SDN/NFV: Software Defined Networking & Network Function Virtualization

Christian Esteve Rothenberg (University of Campinas)
Rodrigo Fonseca (Brown University)

Monday, August 22, 2016
SDN & NFV :: Network Programmability / Flexibility

A means to make the network more flexible and simple by minimising dependence on HW
Why NFV/SDN?

1. **Virtualization**: Use network resource without worrying about where it is physically located, how much it is, how it is organized, etc.
2. **Orchestration**: Manage thousands of devices
3. **Programmability**: Should be able to change behavior on the fly.
4. **Dynamic Scaling**: Should be able to change size, quantity, as a F(load)
5. **Automation**: Let machines / software do humans’ work
6. **Visibility**: Monitor resources, connectivity
7. **Performance**: Optimize network device utilization
8. **Multi-tenancy**: Slice the network for different customers (as-a-Service)
9. **Service Integration**: Let network management play nice with OSS/BSS
10. **Openness**: Full choice of modular plug-ins

**Note**: These are exactly the same reasons why we need/want SDN/NFV.

**Obs**: Differences on the (complementary) SDN and NFV approaches on how. (SDN :: decoupling of control plane, NFV : decoupling of SW function from HW)

Source: Adapted from Raj Jain
NFV vs. SDN

**SDN** >>> **flexible** forwarding & steering of traffic in a physical or virtual network environment

[Network Re-Architecture]

**NFV** >>> **flexible** placement of virtualized network functions across the network & cloud

[Appliance Re-Architecture] (initially)

>>> **SDN & NFV** are **complementary** tools for achieving full **network programmability**
Networking as Learned in School (text books)

Source: Martin Casado CS244 Spring 2013, Lecture 6, SDN
Networking in Practice

“in theory, theory and practice are the same; in practice they are not...”

Source: Martin Casado CS244 Spring 2013, Lecture 6, SDN
Problem with Internet Infrastructure

- Hundreds of protocols
- 6,500 RFCs
- Tens of Millions of lines of code
- Closed, proprietary, outdated
- Billions of gates
- Power hungry and bloated

Vertically integrated, complex, closed, proprietary

Not good for network owners and users

Source: ON.LAB
Trend

“Mainframe”

Computer Industry

Virtualization layer

Windows (OS)
Linux
Mac OS

x86 (Computer)

Network Industry

Virtualization or “Slicing”

NOX (Network OS)
Network OS

OpenFlow

Source: ON.LAB
SDN to the rescue!

Software Defined Networking

Warning: Contains optimism
(Plug to http://PacketPushers.net for Unicorn Humor!)

What is SDN?

A: Network Virtualization  B: Openflow
C: Layer-2 Technology  D: Depends on the Day
So, What is SDN?

“OpenFlow is SDN, but SDN is not OpenFlow”
(Does not say much about SDN) — Networking community

“Don’t let humans do machines’ work”
(probably right...) — Networking Professional

“Let’s call SDN whatever we can ship today”
(aka SDN washing) — Vendor X

“SDN is the magic buzzword that will bring us VC funding”
(hmmm... N/A, N/C) — Startup Y

“SDN is the magic that will get my paper/grant accepted”
(maybe but not at SIGCOMM?) — Researcher Z
What is SDN?

In the SDN architecture, the control and data planes are decoupled, network intelligence and state are logically centralized, and the underlying network infrastructure is abstracted from the applications.

— Open Networking Foundation white paper

Software Defined Networking (SDN) refactors the relationship between network devices and the software that controls them. Open interfaces to network switches enable more flexible and predictable network control, and they make it easier to extend network function.

— HotSDN CFP
SDN definitions

• With the original (OpenFlow) definition, SDN represented a network architecture where the forwarding state is solely managed by a control plane and is decoupled from the data plane.

• The industry, however, has moved on from the original academic purist view of SDN to referring to anything disruptive or fundamentally new as part of SDN.

