Software-Defined Network Security

Andreas Schmidt

Telecommunications Lab, Saarland Informatics Campus, 20th Jul. 2017

Who am I?

Andreas Schmidt is ...

... a **PhD student** at the Telecommunications Lab of **Prof. Thorsten Herfet**.

... doing research in the area of **fast and reliable** networking (larn.systems).

LARN

Latency- and Resilience-Aware Networking

... developing solutions using **Software-Defined Networks** (www.on.uni-saarland.de).

... teaching the **Hands-On Networking** practical course (next iteration: Spring 2018).

... reachable via schmidt@nt.uni-saarland.de.

... maybe the only thing in this slide deck that is not *software-defined*.

Software-Defined Introduction

Enter the Time Machine and ...

... go back to the year 2008.

State-of-the-Art in Networking

Enterprises mostly use proprietary hardware and software solutions (Cisco, Juniper, HP, ...).

Network innovation impeded by standardization process (IETF, ITU, et al.). Might take several years for new ideas to turn into actual products.

Another Significant Tech Event

First iPhone has been released (Nov'07) and the "App Market" is opened (summer'08).

Most Significant Events in 2008

- Obama elected as the first Afro-American US president.
- Bill Gates has his last day as CEO of Microsoft.
- Fidel Castro retires as president of Cuba.

Challenges since then...

- Increasing traffic **volume**.
  Streaming Services (NetFlix, Youtube, ...), Big-Data and IoT applications.
- Increasing traffic **diversity**.
  IoT provides various sensor data, more applications require different things.
- Increasing **mobile usage** of the web and online services.
  - Handhelds (Palm, Blackberry, etc.) could access the Internet (nobody used it).
  - iPhone lead to a broad adoption of the mobile web.
  - Apps increased the possibilities of mobiles (99% involve communication).
- Quickly **changing demands**.
  Think of Pokemon Go, which nearly everyone was suddenly playing.

How to serve all these new cat videos to smartphones across the globe?

How to Tackle These Challenges?

Scalability

- Services might start out small, but gain traction very quickly.
  Think of Pokemon Go or the latest cat video distribution platform.
- ISPs, XaaS-providers et al. need to quickly scale up (and down) infrastructure.
  In order to provide appropriate user experience and be economic at the same time.

Flexibility

- Consumption changes quickly: No time to fully design your network upfront.
- Network devices should be flexible to change when demands are changing.

Programmability

- For *flexibility* it would be great to have something like "Network Apps".

Software-Defined Networking (SDN)

- Initial efforts around **Active Networking** (AN) in the 90s (did not take off).
- First paper on **OpenFlow** *[McKeown2008]* published at ACM SIGCOMM 2008.
  - Laid the foundation to get *flexiblity*, *scalability* and *programmability*.
  - Provided a solution that was practical for industry. (AN was too academic)
- Liberalized the networking hardware market.
  Every device with OpenFlow implements same *semantic*, though it might have different *performance*. But: No more special purpose [your vendor here] protocols.
- Shortcuts standardization processes to accelerate innovation.
  "Network apps" can be deployed without changing and implementing standards.
- Saves real money.
  Think of data centers and the provided flexibility.

Separation of Concerns

❓ What is *routing* and what is *forwarding*?

✔ Two distinct functions that are usually executed together and entangled.

👁 Control Plane

Routing

- *Dictates* who can connect to whom.
- *Implements* policies and access control.
- *Store* state and information about the network's composition.

