Low-overhead packet loss and one-way delay measurements in Service Provider SDN

Wolfgang John and Catalin Meirosu
Ericsson Research, Stockholm, Sweden

In service provider networks, operators will take advantage of the possibilities to aggregate traffic by applying coarse-grain flow definitions, which can be realized in modern OpenFlow [1] by applying wildcards in certain fields of the flow definition table. Service provider SLAs depend on the ability to monitor performance metrics, including packet loss rates, delay, etc. Such measurement capabilities provide operators with greater visibility into the performance characteristics of their networks, thereby facilitating planning, troubleshooting, and network performance evaluation. While this capability is important, operators try to limit the overhead of management-specific traffic in operational networks as much as possible. Good practices call for the entire management traffic to not exceed about 5% of the total aggregated data traffic. Another aspect of the management overhead associated with monitoring is the signaling related to setting up and tearing off monitoring sessions, exemplified by protocols such as OWAMP and TWAMP in packet networks or GMPLS extensions for setting up OAM tools in MPLS and carrier Ethernet networks.

1. State-of-the-Art SDN monitoring

OpenFlow counter based solutions of [4-6] do not specify how aggregated and wild carded flows could be used for accurate loss measurements. The selection of specific microflows (i.e. exact flow definitions without wildcards) for retrieving packet-loss samples is not covered by any of the solutions in [2, 4-6]. The validity of the packet loss calculations made by directly comparing OpenFlow per-flow packet counters is negatively affected by grouping flows using wildcard rules in two ways:

a. There is no guarantee that wild-carded flow definitions at one switch correspond to the same definition on other switches, i.e. wild-carded flow definitions do not reflect the same set of aggregated microflows. This can be caused by varying flow definitions on different locations along a path, or use of microflows at intermediate switches, diverting parts of the original aggregated flow to other network paths.
b. Pull-based mechanisms applied during flow life-time create synchronization problems due to the unsure state of in-flight packets when comparing packet counters for packet-loss measurements. Furthermore, wildcarded-rules providing flow aggregation are often pre-installed and are unlikely to timeout at all, or at least very seldom. As a result, start and end of aggregated flows are not clearly defined, providing not sufficient data points for continuous packet-loss measurements.

DevoFlow [3] is focused around throughput and link utilization measurements. It runs the risk of generating large amounts of management traffic when an elephant flow receives the same automatic threshold for generating notifications towards the controller as a mice flow. Per-flow single triggers would need to be configured based on the nature of the flow (elephant or mice), meaning automated adaptation is difficult without controller intervention and require even more complexity on the data plane switch. Automatic rule cloning is not controlled in any way, which may translate onto requirements for large flow tables, potential resource exhaustion or aggressive eviction of controller-installed flows. It is difficult to accurately calculate delay since timestamping has to be performed in the controller when reports arrive (hence the interval would include the delay between the switch and the controller, affecting mice as well as intermediary periodic reports from elephants).

FlowSense [2] presents a method for monitoring network utilization by making use of message exchanges typical to the OpenFlow protocol. The method thus introduces no additional traffic into the network, hence the claim of “zero cost”. Service provider SDN scenarios with pre-provisioned and aggregated flow definitions (using wildcard rules) cannot be covered by straightforward extensions of this method, because such flows are pre-installed and do not expire. Thus the markers needed to determine the lifetime of the flow are not generated. Also, the method does not describe how to identify the same packet or packets sequence at different monitoring points, which is required for loss measurements.

2. Low-overhead packet loss measurements

We depict our method for estimation of packet loss (and optionally one-way delay) in Fig. 1. The method is taking advantage of user traffic transported using aggregated flow descriptors, as common in SP-SDN scenarios. An edge node identifies at the starting point of the measurements a microflow part of an aggregated flow descriptor according to policies delegated by a controller. The edgenode then automatically creates the required per-microflow descriptors within the switch and configures the entry (e.g. time-out values) in a way that adapts to the nature of the microflow (e.g. elephant vs mice). Next, the edgenode coordinates further measurement points via the controller with other switches in the measurement path. Once the microflow expires, notifications including per-microflow packet counts are generated towards the controller from all measurement points along the flow path (i.e. switches where the microflows have been configured). The controller estimates the packet loss based on the received messages. One-way delay measurements could also be performed by including timestamps in the notifications. Due to space constraints, we will briefly mention here only certain key parts of our method.

Microflows to be used for packet-loss measurements are selected based on policies configured by the controller on the edge switch. Examples of policies can be ALL microflows, sampled subsets of existing microflows (random, systematic, etc.), or based on specific packet header field values. The OpenFlow messaging protocol is extended to notify the controller of microflows selected by the edge node that starts the measurement. For the selected microflows, at least the flow definition (match structure) needs to be communicated to the controller.

