Advanced Internet and IoT Technologies

- Introduction to Software Defined Networking-

Prof. Dr. Thomas Schmidt

http://inet.haw-hamburg.de | t.schmidt@haw-hamburg.de

Agenda

Motivation: The SDN Perspective

SDN Architecture

OpenFlow

SDN Applications

MOTIVATION

Building Blocks of Traditional Computer Networks

Many, cheap Switches that forward packets fast based on fixed simple rules

- Switch control functions are static and not open for programming

Few expensive Routers that operate complex control following routing algorithms

- Routing control functions adapt by dynamic routing protocols

The network Control Plane is fully distributed across the devices

Observation 1:

Switches could be more versatile if forwarding instructions were changeable by programs

Making the forwarding plane programmable opens perspectives for research on and rethinking of operative networking

If network programming became part of the control plane, the networking logic could be accessible by application programs

Observation 1:

Making the forwarding plane programmable turns the networking logic into an application

Switches could be more versatile if forwarding instructions were changeable by programs

Making the forwarding plane programmable opens perspectives for research on and rethinking of operative networking

If network programming became part of the control plane, the networking logic could be accessible by application programs

Observation 2:

Forwarding decisions are identical for packets that belong to the same data flow

Decisions mainly apply to a bulk of packets and can be cached

Controlling a flow allows for richer semantic in networking than only the destination address

Observation 2:

Networks can be operated according to a flow semantic that is unique across device types

Forwarding decisions are identical for packets that belong to the same data flow

Decisions mainly apply to a bulk of packets and can be cached

Controlling a flow allows for richer semantic in networking than only the destination address

Observation 3:

Programmable control logic can be externalized: forwarding devices can cache instructions received from a controller

Splitting devices leads to a strict separation of the control from the forwarding plane

Administrative domains (interior networks) can centralize the control plane in a Network Controller or Network Operating System

Observation 3:

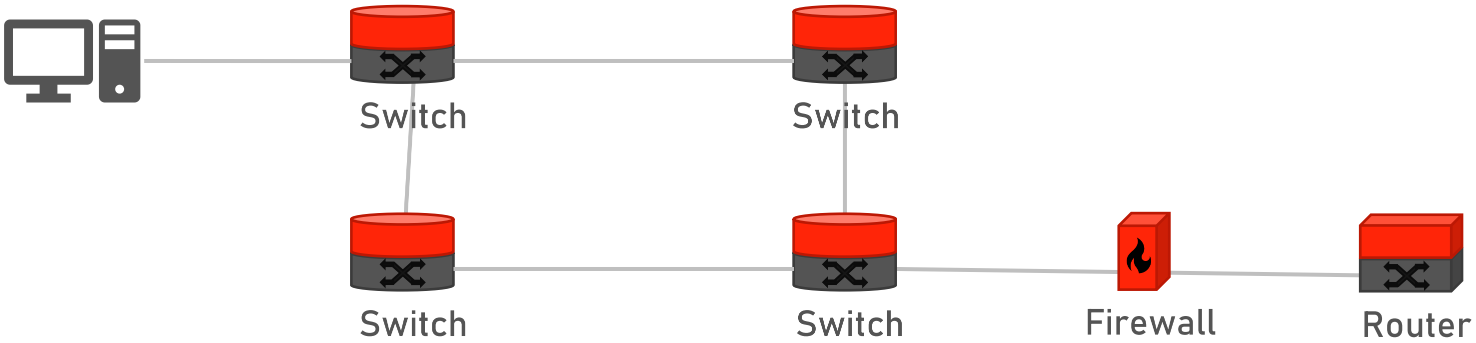
Interior domains can centralize network control in a dedicated Network Operating System (NOS)

Programmable control logic can be externalized: forwarding devices can cache instructions received from a controller

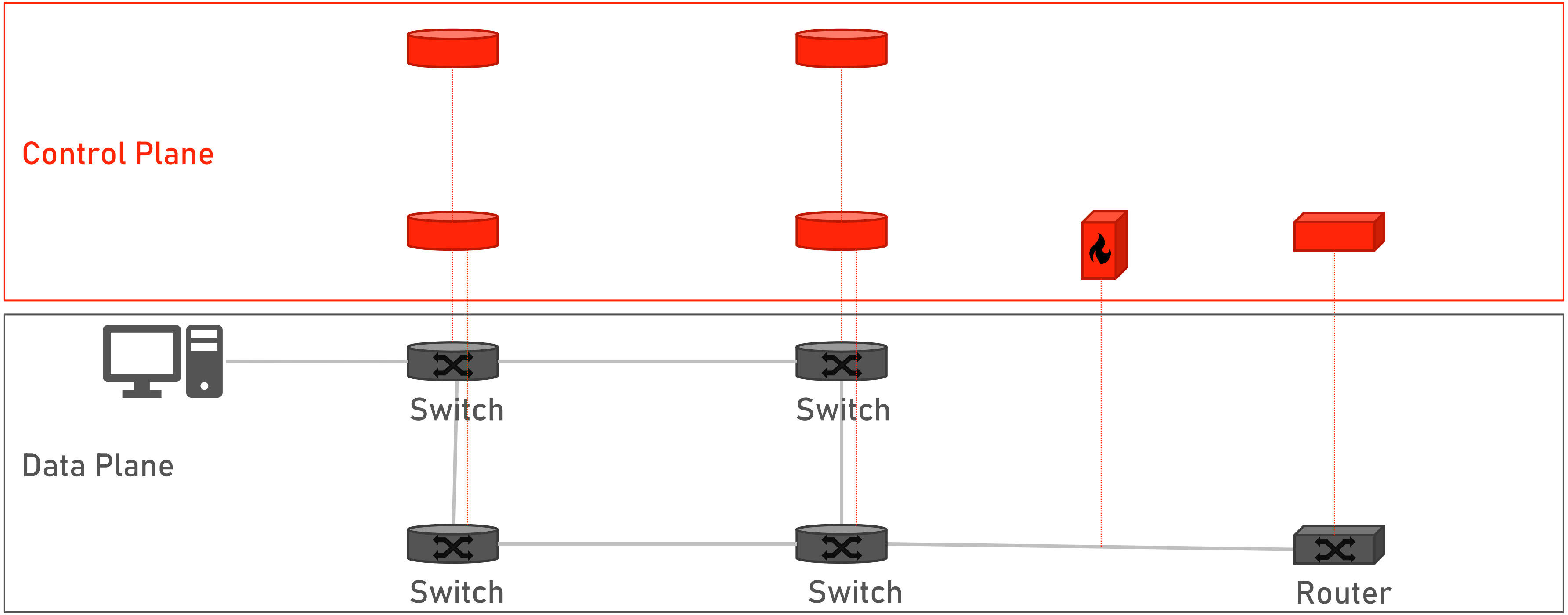
Splitting devices leads to a strict separation of the control from the forwarding plane

Administrative domains (interior networks) can centralize the control plane in a Network Controller or Network Operating System

