Electronic Communications of the EASST
Volume 17 (2009)

Workshops der Wissenschaftlichen Konferenz Kommunikation in Verteilten Systemen 2009 (WowKiVS 2009)

Flowstream Architectures

Adam Greenhalgh, Mark Handley, Mickaël Hoerdt, Felipe Huici, Laurent Mathy and Panagiotis Papadimitriou

5 pages

Guest Editors: M. Wagner, D. Hogrefe, K. Geihs, K. David
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Flowstream Architectures

Adam Greenhalgh[1], Mark Handley[2], Mickaël Hoerdt[3], Felipe Huici[4], Laurent Mathy[5] and Panagiotis Papadimitriou[6]

[3] m.hoerdt@lancaster.ac.uk [5] l.mathy@lancaster.ac.uk [6] p.papadimitriou@lancaster.ac.uk
Computing Dept., Lancaster University, UK

[1] a.greenhalgh@cs.ucl.ac.uk [2] m.handley@cs.ucl.ac.uk
Dept. of Computer Science, University College London, UK

[4] felipe.huici@nw.neclab.eu
NEC Europe, Heidelberg, Germany

Abstract: The Internet has seen a proliferation of specialized middlebox devices that carry out crucial network functionality such as load balancing, packet inspection or intrusion detection, amongst others. Traditionally, high performance network devices have been built on custom multi-core, specialized memory hierarchies, architectures which are well suited to packet processing. Recently, commodity PC hardware has experienced a move to multiple multi-core chips, as well as the routine inclusion of multiple memory hierarchies in the so-called NUMA architectures. While a PC architecture is obviously not specifically targeted to network applications, it nevertheless provides high performance cheaply. Furthermore, a few commodity switch technologies have recently emerged offering the possibility to control the switching of flows in a rather fine grained manner. Put together, these new technologies offer a new network commodity platform enabling new flow processing and forwarding at an unprecedented flexibility and low cost.

Keywords: virtualization, router, platform architecture, commodity hardware

1 Introduction

The Internet has seen a proliferation of specialized middlebox devices that carry out crucial network functionality such as load balancing, packet inspection or intrusion detection, amongst others. At the same time, we experience a trend towards the commoditization of hardware, which allows for cheap and extremely capable switching and processing components (e.g., multi-core chips). A few commodity switch technologies have recently emerged offering the possibility to control the switching of flows in a rather fine grained manner [1]. Put together, these new technologies render commodity hardware a viable platform for flow processing and forwarding at an unprecedented flexibility and low cost. In this context, we propose a generic network control, forwarding and flow processing platform built from commodity switch hardware and a small cluster of servers. Such a platform is inexpensive, very flexible, scalable, and failure tolerant.

2 Flowstream Architectures

We call such platforms “Flowstream Architectures”, for reasons that should be clear shortly. Platforms built according to the Flowstream architecture can be characterized by the following properties:

- The core of the platform consists of an ethernet switch configured to route flows. A flow is defined in the OpenFlow sense, as packets that match a (possibly wildcarded) tuple of source and destination addresses and ports.
- Streams of data from these flows are then routed to one of a number of attached commodity server boxes for additional processing, before being forwarded on to the final destination. The server boxes can also act as traffic sinks.
- Software running on the server boxes can be composed to provide processing pipelines of modules.
- These modules are virtualized, in the sense that they can be moved between the servers to balance load and provide robust service in the presence of failures.
- The switch and servers are managed as a single platform from the point of view of the operators.

2.1 Description of a Platform

The platform consists of a flow-based switch, the servers (which we call *module hosts* to distinguish them from traditional servers), and a controller (see figure 1). Each host runs a number of *processing modules* where all of the actual flow processing takes place except for basic forwarding which can be done by the switch. Further, hosts contain a special module called a *control module*, which receives commands from the platform’s controller to remove, install or migrate modules, as well as to provide monitoring information about the host’s load and performance.

There are three main technologies available to us for implementing a module:

- A virtual machine running its own OS and module application.
- A process running on a virtual machine shared with other modules.
- A set of kernel forwarding elements instantiated in the kernel of the device driver domain on one of the module hosts.

The first of these options is the most general and provides the best inter-module isolation, whereas the third will provide the highest performance for traffic that needs to traverse several modules in the same module host. We envisage different applications will use different implementation options, often on the same Flowstream platform.

