Update on the exam and the short paper

Exam:

- We are planing an online **written exam**
- The current date is: 2021-07-16 1:00 pm
- The Platform will be LEXA: `https://lexa.h-brs.de/`

Short paper:

- Clock is ticking.

Paper/Source Code Upload

Please upload your final Version of the paper and the source code here.

Übungseinheiten Info Einstellungen Abgaben und Noten

Zeigen Bearbeiten

Paper Upload (Verpflichtend)

Verbleibende Bearbeitungsdauer: 1 Monat, 12 Stunden, 46 Minuten Abgabetermin: 01. Jul 2021, 23:55

Source Code Upload (Verpflichtend)

Verbleibende Bearbeitungsdauer: 1 Monat, 12 Stunden, 46 Minuten Abgabetermin: 01. Jul 2021, 23:55

Communication in Distributed Systems

9 Software Defined Networking

Dr.-Ing. Michael Rademacher

2020-06-15

Discussion of the Reading

IEEE 802.11s: The WLAN Mesh Standard

Guido R. Hiertz, RWTH Aachen University
Dee Denteneer, Philips Research
Sebastian Max, RWTH Aachen University
Rakesh Taori, Samsung Electronics Co. Ltd.
Javier Cardona, cozybit Inc.
Lars Berlemann, T-Mobile International
Bernhard Walke, RWTH Aachen University

Talking Points:

- What is the goal of this paper?
- What was the motivation for developing the IEEE802.11s Standard.
- When IEEE802.11 is used in a Distribution System (DS) what is difference between source/destination and transmitter/receiver addresses? Why does IEEE802.11s has the need for six addresses?
- Why does the throughput decreases rapidly with the number of hops in the fully connected topology of their experiment?

The Networking “Planes”

The Networking "Planes"

- Abstract, logical concepts to represent an overall network architecture
  - similar to the OSI Model

1. **Data plane:**
   - processing and delivery of packets with local forwarding state
   - *Moves packets from input to output*
2. **Control plane:**
   - computing the forwarding state in routers/switches
   - *determines how packets should be forwarded*
3. **Management plane:**
   - Methods of configuring the control plane
   - *Command Line Interface (CLI), Simple Network Management Protocol (SNMP), etc.*

	Data	Control	Management
Time-Scale	A packet (10 ns)	An event (10 ms to 10 s)	An interaction (min to hours)
How	NIC	Switch/Router Software	Humans/scripts

The Networking "Planes"

Data Plane:

- Specialized Application-Specific Integrated Circuit (ASIC)
- High speed, low latency
- Traffic is forwarded through a device
- Matching on some header bits and perform an action
- Switching, IP-Forward, Queuing, Firewall

Control Plane:

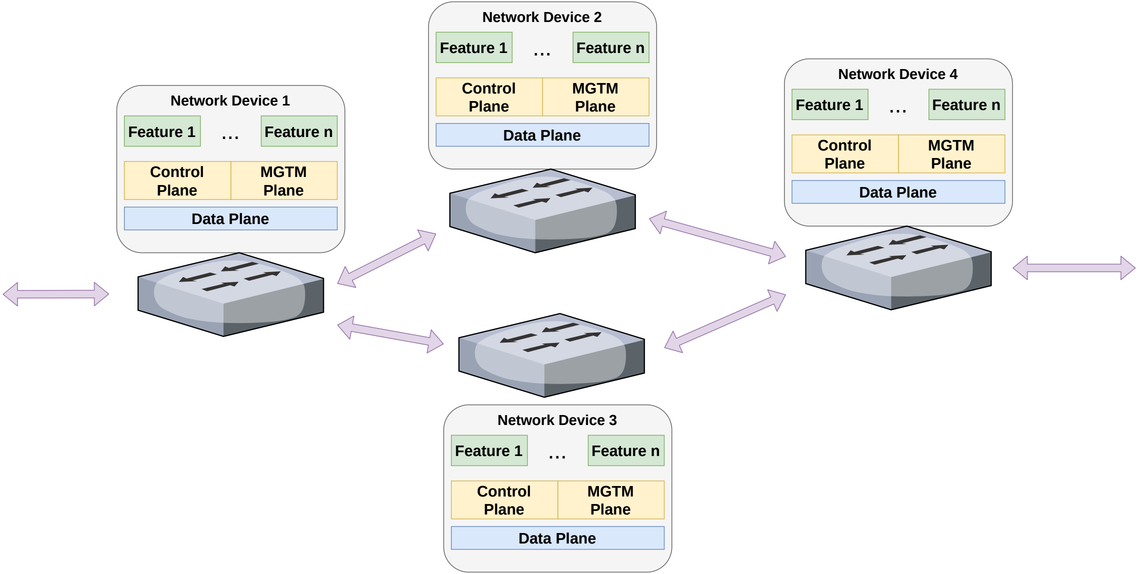
- Software based, uses a CPU
- Traffic is sent to a device, or generated on a device
- discover adjacent devices
- populate forwarding tables
- OSPF, BGP, ...