At least two definitions for SDN:

1. academic
   (purist view: strict decoupling of the data and control plane)

2. industry
   (many-fold business-driven views)

    SDN :: Evolving Definition
Rethinking the “Division of Labor”
Traditional Computer Networks

Data plane: Packet streaming
Forward, filter, buffer, mark, rate-limit, and measure packets

Source: Adapted from J. Rexford
Rethinking the “Division of Labor”
Traditional Computer Networks

Control plane:
Distributed algorithms

Track topology changes, compute routes, install forwarding rules

Source: Adapted from J. Rexford
Rethinking the “Division of Labor”
Traditional Computer Networks

Management plane:
Human time scale

Collect measurements and configure the equipment

Source: Adapted from J. Rexford
Software Defined Networking (SDN)

- Logically-centralized control
- API to the data plane (e.g., OpenFlow)

Smart, slow

Dumb, fast

Switches

Source: Adapted from J. Rexford
SDN: Definitions, Concepts, and Terminology

SDN refers to software-defined networking architectures where:

- Data- and control planes **decoupled** from one another.
- Data plane at **forwarding devices managed and controlled** (remotely) by a “controller”.
- Well-defined **programming interface** between control- and data planes.
- **Applications** running on controller manage and control underlying (abstract) data plane.

• **Control plane:** controls the data plane; logically centralized in the “controller” (a.k.a., network operating system).

• **Southbound interface:**
  (instruction set to program the data plane +
  (protocol btw control- and data planes).
  E.g., OpenFlow, POF, Forces, Netconf

Source:
“Software-Defined Networking: A Comprehensive Survey”,
Kreutz et al., In Proceedings of the IEEE, Vol. 103, Issue 1, Jan. 2015..
SDN: Definitions, Concepts, and Terminology

- **Data plane**: network infrastructure consisting of interconnected forwarding devices (a.k.a., forwarding plane).

- **Forwarding devices**: data plane hardware- or software devices responsible for data forwarding.

- **Flow**: sequence of packets between source-destination pair; flow packets receive identical service at forwarding devices.

- **Flow rules**: instruction set that act on incoming packets (e.g., drop, forward to controller, etc)

- **Flow table**: resides on switches and contains rules to handle flow packets.

SDN: Definitions, Concepts, and Terminology

- **Northbound interface**: API offered by control plane to develop network control- and management applications.

- **Application Layer / Business Applications (Management plane)**: functions, e.g., routing, traffic engineering, that use Controller functions / APIs to manage and control network infrastructure.

One SDN to rule them all, with a discovery app to find them, One SDN controller to tell them all, on which switchport to bind them. In the Data Center, where the packets fly.

One SDN to rule them all

Actually not, different reasonable models and approaches to SDN are being pursued

Further reading: [http://theborgqueen.wordpress.com/2014/03/31/the-legend-of-sdn-one-controller-to-rule-them-all/](http://theborgqueen.wordpress.com/2014/03/31/the-legend-of-sdn-one-controller-to-rule-them-all/)
Different SDN Models

- Canonical/Open SDN
- Compiler
- Hybrid
- Overlay

Legend:
- Control-plane component(s)
- Data-plane component(s)
SDN asks (at least) three major questions

1. Where the control plane resides “Distributed vs Centralized”?
2. How does the Control Plane talk to the Data Plane?
3. How are Control and Data Planes programmed?

Source: Adapted from T. Nadeu, slides-85-sdnrg-5.pptx
Different SDN Models to Program / Refactor the Stack

- **Orchestrator**
  - VNFM (Manager)
  - VIM (Infra-M)
  - Legacy
- **OSS/BSS**
  - Mgm. APIs
  - Mgm. Apps
  - Business / Control Apps
- **SDN**
  - Network Controller / OS
    - Northbound APIs
    - Southbound APIs/Plugins
    - Southbound Protocol (e.g. OF)
  - Virtualization
    - HW Resources
      - GP-CPU (x86, ARM)
    - Virtualization
      - Mgmt.
  - APIs
  - Mgm.
  - Distributed L2/L3 Control Plane
    - Southbound Agent (e.g. OF)
  - HAL APIs / Drivers
  - Data Plane
  - P4

Program

Compiler

Target Binary

Auto-Generated

APIs

Network Function Virtualization (NFV)

- VNF
  - Mgmt. CP DP
- Mgmt. APIs
- OSS/BSS APIs
  - Southbound APIs/Plugins
**11:00am - 12:40pm Session 1 - SDN & NFV I**