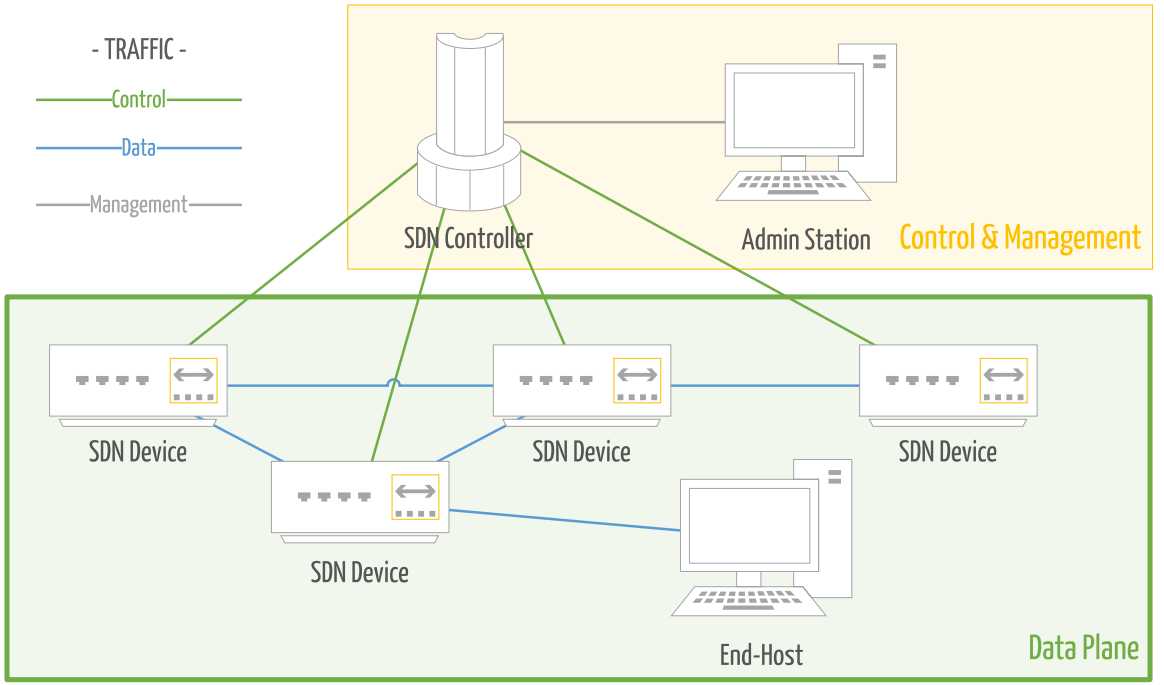
⟷ Data Plane

Forwarding

- *Move* packets from port to port.
- *Duplicate* packets (multicast).
- *Manipulate* packet header information (NAT, QoS tagging).
- *Drop* packets (firewall, IPS, ...).

👍 Separation makes hardware simpler and general-purpose.

SDN - The Big Picture



Similar graphic to *[Kreutz2013]*.

Centralized and Distributed Routing

Centralized Routing

Forwarding tables are propagated by a **central coordinator**.

Benefits

- Single point of **truth**.
- Updates **propagate quickly**.

Drawbacks

- Single point of **failure**.
- **Scales badly** (response times).

Distributed Routing

Forwarding tables are propagated by using specialized **routing protocols**.

Benefits

- More **resilient**.
- **Scales** well.

Drawbacks

- **Convergence** might take time.
- State updates **propagate slowly**.

SDN implements...

Centralized Routing

Motivation

- Single point of truth is important.
  Quickly reconfigure network.
- Networks change quickly.
  Hence long propagation times are detrimental.

Coping with Drawbacks

- Single Point of Failure
  SDN controllers can be built *dependable*: e.g. hot standby system.
- Scaling
  SDN controllers can be built *distributed*: e.g. share state between controllers.

Software-Defined Architecture

SDN | Nodes and Hosts

Node: Network device (virtual or physical) with ports and a flow table.

Subsumes all device types you got to know already (switch, bridge, router, ...)

Physical

- Cisco Catalyst 4500
- HP 2920 Switch Series
- IBM RackSwitch G8264T
- ...

Virtual

- Open vSwitch
- LINC
- ...

Hosts: Ordinary devices that are connected to nodes. They do **not** require support for Software-Defined Networking.

SDN | Matches

Match: Specific values, ranges or wildcards for packet header parameters.

Information from all layers that we have covered so far, and many more.

Link Layer

- MAC address (Src, Dst)
- Ether Type
- VLAN Tags and IDs
- ARP fields (target IP, etc.)
- ...

Transport Layer

- Ports (Src and Dst)
- ...

