Technical Report of NetView

I. Deisgn of NetView

A. Metadata and API

To simplify telemetry policy enforcement for operators, we provide a suite of expressive and convenient telemetry primitives, including telemetry metadata and query primitives. Then, we present several non-exhaustive network telemetry applications to show the usefulness of NetView.

1) Supported Metadata: NetView supports two kinds of metadata: primary metadata and derived metadata. (1) Primary Metadata. As is shown in Figure 1, primary metadata is the type of metadata directly supported by data plane [1], comprising switch metadata, ingress metadata, egress metadata, and buffer metadata. In this paper, we take all of the primary metadata into consideration, towards fine-grained visibility of the network. (2) Derived Metadata. To further facilitate operators to write telemetry queries, we propose high-level empirical derived metadata which is not directly supported by data plane and can be calculated by primary metadata. As is shown in Figure 2, it contains path-level metadata, node-level metadata, and probe-level metadata. Particularly, we can tackle path-level metadata by approximating it into primary metadata at the node. As for the delay, it consists of four parts: transmission delay, propagation delay, queueing delay, and processing delay. The queueing delay and processing delay represents network performance deviation among the above four parts. As for throughput, the path throughput is decided by the minimal port throughput along the path. Therefore, we can tackle telemetry metrics on paths by approximating it into node metrics. Without measuring delay/throughput along the path, instead, NetView measures queueing delay/port throughput of each node, and use node-level queueing delay/port throughput to compose delay/throughput for different paths.

2) Network Telemetry API: NetView provides declarative APIs that can express queries for a wide range of telemetry tasks and also frees the network operator from reasoning about where or how the query will execute. We briefly introduce the APIs as follows.

- PathQuery("*SrcNode:SrcPort*", "*DstNode:DstPort*") means that query the information of (a) path(s) between the port of source node and the port of destination node.
- NodeQuery("*Node:Port*") means that query the information of (a) given node(s).
- ProbeQuery("*TerminalIP*") means that query the information of probes from a single terminal.
- Where("*Scope*") means that specify the metadata scope of the query.
- Period("*DesiredTime*") means that setting a timer to query network periodically.

Category	Primary Metadata	Description
Switch Metadata	switch_id	The unique id of a switch within a management domain.
	control_plane_state_version	Whenever a control plane state changes (e.g., IP FIB update), the control plane can also update this version number in the data plane.
Ingress Metadata	ingress_port_id	The port on which the packet was received.
	ingress_timestamp	The device local time when the packet was received on the ingress port.
	ingress_port_rx_pkt_count	Total number of packets received on the ingress port where the packet was received.
	ingress_port_rx_byte_count	Total number of bytes received on the ingress port where the packet was received.
	ingress_port_rx_drop_count	Total number of packet drops occurred on the ingress port where the packet was received.
	ingress_port_rx_utilization	Current utilization of the ingress port where the packet was received.
Egress Metadata	egress_port_id	The id of the port forwarding the packet.
	egress_timestamp	Device local time when the packet leaves the egress port.
	egress_port_tx_pkt_count	Total number of packets forwarded through the egress port.
	egress_port_tx_byte_count	Total number of bytes forwarded through the egress port where the packet was forwarded.
	egress_port_tx_drop_count	Total number of packet drops occurred on the egress port where the packet was forwarded.
	egress_port_tx_utilization	Current utilization of the egress port via which the packet was sent out.
Buffer Metadata	queue_id	The id of the queue the device used to serve the packet.
	instantaneous_queue_length	The instantaneous length (in bytes, cells, or packets) of the queue the packet has observed in the device while being forwarded.
	average_queue_length	The average length (in bytes, cells, or packets) of the queue via which the packet was served.
	congestion_status	The ratio of the current queue length to the configured maximum queue limit.
	queue_drop_count	Total number of packets dropped by the queue.

Figure 1. Primary metadata of NetView.

- Return("*Metadata*") means that return a derived/primary metadata as the telemetry result.

3) Network-Telemetry-Based Applications: With the support of the primary metadata and the derived metadata, NetView can support various telemetry tasks, which are useful for network operation. As shown in Figure 3, we introduce four applications as examples. (a) Prove Network Innocence. Prove network innocence when the application quality gets worse. With NetView, network operators can write queries and let NetView periodically check whether end-to-end throughput conforms to SLA, identifying what parts of the system (end hosts or what parts of the network) should be blamed for a network glitch or service interruption. (b) Network Planning. Network operators are motivated to optimize their network utilization for better ROI or lower CAPEX, as well as differentiation across services and/or users of a given service. The first step is to know the real-time network conditions

Category	Derived Metadata	Description
Path-level Metadata	path_length	The number of hops in the given path.
	path_trace	Each hop information in the given path in sequence, i.e., "node:port, node:port…".
	path_rtt	Round trip time in the given path.
	path_utilization	Link utilization of each link in the given path.
	path_throughput	Instantaneous throughput of each link in the given path.
Node-level Metadata	switch_workload	Total traffic on all in/out ports of the switch.
	forward_packet_count	Total number of packets forwarded in the switch.
	drop_packet_count	Total number of packets dropped in the switch.
	alive_port_count	Total number of alive ports in the switch.
Probe-level Metadata	tx_probe_count	Total number of sent probes.
	tx_probe_timestamp	The timestamp of the first/last sent probe.
	rx_probe_count	Total number of received probes.
	rx_probe_timestamp	The timestamp of the first/last received probe.

Figure 2. Derived metadata of NetView.

Application	Query	Description
End-to-end Throughput Measurement	Q = PathQuery("1:1", "2:1") .Where("path_throughput < 1Gbps") .Period("2s") .Return("path_throughput ")	Measure the path throughput between the port 1 of switch 1 and the port 1 of switch 2, which conforms to the limitation.
Queue Length Measurement	Q = NodeQuery("*:*") .Period("5ms") .Return("instantaneous_queue_length")	Measure the instantaneous length of the queue in all of switches every 5ms.
Node Black Hole Discovery	Q = NodeQuery("*:*") .Where("switch_workload == 0") .Period("10s") .Return("switch_id")	Locate switch id of the node black hole by discovering the switch not transmitting packets.
Random Packet Drop Detection	Q = NodeQuery("*:*") .Where("ingress_port_rx_drop_count > 0" ||"egress_port_tx_drop_count > 0") .Period("1ms") .Return("switch_id" ,"ingress_port_id" , "egress_port_id")	Detect ports where packet drop happens.

Figure 3. Sample applications of NetView.

before applying policies to steer the user traffic or adjust the load balancing algorithm. With NetView, network operators can accumulate a large volume of telemetry data (*e.g.*, queue length) for the long term network capacity planning and topology augmentation. (c) Failure Location. Network failure often involves a sequence of chained events, and the source of the failure is not straightforward to identify, especially when the failure is sporadic. With NetView, network operators can write queries for parts of network prone to failure or even the whole network within the management domain to detect the failure at the first time. (d) Event Tracking. Numerous network events are of interest to network operators. For example, Network operators always want to learn where and when packets are dropped for an application flow in time. With NetView, network operators can easily get a vital clue (*e.g.*, random packet drop count) about non-reproducible events.

REFERENCES

[1] Ed Doe Hugh Holbrook Anoop Ghan-wani Dan Daly Mukesh Hira Changhoon Kim, Parag Bhide and Bruce Davie. In-band network telemetry. Website, 2016. https://p4.org/assets/INT-current-spec.pdf.