**Session Chair: Nate Foster (Cornell University)**

**ClickNP: Highly Flexible and High Performance Network Processing with Reconfigurable Hardware**
Bojie Li (USTC / Microsoft Research), Kun Tan (Microsoft Research), Layong (Larry) Luo (Microsoft), Yanqing Peng (SJTU / Microsoft Research), Renqian Luo (USTC / Microsoft Research), Ningyi Xu (Microsoft Research), Yongqiang Xiong (Microsoft Research), Peng Cheng (Microsoft Research), Enhong Chen (USTC)

**Packet Transactions: High-Level Programming for Line-Rate Switches**
Anirudh Sivaraman (MIT CSAIL), Alvin Cheung (University of Washington, Seattle), Mihai Budiu (VMWare Research), Changhoo Kim (Barefoot Networks), Mohammad Alizadeh (MIT CSAIL), Hari Balakrishnan (MIT CSAIL), George Varghese (Microsoft Research), Nick McKeown (Stanford University), Steve Licking (Barefoot Networks)

**SNAP: Stateful Network-Wide Abstractions for Packet Processing**
Mina Tahmasbi Arashloo (Princeton University), Yaron Koral (Princeton University), Michael Greenberg (Pomona College), Jennifer Rexford (Princeton University), David Walker (Princeton University)

**Programmable Packet Scheduling at Line Rate**
Anirudh Sivaraman (MIT CSAIL), Suvinay Subramaniam (MIT CSAIL), Mohammad Alizadeh (MIT CSAIL), Sharad Chole (Cisco Systems), Shang-Tse Chuang (Cisco Systems), Anurag Agrawal (Barefoot Networks), Hari Balakrishnan (MIT CSAIL), Tom Edsall (Cisco Systems), Sachin Katti (Stanford University), Nick McKeown (Stanford University)
ClickNP: Highly Flexible and High Performance Network Processing with Reconfigurable Hardware

Contributions

• Accelerating NFs with programmable HW (FPGA)
• ClickNP: C-like DSL & toolchain
• 40 Gbps line rate
• Five demonstration NFs: (1) traffic capture and generator, (2) a firewall, (3) IPSec gateway, (4) Layer-4 load balancer, (5) pFabric scheduler

Topic Challenges

• High-performance programmable DP implementation
• Programmer-friendly high-level DSL for networking

How are Control & Data Planes programmed?
Contributions

• Program data-plane algorithms in a high-level language and compile them
• Domino, a C-like imperative language + compiler
• Banzai machine model for DP

Topic Challenges

• High-performance programmable DP implementation
• DP algorithms create and modify algorithmic state
• SW algorithms on programmable line-rate HW

How are Control & Data Planes programmed?
Contributions

• Programmable scheduler using a single abstraction: the push-in first-out queue (PIFO)
• HW design for a 64-port 10 Gbit/s switch
• Verilog code available at http://web.mit.edu/pifo/

Topic Challenges

• High-performance programmable DP implementation
• Scheduling algorithms—potentially algorithms that are unknown today—to be programmed into a switch without requiring hardware redesign
• How will programmable scheduling be used in practice?
Contributions

- Programming language with persistent global arrays, transactions, one-big-switch illusion
- Compiler that decides where to place state, how to route traffic (through MILP)
- 20 Example applications

Topic Challenges

- Managing distributed state
- Consistency of state
- Efficient use of routes, switch resources

Scope

SNAP program

Compiler
- Distributing state
- One Big Switch abstraction

Data Plane

Stateful

#programmability
#visibility
#automation
#virtualization
<table>
<thead>
<tr>
<th>Session Time</th>
<th>Session Title</th>
<th>Chair</th>
</tr>
</thead>
<tbody>
<tr>
<td>3:45pm - 5:00pm</td>
<td>Session 11 - SDN &amp; NFV II</td>
<td>Aditya Akella</td>
</tr>
<tr>
<td></td>
<td><strong>OpenBox: A Software-Defined Framework for Developing, Deploying, and Managing Network Functions</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Anat Bremler-Barr (The Interdisciplinary Center, Herzliya), Yotam Harchol (The Hebrew University of Jerusalem), David Hay (The Hebrew University of Jerusalem)</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>PISCES: A Programmable, Protocol-Independent Software Switch</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Muhammad Shahbaz (Princeton University), Sean Choi (Stanford University), Ben Pfaff (VMware), Changhoon Kim (Barefoot Networks), Nick Feamster (Princeton University), Nick McKeown (Stanford University), Jennifer Rexford (Princeton University)</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Dataplane Specialization for High-performance OpenFlow Software Switching</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>László Molnár (Ericsson Research, Hungary), Gergely Pongrácz (Ericsson Research, Hungary), Gábor Enyedi (Ericsson Research, Hungary), Zoltán Kis (Ericsson Research, Hungary), Levente Cskor (Budapest University of Technology and Economics), Ferenc Juhász (Budapest University of Technology and Economics), Attila Körösi (Budapest University of Technology and Economics), Gábor Retvári (Budapest University of Technology and Economics)</td>
<td></td>
</tr>
</tbody>
</table>
Contributions

