Δίκτυα Καθοριζόμενα από Λογισμικό
Ενότητα 3.2: SDN Switches: Αρχιτεκτονική και Σχεδιασμός

Ξενοφώντας Δημητρόπουλος
Τμήμα Επιστήμης Υπολογιστών
SDN Switches: Architecture and Design

Xenofontas Dimitropoulos
24/11/2014

Slides prepared by: Markus Happe

Also credits to online material by Raj Jain, Nick Bastin, Rui Miao, Nick McKeown, Lorenzo de Carli, and the Open Networking Foundation
Software-Defined Networking: Switches

Well-defined open API

Constructs a logical map of the network

Network OS

Open vendor agnostic protocol

Simple Packet Forwarding Hardware

OpenFlow
## SDN Switch: Simple Packet Forwarding Hardware

### Flow Table

<table>
<thead>
<tr>
<th>MAC src</th>
<th>MAC dst</th>
<th>IP Src</th>
<th>IP Dst</th>
<th>TCP sport</th>
<th>TCP dport</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>5.6.7.8</td>
<td>*</td>
<td>*</td>
<td>port 1</td>
</tr>
</tbody>
</table>

### Diagram

- **Software Layer**
  - OpenFlow Agent
  - Flow Table
  - OpenFlow Protocol

- **Hardware Layer**
  - Port 1
  - Port 2
  - Port 3
  - Port 4

- **Network Devices**
  - 5.6.7.8
  - 1.2.3.4
SDN Switch: Simple Packet Forwarding Hardware

- **Controller**
  - writes forwarding table(s) of the switch

- **Switch**
  - forwards packets to controller, if there is no matching flow table entry
  - needs to forward packets according to flow table(s)

  - multiple full-duplex Ethernet ports: e.g. 4, 8, 24, 48, etc.
  - where each port has 1GbE, 10GbE, etc.
  - → back plane needs to process millions of packets per second

How can an SDN switch process millions of packets per second?
Lecture Overview

Part I: Efficient Flow-Action Matching

Part II: Architecture and Design of SDN Switches

Part III: Configuration and Management of SDN Switches

Part IV: Next Generation of SDN Switches
Part I: Efficient Flow-Action Matching (How to Match Packets to Flow Tables?)
Efficient Flow-Action Matching Types in SDN

- **Exact rules**
  - all (selected) header fields are defined in flow table
  - incoming packet can be matched to a unique exact rule

- **Longest prefix rules**
  - select flow rule with longest matching prefix
  - e.g. 200.124.12.*, 200.124.*.*, 200.*.*.*
  - example: IPv4/IPv6 destination address lookup

- **Wildcard rules**
  - some header fields contain wildcards (*)
  - example: access-control list lookup (firewall)

- **Multiple rules might match incoming packet**
  - prioritization required to identify matching rule
Exact Flow-Action Matches: Naive Approach

- Flow table stored in memory (e.g. SRAM)
  - assumption: flow table entries are unsorted
  - linear search of table entries in memory
  - stop when match is found (or when reaching final flow table entry)
  - **but: low performance for long tables**
  - we only have few clock cycles for matching

<table>
<thead>
<tr>
<th>no.</th>
<th>header 1</th>
<th>header 2</th>
<th>stats</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>11</td>
<td>01</td>
<td>stats0</td>
<td>act0</td>
</tr>
<tr>
<td>1</td>
<td>00</td>
<td>10</td>
<td>stats1</td>
<td>act1</td>
</tr>
<tr>
<td>2</td>
<td>01</td>
<td>01</td>
<td>stats2</td>
<td>act2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>N</td>
<td>01</td>
<td>01</td>
<td>statsN</td>
<td>actN</td>
</tr>
</tbody>
</table>
## Exact Flow-Action Matches: Naive Approach

- Flow table stored in memory (e.g. SRAM)
  - assumption: flow table entries are unsorted
  - linear search of table entries in memory
  - stop when match is found (or when reaching final flow table entry)
  - **but: low performance for long tables**
  - we only have few clock cycles

