FlashFlow: a GPU-based Fully Programmable OpenFlow Switch

Ghulam Memon, University of Oregon
Matteo Varvello, Alcatel Lucent
Rafael Läufer, Alcatel Lucent
T.V. Lakshman, Alcatel Lucent
Jun Li, University of Oregon
Mingwei Zhang, University of Oregon, Poster Author

Introducing FlashFlow

OpenFlow[1] is a communication protocol between network switches and controllers, which decouples the data plane from the control plane in today's networks. OpenFlow was designed to allow researchers to experiment with novel protocols and algorithms within campus network. However, recently, OpenFlow has been adopted by larger and faster networks (e.g., data-centers), and for more complex operations (e.g., traffic management). Several vendors (CISCO, HP and Juniper) now offer OpenFlow support in their switches. Ironically, the proprietary nature of commercial OpenFlow switches creates a new barrier for the research community: researchers cannot experiment with the design of the switches. Motivated by this observation, we hereby design, prototype and evaluate FlashFlow, the first Software-based OpenFlow switch.

FlashFlow leverages GPU acceleration to operate at high-speed: our prototype aims to reach 20 Gbps when working with a flow table that contains 1,000,000 exact match entries and 1,000 wild-card entries, the largest flow table supported by a state-of-the-art OpenFlow hardware switch. Most importantly, FlashFlow is programmable.As a proof of concept, we enhance FlashFlow with several mechanisms that researchers have recently proposed to address some OpenFlow's limitations, but that could not be directly tested in hardware switches.

Why FlashFlow?

High-Speed

FlashFlow aims to reach a speed of 20 Gbps while dealing with flow tables that contain 1 Million exact match entries and 1,000 wildcard entries.

Programmability

We enhance FlashFlow with several features from DevoFlow[4] which could not be integrated in an hardware switch.

Overview of FlashFlow

This work designs, implements and evaluates FlashFlow, the first Software-based OpenFlow switch. FlashFlow solves current limitations of OpenFlow switches while allowing for complete programmability in a SDN spirit.

The rationale of FlashFlow is to leverage the high parallelism of GPU to perform high-speed packet processing, whereas delegating to the CPU all other OpenFlow operations, e.g., statistics maintenance and communication with the controller.