For composing kernel forwarding elements, the Click modular router [4] provides a suitable set of building blocks. For example, a module can be composed of a predefined set of Click elements, and under the control of the operator, cascades of such modules can be plumbed together at run-time.

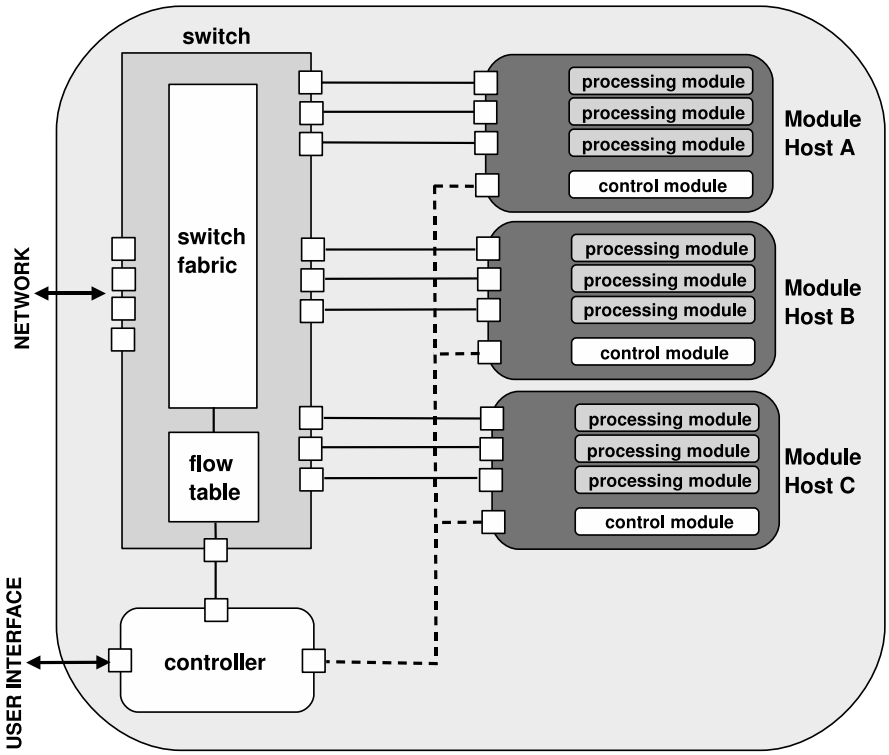
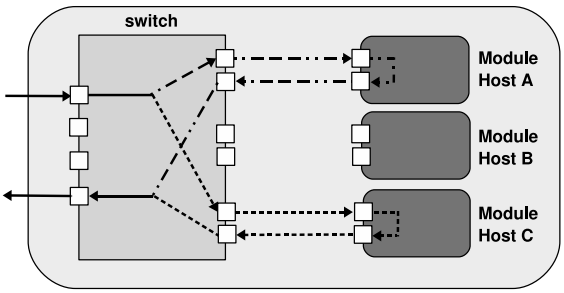


Figure 1: Overview of a Flowstream platform.

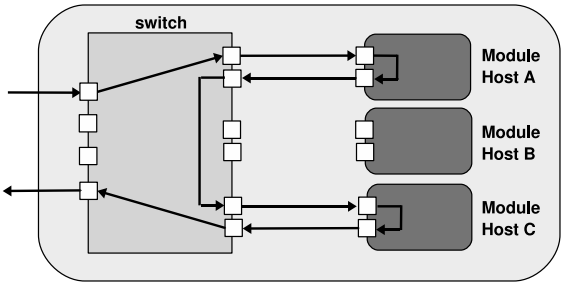
A Flowstream platform's second main component is the Openflow switch [1], providing the basic connectivity between module hosts and the network. In addition to this, the switch contains a flow table which is configured by the controller at runtime, allowing different flows to be directed to any of the ports on the switch. It is worth pointing out that while figure 1 shows a single switch, it would be certainly possible to scale the platform's port density by including additional switches.

The final component is the controller. Essentially, this is the brains of the platform and also its user interface to the outside world. When the operator makes a request (for instance, running an IDS on flows to a particular web server), the controller begins by choosing the module host or hosts to install the processing module(s) on. Such a decision could be based on the hosts' current load, information that the controller retrieves periodically from the control modules. Having selected a host, the controller then instructs the control module to install the requested processing module. Once this is done, the controller configures the switch's flow table so that the corresponding flows are directed to the right processing module.