<table>
<thead>
<tr>
<th>number</th>
<th>header 1</th>
<th>header 2</th>
<th>stats</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>11</td>
<td>01</td>
<td>stats0</td>
<td>act0</td>
</tr>
<tr>
<td>1</td>
<td>00</td>
<td>10</td>
<td>stats1</td>
<td>act1</td>
</tr>
<tr>
<td>2</td>
<td>01</td>
<td>01</td>
<td>stats2</td>
<td>act2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>N</td>
<td>01</td>
<td>01</td>
<td>statsN</td>
<td>actN</td>
</tr>
</tbody>
</table>
Binary Content-Addressable Memory (CAM)

- Idea: parallel search of all memory entries
- Can be used for exact matches (and prefix matches)
  - e.g.: use multiple CAMs used for different 8/16/24-bit prefixes
- Advantage: matches packet to flow rule in a single operation
- Expensive and power hungry
Ternary Content-Addressable Memory (TCAM)

- Similar to CAM, but each header bit is encoded in two bits
  - 0 → 01, 1 → 10, don’t care (x) → 11
  - support for wildcards and prefixes
- Can be used for all kind of matches
- Very expensive, very power hungry

Algorithmic Approach: Hash Table

- Computes table position of rule from packet header
- Use hash function to map headers to flow table
- Can be used for exact matches
- But: flow table is much smaller than header space
  - collision: multiple headers have same hash value
  - use two independent hash functions to resolve collisions
  - alternative: use multiple flow tables, check them in parallel
Algorithmic Approach: Trie

- Can be used all match types
- Form trie from prefixes or header fields
- Packets traverse trie in a pipeline (pipeline stage = trie stage)
- Matching requires several operations (= trie depth)
- Trie can be compressed to save resources

Example flow table

<table>
<thead>
<tr>
<th>Rule</th>
<th>Prio</th>
<th>Field x</th>
<th>Field y</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>1</td>
<td>00~01</td>
<td>00~00</td>
</tr>
<tr>
<td>R2</td>
<td>2</td>
<td>00~01</td>
<td>00~11</td>
</tr>
<tr>
<td>R3</td>
<td>3</td>
<td>10~10</td>
<td>00~11</td>
</tr>
<tr>
<td>R4</td>
<td>4</td>
<td>11~11</td>
<td>11~11</td>
</tr>
<tr>
<td>R5</td>
<td>5</td>
<td>11~11</td>
<td>00~11</td>
</tr>
</tbody>
</table>
Summary: Efficient Flow-Action Matching

- **Challenge**
  - match millions of packets per second to long flow tables
  - only few clock cycles for matching

- **Content-addressable memory**
  - fastest, but also most expensive solution (power, area)
  - preferred in ASICs

- **Algorithmic approaches**
  - require few clock cycles, less expensive (power, area)
  - preferred for general-purpose and network processors

- **Further approaches**
  - optimized versions of algorithms or CAMs
  - combination of presented solutions
  - other solutions?
Part II: Architecture and Design of SDN Switches
OF Switch Design and Architectures

A) Software Test Switches

B) Commodity Hardware Switches: Merchant Silicon

C) Commodity Hardware Switches: Network Processors
A) Software Test Switches
Open vSwitch: Software Switch

- OpenFlow capable virtual software switch
  - used with hypervisors to interconnect to virtual machines within a host
  - and virtual machines between different hosts across networks
  - open source: www.openvswitch.org
  - included in Linux 3.3 per default
  - written in C / Python

- Features:
  - integrate well with virtual machine managers
  - supports tunnels, remote control, NetFlow, sFlow
  - default switch in XenServer, Xen Cloud Platform
  - supports Xen, Virtualbox, Proxmox VE, KVM
Open vSwitch Internals