Management Plane:

- (Have been) human-centric
- Fault, Configuration and monitoring management
- Traffic Engineering: Setting the weights
- Config. of VLANs, MPLS

Traditional Network Architecture

Traditional Network Architecture — hardware-centric model



Challenges for traditional network architectures

- Traditional network architectures imply a good network performance and resilience but are very complex and hard to manage.
- Each network device is configured individually using low-level and often vendor-specific commands.
- Control plane and data plane are bundled inside the networking devices. Reducing flexibility and hindering innovation and evolution of networking infrastructure:
  - The transition from IPV4 to IPV6 started more than a decade ago.
  - A new routing protocol can take up to 10 years to be fully designed, evaluated and deployed.
  - Have you tried changing the internet architecture?

A command sequence to configure an EtherSwitch router for VLAN switching might be similar to the following:

```
enable
vlan database
vlan 2 name sales
vlan 3 name rd
exit
configure terminal
interface vlan 1
ip address 192.168.1.10
no shutdown
ip default-gateway 192.168.1.1
interface fastethernet1/0
switchport mode trunk
interface range fastethernet1/1
switchport access vlan 2
interface range fastethernet1/3
switchport access vlan 3
exit
exit
show vlan-switch brief
show interface trunk
```

Software Defined Networking (SDN)

Software Defined Networking (SDN)

SDN is *not* a revolutionary technology, but just a new way for organising network functionality.

1. It breaks the vertical, hardware-centric model by separating the control logic (control plane) from the underlying devices that forward the traffic (data plane).
2. Network switches become simple (vendor agnostic) forwarding devices and the control logic is implemented in a logically centralized controller (or network operating system) simplifying innovation and evolution.

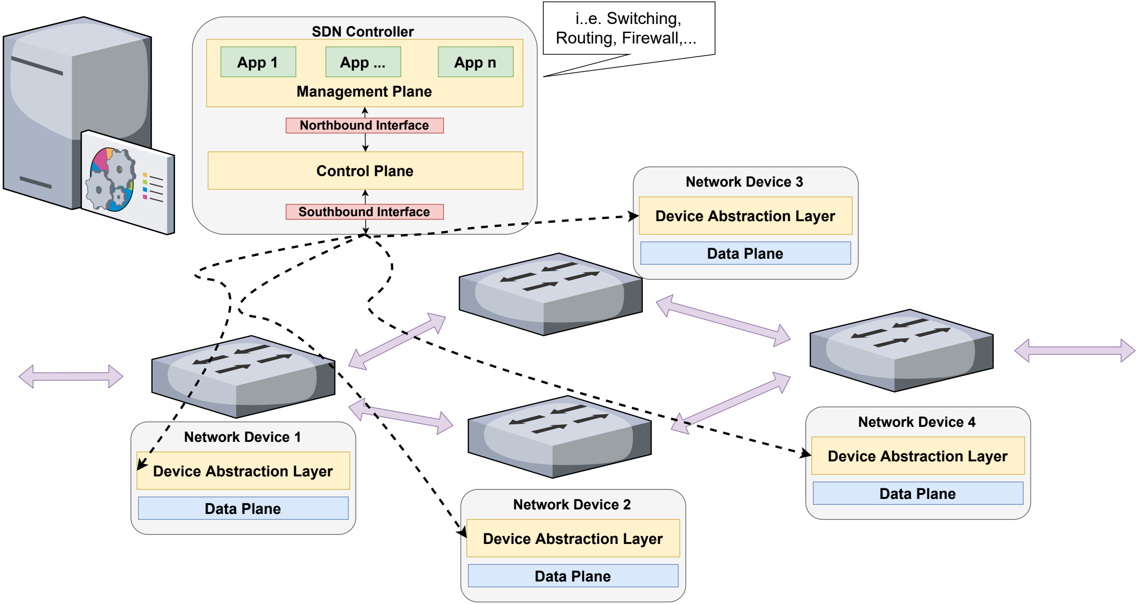
History:

- $\sim$ 2004 Research on new network management paradigms
- 2008 Software Defined Networking
- 2011 Open Networking Foundation (ONF)

Status today:

- **Accepted as the right way for networking**
- Lots of products, in use at Google, Microsoft, Amazon, Internet Service Provider (ISP) (partly)
- The 5G core network builds upon the idea of Software Defined Networking (SDN)
- Current Question: *Which* SDN solution will dominate networking?