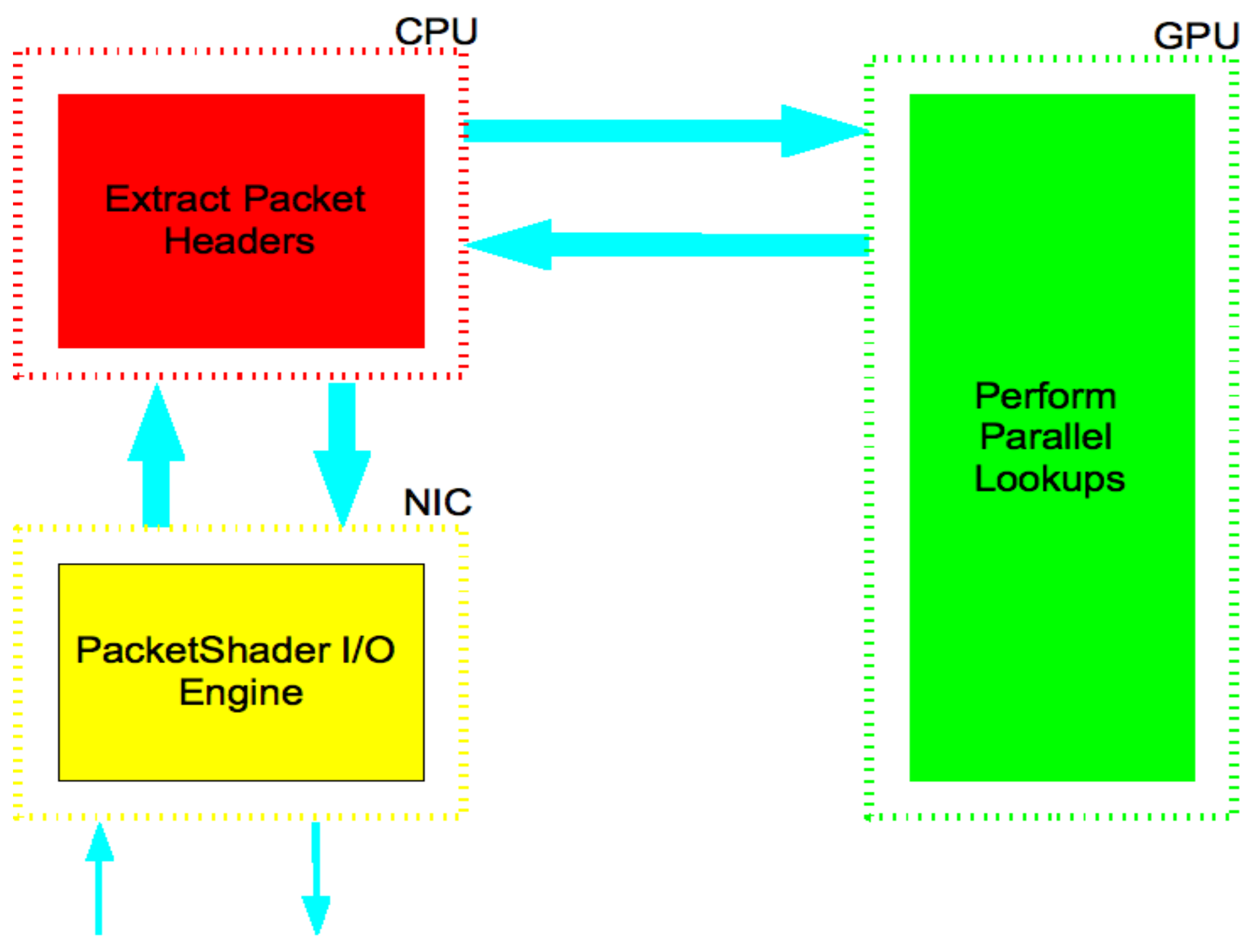

Fig. 1 FlashFlow basic structure

Work flow of FlashFlow

In the CPU module, a group of packets is received(PacketSharder[2]), and their headers are extracted (2). This set of headers is then sent to the GPU module (3) where exact matching is performed (4). The set of rules derived from exact matching are sent back to the CPU module (5) where (6): a) the actions associated to these rules are retrieved, b) eventual absence of actions for a set of headers are verified. If a), the action associated to each rule is performed, e.g., a packet is forwarded (7a). If b), a wildcard matching operation is required; accordingly, this set of headers is sent back to the GPU module (7b) where wildcard matching is performed (8).
As above, the rules obtained via wildcard matching are sent back to the CPU module (9) in order to retrieve the corresponding actions (10). Differently from above, in case of absence of action for a given header, the intervention of the controller is needed: the CPU module SSL encrypts the packet associated to this header (11) and sends it to the controller (12). The controller identifies the <rule,action> pair for this packet (13) and informs the CPU module (14) which updates its set of rules accordingly.