- `ovs-ofctl`
- `ovs-dpctl`
- `ovs-appctl`
- `ovs-vsctl`
- `ovsdb-client`
- `ovsdb-tool`
- `ovs-vswitchd`
- `ovsdb-server`
- `ovsdb`
- `netlink`
- `source: de.slideshare.net/rajdeep/openvswitch-deep-live`
Open vSwitch

- **openvswitch_mod.ko**: kernel-space packet processing
  - in limited time due to hashing
  - if match: apply set of actions, update counters
  - if no match: go to user-space and eventually to the controller

- **ovs-vswitchd**: user-space packet processing
  - first packets of a flow are handled here (→actions, counters)
  - put new exact flow table rules to kernel hash tables
  - also: linear search in wildcard flow table (→actions, counters)

- Can be installed on some commodity switches
  - enables OpenFlow, but with poor performance
### Further Software Switches

<table>
<thead>
<tr>
<th>Switch</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indigo</td>
<td>open-source implementation that runs on physical switches and uses features of the ASICs to run OpenFlow</td>
</tr>
<tr>
<td>LINC</td>
<td>open-source implementation that runs on Linux, Solaris, Windows, MacOS and FreeBSD</td>
</tr>
<tr>
<td>Pantou</td>
<td>turns a commercial wireless router/access point to an OpenFlow-enabled switch. OpenFlow runs on OpenWRT supported generic Broadcom and some models of LinkSys and TP-Link access points with Broadcom and Atheros chipsets</td>
</tr>
<tr>
<td>Of13softswitch</td>
<td>user-space software switch (based on Ericsson TrafficLab 1.1 softswitch)</td>
</tr>
<tr>
<td>XORPlus</td>
<td>open-source switching software to drive high-performance ASICs. supports STP/RSTP/MSTP, LCAP, QoS, VLAN, LLDP, ACL, OSPF/ECMP, RIP, IGMP, IPv6, PIM-SM</td>
</tr>
</tbody>
</table>
Summary Software Test Switches

- **Advantages**
  - maximum flexibility: develop novel protocols, routing algorithms, etc.
  - unlimited flow table size, unlimited number of flow tables
  - simple implementation effort
  - simulate entire networks on single computer

- **Disadvantages**
  - slow flow matching performance
  - usually not used as switches in actual networks

- **Therefore...**
  - hardware support required to support switches with many ports at high line rates of 1GbE, 10 GbE, 100 GbE, 1 TbE, etc.
B) Commodity Hardware Switches: Merchant Silicon
## OpenFlow Vendors and Solutions

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Model / Series</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arista</td>
<td>7050, 7150, 7300, 7500</td>
<td>1.0</td>
</tr>
<tr>
<td>Brocade</td>
<td>ICX, VDX</td>
<td>1.3</td>
</tr>
<tr>
<td>Brocade</td>
<td>MLXe, CER, CES</td>
<td>1.3</td>
</tr>
<tr>
<td>Brocade</td>
<td>NetIron XMR</td>
<td>1.3</td>
</tr>
<tr>
<td>Extreme Networks</td>
<td>BlackDiamond 8000/X8, Summit X670</td>
<td>1.0</td>
</tr>
<tr>
<td>HP</td>
<td>2029, 3500/3500yl, 3800, 5400zl</td>
<td>1.0, 1.3</td>
</tr>
<tr>
<td>IBM</td>
<td>Programmable Network Controller, RackSwitches</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>G8264, G8264T, G8332, G8052, G8316</td>
<td></td>
</tr>
<tr>
<td>Juniper</td>
<td>EX, MX</td>
<td>1.0</td>
</tr>
<tr>
<td>NEC</td>
<td>PF5240, PF5820, PF1000</td>
<td>1.0</td>
</tr>
<tr>
<td>NEC</td>
<td>ProgrammableFlow Network Controller PF 6800</td>
<td>1.0, 1.3</td>
</tr>
<tr>
<td>Pica-8</td>
<td>P-3290, P-3295, P-3780, P-3920</td>
<td>1.4</td>
</tr>
</tbody>
</table>