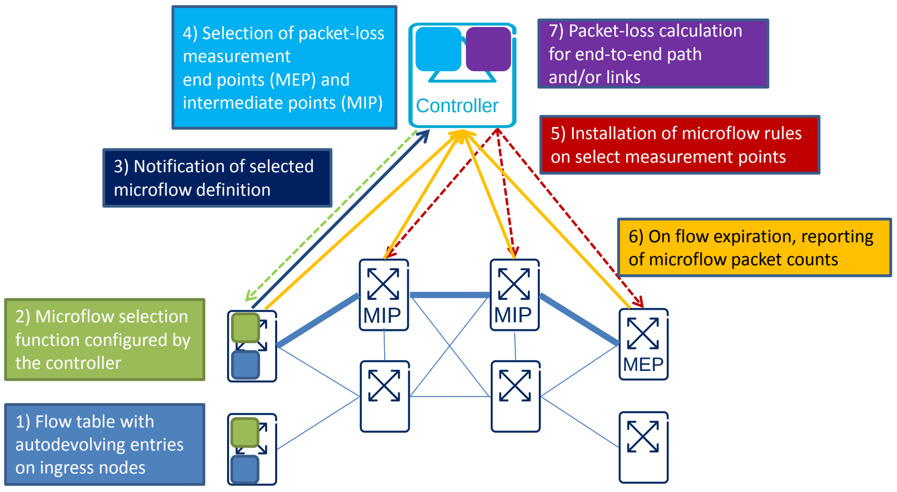
4) Selection of packet-loss measurement end points (MEP) and intermediate points (MIP)
Controller
7) Packet-loss calculation for end-to-end path and/or links
3) Notification of selected microflow definition
5) Installation of microflow rules on select measurement points
6) On flow expiration, reporting of microflow packet counts
MIP
MIP
MEP
2) Microflow selection function configured by the controller
1) Flow table with autodevolving entries on ingress nodes

Figure 1: Components and processes realizing low-overhead packet loss measurements.

Within a switch, we define a new flow cloning mechanism adapted to multiple tables as specified in modern OpenFlow. The auto-devolving functionality can be applied to any table in the pipeline for the multiple table abstraction. However, it needs to be ensured that the instructions associated with the wildcard rules (e.g. *goto-table* instructions) are inherited by the auto-devolved microflows, so that the path through the pipeline remains unaltered. By choosing which table in the pipeline is to perform auto-devolvement for a particular aggregate rule, the controller is allowed to select the level of granularity of the devolved flows, since rules in preceding tables might already have acted as filters of which subset of flows to devolve. In addition to the standard OpenFlow 1.4 eviction and vacancy mechanisms, each aggregate rule is allocated a certain devolvement space in the table in order to avoid overconsumption of the available flow table space. This is configured by the controller when the rule is installed and it can be chosen to reflect knowledge about the level of aggregation expected within this flow. The devolvement process for a particular rule would then only use this space.

On expiration of a microflow on measurement points (e.g. through idle-timeout), a *flow_removed* notification is sent to the controller. This is existing OpenFlow behavior and no extensions are required. The *flow_removed* message includes among others packet counts at the time of removal. An application on the controller has thus accurate values to retrieve packet-loss rates for the specific flow by subtraction of per-flow packet counters.

A way to realize unidirectional flow delay measurements is based on the assumption that switches are time synchronized (e.g. through NTP, IEEE 1588, etc.), allowing them to timestamp flow expirations. Assuming that flow inactivity timers on all nodes are known by the controller, the delay could be obtained by calculating the differences between the flow expiration timestamps, accounting for potentially different inactivity timeouts.

We are working towards implementing and evaluating low-overhead packet loss measurements for wildcarded flows as part of our contribution to the FP7 UNIFY project (http://www.fp7-unify.eu/). One of the objectives of the WP on Service Provider DevOps is to develop scalable and programmable SDN monitoring capabilities [7], where this work will contribute.

References

[1] Openflow Switch Specification, v.1.4.
Online: http://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf

[2] Curtis Yu et al., “FlowSense: Monitoring Network Utilization with Zero Measurement Cost”, in PAM, 2013

[3] Jeffrey C. Mogul et al., “DevoFlow: cost-effective flow management for high performance enterprise networks”, in Hotnets-IX, 2010

[4] M. Castrucci and A. Simeoni, “Extension of the OpenFlow framework to foster the Software Defined Network paradigm in the Future Internet”, GARR Conference, 2011

[5] V.N. Gourov, “Network Monitoring with Software Defined Networking”, master Thesis, TU Delft, 2013

[6] D.M.F. Mattos et al. "OMNI: Openflow management infrastructure", in NoF, 2011

[7] A. Császár et al. "Unifying Cloud and Carrier Network", in DCC, 2013