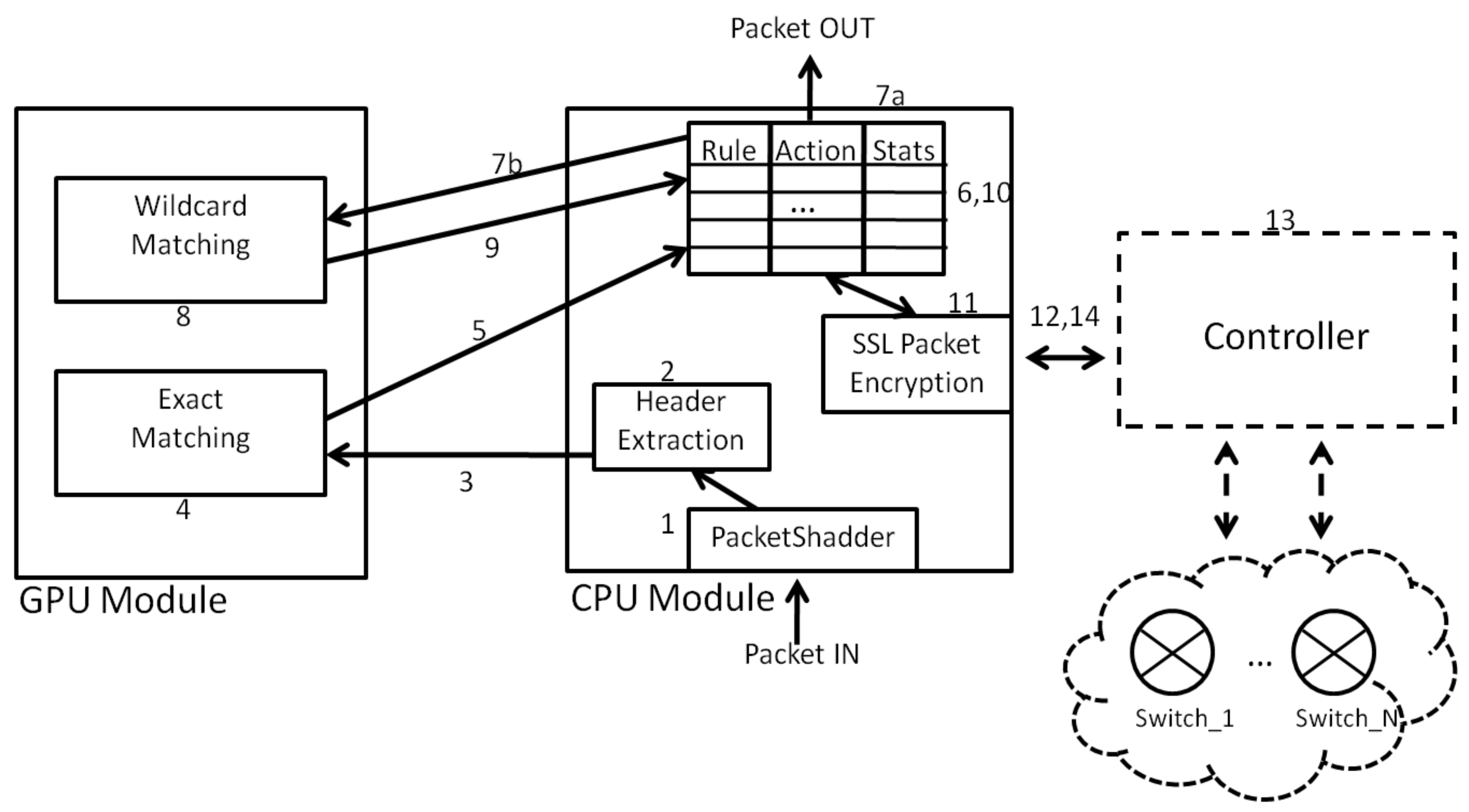

Fig. 2 FlashFlow's design and operational flow

Implementation

We have implemented FlashFlow on a commodity machine powered by
*) Two quad-core Xeon X5550 processors
*) Two NVIDIA GTX580 cards
*) Two Dual Port 10G NICs with Intel 82599EB Chipset

FlashFlow leverages the high parallelism of Graphic Processing Unit (GPU) to accommodate:
1) large exact matching and wild-card matching tables,
2) high number of new flows per second,
3) high programmability.

References:

1. McKeown, Nick, et al. "OpenFlow: enabling innovation in campus networks."ACM SIGCOMM Computer Communication Review 38.2 (2008): 69-74.

2. Han, Sangjin, et al. "PacketShader: a GPU-accelerated software router." ACM SIGCOMM Computer Communication Review 40.4 (2010): 195-206.

3. Rizzo, Luigi. "netmap: a novel framework for fast packet I/O." USENIX ATC. 2012.

4. Curtis, Andrew R., et al. "DevoFlow: scaling flow management for high-performance networks." SIGCOMM-Computer Communication Review 41.4 (2011): 254.