*source: [www.tomsitpro.com/articles/pica8-openflow-1.4-sdn-switches,1-1927.html](https://www.tomsitpro.com/articles/pica8-openflow-1.4-sdn-switches,1-1927.html)*

*Full list of switches: [https://www.sdncentral.com/comprehensive-list-hardware-switching-routing/](https://www.sdncentral.com/comprehensive-list-hardware-switching-routing/)*
Commodity Hardware Switches

- Widely adopt single switching chip design
- Greatly simplifies switch design and reduces cost
- Switch vendors depend on merchant silicon switch ASICs
Flow Matching on ASIC-based Switches

- Observation: 10% of flows account for over 80% of traffic [1]
  - elephants: long-lived, high-bandwidth flows
  - mice: short-lived, low-bandwidth flows

- Elephant flows $\rightarrow$ HW (TCAM), mice flows $\rightarrow$ SW (SRAM)

OpenFlow Switches based on Merchant Silicon

- Run software switch on CPU
  - e.g. Open vSwitch (OVS)
  - Linux running on merchant silicon device drivers
  - *dpif* layer communicates to TCAM

- Packet match
  - not in TCAM: forward packet to CPU
  - *ofproto* communicates with controller and writes new flow rule to TCAM
Switches based on Merchant Silicon

- Most commodity switches use ASIC from a single vendor
  - merchants: Broadcom, Marvell, Fulcrum (Intel), Centec
  - advantage: lower production costs
Merchant Silicon

- Every 18-24 month new generation of merchant silicon
  - twice as many ports, 50% lower forwarding latency
  - lowers power consumption, reduces latency jitter, etc.

- Designed as general networking switches with standard throughputs and configurable feature sets
  - 176 Gbps supports 48x 1GbE, 4x 10GbE
  - 1.28 Tbps supports 48x 10GbE, 4x 100GbE

- Built with traditional networking in mind
  - limited flexibility
General Hardware Processing Pipeline of a Switch

**Input processing**
- parser: extract header fields
- (pipelined) flow-action matching

**Buffering / Queuing**
- action: insert packet to queue/s (QoS, unicast, multicast) or drop

**Output processing**
- action: rewrite header fields (pipelined)
Merchant Silicon: Input Processing

- **Counter**
- **L2 CAM**: 128k x 48b store VLAN, dst MAC addr.
- **L3 CAM**: 16k x 32b Host learning match
- **TCAM**: 4k Wildcard match
- **Packet Parser**
- **VLAN Processor**
- **L2 Match / Learning**
- **L3 Match / Learning**
- **ACL Processing**
- **MST Storage**
- **Output Buffering**
Merchant Silicon: Input Processing

- Broadcom: OpenFlow Data Plane Abstraction (OF-DPA)
  - 2014: Broadcom released specification for StrataXGS ASICs
  - OpenFlow data plane abstraction networking software
  - supports OpenFlow 1.3.1 combined with Indigo 2.0 software switch
  - tables do not necessary directly correspond to hardware tables

Source: Broadcom OpenFlow Software OF-DPA: OpenFlow 1.3.1 Switch Pipeline Specification and Software
Merchant Silicon: Processing Pipeline

Input processing
- parser: extract header fields
- (pipelined) flow-action matching

Buffering / Queuing
- action: insert packet to queue/s (QoS, unicast, multicast) or drop

Output processing
- action: rewrite header fields (pipelined)
Merchant Silicon: Buffering/Queuing

- Packets are buffered until they are sent to output ports
- Several different queues: multicast queues, per port queues
  - queues can have different quality-of-service features (e.g. bandwidth)

(source: Cisco Understanding Queuing With Hierarchical Queueing Framework (HQF), June 2012)
Merchant Silicon: Processing Pipeline