• Framework for network-wide development, deployment, and management of network functions (NFs).
• OpenBox Protocol & Controller

Topic Challenges

• Flexibility/programmability of SDN/NFV
• Management & DP Performance of Service Function Chains

How are Control & Data Planes programmed?

How does the Control Plane talk to the Data Plane?
Contributions

• Software switch derived from Open vSwitch (OVS) with behavior customized using P4: https://github.com/P4-vSwitch

• Compiler to optimize forwarding performance

• Programs are about 40x shorter than equivalent OVS ones

Topic Challenges

• High-performance SW-based DP implementation

• Flexible hypervisor switches ("hard-wired" today)
Contributions

- **ESWITCH** switch architecture using on-the-fly template-based to compile OpenFlow pipeline into efficient machine code
- A case against flow caching and general purpose switch fast paths → dataplane specialized with respect to the workload
- 100+ Gbps on a single Intel blade and 100Ks flow entries, while supporting fast updates

Topic Challenges

- High-performance SW-based OpenFlow/DP implementation

How are Control & Data Planes programmed?
<table>
<thead>
<tr>
<th>Time</th>
<th>Session</th>
<th>Chair</th>
<th>Presenters</th>
</tr>
</thead>
</table>
| 4:15pm - 5:30pm | **Session 3 - Monitoring and Diagnostics** | **Jeff Mogul (Google)** | **One Sketch to Rule Them All: Rethinking Network Flow Monitoring with UnivMon**
|              |                                  |                  | **Trumpet: Timely and Precise Triggers in Data Centers**
|              |                                  |                  | Masoud Moshref *(University of Southern California)*, Minlan Yu *(University of Southern California)*, Ramesh Govindan *(University of Southern California)*, Amin Vahdat *(Google)* |
|              |                                  |                  | **The Good, the Bad, and the Differences: Better Network Diagnostics with Differential Provenance**
|              |                                  |                  | Ang Chen *(University of Pennsylvania)*, Yang Wu *(University of Pennsylvania)*, Andreas Haeberlen *(University of Pennsylvania)*, Wenchao Zhou *(Georgetown University)*, Boon Thau Loo *(University of Pennsylvania)* |
Contributions

- Universal Streaming implementation using P4
  - Heavy hitters on successive sampled substreams
- One-big-switch abstraction for monitoring sketches
- Comparable accuracy to custom sketches

Topic Challenges

- Several algorithm and sketches exist for specific problems
  - Data structures and algorithms specific to desired metric
- Solution that is both general and accurate is an open problem

Scope

Monitoring with limited resources

Sketches/Streaming algorithms:

Single or constant passes over data, sublinear space, approximate given statistical measure (mean, median, moments,..)

Seminal paper AMS paper (ref [9])
Contributions

• Finding root causes by differential provenance
  • Given a reference (good) provenance tree, and a bad one, find the events you have to change in the bad one to make it good

Topic Challenges

• Provenance produces sufficient, but extensive information to diagnose root causes

Scope

Diagnostics of networked systems based on provenance

SDNs one use case in which programmability helps with recording of provenance and replay of events
Contributions

• Language for specifying network-wide predicates
• Leverage end-host CPU resources to achieve the goals
  • Many useful optimizations for processing

Topic Challenges

• Scale
  • Volume of traffic
  • # of events
  • # of endpoint
  • 70ns/packet (64b @ 10G)

Scope

• Control loop for monitoring and acting on the network
  • Programmability enables software control loop (not human timescale)
  • Datacenter active monitoring
    • Faults detection, network planning, traffic engineering, performance diagnosis
• Goals:
  Network-wide predicates over every packet with μs reaction time
SDN/NFV: The Frontier of Networking