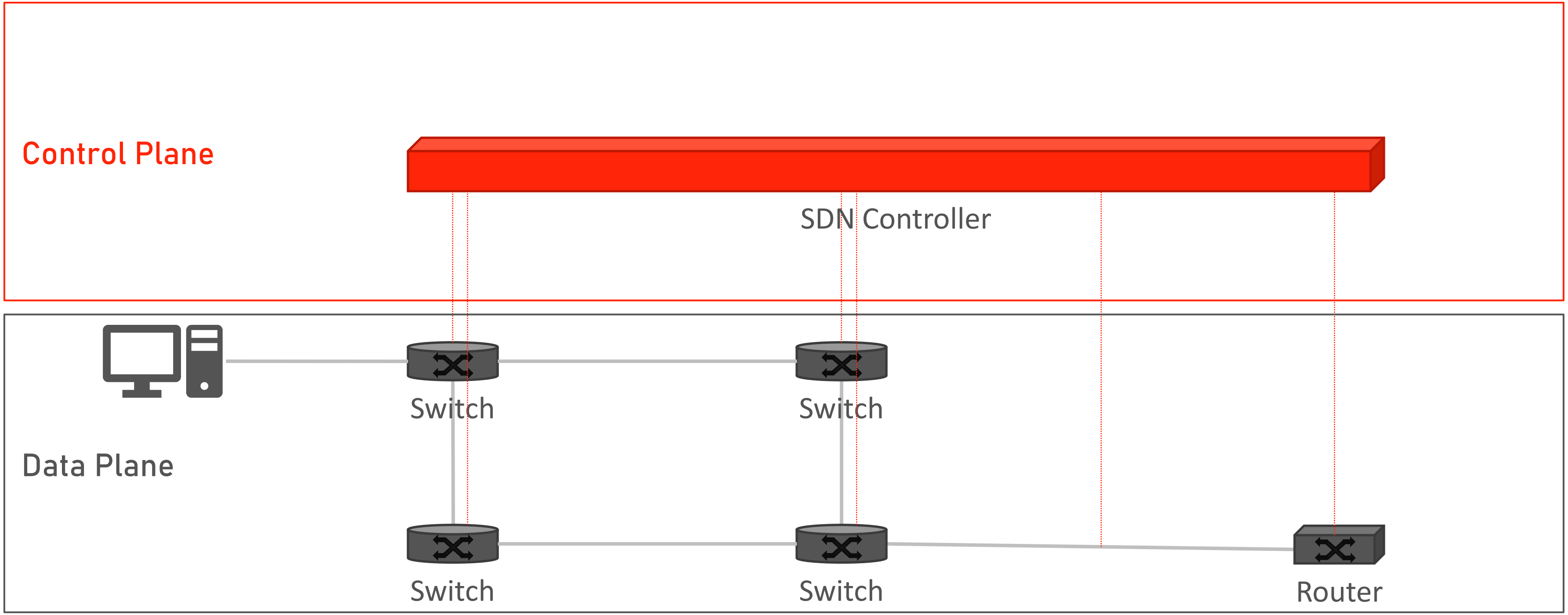
Combining these Steps



Combining these Steps



Combining these Steps



SDN Objectives

Break up the relatively static, rigid architecture of traditional IP networks to allow for:

- A programmable data plane
- Make forwarding devices more versatile and adaptable to specific use cases
- A simplified, comprehensive view of the control plane
- Network abstractions to ease high-level network management

SDN ARCHITECTURE

Software Defined Networking

Problem:

Network components are mainly closed systems

- Vendor hardware only allows “configuration”
- Difficult to define and probe new behavior

Idea of OpenFlow:

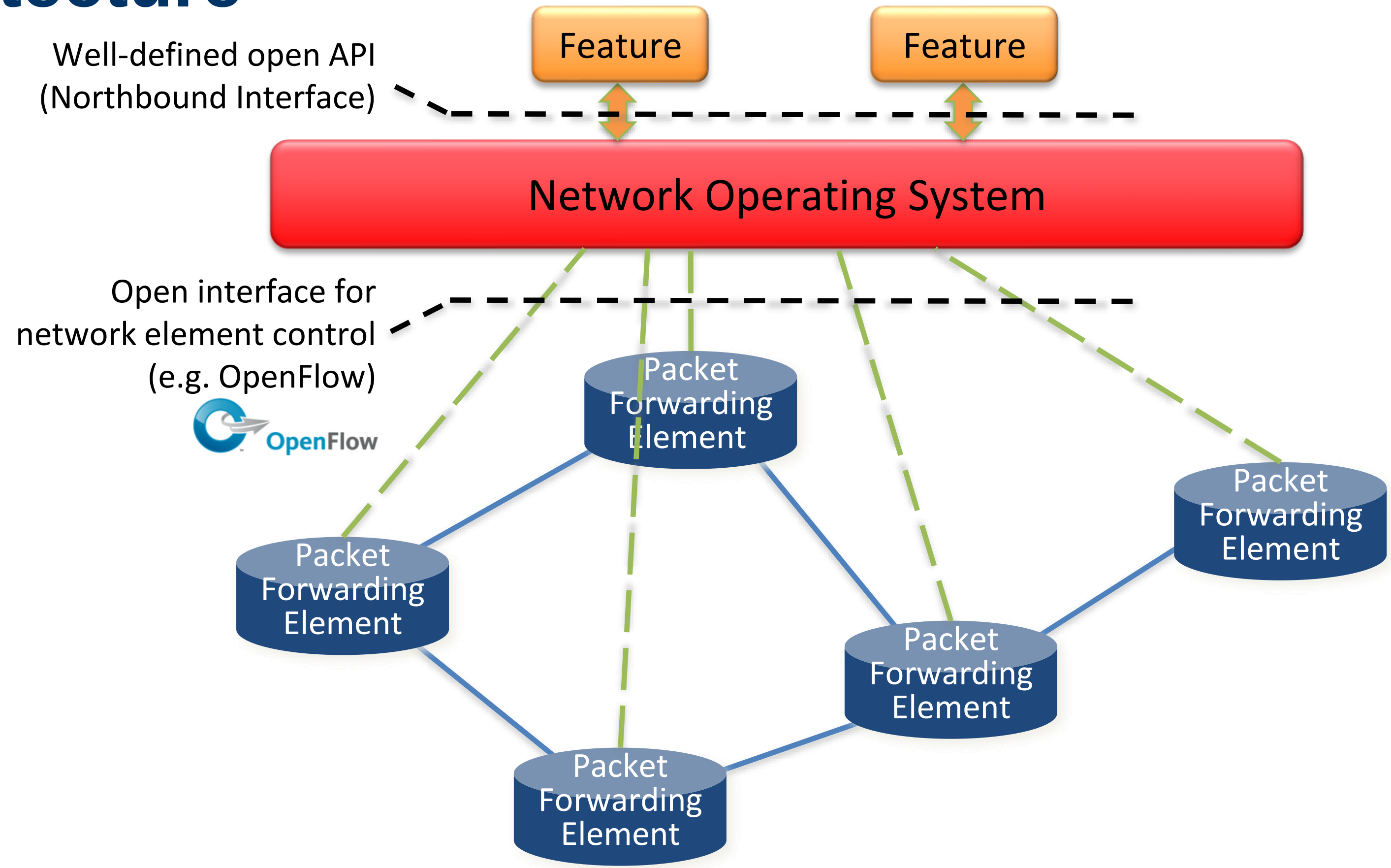
Open network fabrics to a programming interface

- Speed, scale, performance of vendor hardware
- Flexibility and control of software switches

Alternative formulation

- Make switch fabrics (TCAMs) accessible to high-level intelligence
- Separate out a (central) control plane

SDN Architecture



Entities of the SDN Architecture

Forwarding Device: Hardware or software that performs well-defined operations on arriving packets as instructed by flow rules

SDN Controller: Logical function that deduces network control logic from configurations, applications, and observations. It implements flow rules dynamically or statically in forwarding devices

