Cloud Networking & Infrastructure As A Service (SDN & NFV)
Contents at Glance

• Introduction to the Nuts and Bullets of SDN/NFV Acronyms
• History and Evolution of SDN
• SDN Architecture
• SDN Global Integration
• The Importance of NFV
• Virtualizing Networking Nodes via NFV
• Service Providers – Visions and Innovations
Introduction to the Nuts and Bolts of SDN/NFV acronyms
The Internet Has Gone Through Four Distinct Evolutions
The Internet will Extend to Billions of New Devices
Number of Connected Objects Expected to Reach 50bn by 2020
Turning Data into Wisdom
Open Mind Evolution

- Supply of Innovative Applications
- Open Innovation
- Reduce Space & Power Consumption
- Network Functions
- Virtualization
- Software Defined Networking
- Create Network Abstraction for faster innovation
Warning, this could get confusing
SDN

Software Defined Networking (SDN) – an emerging software architecture for networking infrastructure that separates the data plane, control plane and application plane and centralizes their management for greater flexibility, lower cost, improved process efficiency and simplified management.
NFV

Network Function Virtualization (NFV) – an alternative design approach for building complex telecom or service provider solutions that virtualizes entire classes of functions into building blocks that may be connected, or chained, together to create services and deployed on industry standard hardware. Examples include border gateways and routers, session controllers and voice service, security appliances and application delivery systems.
<table>
<thead>
<tr>
<th>Feature Set</th>
<th>Software Defined Networking (SDN)</th>
<th>Network Function Virtualization (NFV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Concept</td>
<td>Separate control and data planes, centralize control and programmability of network</td>
<td>Relocate network functions from dedicated appliances to generic servers</td>
</tr>
<tr>
<td>Target Location</td>
<td>Campus, Data Center / Cloud</td>
<td>Service Providers/Telco Providers/Cloud Providers</td>
</tr>
<tr>
<td>Target Devices</td>
<td>Commodity servers, routers, switches</td>
<td>Commodity servers</td>
</tr>
<tr>
<td>Initial Applications</td>
<td>Network Programmability</td>
<td>Cloud Orchestration</td>
</tr>
<tr>
<td>New Protocols</td>
<td>Openflow, OpenDaylight</td>
<td>OpenStack API’s</td>
</tr>
<tr>
<td>Formalization</td>
<td>Open Networking Foundation (ONF)</td>
<td>European Telecommunications Standards Institute / NFV Working Group</td>
</tr>
</tbody>
</table>
SDN & NFV as One
History and Evolution of SDN
But How did it all start ...

Major Leaders in Architecting SDN worldwide

Inventor of SDN

Martin Casado

Nick McKeown

Teemo Koponen
Traditional Networking Drawbacks

- Networks are Complex and Hard to Manage
- Manual Intervention
- Limited visibility that span from users to applications
- Slow Provisioning of Network Services
- Unable to adapt quickly to dynamic changing of business needs
Caveats and Clarifications

• SDN is not really a “technological” advance.
  – It is merely a way of organizing network functionality.

• But that’s all the Internet architecture is …
  – Not clever, but the right design
Why Was SDN Needed?

• Networks are hard to manage

• Networks are hard to evolve

• Network design is not based on formal principles
Paradise Lost?

- Limited Redundancy
- Constrained Topology
- Poor responses to dynamic events
- Scaling limited by operational complexity
Problem: Poor Forwarding

- Fixed function
- Implementation often exposed through API
- Unclear state consistency semantics
Problem: Distributed Computing is Hard
Designing Traffic Engineering

Flows: R1->R6: 20; R2->R6: 20; R4->R6: 20
Convergence After Failure

Flows: R1->R6: 20; R2->R6: 20; R4->R6: 20
Convergence After Failure

Flows: R1->R6: 20; R2->R6: 20; R4->R6: 20

R5-R6 link fails
- R1, R2, R4 *autonomously* try for next best path
Convergence After Failure

Flows: R1->R6: 20; R2->R6: 20; R4->R6: 20

R5-R6 link fails
- R1, R2, R4 autonomously try for next best path
- R1 wins, R2, R4 retry for next best path
- R2 wins this round, R4 retries again
What is the Root Cause?