Existing
• CLIs
• Closed Source
• Vendor Lead
• Classic Network Appliances

New
• APIs
• Open Source
• Customer Lead
• Network Function Virtualization (NFV)

Adapted from: Kyle Mestery, Next Generation Network Developer Skills
Thank you! Questions?
BACKUP
Session 1 - SDN & NFV
1.1 [NFV] ClickNP: Highly Flexible and High Performance Network Processing with Reconfigurable Hardware
1.2 [Programmable data plane] Packet Transactions: High-Level Programming for Line-Rate Switches
1.3 [One big switch] SNAP: Stateful Network-Wide Abstractions for Packet Processing
1.4 [Programmable data plane] Programmable Packet Scheduling at Line Rate

Session 11 - SDN & NFV
11.1 [NFV] OpenBox: A Software-Defined Framework for Developing, Deploying, and Managing Network Functions
11.3 [switch design] Dataplane Specialization for High-performance OpenFlow Software Switching

Session 3 - Monitoring and Diagnostics
3.1 [Monitoring] One Sketch to Rule Them All
3.2 [Monitoring] Differential Provenance
3.3 [Monitoring] Trumpet: Triggers in Data Centers
Architectural Framework [ETSI NFV]

Source: ETSI NFV White Paper 2
NFV

Alternative options to virtualize NFV apps
ONF recursive SDN architecture

Network Programmability Layers

Source: Introducing Network Programmability Fundamentals
Part#: CTOD-SDN-1.0-017141
SDN asks (at least) three major questions

- Where the control plane resides
  “Distributed vs Centralized”?

- What state belongs in distributed protocols?
- What state must stay local to switches?
- What state should be centralized?

- What are the effects of each on:
  - state synchronization overhead
  - total control plane overhead
  - system stability and resiliency
  - efficiency in resource use
  - control loop tightness

SDN asks (at least) three major questions

- Prop. IPC
- OpenFlow (with or w/extensions)
- Open Source south-bound protocols
- Via SDN controller broker and south-bound plug-ins
- Other standardized protocols

What are the effects of each on:
- Interoperability, Evolvability, Performance
- Vendor Lock-in
SDN asks (at least) three major questions

• Levels of Abstraction
• Open APIs
• Standardized Protocols

• What are the effects of each on:
  - Data plane flexibility
  - Integration with legacy
  - Interoperability (CP / DP)
  - Vendor lock-in

NFV Concepts

- **Network Function (NF):** Functional building block with a well defined interfaces and well defined functional behavior
- **Virtualized Network Function (VNF):** Software implementation of NF that can be deployed in a virtualized infrastructure
- **VNF Set:** Connectivity between VNFs is not specified, e.g., residential gateways
- **VNF Forwarding Graph:** Service chain when network connectivity order is important, e.g., firewall, NAT, load balancer
- **NFV Infrastructure (NFVI):** Hardware and software required to deploy, manage and execute VNFs including computation, networking, and storage.
- **NFV Orchestrator:** Automates the deployment, operation, management, coordination of VNFs and NFVI.

Source: Adapted from Raj Jain
NFV Concepts

- **NFVI Point of Presence (PoP):** Location of NFVI
- **NFVI-PoP Network:** Internal network
- **Transport Network:** Network connecting a PoP to other PoPs or external networks
- **VNF Manager:** VNF lifecycle management e.g., instantiation, update, scaling, query, monitoring, fault diagnosis, healing, termination
- **Virtualized Infrastructure Manager:** Management of computing, storage, network, software resources
- **Network Service:** A composition of network functions and defined by its functional and behavioral specification
- **NFV Service:** A network services using NFs with at least one VNF.

Source: Adapted from Raj Jain
NFV Concepts

• **User Service:** Services offered to end users/customers/subscribers.

• **Deployment Behavior:** NFVI resources that a VNF requires, e.g., Number of VMs, memory, disk, images, bandwidth, latency

• **Operational Behavior:** VNF instance topology and lifecycle operations, e.g., start, stop, pause, migration, ...

• **VNF Descriptor:** Deployment behavior + Operational behavior

Source: Adapted from Raj Jain