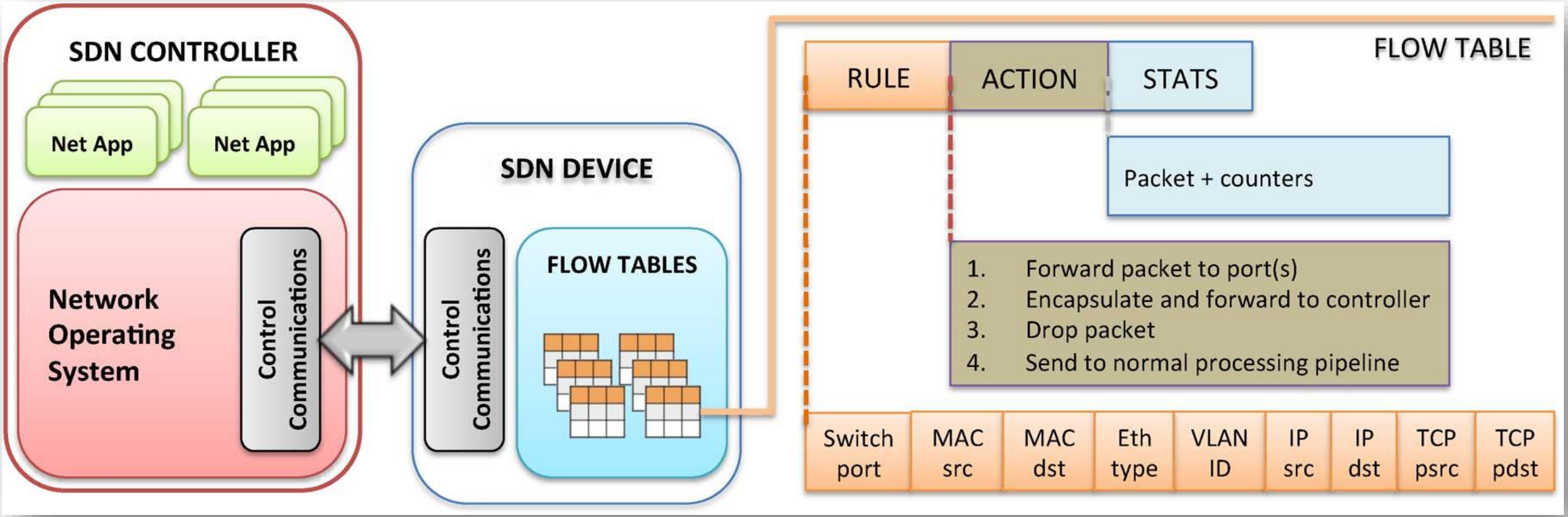
Southbound Interface: The instruction set of the forwarding devices including its corresponding communication protocol (e.g., OpenFlow)

Northbound Interface: API to application developers which abstracts the low-level instruction sets used by southbound interfaces

Network Applications: A set of programs that implement the management logic of the network with its policies

SDN Logical Components

Openflow defines the network interface between devices and controller



Characteristic SDN Operations

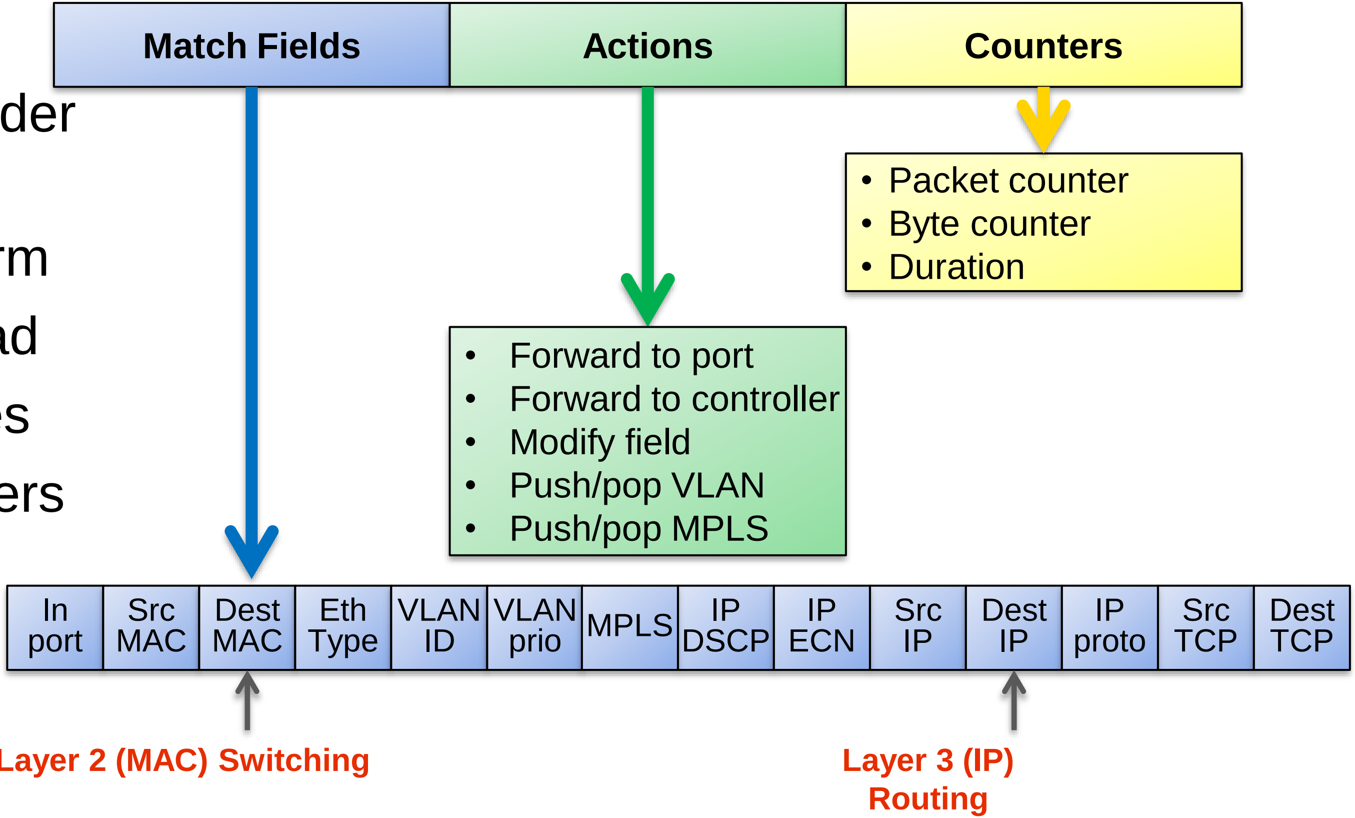
- Packet processing according to policies
  - transferred to header match rules at switches
- Rules distributed by central controller
  - Proactively: at network configuration time
  - Reactively: in response to first unmatched packet
  - Controller provides “global” knowledge
- Intra-domain - not applicable on Internet scale
  - Common deployment: Data-centers, Enterprise Networks

