Improving Network Monitoring and Management with Programmable Data Planes

Posted by P4.org on September 25, 2015 Quicklinks : INT Specification – INT GitHub repository – INT demo video Compute virtualization and the widespread deployment of virtual machines has led to an extension of the network into the hypervisor.

By P4.org

Show original page

Posted by P4.org on September 25, 2015

Quicklinks : INT Specification - INT GitHub repository - INT demo video

Compute virtualization and the widespread deployment of virtual machines has led to an extension of the network into the hypervisor. Network virtualization solutions have emerged that enable rapid provisioning of network services -- logical switches, logical routers, load balancers, firewalls and beyond -- for virtual machine and container network interfaces. VMware NSX is a network virtualization platform that is designed to be decoupled from the physical network so as to enable deployment of virtual services over any physical network infrastructure, the only requirement from the physical network being IP connectivity between the hypervisors.

While the decoupling of physical and virtual topologies has advantages, it is important to have some interaction between the physical and virtual switches to allow for end-to-end monitoring of the entire physical + virtual network from a “single pane of glass” and to help in troubleshooting and fault isolation in complex physical + virtual topologies.

We propose methods for various network elements to collect and report their state in real-time, allowing for improved cooperation between the virtual and physical layers without requiring intermediate layers such as CPU driven control planes. The general term we have applied to these methods is INT: Inband Network Telemetry.

Some examples of useful network state to be reported by network elements are –

(i) <Switch-ID, Input port ID, output port ID> – This allows for determination and monitoring of the different paths between a pair of end-points. Current mechanisms for determining multiple paths between a pair of end-points are based on IP traceroute and can only discover equal-cost layer 3 paths (ECMP routes), but cannot discover or report the existence of multiple links bundled together as a port-channel.

(ii) Link Utilization – Link utilization along different paths between a pair of end-points may be used simply for the purpose of monitoring, or may potentially be used for intelligently choosing which equal-cost path a new flow/flowlet will take in the physical network -- as opposed to blindly hashing the new flow to one of the equal-cost paths. Intelligent utilization-aware routing will result in better administrative control, higher throughput, and lower job-completion times.

(iii) Latency – End-to-end latency along different paths between a pair of end-points may also be used simply for the purpose of monitoring, or can potentially be used for intelligent routing of latency-sensitive application traffic.

The premise of INT is quite simple: the traditional methods for managing networks are almost exclusively based on a client/server or “pull” model. A management device periodically polls the devices of interest for specific data, these requests are handled by the local control plane stack, and the data is collected and sent back. Generations of networks have proven the shortcomings of this model arising from the limitations of CPU-based control planes and an explosive increase in the number of network elements and ports (both virtual and physical) and link bandwidths.

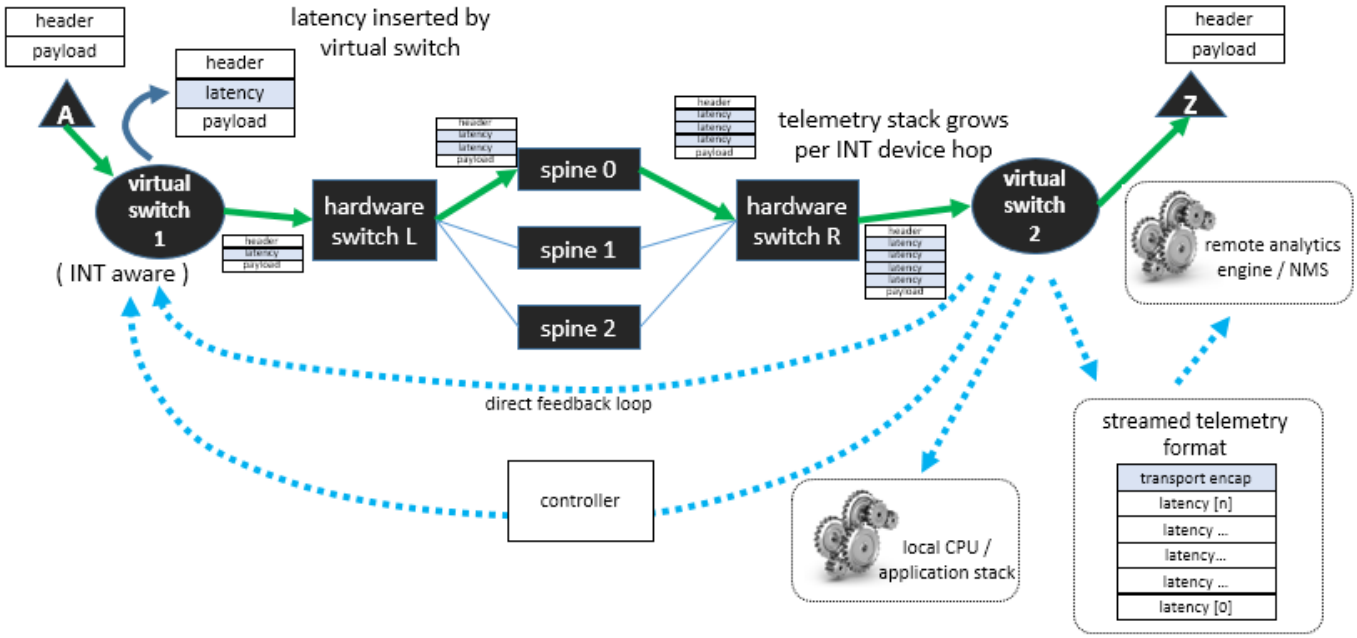
In the INT model, information from the data plane of the network is extracted and exported directly ***from*** the data plane – without the overhead or scale limitations of a control plane CPU. Moreover, the information that resides in the data plane is often ephemeral in nature: state on such things as queue depths, packet drops, and routing / ECMP path selections often change within milliseconds, and if they are not tracked or made visible at the data plane the information (or potential problem) goes unnoticed. Having a programmatic interface directly into the data plane is thus a key requirement for the development of these types of applications.