Network Layer

- IP address
- IP protocol (UDP, TCP, ...)
- ...

SDN | Flows

➔ **Flow:** Sequence of packets with shared traits.

Examples

- One TCP connection.
  - Same 4-tuple (sender IP and port as well as receiver IP and port).
  - Could treat both direction individually (if needed).
- One sender.
  - Evaluate IP and MAC address.
- One stream.
  - RTP packet stream with same stream-ids, same sender.
  - e.g. one VoIP call.

SDN | Actions

Output

- Specify the data to be sent.
  Could also be a packet buffered at the node.
- Specify on which physical port to send.

Drop Packet

- As the name suggests...

Modify Packet Headers

- All header information on all layers can be rewritten.
- Allows to implement features as e.g. NAT.

Ask Controller

- Fallback action if not specified differently.
- Used with monitoring solutions.
  "interesting" packets are forwarded to the controller for further inspection.

SDN | Network Instruction Set

Flow Table: Realization of switching unit's semantics.

Flow Entry: Match + Actions. Entered into the node's flow table.

Match				Action
Source IP	Destination IP	IP Protocol	Destination Port	
192.168.2.10	192.168.2.20	*	*	Output (2)
192.168.2.15	131.245.50.10	TCP (6)	80	Rewrite (Src IP = 134.96.50.1)
6.6.6.6	*	*	*	Drop

Flow Table concept also called *Network Instruction Set*.
Elementary set of operations provide everything to implement any function.

(cmp. general purpose CPU instruction set)

SDN | Network Controller

Controller: Central (potentially replicated) instance that manages the network.

Event: Can be handled by the controller when something happens.

- Packet In
  Upon reception on physical link.
- Flow Removed
  For instance due to timeout.
- Port Status
  Device disconnected, reconnected, ...
- Error
  Special cases...

Message: Instruction sent by the controller to the node to advise it.

- Packet Out
  Send packet on specific port.
- Flow Modification
  Add/update/delete flow table entry.
- Table Modificaton
  Add/update/delete complete flow table.
- Meter Modification
  Add/update/delete sensors.

SDN | NIS Example

Scenario: A switch S receives a packet P, not knowing where to send it.

`Packet_In` (Event)

- Switch identity (S) and physical port the packet arrived at.
- Reason why it was sent to the controller (no match).
- Packet Header Information:
  - Src: IP, Mac, Port
  - Dst: IP, Mac, Port
  - IP Protocol, ToS field, ...
- Packet Payload

`Flow_Mod` (Message)

- Command (e.g. Add)
- Idle/Hard Timeouts (default: 10sec).
- Out-Port: Send the buffered packet to a certain port (determined by routing).
- List of actions (e.g. for all packets from SrcIP(P) to DstIP(P) output packets on certain port).

Programming SDNs

Controller Softwares

Floodlight

- One of the older, still maintained solutions.
- Used by our lab for research and development.
- Language: Java

ONOS

- Focus on Internet-Service Providers (ISP) to manage access networks, autonomous systems etc.
- Language: Java

OpenDaylight

- Focus on data center applications to manage clouds.
- Language: Java

★ Ryu

- Agile, component-based SDN framework.
- **Used in the following.**
- Language: Python

Many more...

Communications with SDN

Goal: Controller is taking responsibility for establishing end-host connectivity.

Approach: Replicate functionality of link layer devices.