Flow Table Operations

Flow rules can match on any header entry

Actions can perform

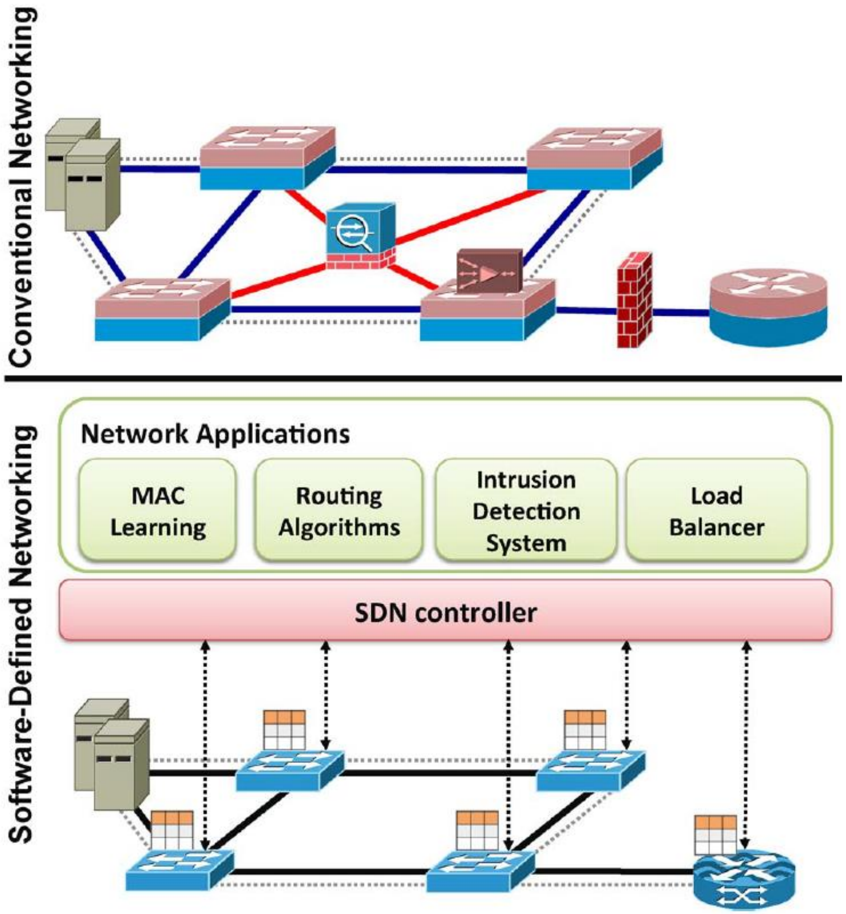
- decision on read
- header changes
- changes of layers
- ...
- introduce high complexity



SDN vs. Conventional Networking

Conventional: Networking functions are implemented and deployed in boxes

SDN: Network functions plus additional applications are virtualized – they signal configuration requests via an API to the SDN controller



OPENFLOW PROTOCOL

About OpenFlow

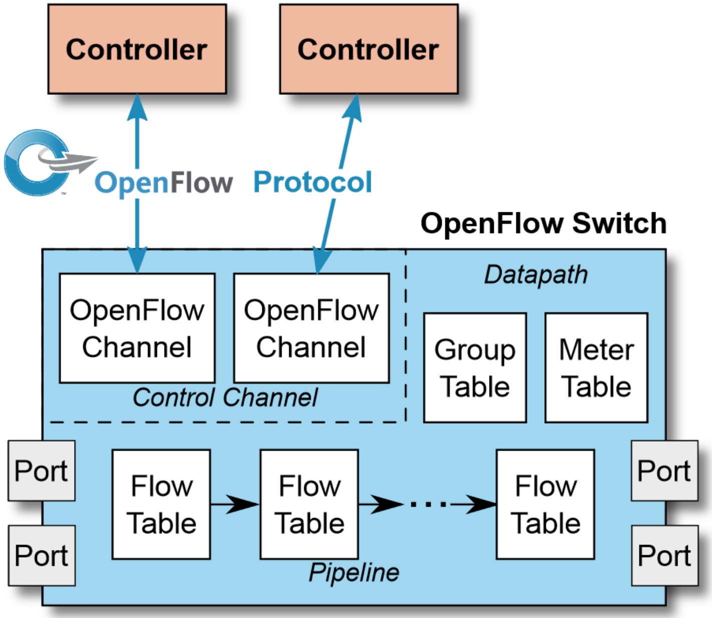
First presented in 2008

OpenFlow® Switch Specification of the Open Networking Foundation (ONF)

- 12/2009 – TS-001: Specification 1.0.0
- Current version: 04/2015 – TS-025: Specification [Ver 1.5.1]

OpenFlow Protocol used as channel between switches and controller

Also specifies a flow table matching pipeline for switches



OpenFlow Flow Entries

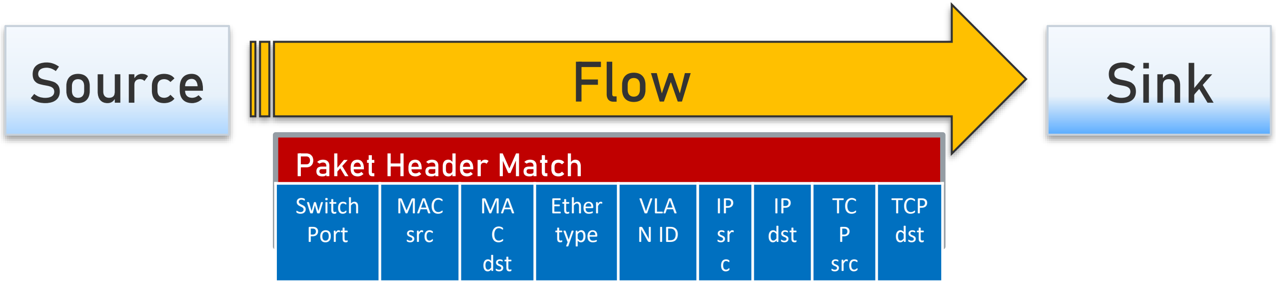
Match fields: to match against arrival port and packet headers

Priority: matching precedence

Instructions: to modify, forward, or dropt the packet

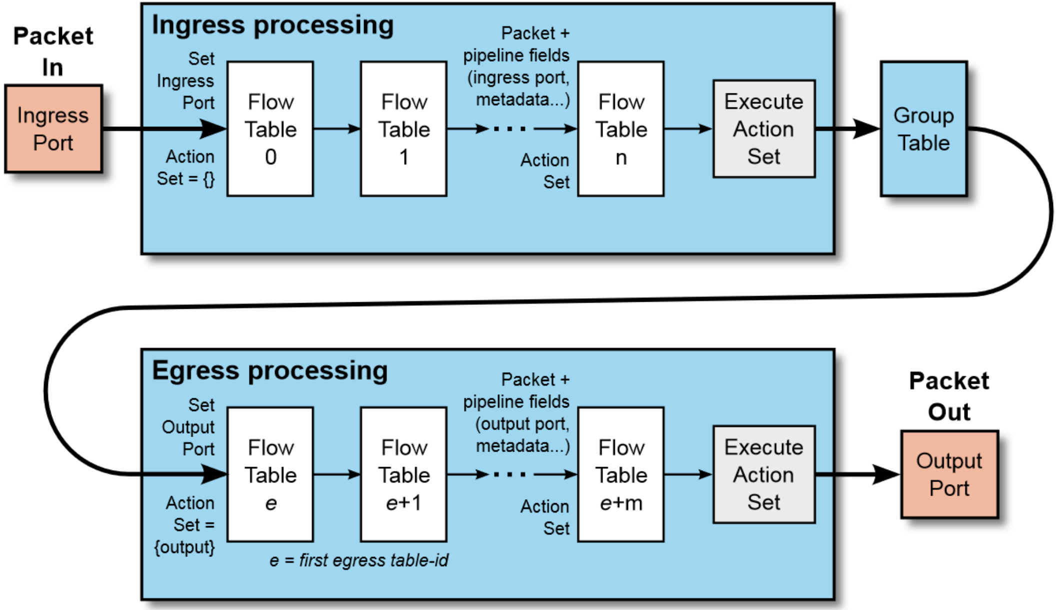
Timeouts: time or idle time before flow entry expires

Counters, cookie, and flags: statistics and controller information



OpenFlow 1.5 Dataplane

1. Packets arrive at a port
2. Collect metadata: ingress port, header fields
3. Select a table for search
4. Select highest priority flow entry that matches
5. Apply action(s): set output or select next table



OpenFlow Protocol Operations

Switch to Controller

Initiate a connection

Inform about state changes

Forward incoming packets (because of table miss or flow entry action)

Controller to Switch

Request supported features

Set or query the configuration

Modify or read the state

Send packets out on a port

- Openflow uses TCP or TLS with default port 6653
- Switch is client, Controller is Server