**Input processing**
- **parser**: extract header fields
- *(pipelined) flow-action matching*

**Buffering / Queuing**
- **action**: insert packet to queue/s (QoS, unicast, multicast) or drop

**Output processing**
- **action**: rewrite header fields *(pipelined)*
Merchant Silicon: Output Processing

- Field processor makes modification to the headers
  - as defined by the action set, which is build at input processing

- Less complex than input processing
  - perform the actions which are selected during input processing

- Various ASICs support various output actions
  - cheapest ASICs: output packets on any port, no support for rewrites
  - few ASICs: interleave output and rewrite actions
Shortcomings of Merchant Silicon

- Slow production cycles
  - usually 18-24 months or more
  - vendors need to wait until the new merchant silicon is released

- Focus on lower-layer networking services (L2-L3)
  - meet expectations of large number of different customers
  - focus on: throughput, port number, latency, power consumption
  - but not on higher-layer services (L4-L7)

- Furthermore...
  - small sizes of useful tables that can implement SDN data planes
  - usually slow bus speeds between ASIC and CPU/NPU
  - often: small/slow on-chip CPUs
  - lack of flexible actions support
Addressing Shortcomings of Merchant Silicon

- Vendors try to compensate shortcomings

- More-advanced commodity switches
  - high performance multi-core CPU or network processor array
  - high-bandwidth connection between CPU and ASIC (PCIe, custom)

- Interesting solution: hybrid hardware switching architecture
  - hybrid: merchant ASICs and custom vendor ASICs
  - custom ASIC: focus on higher value network services
  - merchant ASIC: focus on forwarding and power consumption

- Example hybrid hardware switch: Cisco Nexus 9000
  - 2 custom ASICs (28nm): VXLAN routing
  - 2 Broadcom Trident II ASICs (40nm): L2/L3 forwarding
C) Commodity Hardware Switches: Network Processors
Network Processors (NPUs)

- Network processors
  - alternative to merchant silicon (on the fast path)
  - integrated circuit, feature set specifically targeted at networking domain
  - software programmable but with high performance

- Improved time to market
  - software-only changes should require less time to develop, test, and deploy than hardware or mixed hardware/software changes.

- Reduced development cost
  - software-only changes should take less effort and expense to develop, test, and deploy than hardware or mixed hardware/software changes.

- Increased time in market
  - ability to support new features, services and protocols with software-only upgrades increases the useful life of a system and the amount of revenue the system can generate over its useful lifetime.
EZ Chip: NP-4 100-Gbit Network Processor

- supports three types of lookup tables: direct access tables, hash tables and tries (stored in DRAM)
- longest prefix match and wildcards are usually supported in tries
- optional: external TCAM

source: EZ Chip NPU-4 Product Brief
EZ Chip: NP-4 Main Functional Blocks

- Task optimized processors (TOPs)
  - many high-performance processors, each optimized for a specific task
  - perform: packet classification, forwarding and modification

- Control CPU
  - extends flexibility for monitoring, management offload, statistics

- Traffic manager
  - for ingress/egress paths, frame queuing, supports QoS mechanisms

- QoS CPU
  - monitor and control traffic managers

source: EZ Chip NPU-4 Product Brief
Summary: Architecture and Design of Switches

- **Software test switches**
  - most flexible, easy to program, ‘unlimited’ table sizes, low performance

- **Merchant silicon + CPU**
  - wide-spread, cheap, fast, inflexible
  - limited programmability, hardware is fixed
  - limited flow table size on fast path (TCAM)
  - long production cycles for silicon

- **NPUs + CPU**
  - fast, flexible, more expensive
  - software programmable (C/C++)
  - large flow tables possible (tries)
  - can support new protocols on fast path
  - but: processors highly optimized for current network protocols
  - NPU merchant-specific software development kits, APIs, toolflows
Part III: Configuration/Management of OF Switches
How to Configure/Manage OpenFlow Switches?