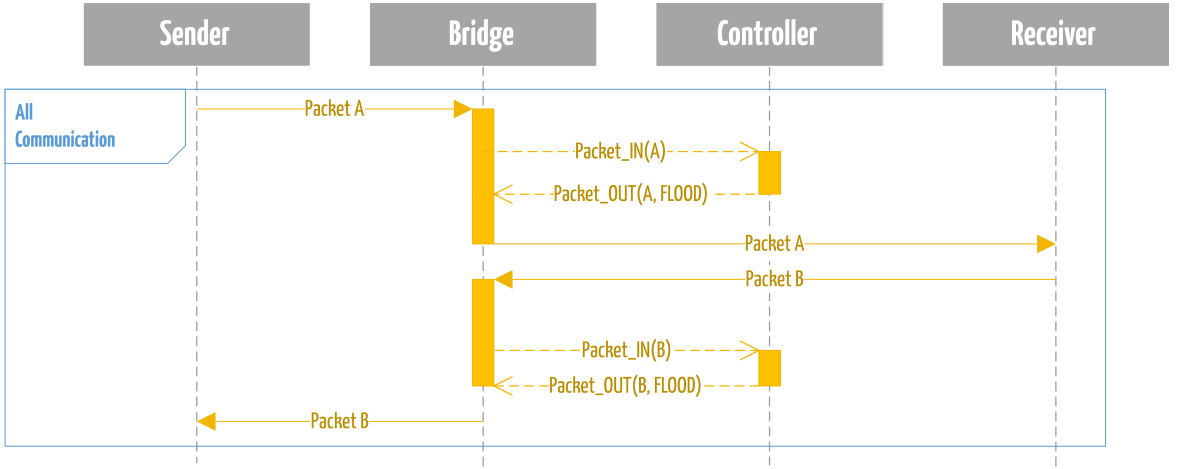
Bridge

- Used to separate collision domains in bus-based networks.
- Floods packets from incoming network to all others.
- Rarely used today, but concept still relevant.
- Half-duplex.

Switch

- Used to create star-based topologies.
- Store-and-forward packets from incoming to outgoing ports.
- Learning switches remember source and destination MAC addresses, so that succeeding packets don't need to get flooded.
- Full-duplex.