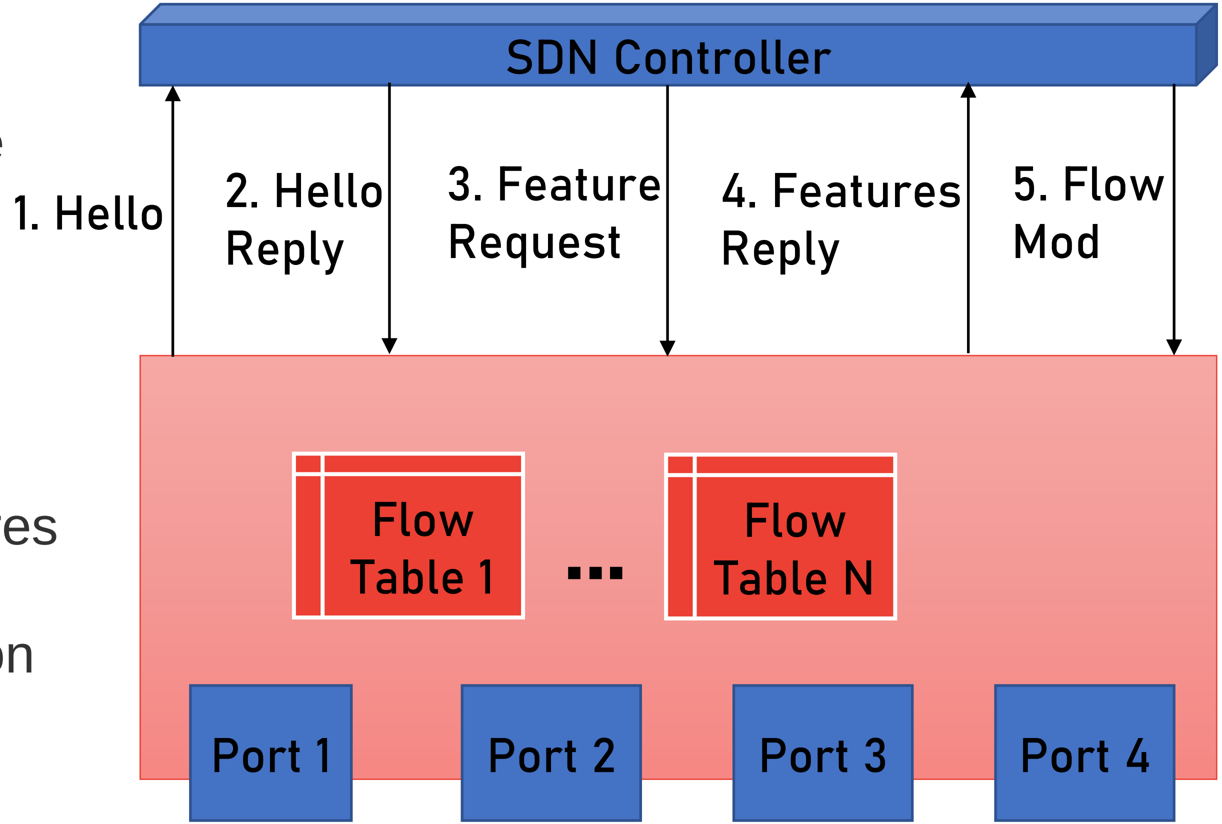
Initial Set-Up

Hello Handshake to agree on OpenFlow version

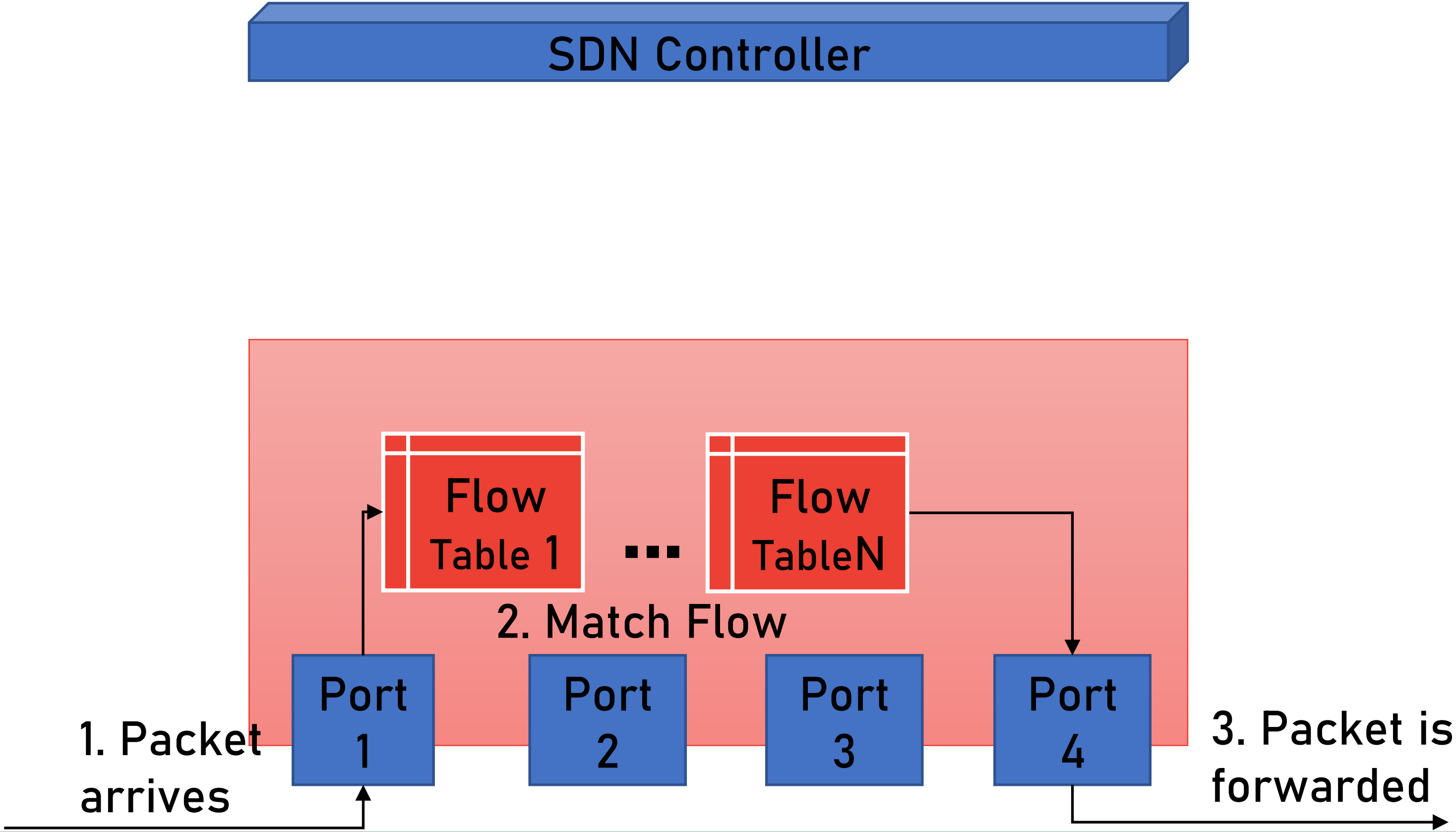
Switch tries periodically to connect

Exchange available features

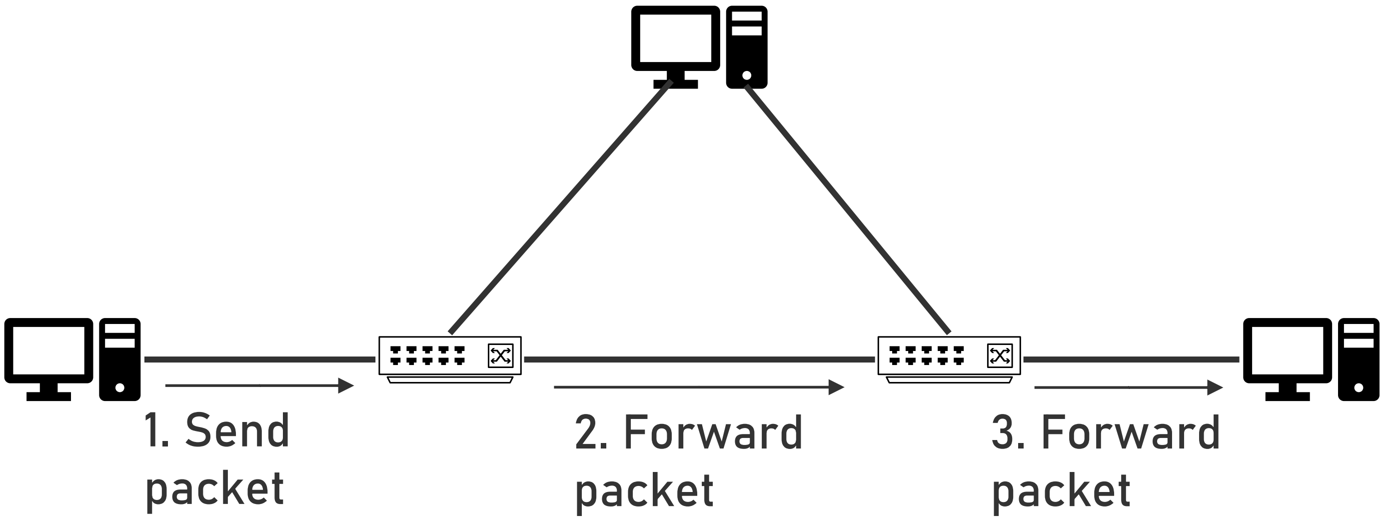
Set initial flow configuration



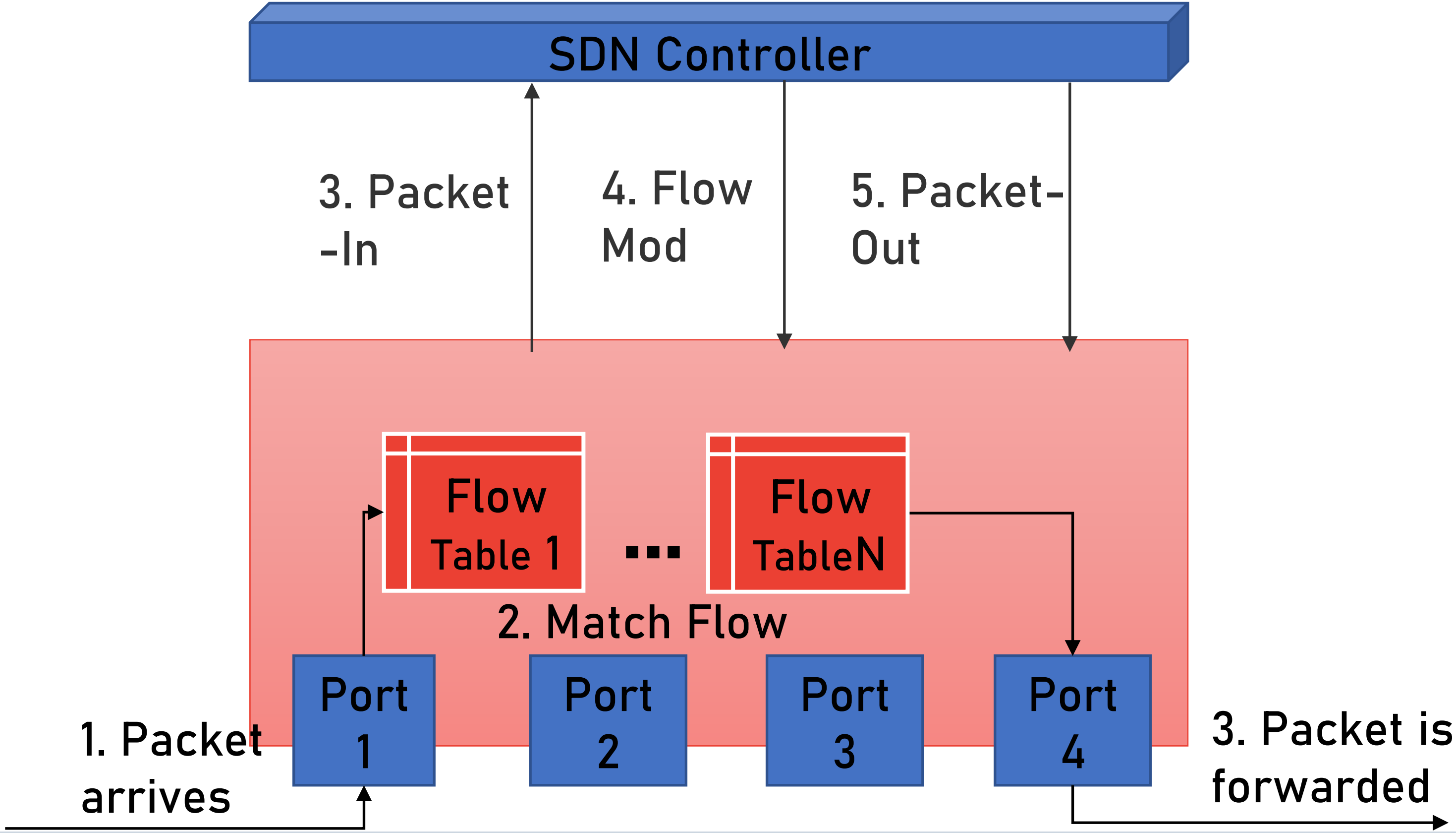
Forwarding Known Flows



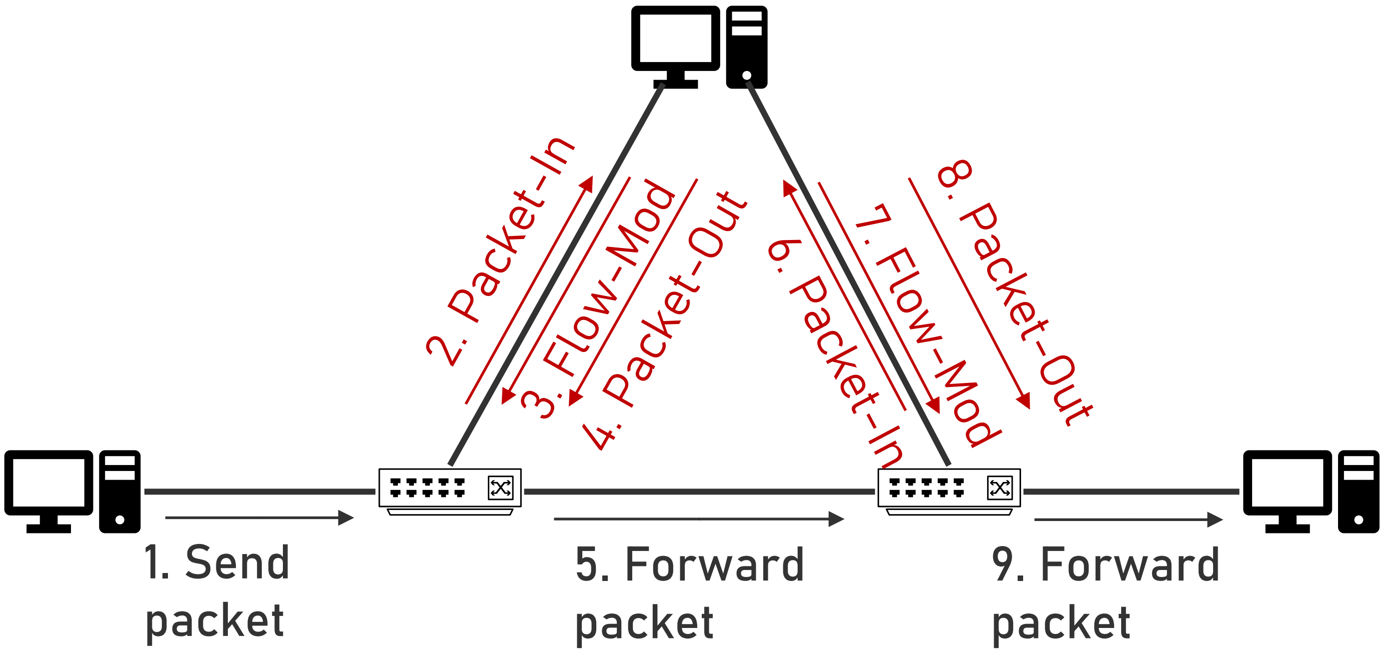
Forwarding Known Flows: Network View



Forwarding Unknown Flows



Forwarding Unknown Flows – Network View



Standardisation and Deployment

SDN development and standardization initiated the Open Networking Foundation

- Standardized OpenFlow and the successor language P4

Various other bodies are involved with SDN

- IETF/IRTF (e.g., NETCONF)
- IEEE
- ITU-T (e.g., broadband transport)