In network deployments with INT, the source vSwitch embeds “instructions” in packets that specify the desired network state that intermediate INT-aware network elements insert in the packet. The destination vSwitch collects the reported state. The destination vSwitch may export the collected data to a local CPU for consumption on a local device, export the data to a remote server, or feed back the data to the sending vSwitch for it to take certain actions based on the data. A combination of these actions may also be taken.

P4 is ideally suited for developing these types of INT applications: it is general enough to express multiple packet formats and header fields, it is very flexible in how this

information can be encapsulated and/or encoded, and provides primitives such as registers to perform meaningful analysis, state management, and thresholding -- all within the data plane. The ability to manipulate headers in a flexible way is a key part of the solution, another is the information made visible by the data plane itself. This model can be further extended when multiple devices are involved: the register capabilities in P4 allow for functions such as comparison, thresholding, and stateful processing of the telemetry information.

A simple INT use case:
Measuring and reporting end-to-end latency between virtual switches



An illustrative example of INT is described here. Source vSwitch (vSwitch 1) embeds instructions for each network element to report the latency that the packet encounters at the network element (delta between local egress timestamp and local ingress timestamp). The receiving vSwitch (vSwitch2) can compute the end-to-end latency as a sum of the per-hop latencies (under the assumption that switching and queueing latencies dominate and propagation delays are minimal, which is typically true in today's networks). Per-hop latencies in the packet received at the destination vSwitch can also be used to determine which network element(s) contributed most to the end-to-end latency.

While the above example illustrates collection of a single piece of network state, multiple types of network state could be collected together. For example, <switch-ID, input-port, output-port, egress-link-utilization> may all be collected together to discover the different paths between a pair of end-points and the congestion level along each path.

There is substantial value in being able to insert or modify headers with custom metadata, but it is also important to be able to somehow filter or constrain how much of this information is to be acted upon. The P4 language allows basic stateful operations

that can check for thresholds on this data, allowing the user to filter out “normal” traffic, only taking actions when specific conditions are met or specific dataplane events occur. Once these foundational capabilities have been added into the network, a whole new range of applications, debugging, and optimizations become possible. Dynamically monitoring link utilization (and reacting do that information instantly), automatic fast reroute (around not just total failures but also around partially degraded links), and hardware-accurate path and latency tracing: these and other applications are enabled by a programmable data plane, the high level language, and the INT building blocks.

While this blog mainly focuses on use of INT in virtualized network deployments with vSwitches at the edge, INT provides immense value in networks in general. The visibility provided by INT enables real-time debugging of network issues, and the automation of such tasks: think “self healing” networks. This telemetry is also the baseline information for the next generation of network management applications and dynamic optimization/allocation of network resources such as link bandwidth, forwarding memories, and packet buffers. As networks continue to scale out in terms of numbers of links, devices, and scope, one of the main challenges for operators has become monitoring, maintaining and supporting these critical infrastructures. It is increasingly obvious that the traditional ways of managing networks simply do not provide the flexibility, visibility, or collection efficiency required for today’s networks.

Hopefully we’ve described how P4 adds the ability to easily and incrementally write programs that can gather and distribute information directly from the data plane in a customizable, programmable way. The ability to publish this information in an arbitrary format obviously has great value, but there is also value in having a baseline format or specification for how the data might be collected and exported. To this end, we have created an initial version of the INT specification, which provides a framework of packet formats and encapsulation methods that can be used to implement these types of telemetry and monitoring features. The current version of the INT spec can be accessed at http://p4.org/wp-content/uploads/fixed/INT/INT-current-spec.pdf. Additionally, an implementation of INT using the open-source P4 behavioral model has been posted at https://github.com/p4lang/p4factory/tree/master/apps/int. We also have an introductory video that describes some of the concepts and provides another visualizer of an application using INT.

The INT source code is permissively licensed, so feel free to go pull a clone and start thinking about your own telemetry based applications.

About the authors: Mukesh Hira is a Staff Software Engineer in the Networking and Security Business Unit of VMware where he leads some of the NSX data plane projects. He holds a PhD in Electrical Engineering from Stanford University and has over 15 years of experience in design and development of networking software. LJ Wobker is a Technical Marketing Engineer at Barefoot Networks, working on programmable dataplanes and network monitoring and instrumentation. Prior to Barefoot, he worked

for 15+ years at Cisco on scaling very large networks, distributed routing system design, and high speed packet forwarding silicon.