• How do we build systems that work well?
  – We break them into tractable components

  “Modularity based on abstraction is the way things get done.” – Barbara Liskov

• If you can’t **manage**, **evolve**, or **understand** a system, probably don’t have the right abstractions

• What abstractions do we need for networks?
The Two Networking “Planes”

- Data plane: process packets with local forwarding state
  - Forwarding state + packet header -> forwarding decision
- Control plane: Compute the forwarding state
  - Distributed protocols, manual configuration, etc.
- These planes solve very different problems
  - Therefore they require different abstractions
Data Plane Abstraction Layers

Applications
Built on...

Reliable/Unreliable Transport
Built on..

Best Effort global packet delivery
Built on...

Best Effort local packet delivery
Built on...

Local physical transfer of bits
Layers Key to Internet’s Success

• Broke problem into tractable pieces
  – Don’t have to solve all networking problems at once.

• Enable rapid innovation in each layer
  – Can evolve independently, if interface unchanged

• Survived many orders-of-magnitude change in
  – Speed
  – Scale
  – Diversity of uses
The Myth of the Internet’s Dataplane

- The Internet only provides best effort delivery
- This minimal functionality is enough for users
Today’s Control Plane: No Abstraction

- Variety of mechanisms and goals:
  - Routing: Distributed routing algorithms
  - Isolation: ACL’s, VLANs, Firewalls …
  - Traffic Engineering: adjust weights MPLS

- No modularity, limited functionality

- Control plane mess: mechanism without abstraction – impossible to program networks without technology abstraction
Abstractions For Control Plane

- Abstractions: identify reusable components
- To accomplish its task, the control plane must:
  - Figure out what network looks like (topology)
  - Figure out how to accomplish goal on given topology
  - Tell the switches/routers what to do (configure forwarding state)
- What components do we want to reuse?
  - Determining the topology information
  - Configuring forwarding state on routers/switches

*These are the roots of SDN*
SDN Architecture
SDN: Two Control Plane Abstractions

• Global network view
  – Provides information about current network
  – Implemented with “Network Operating System”
    • Software running on servers in network

• Forwarding model:
  – Provides standard way of defining forwarding state
  – This is OpenFlow
    • Specification of <header, action> flow entries
Decouple Distribution Model from Topology
Clean Separation of Concerns

• Control program: express operator goals
  – Implemented on global network view abstraction
  – Computes forwarding state for each router/switch
• NOS: links global view and physical routers/switches
  – Gathers information for global network view
  – Conveys configurations from control program to routers/switches
• Routers/Switches: merely follow orders from NOS

Enables independent innovation in “layers”
Major Change in Paradigm

• Control mechanism is now programmed using NOS API
  – Not a distributed protocol, just a graph algorithm
  – Much easier to manage, evolve and understand
• Clean separation of control and data planes
  – Not packaged together in proprietary boxes
  – Enables use of commodity hardware, 3rd party software
  – Supports better testing and troubleshooting
The Role of Control Programs

- Control programs are how operators express their network requirements and policies
  - Connectivity, isolation, access control etc.
- Control programs should **not** be responsible for implementing these requirements and policies
- Reason: Control Program should be relatively simple
  - Push complexity into reusable code (NOS, etc.)
## SDN Architecture

| SDN Architecture | 
|------------------|---|
| Application Layer | 
| Control Layer | 
| Infrastructure Layer | 

[Diagram of SDN Architecture showing layers and interactions with business applications, cloud orchestration, SDN controller, and network devices.]
Simple Example: Access Control

• Operator goal: prevent A’s packets from reaching B
• Control program does so with access control entries.
Virtualization Simplifies Control Program
Hypervisor then inserts flow entries as needed.
Centralized Traffic Engineering

Simple topology

Flows:
- R1->R6: 20; R2->R6: 20; R4->R6: 20
Centralized Traffic Engineering

Simple topology

Flows:
- R1->R6: 20; R2->R6: 20; R4->R6: 20
- R5-R6 fails
  - R5 informs TE, which programs routers in one shot
Centralized Traffic Engineering

Simple topology

Flows:
- R1 -> R6: 20; R2 -> R6: 20; R4 -> R6: 20
- R5-R6 link fails
- R5 informs TE, which programs routers in one shot
- Leads to faster realization of target optimum
Software Defined Network
Software Switches are Common
Datacenter Evolution Design
SDN Global Integration
Virtualization – SDN Killer Application