- ETSI (e.g., orchestration of network functions)

SDN technologies see numerous deployments

APPLICATION EXAMPLES

SDN Applications

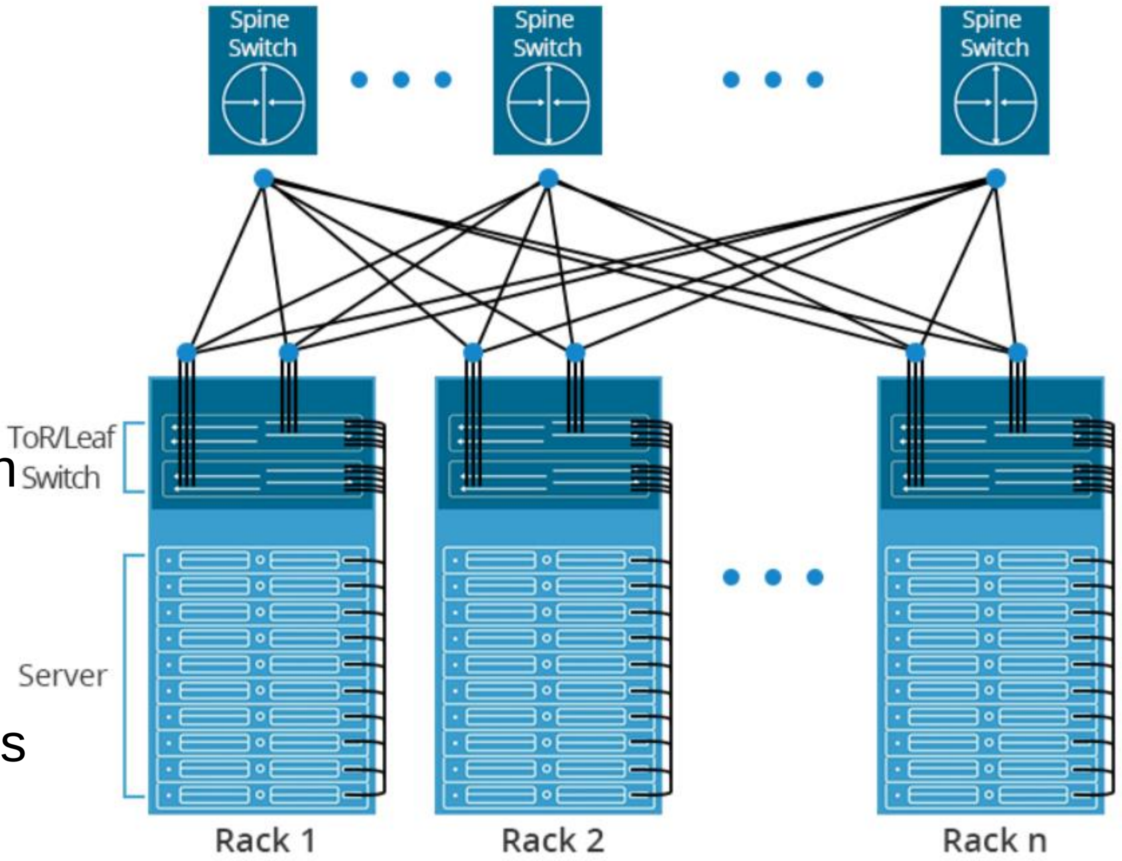
- Traffic engineering
- Mobility and wireless
- Data-center networking
- Security and dependability
- Measurement and monitoring

Data Center Networks

Top of Rack (ToR) switches interconnect servers in a single rack

Leaf switches connect racks with the Spine switch layer

Spines connecting to all Leafs generate equal-length paths between racks



SDN Assistance in Data Centers

Fast flow setup –
without MAC learning

Rapid path repair

Live network migration

Network-aware VM placement

SDN Assistance in Data Centers

Fast flow setup –
without MAC learning

Rapid path repair