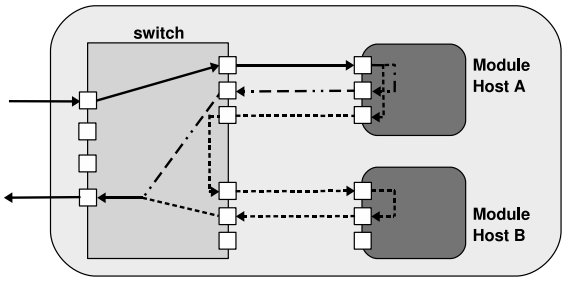
With all of these components in place, a Flowstream architecture provides a powerful platform for flow processing. The fact that it is built upon commodity yet, as shown in previous work [2, 3], high performance hardware should result in significant cost savings. In addition, a Flowstream setup can be easily expanded and contracted dynamically by adding or removing module hosts, something that cannot be easily accomplished on conventional routers or middleboxes. Further, when required, the isolation provided by virtualized module hosts allows



(a) Parallel processing (load-balancing) scenario.



(b) Serial processing scenario.



(c) Flow splitting scenario.

Figure 2: Basic platform usage scenarios.

several different flow processing operations to be performed simultaneously while minimizing negative interactions. The controller can also migrate modules as required to ensure that a processing task does not significantly degrade the performance of others. Last but not least, using general-purpose processors and allowing operators to install their own flow processing modules yields great flexibility. So long as modules have access to well-defined flow APIs, a Flowstream platform can accommodate a wide range of existing and even future network applications. It is precisely the usage of the platform that we discuss next.

2.2 Usage Scenarios

In the most basic case, the operator places a request to a Flowstream platform's controller asking it to apply a certain processing module to a subset of the traffic being forwarded. The controller then chooses a module host with appropriate load levels and installs the module on it, also config-

uring the switch's flow table. The flow then travels from the switch to the module for processing, before being sent back to the switch and then out onto the network.

Beyond the simple case, there are two more interesting usage scenarios, depending on whether modules act on flows in parallel or serially. In parallel processing (see figure 2(a)), flows are load-balanced, pushing different flows to different module hosts but processing each of them equally. In serial processing or pipelining (see figure 2(b)), the operations performed on flows are split across several module hosts and done one at a time. Combinations of serial and parallel are certainly possible.

A more complex usage scenario is *flow splitting*, whereby a processing module is used to split a subset of traffic from a flow aggregate to another module for further processing (see figure 2(c)). As long as its capabilities are sufficient, the switch can also be used to split traffic.

2.3 Module Migration

Flowstream architectures fit firmly into the trend of using arrays of cheap and potentially unreliable hardware, but providing robustness in software. To provide such robustness, we need to be able to migrate modules between hosts, both to manage changing load and to adapt to failures. It is perhaps this ability to migrate processing functions between hardware while simultaneously re-plumbing the switch's flow table to match, that perhaps best illustrates the flexibility of Flowstream architectures. This flexibility can even be used to power down underused module hosts during quiet hours to save on electricity costs.

3 Conclusions

We presented Flowstream, a new class of system architectures for in-network processing platforms that has emerged from the confluence of the commoditization of switch and x86 server hardware. Because they are inexpensive, very flexible, scalable and failure tolerant we believe that such platforms can be used to implement the functionality of the middleboxes that are currently required for the Internet to operate, as well as future ones.

Bibliography

[1] Open Flow Switch Consortium. Open flow switch. http://www.openflowswitch.org.

[2] Norbert Egi, Adam Greenhalgh, Mark Handley, Mickael Hoerdt, Felipe Huici, and Laurent Mathy. Towards high performance virtual routers on commodity hardware. In *Proceedings of ACM CoNEXT 2008*, Madrid, Spain, December 2008.

[3] Norbert Egi, Adam Greenhalgh, Mark Handley, Mickael Hoerdt, and Laurent Mathy. Virtual router project. http://nrg.cs.ucl.ac.uk/vrouter/.

[4] Eddie Kohler, Robert Morris, Benjie Chen, John Jahnotti, and M. Frans Kasshoek. The click modular router. *ACM Transaction on Computer Systems*, 18(3):263–297, 2000.