- **OpenFlow protocol**
  - used for communication between switch(es) and controller(s)
  - e.g.: add/modify/remove flow table entries
  - access flow table statistics
  - operates on a timescale of a flow
  - see previous lecture about OpenFlow, controllers, etc.
How to Configure/Manage OpenFlow Switches?

- OpenFlow protocol
  - used for communication between switch(es) and controller(s)
  - e.g.: add/modify/remove flow table entries
  - access flow table
  - operates on a timescale of a flow
  - see previous lecture about OpenFlow, controllers, etc.

But: Separate protocol for configure/manage the operational context of a switch
How to Configure/Manage OpenFlow Switches?

- OpenFlow management and configuration protocol
  - enables the remote configuration and management of OF switches
  - no assumption about configuration point (service in controller, )
  - bootstrapping: switch initiates connection to controller
    - controller’s IP address, port, TLS/TCP, ...
  - detect and update the topology between OF switches
  - allocate resources within switches: ports, queues (enable/disable ports)
  - operates on a slow timescale
Physical vs. Logical OpenFlow Switches

- physical switch = one or more logical switches
- OF-Config allows for configuration of multiple logical switches
- resources: ports/queues/tables are partitioned between logical switches
- logical switch assumes to have full control over its assigned resources
How to Configure/Manage OpenFlow Switches?

- **OF notification framework**
  - event triggered messages: report link failures, etc.
  - publish/subscribe model
    - switch = publisher
    - controller and configuration points can subscribe to selected events
  - examples
    - attribute value change, communication alarm, QoS alarm, processing error alarm, state change, etc.

Diagram:
- OpenFlow Controller
- OpenFlow Configuration Point
- OpenFlow protocol
- Operational Context
- OpenFlow Switch
- OF-Config
OF-Config 1.2: Further Functionalities

- **Monitoring**
  - monitor physical network of physical switches
  - monitor logical network of logical switches

- **Configuration**
  - configuration of queues and ports
  - ability to remotely change some aspects of ports (up/down)
  - configuration of certificates for secure communication between logical switches and controllers
  - configuration of a set of specific tunnel types (VxLAN, etc.)

- **Versioning**
  - negotiation of which OF-Config versions are supported
  - support for OpenFlow versions 1.0 – 1.3.1
Summary: Configuration/Management of Switches

- **Configuration points**
  - initialize switches using OF_Config protocol
  - establish connections between switch and controllers

- **Physical vs. logical switches**
  - one physical switch can instantiate several logical switches
  - physical resources partitioned between logical switches
Part IV: Next Generation of SDN Switches
Next Generation of SDN Switches

- Protocol-independent packet forwarding
  - Huawei’s approach
  - towards OpenFlow 2.0

- Industrial trends
  - open hardware switch developed for Facebook
  - Intel dreams to replace ASICs/NPUs by CPUs
Protocol-Independent Forwarding

- OpenFlow 1.x Limitations
  - OpenFlow protocol is limited to fixed set of protocols
  - version 1.4 already contains 41 different header fields
  - adding user-defined protocols requires significant effort
  - OpenFlow constraints development of new protocols
  - switch cannot express its capabilities to the controller

- Solution: protocol-independent packet forwarding
  - Huawei’s protocol-oblivious forwarding (POF)
  - towards OpenFlow 2.0
Huawei Protocol-Oblivious Forwarding (POF)

- Generic instructions for packet field parsing and handling
  - Table search keys are defined as \{offset, length\} tuples
  - Instructions/Actions access packet data or metadata using \{offset, length\} tuples
  - Include other math, logic, move, branching, and jump instructions

- Proposed in 2013

Current OpenFlow
- \{offset, length\} covers any frame based formats
- \~40 matching header fields defined yet still many uncovered protocols/headers

POF
- POFAT_SET_FIELD
- POFAT_ADD_FIELD
- POFAT_DELETE_FIELD
- POFAT_MOD_FIELD
- Period.