Live network migration

Network-aware VM placement

Path diversity management

Transport protocol adaption

Real-time network monitoring

Energy saving

Security in Vehicular Networks

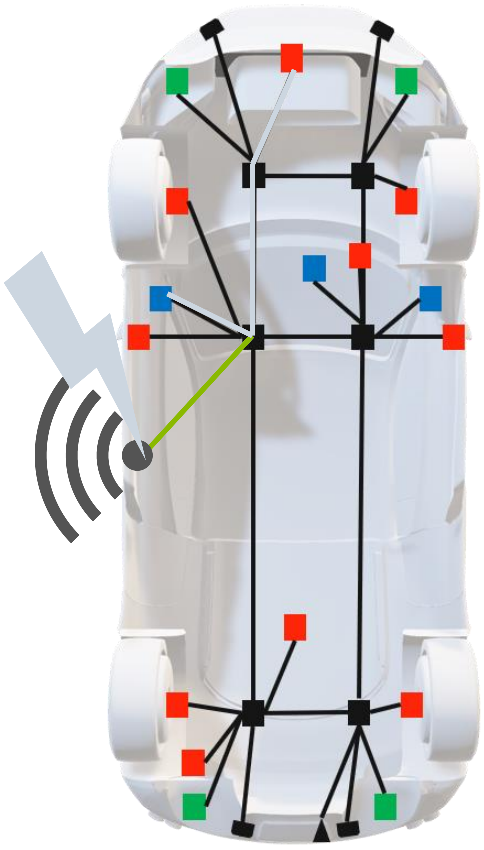
Evolution to Ethernet

Gateways integrate legacy buses

Time-Sensitive Networking (TSN)

Part of global networking (V2X)

Attack surface widens – with fatal possible consequences



Network Security by SDN Flow Separation

Chassis Control: Safety-critical communication in cars is well defined

Currently CAN: Message type specifies operations and legitimate senders – receivers

Tomorrow SOME/IP: Message type marked in application payload

SDN Controller: Can monitor message flows – and prevent illegitimate message distribution