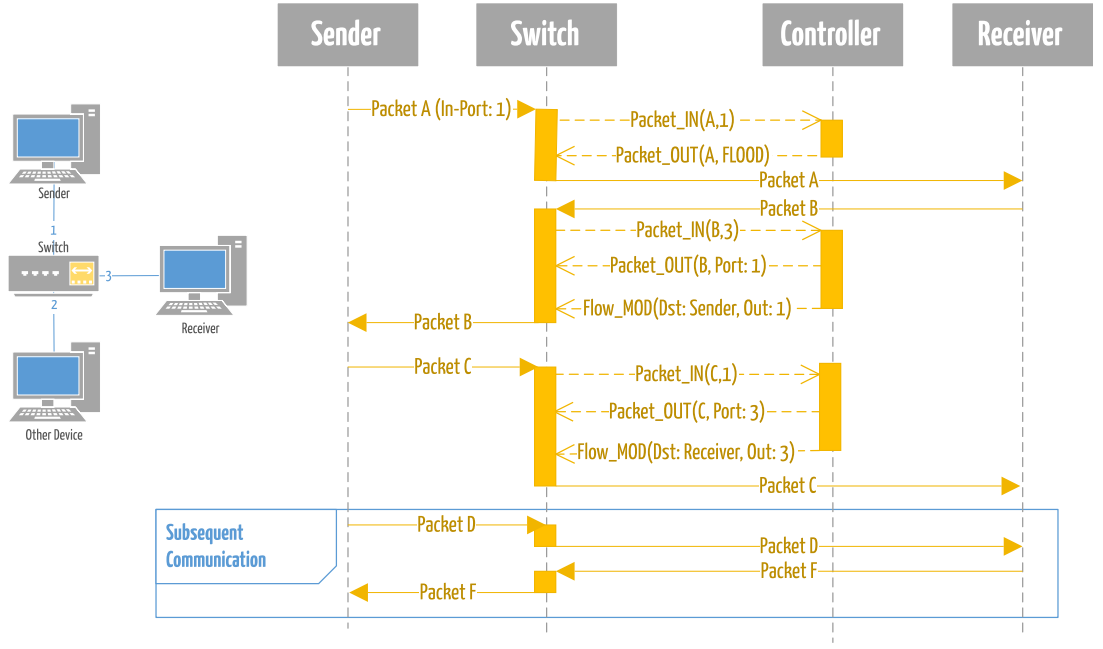
Bridging with SDN



Bridge - Code

```
from ryu.base import app_manager
from ryu.controller import ofp_event
from ryu.controller.handler import MAIN_DISPATCHER
from ryu.controller.handler import set_ev_cls

class Bridge(app_manager.RyuApp):
    def __init__(self, *args, **kwargs):
        super(Bridge, self).__init__(*args, **kwargs)

    @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
    def packet_in_handler(self, ev):
        msg = ev.msg
        dp = msg.datapath
        ofp = dp.ofproto
        ofp_parser = dp.ofproto_parser

        actions = [ofp_parser.OFPActionOutput(ofp.OFPP_FLOOD)]
        out = ofp_parser.OFPPacketOut(
            datapath=dp, buffer_id=msg.buffer_id, in_port=msg.in_port,
            actions=actions)
        dp.send_msg(out)
```

Switching with SDN



Code I

```
@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def _packet_in_handler(self, ev):
    msg = ev.msg;  datapath = msg.datapath; ofproto = datapath.ofproto

    pkt = packet.Packet(msg.data)
    eth = pkt.get_protocol(ethernet.ethernet)

    dst = eth.dst
    src = eth.src

    dpid = datapath.id
    self.mac_to_port.setdefault(dpid, {})

    # learn a mac address to avoid FLOOD next time.
    self.mac_to_port[dpid][src] = msg.in_port

    if dst in self.mac_to_port[dpid]:
        out_port = self.mac_to_port[dpid][dst]
    else:
        out_port = ofproto.OFPP_FLOOD

    actions = [datapath.ofproto_parser.OFPActionOutput(out_port)]

    # install a flow to avoid packet_in next time
    if out_port != ofproto.OFPP_FLOOD:
        self.add_flow(datapath, msg.in_port, dst, actions)

    data = None
    if msg.buffer_id == ofproto.OFP_NO_BUFFER:
        data = msg.data

    out = datapath.ofproto_parser.OFPPacketOut(datapath=datapath, buffer_id=msg.buffer_id,
        in_port=msg.in_port, actions=actions, data=data)

    datapath.send_msg(out)
```

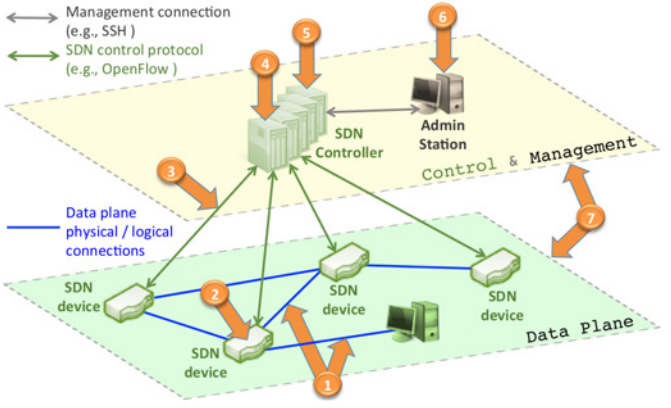
Code II

```
def add_flow(self, datapath, in_port, dst, actions):
    ofproto = datapath.ofproto

    match = datapath.ofproto_parser.OFPMatch(
        in_port=in_port, dl_dst=haddr_to_bin(dst))

    mod = datapath.ofproto_parser.OFPFlowMod(
        datapath=datapath, match=match, cookie=0,
        command=ofproto.OFPFC_ADD, idle_timeout=0, hard_timeout=0,
        priority=ofproto.OFP_DEFAULT_PRIORITY,
        flags=ofproto.OFPFF_SEND_FLOW_REM, actions=actions)

    datapath.send_msg(mod)
```

Software-Defined Security

Attacking SDNs

1. Forged traffic.
2. Switch vulnerabilities.
3. Manipulate control plane communication. 
4. Controller vulnerabilities. 
5. Missing trust between controller and "network applications". 
6. Vulnerabilities in admin stations.
7. Missing trusted resources for forensics and remediation.



Source: [Kreutz2013]

 Specific to SDN

Defending SDNs - State of the Art

Security

- Control traffic is secured (TLS).
- Encryption to ensure confidentiality.
  Meta-data can leak information.
- Signatures to ensure authenticity.
  Controllers only talk to legitimate switches and vice-versa.

