Software Defined Networking
Data centre perspective: *Scalability & Resilience*

Seminar: Prof. Timothy Roscoe & Dr. Desislava Dimitrova (Desi)
Networks

- Why do we care about networks?
  - The neural system of the data centre

- Why do we care to advance them?
  - Traffic volume and traffic diversity increase
  - Traffic QoS demands change
  - Server virtualization happened

- How to advance them?
The state of the Network

- **Networks are hard to manage**
  - Heterogeneous hardware firmware and functionality

- **Networks are hard to evolve**
  - Proprietary network devices
  - Slow development and long time-to-market
  - Easy to break

- **There are no formal principles**

Scott Shenker - Stanford Seminar - Software-Defined Networking at the Crossroads
Networks
Networks

- Set of devices to forward traffic between hosts
- Implement a network protocol stack for communication
Traditional networking

- Data plane – packet fwd with local state
- Control plane – compute fwd state
- Distributed control plane
Traditional networking

- **Advantages**
  - Fast packet forwarding rate
  - Established, tested protocols

- **Disadvantages**
  - Convergence time
Traditional networking

- Advantages
  - Fast packet forwarding rate
  - Established, tested protocols

- Disadvantages
  - Convergence time
  - Slow behaviour updates
Traditional networking

- **Advantages**
  - Fast packet forwarding rate
  - Established, tested protocols

- **Disadvantages**
  - Convergence time
  - Slow behaviour updates
  - Limited flexibility
We need better modularity and abstractions for the control plane
Software Defined Networking

- Separates data and control plane
- (Logically) Centralized control plane
Software Defined Networking: Abstractions

- Global network view – topology abstraction
  - From data plane discovery and flow statistics
  - Periodically updated
  - Optimal flow management decisions

- Configuring fwd state – forwarding abstraction
  - Standard instructions
  - Based on flow rules
Software Defined Networking: Control plane

- Topology abstraction
  - Information hiding
  - Protection of lower layers
  - Standard interface

- Example: Routing with link reservation
  - Physical view: Links and load
  - Logical view A: available links
  - Logical view B: single switch
Control plane
Software Defined Networking: Control channel

- Implements the OpenFlow protocol [OF14]
  - Messages, incl. flow rules and corresponding actions
  - Switch & controller behaviour
Software Defined Networking: Control plane

- aka Network Operating System (NOS)

- Implements control functionality
  - Topology discovery & abstraction
  - Generates co-existing flow rules

- Communicates with the data plane
  - Via Open Flow protocol
  - ↑ Discovery messages and flow stats
  - ↓ Flow rules
Software Defined Networking: Control plane

- Advantages
  - Fast reaction time (Efficiency)
Software Defined Networking: Control plane

- Implementation: Single- vs multi-threaded

- Deployment
  - Centralized
  - Physically distributed [KoT10, ToA10]
  - Logically distributed [ShR10]
  - Hierarchical controllers [YaS12]

- Flow management mode
  - Proactive mode [ReJ10]
  - Reactive mode [CaM07]
Software Defined Networking: Control plane

- Top performance
  - POX Controller http://www.noxrepo.org/pox/about-pox/
  - Ryu http://osrg.github.io/ryu/
  - Trema http://trema.github.io/trema/
  - FloodLight http://www.projectfloodlight.org/floodlight/
  - OpenDaylight http://www.opendaylight.org/

- Special tasks
  - RouteFlow https://sites.google.com/site/routeflow/
  - Flowvisor https://openflow.stanford.edu/display/DOCS/Flowvisor
Software Defined Networking: Control plane

- Frozen development/Immature:
  - **FlowER** https://github.com/travelping/flower/blob/master/README.md
  - **MUL SDN Controller** http://sourceforge.net/projects/mul/
  - **Jaxon** http://jaxon.onuos.org/
  - **Beacon** https://openflow.stanford.edu/display/Beacon/Home
  - **Network Development and Deployment Initiative Open Exchange Software Suite (NDDI OESS)** http://code.google.com/p/nddi/wiki/README
  - **NodeFlow** https://github.com/dreamerslab/node.flow
  - **Open vSwitch Controller (ovs-controller)** http://openvswitch.org
Software Defined Networking: Control plane

- References
  - [ReJ10] Rexford, M.J. Freedman, and J. Wang. Scalable flow-based networking with DIFANE. In SIGCOMM 2010
Above the control plane
Software Defined Networking: Control applications

- Implement diverse control functionality
  - Routing
  - Bandwidth provisioning
  - Load balancing
  - Packet inspection
  - Monitoring
  - ...

- Communicate with the controller
  - ↑ Topology and flow stats
  - ↓ Flow control decisions

Diagram:
- SDN Controller
  - Open Flow protocol
  - Local CPU/NPU
  - Switching fabric

Control applications
Software Defined Networking: Control applications

- **Advantages**
  - Fast reaction time (Efficiency)
  - Quick behaviour updates (Scale)

- **Example**
  - New Application X
  - Tag X - new set of packet headers

![Diagram of SDN Controller and Flow rule: match on TagX]
Software Defined Networking: Control applications

- Advantages
  - Fast reaction time (Efficiency)
  - Quick behaviour updates (Scale)
  - Flexible flow management (Innovation)

- Example
  - Unequal Load Balancing

![Diagram of SDN Control applications]

SDN Controller

Load Balancer

(h1,h2) -> Black
(h1,h3) -> Green

h1

LB

h2

h3
Modularity and compositions are the key to scalability & performance

- Single module: Multiple functions >> hard to program, debug, tests, enhance
- Single module: Single function      >> easier to program, debug, test; reuse
- Multiple modules: Same flow        >> complementary actions
Software Defined Networking: Control applications

How to combine different modules for a complete network policy?