SDN - Architecture



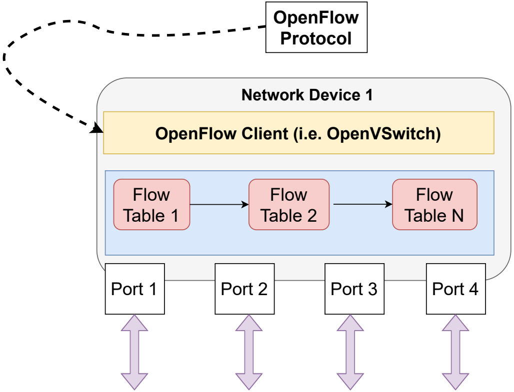
OpenFlow

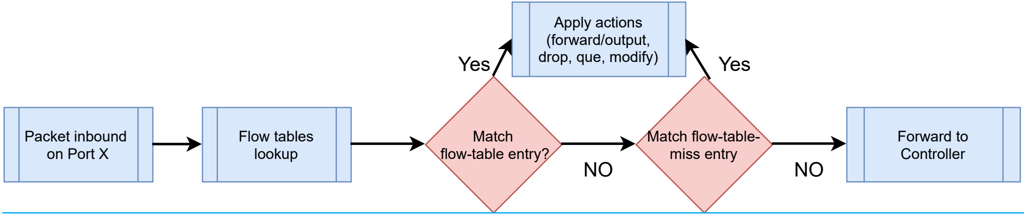
OpenFlow - Motivation and Overview

- **General Myth:** OpenFlow is Software Defined Networking (SDN).
- **Reality:** OpenFlow is an open API that provides a standard interface for programming the data plane switches. OpenFlow mostly defines the **Southbound Interface.**
- A x86 instruction set for network-equipment (i.e. Routers, L2/L3 switches)
- Separation of control plane and data plane
  - The datapath of an OpenFlow switch consist of a **flow-table**, and an action associated with each flow entry
  - The control path consists of a controller which programs the flow entry in the flow-table

OpenFlow Logical Switch

- An OpenFlow Logical Switch consists of **one or more flow tables** and a group table, which perform **packet lookups and forwarding**, and one or more OpenFlow channels to an external controller. **The switch communicates with the controller and the controller manages the switch via the OpenFlow switch protocol.** [4]





Flow-Table Entries

Matching Rule	Action	Stats

Packet + byte Counters

Required:	Optional:
- Output	- Set-Que
- Drop	- Push/Pop Tag
- Group	- Set-Field

Switch Port	VLAN ID	VLAN PCP	VLAN PCP	MAC src	MAC dst	VLAN PCP	Eth type	IP Src	IP Dst	IP ToS	IP Prot	L4 sport	L4 dport

Flow-Table Entries Examples

Switching

Switch Port	VLAN ID	VLAN PCP	VLAN PCP	MAC src	MAC dst	VLAN PCP	Eth type	IP Src	IP Dst	IP ToS	IP Prot	L4 sport	L4 dport	**Action**
*	*	*	*	*	00:1f:..	*	*	*	*	*	*	*	*	port 3

Flow Switching

Switch Port	VLAN ID	VLAN PCP	VLAN PCP	MAC src	MAC dst	VLAN PCP	Eth type	IP Src	IP Dst	IP ToS	IP Prot	L4 sport	L4 dport	**Action**
port 3	vlan1	*	*	00:20	00:1f:..	*	0800	1.2.3.4	5.6.7.8	*	4	17264	80	port 4

Firewall

Switch Port	VLAN ID	VLAN PCP	VLAN PCP	MAC src	MAC dst	VLAN PCP	Eth type	IP Src	IP Dst	IP ToS	IP Prot	L4 sport	L4 dport	**Action**
*	*	*	*	*	*	*	*	*	*	*	*	*	22	drop

Routing

Switch Port	VLAN ID	VLAN PCP	VLAN PCP	MAC src	MAC dst	VLAN PCP	Eth type	IP Src	IP Dst	IP ToS	IP Prot	L4 sport	L4 dport	**Action**
*	*	*	*	*	*	*	*	*	5.6.7.8	*	*	*	*	port 9

Simplified SDN-Load-Balancer Application

- Set static *virt-ip* of the virtual public load-balancer (5.6.7.8)
- Init list of server-ips (10.0.0.0/24)
- For each *packet-in* for *virt-ip* from *client-ip*
  - Pick server-ip in **round-robin fashion**
  - Insert-flow:
    - Match: *virt-ip*
    - Action:
      - Rewrite *virt-ip* to *server-ip*
      - Forward to MAC-TO-PORT(*server-ip*)
  - Proactively insert reverse flow:
    - Match: SRC-IP == *server-ip* AND DST-IP == *client-ip*
    - Action:
      - Rewrite *server-ip* to *virt-ip*
      - Forward to MAC-TO-PORT(*client-ip*)
- All subsequent packets of the request will directly be sent to the chosen server and not be seen by the controller.