Dependability

- Control data is replicated.
- Naive Approach: Hot-standby controller to avoid network downtime.
- Intelligent Approach: Share state and load between controller instances.

More details in *[Kreutz2013]*.

Defending SDNs - Opportunities

Diversity

- Use different vendors, controllers, switches.
- Avoid one bug making the complete network vulnerable.

Dynamic Device Association

- Managed switch should have backup controller.
- Multiple controller make decisions (majority vote).

Trust

- Controller apps should not be malicious or broken.
- Switch and controller should have mutual trust.
- Use e.g. a trusted computation base (TCB).

Fast and Reliable Software Updates

- No software is (and will be) perfect.
- Regular updates are mandatory.

and more ...

Defenses with SDNs

Approach: Recreate security applications as *Virtual Network Functions* (VNF).

Benefits

- Easier deployment (no hardware involved, less software updates).
- Finer tuning of traffic captured (aggregate statistics, no need to dig through packets).
- No need for vendor support.
- Additional perspectives (e.g. place functions close to end-hosts).

Functions

- Firewall
- Monitoring
- Wiretap / Packet Sniffer
- **Intrusion-Detection System (IDS)**
- Intrusion-Prevention System (IPS)
- Honey Pot / Net

Intrusion Detection

Malicious Actions

- Reconnaisance
  Port scans, network enumeration.
- Backdoors
  Port knocking, accessing uncommon ports.
- Denial-of-Service (DoS)
  Malformed packets, high number of requests.

Anomaly Detection

- Port Scans
  High rate of `(SrcHost,DstHost,DstPort)` tuples.
- Backdoors
  Access uncommon `(DstHost,DstPort)`.
- DoS
  High number of incomplete requests to a destination `(DstHost,DstPort)`.

Requirements

- *Efficiency:* Legitimate traffic should not suffer (no significant delays).
- *Effectiveness:* Flows are properly classified (false positive / negative rate low).

SDN Mechanisms to Counter Intrusions

Detection

- Uncommon packets are sent to the controller first.
- Installed flows are automatically monitored:
  - Statistics (bytes send/recv).
  - Number of connections per client.
  - Traffic volume caused by client.
  - Service consumed (HTTP, SMTP) and protocols used (TCP, UDP).

Prevention

- Only use routes that are *proactively* defined and legitimate.
- Establish routes *reactively*, but inspect first packets.
- *Rate-limit* users that open too many connections or send too much data.

Advantages of SDN

- Networks are **more flexible**:
  - Data centers purely work with VMs or containers, which can be easier integrated and migrated when using SDN.
    e.g. spin up more instances of a video-distribution service for streaming.
- Networks are **more resilient and cheaper** to create and operate:
  - In data centers, some hardware is always broken / offline.
  - Use cheaper hardware instead of expensive limited-purpose equipment.
  - SDN is vendor-agnostic, allowing configuration to work everywhere.
- Networks can be made **more secure**:
  - Not out-of-the-box and not conceptually better.
  - Centralization mitigates practical problems with security implementation.
  - Opportunities to improve and optimize existing approaches.

Wrap-Up

Questions?

Take-Home Messages

- SDN as an approach to manage **flexible, scalable and programmable networks**.
- **All network functions** can be implemented using SDN.
- **Security** is again a major concern, in defense as well as attacking terms.

Further Reading (optional)

- Technical Documentation
  - OpenFlow 1.5 Specification
  - Ryu Documentation
- Publications
  - **[McKeown2008]** N. McKeown, et al. - "OpenFlow: enabling innovation in campus networks", SIGCOMM Review'08
  - **[Kreutz2013]** D. Kreutz, F. Ramos, P. Verissimo - "Towards Secure and Dependable Software-Defined Networks", HotSDN'13
  - **[He2016]** L. He, C. Xu and Y. Luo - "vTC: Machine Learning Based Traffic Classification as a Virtual Network Function", SDN-NFVSec'16