Network policy: the set of rules which a particular flow should obey while passing through the network
Software Defined Networking: Control applications

### Parallel composition

- **LB** + **Monitor**
  - **SDN Controller**
  - dstip = 1.2.3.4 → count
  - dstip = 1.2.3.4, vlan = A → path 1
  - dstip = 1.2.3.4, vlan = C → path 2

### Sequential composition

- **FW** → **Route**
  - **SDN Controller**
  - P1: dstptcp = xx → drop
  - P2: dstptcp != xx → fwd(1)
Software Defined Networking: Languages

- Network programming languages
  - Abstraction for declarative network policy
  - Boolean predicates for composition
  - Simple operations (match, modify)

- Controller converts policies to OpenFlow rules

- Languages
  - Frenetic >> Pyretic
  - Merlin, Nettle, Maple

Network policy
\[ \text{src} \gg \text{LB} \gg \text{dst} \]
\[ \text{src} \gg \text{FW.BW} \gg \text{dst} \]

OpenFlow rules
\[ \text{dstip} = 5.6.7.8 \rightarrow \text{path 1} \]
\[ \text{dstip} = 5.6.7.8 \rightarrow \text{path 2} \]
Software Defined Networking: Control applications

- References
Software Defined Networking: Control applications

References

Data plane
Software Defined Networking: Data plane

- Streaming algorithms on packets
  - Matching on some headers (40 fields)
  - Perform some actions

- Wide range of functionality
  - Mapping header fields
  - Buffering and marking
  - Forwarding
  - Traffic monitoring
  - Access control
  - …

\[ \text{OF rules} = (\text{match}, \text{action}) \]

- Local CPU/NPU
- Switching fabric

match \quad \rightarrow \quad \text{action}
Software Defined Networking: Data plane

- Packet handling
  - Based on (match, action) pairs
  - Multi-stage matching tables
  - Different functionality -> different tables

- Known flow

OF rule = (match XYZ, fwd)

action = fwd
Software Defined Networking: Data plane

- New flow

![Diagram showing SDN data plane](image-url)
Software Defined Networking: Data plane

- Technologies
  - ASIC – fast but expensive to change
  - FPGA and NPU – fast, reconfigurable, hard to program
  - GPU – massive parallel computation
  - CPU – easy to program but slow performance

- Modular composition
  - Line-rate hardware for active flows
  - Slow CPU processing for dormant flows
  - CPU buffers OF rules from controller

\[ \text{OF rules} = (\text{match, action}) \]
Software Defined Networking: Data plane

- **Performance: Data processing aspect**
  - *known*
  - **Line-rate** – 10Gbps, 40 Gbps, 100Gbps
  - **Small packets** – 40-byte TCP ACK packets
  - **Small time per packet** – 8 ns to process a packet

- **Performance: Control aspect**
  - *experimental*
  - Rule installation, modification, deletion
  - Stats collection
  - # flows (rules)

\[ \text{OF rules} = (\text{match}, \text{action}) \]
Software Defined Networking: Data plane

- Performance: Control aspect
  - Hardware ASIC switches (HP, NEC, Juniper, Pica8)
    - Up to 4k flow entries
    - 10-40 ms for a single flow insertion
  - Hardware NetFPGA switches
    - 32k exact match entries (in SRAM) at line rate
    - 11μs for a single flow insertion
  - Software switches: Open vSwitch (OVS)
    - Theoretically unlimited matching possibilities
    - 65k flow entries for OVS
    - 1ms for a single flow entry
Software Defined Networking: Data plane

References

SDN in practice
Software Defined Networking: Challenges

- Scalability
  - Decision elements responsible for many routers
- Response time
  - Delays between decision elements and routers
- Reliability
  - Surviving failures of decision elements and routers
- Consistency
  - Ensuring multiple decision elements behave consistently
- Security
  - Network vulnerable to attacks on decision elements
- Interoperability
  - Legacy routers and neighboring domains
Challenges: Scalability

The ability to remain performant as the #flows grows.

- SDN Controller
  - Processing delay
  - Bandwidth
  - Updates rate
  - Switch CPU

- Local CPU/NPU
  - Flow table size
  - Rule processing time
  - Bandwidth
  - Delay
  - Lookup speed
  - Buffer size

- Switching fabric
  - Bandwidth

The ability to remain performant as the #flows grows.
Challenges: Scalability
Challenges: Resilience

The ability to recover control logic after a failure.

- Consistent state & synchronization
- Flow rule validation
- Quick recovery from link failures
- In-band vs Out-of-band control channel
- Flow rule verification
- Behaviour verification
- Local CPU/NPU
- Switching fabric
- SDN Controller
Software Defined Networking: Challenges

- **SDN**: Ethane and OpenFlow [1-2]
- **Scalability** [3]
  - What to handle: Hedera vs Mahout [4]
  - Where to handle: DevoFlow vs Difane [5]
  - How to handle: Fibbing [6]
  - Switch: P4 vs POF [11]
  - Controller: Placement, FlowVisor [12]
- **Resilience** [3]
  - SDN and programmability [7]
  - Correctness: SDNRacer, NICE [8]
  - Fault tolerance: Ravana [9]
  - Version control: GitFlow [10]
Contact information and credits

ETH Zurich
Systems Group
Universitätstrasse 6
8092 Zürich

www.ethz.ch/en.html

© ETH Zurich, February 2016
Software Defined Networking: Research questions

- Which SDN components need standards?
- Data plane design
  - How to improve performance?
  - What functionality is appropriate?
  - Which technologies, combined how?
- Protocol design – functionality vs complexity
- Controller design
  - Centralized vs Distributed
  - What abstractions to use?
- What are interesting SDN applications?
- What is a good “division of labor”??