An OpenFlow Switch Element for Click*

Yogesh Mundada†, Rob Sherwood*, Nick Feamster†
† Georgia Tech * Deutsche Telekom Inc. R&D Lab

1. Summary

Conventional network middleboxes process packets either per-flow (e.g., forwarding tables, MPLS) or per-packet (e.g., DPI, IDS, firewalls). Each approach offers protocol designers different levels of control. Flow-processing simplifies "macro"-level decisions by abstracting away the packet-by-packet details, while packet-processing exposes those details, providing "micro"-level control. Currently, due to architectural or hardware constraints, network protocols must typically use one paradigm or the other.

We will present an OpenFlow element for Click, which allows *hybrid* packet and flow processing. Such a hybrid model could offer the best of both worlds: the flexibility of packet-based processing, and the simplicity flow-based processing. The talk will include motivation, a description of the design, and a demonstration of the element in use.

The recently proposed OpenFlow protocol [1] provides a common interface to control how packets are forwarded. Using OpenFlow, a centralized controller manipulates the flow processing properties across the network. Click allows a researchers to create customized packet processing devices by interconnecting various elements. but the element paths traversed by the packets at run time are determined statically and cannot be changed dynamically.

To create a hybrid of the two models, we have added an OpenFlow interface to the Click router with the `OpenFlowClick` element [2]. This element allows a controller to install rules to make packets traverse different element paths. It also allows a single controller to control multiple Click routers at the same time.

This interface opens up many possibilities for new classes of traffic-processing applications. For example, hybrid elements could eliminate duplicate packets, take specific actions based on pre-determined packet-based rules, perform exception handling and anomaly detection based on periodic inspection of payloads, and generally combine decentralized packet processing with centralized control. Figure 1 shows a generic use case scenario. Despite these possibilities, this hybrid design also introduces several challenges, including deciding which version of Click to use to implement an OpenFlow element, modifying specific kernel mechanisms, and abstracting various functions into the appropriate classes.

2. Design and Implementation

Currently, the `OpenFlowClick` runs as the Click kernel module. The secchan and dpctl utilities from OpenFlow are used with this element without modification. Secchan establishes secure communication channel with the controller.

*Yogesh Mundada performed this work as an intern at Deutsche Telekom.

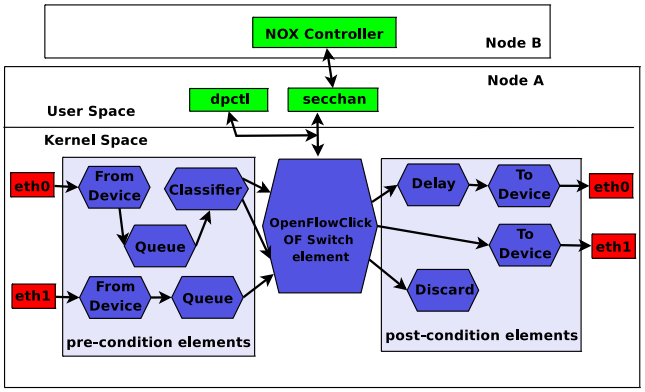

Figure 1: An example OpenFlow+Click network.

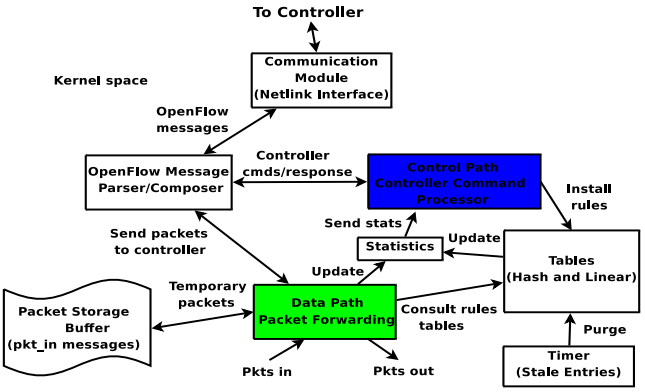

Figure 2: OpenFlowClick Architecture

Dpctl passes commands to the element from user space. Figure 2 shows the internal architecture of this element.

Communication between the OpenFlowClick element in kernel space and secchan in user space occurs over a netlink interface. The control path installs packet forwarding rules received from controller. The data path module actually matches rules and forwards packets. Linear and hash tables are used to store wild card and exact match rules respectively. The packet buffer stores the packets awaiting decision from the controller and a periodic timer module deletes the timed out rules from the table.

REFERENCES

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: enabling innovation in campus networks. 38(2):69–74, April 2008.

[2] OpenFlowClick. `http://www.openflowswitch.org/wk/index.php/OpenFlowClick`, 2009.