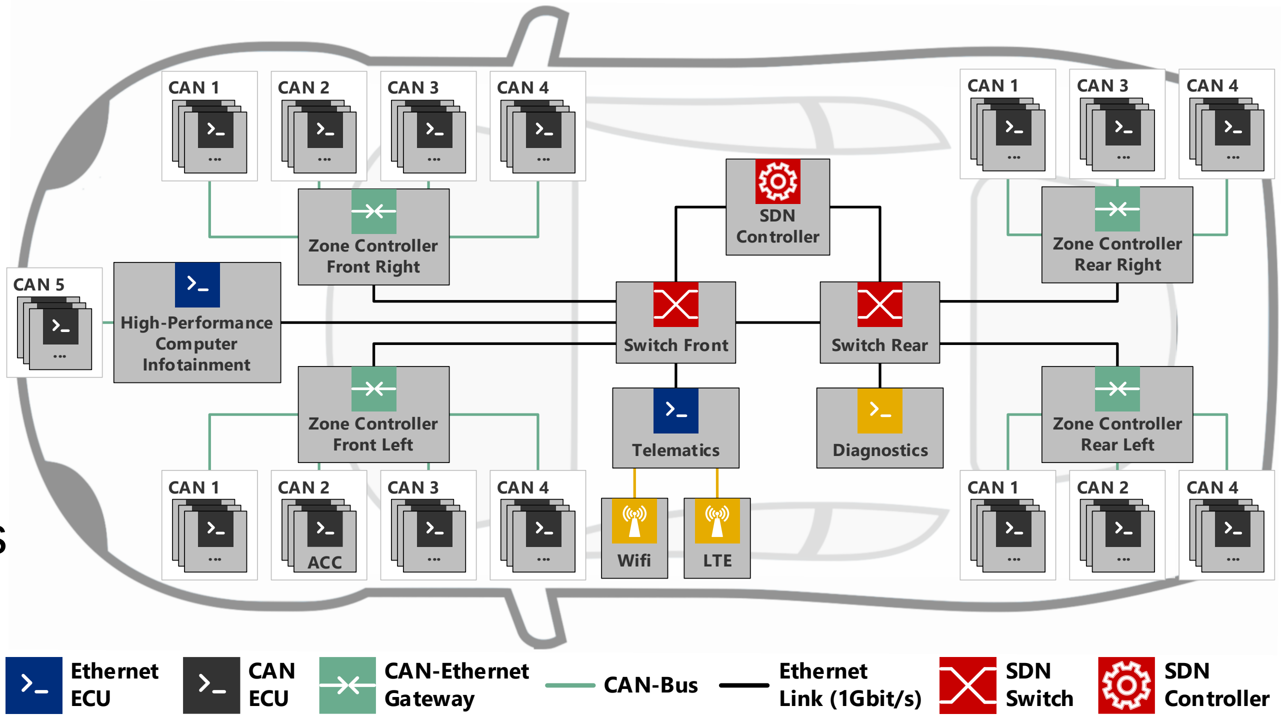
SDN flow admission can act as security guard

In-Car Network

Switched zone topology

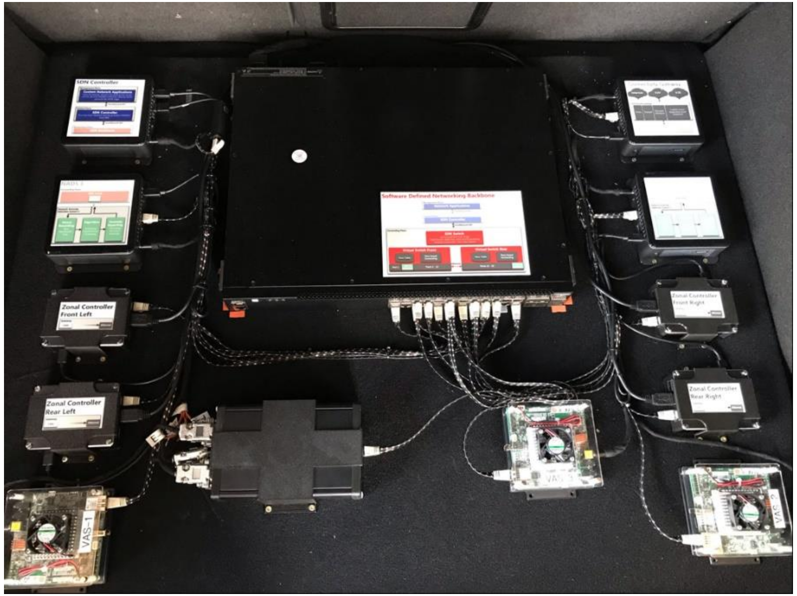
Zone controllers act as gateways for legacy

CAN message type enables flow separation between zone gateways



Our Prototype Car

2016‘ Seat Ateca Prototype



Installation in the trunk

Literature

Timo Häckel, Anja Schmidt, Philipp Meyer, Franz Korf, Thomas C. Schmidt, **Strategies for Integrating Controls Flows in Software-Defined In-Vehicle Networks and Their Impact on Network Security**, **In:** *2020 IEEE Vehicular Networking Conference (VNC) (IEEE VNC 2020),* IEEE, December 2020. https://doi.org/10.1109/VNC51378.2020.9318372

Strategies for Integrating Control Flows in Software-Defined In-Vehicle Networks and Their Impact on Network Security

Timo Häckel, Anja Schmidt, Philipp Meyer, Franz Korf, and Thomas C. Schmidt
Dept. Computer Science, Hamburg University of Applied Sciences, Germany
{timo.haeckel, anja.schmidt, philipp.meyer, franz.korf, t.schmidt}@haw-hamburg.de

***Abstract*—Current In-Vehicle Networks (IVNs) connect Electronic Control Units (ECUs) via domain buses. A gateway forwards messages between these domains. Automotive Ethernet emerges as a flat, high-speed backbone technology for IVNs that carries the various control flows within Ethernet frames. Recently, Software-Defined Networking (SDN) has been identified as a useful building block of the vehicular domain, as it allows the differentiation of packets based on all header fields and thus can isolate unrelated control flows.**

In this work, we systematically explore the different strategies for integrating automotive control flows in switched Ether-networks and analyze their security impact for a software-defined IVN. We discuss how control flow identifiers can be embedded on different layers resulting in a range of solutions from fully exposed embedding to deep encapsulation. We evaluate these strategies in a realistic IVN based on the communication matrix of a production grade vehicle, which we map into a modern Ethernet topology. We find that visibility of automotive

increasingly interconnected ECUs this can open new attack vectors as every gateway participating in the tunneling layer is able to send and receive any message in that tunnel. In contrast, control messages can be directly embedded in Ethernet frames using multicast destination addresses specific to message types and domain specific VLAN IDs. This keeps communication contexts exposed to the switching infrastructure and each vehicular control flow can be precisely identified.

Software-Defined Networking (SDN) was first introduced in campus networks [2] and promises to reduce the complexity of network control while increasing adaptability. In SDN, a central controller manages the control plane of the network. Network devices decide on the forwarding of packets based on a programmable matching pipeline managed by the controller via the OpenFlow protocol. The OpenFlow forwarding

Resumé

Software Defined Networking arose from a research initiative to open up the network forwarding layer by a programming interface

Its central view on the control plane reduces complexity and facilitates management

Intra-domain networks can be optimized by highly adaptive forwarding functions

Caveat: Complex forwarding rules can turn networks into incomprehensible systems

The central controller introduces a new vulnerability to the network infrastructure

Bibliography

1. McKeown et al., “OpenFlow: Enabling Innovation in Campus Networks,” ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, pp. 69–74, 2008
2. Kreutz et al., „Software-Defined Networking: A Comprehensive Survey,” Proceedings of the IEEE, vol. 103, no. 1, pp. 14-76, Jan. 2015
3. ONF, Standard TS-025, “OpenFlow® Switch Specification Ver 1.5.1,” 2015, [Online: https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf]
4. Flowgrammable Website [Online: http://flowgrammable.org]