- Instead of **round-robin** we could use:
- CPU-Load
- Memory-Load
- ...

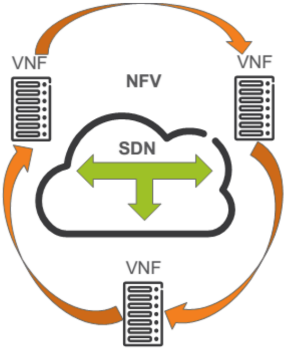
Some SDN challenges and drawbacks

- **Controller placement** and **connection**, **reliability**, and **stability**.
- **Scalability**: One controller for how many nodes?
- **Bootstrapping**: How does the Open Flow controller communicate with OpenFlow switches before OpenFlow has set up the network? (in-band or out-of-band)
- **Fault-tolerance:** What happens if the controller breaks down?
- **Integration with traditional network**: Distributed protocols and SDN mixed?
- **Security**: The SDN controller and the OpenFlow protocol are nice targets.
- **Wireless SDN** [2]
- **Many new ways to mess up a network...**

Software Defined Networking (SDN) vs. Network Function Virtualization (NFV)

SDN vs. NFV

- Software Defined Networking (SDN) advocates replacing standardized networking protocols with centralized software applications that configure all the Network Elements in the network.
  - Flexible forwarding and steering of traffic in a physical or virtual network environment
- Network Function Virtualization (NFV) advocates replacing hardware network elements with software running on COTS computers that may be housed in datacenters.
  - Flexible placement of virtualized network functions across the network.



SDN, NFV and their role in 5G, Katia Obraczka

Reading for next week

Reading for next week

P4: Programming Protocol-Independent Packet Processors

Abstract: P4 is a high-level language for programming protocol-independent packet processors. P4 works in conjunction with SDN control protocols like OpenFlow. In its current form, OpenFlow explicitly specifies protocol headers on which it operates. This set has grown from 12 to 41 fields in a few years, increasing the complexity of the specification while still not providing the flexibility to add new headers. In this paper we propose P4 as a strawman proposal for how OpenFlow should evolve in the future. We have three goals: (1) Reconfigurability in the field: Programmers should be able to change the way switches process packets once they are deployed. (2) Protocol independence: Switches should not be tied to any specific network protocols. (3) Target independence: Programmers should be able to describe packet-processing functionality independently of the specifics of the underlying hardware. As an example, we describe how to use P4 to configure a switch to add a new hierarchical label.[1]

Thank you for your attention.
Are there any questions left?

Room K331
Rathausallee 10
Technopark
Sankt Augustin

michael.rademacher@h-brs.de
www.mc-lab.de
https://michael-rademacher.net

References

[1] Bosshart, P., Daly, D., Gibb, G., Izzard, M., Mckeown, N., Rexford, J., Schlesinger, C., Talayco, D., Vahdat, A., Varghese, G., Walker, D., Mellia, M., Bosshart, P., Daly, D., Gibb, G., Izzard, M., Mckeown, N., and Rexford, J.
Public Review for Programming Protocol-Independent Packet Processors a c m s i g c o m m P4 : Programming Protocol-Independent Packet Processors.
87–95.

[2] Rademacher, M., Jonas, K., Siebertz, F., Rzyska, A., Schlebusch, M., and Kessel, M.
Software-Defined Wireless Mesh Networking: Current Status and Challenges.
Comput. J. (July 2017), 1–16.

[3] Rademacher, M., Kessel, M., and Jonas, K.
Experimental results for the propagation of outdoor ieee802.11 links.
In *Mobilkommunikation - Technologien und Anwendungen. Vorträge der 21. ITG-Fachtagung, 11. - 12. Mai 2016 in Osnabrück. ITG-Fachbericht, Bd. 263* (2016), pp. 32 – 36.

[4] Specification, O. S.
Version 1.0. 0 (wire protocol 0x01).
Open Networking Foundation (2009).