T1 has no obvious change. We can infer that A2 and T2 is on the shared path.

**ACKNOWLEDGEMENT**

This research is supported by National Key R&D Program of China (2017YFB0801701) and National Science Foundation of China (No.61472213). Jun Bi is the corresponding author.

**REFERENCES**