source: Huawei
Huawei Protocol-Oblivious Forwarding

Controller

- Protocol Specific Application
- Protocol Agnostic Tables/Instructions

OpenFlow+

- Flow Tables
- POF Instructions

POF Data Path

Novel Applications & Services

Programming Languages

Application API

Compiler

Flow Instruction Set

Hardware Abstraction Layer

NPU

- Programmable
- Network optimized

Flex Flow Processor

- Runtime & Remote reprogrammable
- Table driven & protocol blind
- Flow instruction set

CPU

- Flexible
- Generic
- Standard
- Low level instruction set

ASIC

- High performance

source: Huawei
Towards OpenFlow 2.0

“We believe that future generations of OpenFlow should allow the controller to tell the switch how to operate, rather than be constrained by a fixed switch design” [1]

- Protocol independence
  - switches should not be tied to any specific network protocols

- Target independence
  - programmers should describe how switches are to process packets in a way that can be compiled down to any target switch that fits our abstract forwarding model

- Reconfigurability in the field
  - programmers should be able to change the way switches process packets once they are deployed in a network

Programming Protocol-independent Processors

SDN Control Plane

Configuration: P4 Program

Compiler

Parser & Table Configuration → Rule Translator

Populating: Installing and querying rules

Target Switch

Classic OpenFlow
new: programmable parser: support for novel protocols
unlike Huawei POF: not focussed on network processors
multiple match-action stages: in parallel, in series (OF 1.0 → in series)
actions are built from a set of primitives supported by the switch.
Intel Ethernet Switch FM 6000 Series (1/2)

- Protocol-independent, hybrid commodity switch by Intel
  - traditional processing pipeline + OpenFlow processing pipeline
- Input processing: FlexPipe processing architecture
  - programmable parser
Output processing: frame forwarding unit
- used for generic pattern matching
Facebook: WEDGE Switch

- **Open Compute Project**
  - Similar to Google Facebook develops own switches for their data centers

- **Zuckerberg: «Saved 1 Billion Dollars»**

- **Linux-based controller «FBOSS»**

- **many partners:** Broadcom, Intel, Big Switch Network, etc

- **modular hardware**

- **Broadcom Trident 2 ASIC**

- **Intel Microserver**
Intel: Replace Merchant Silicon & NPUs with CPUs

- Intel is very interested to move to data centers and enterprise computing sector
  - Intel bought network silicon vendor Fulcrum in 2011

- Intel Data Plane Development Kit for Open vSwitch
  - goal: accelerate packet processing on Intel CPUs (instead of NPUs)
  - Chrystal forest platform: 160 million packets/s on multi-core CPU

- Target group
  - data centers switches
  - top of the rack switches
  - service providers (e.g. Verizon)

- Repeated history?
  - Intel used x86 PC chips to tackle Sun’s/IBM’s servers
  - today: Broadcom, etc.
Summary: Next Generation of SDN Switches

- Software-defined networking trends
  - more flexibility, stronger separation between control and data plane
  - forwarding hardware should no longer hinder protocol development
  - OpenFlow 2.0: protocol-independent packet forwarding

- Large service providers (Google, Facebook)
  - produce their own networking equipment
  - standard solutions from switch vendors no longer fit

- Shift towards NPUs and high-performance CPUs
  - programmability becomes more important than pure forwarding performance

- Will history repeat itself?
  - Can Intel break the dominance of Broadcom?
Τέλος Ενότητας
Χρηματοδότηση

• Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.

• Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης» έχει χρηματοδοτηθεί μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.

• Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

[Image: EU and ESPA logos]
Σημειώματα
Σημείωμα ακείμενης

- Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά, Μη Εμπορική Χρήση, Όχι Παράγωγο Έργο 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαρτούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράφματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».

[1] http://creativecommons.org/licenses/by-nc-nd/4.0/

- Ως Μη Εμπορική ορίζεται η χρήση:
  - που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
  - που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
  - που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

- Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.
Σημείωμα Αναφοράς