- Consider a multi-tenant datacenter
  - Want to allow each tenant to specify virtual topology
  - This defines their individual policies and requirements
- Datacenter’s network hypervisor compiles these virtual topologies into set of switch configurations
  - Takes 1000s of individual tenant virtual topologies
  - Computes configurations to implement all simultaneously

- This is what people are paying money for...
Long-Term Opportunity

• Enable architectural evolution and diversity
  – Build domains with software at edge, hardware in core
  – Only Software knows about interdomain protocols
  – Internal label-forwarding is protocol agnostic (think L2)

• Architectural evolution via SDN control program
  – Add radical new Internet Architectures

• This eliminates the need for a “narrow waist”
  – Domains can run many architectures in parallel
  – A conceptually radical but technically simple change
Programmable Flow Virtual Tenant Networks
SDN -> The Next Level
Top 5 Telecom and Datacom Network Equipment and Software Vendors in the World

- Cisco: 19%
- Huawei: 11%
- Ericsson: 9%
- ZTE: 7%
- Others: 51%

Based on global revenue


a matrix company
The Importance of NFV
Middleboxes as common as Switches/Routers
The Reality of the Internet’s Dataplane

• Middleboxes commonly used to augment dataplane

• They provide wide variety of needed functionality:
  – Firewalls
  – WAN Optimization
  – Proxies
  – Gateways
  – VPN’s
  – Load Balancers
  – IPS/IDS
Four Important Facts

• Most packets touched by several middleboxes
• Most middlebox functionality is deployed at edge.
• Most middleboxes use x86 packet processing
• The processing is more complex than the forwarding
One Inescapable Conclusion

• SDN should implement middlebox functionality
• This cements the case for edge software forwarding
• Represents a radical shift from hardware to software
Network Function Virtualization

- NFV decouples hardware from software
- A subset of software defined networking (SDN)
- Moves the network functions from specialized appliances to applications that run on COTS equipment (servers, switches, storage)
What is NFV?

Specialized Appliances:
- Firewall
- Load Balancer
- Deep-Packet Inspection (DPI)
- Router

NFV Model:
- Firewall VNF
- DPI VNF
- Load Balancer VNF
- Router VNF
- Virtualization Layer
- COTS Servers

Orchestration
Advantages of NFV

- Flexibility - Open SW and HW
- Scalability/Elasticity
- Localization
- Cost Reduction – reduce CAPEX and OPEX
- Time to Market – Shorter development cycles
- Innovation
Virtualizing NFV
Telecom Cloud Components

**SDN**

Vendors trying to maximize utilization through automation

- Determined to leverage existing hw architectures
- Must deliver cost savings to stay in the game

**NFV**

Operators desiring improved financials

- Determined to follow Enterprise Virtualization Path
- High Availability requirements must still be met

**Where to start?**
SDN Paradigm & “Global Views” using NFV

- Abstraction and Layering
- Openness
- Global “Views” & Explicit Control
- Virtualization
- Automation & Orchestration
- Better Leverage of Policy & Analytics
Service Providers
Visions & Innovations
Why Service Providers Want SDNs & NFV

Top Drivers

• New revenue and operational efficiency drive SDN/NFV
  – Service Velocity – quicker time to new revenue
  – Simpler provisioning over multi-vendor networks
  – Lower OPEX with master view over multi-vendor, multi-layer
  – Global view and network intelligence
  – Global view and subscriber intelligence -> quicker new services and new revenues

• Translation: Need more automation, new paradigm
Nearly every operator will deploy SDN or NFV in some aspect of their network at some point.
Likely Domains for Deployments

Prioritized list of SDN target domains

- DC- related: Intra-DC, Inter-DC, Cloud Services
- Consumer/business services via “virtual CPE”
- Mobile Core, BNG/BRAS, mobile backhaul, metro aggregation
Cloud Datacenter with SDN Virtualization
Top 5 Network Locations Operators Plan to Deploy

- Within data centers
- Between data centers
- Operations and management
- CDNs
- Cloud services

© Infonetics Research, *SDN and NFV Strategies: Global Service Provider Survey, July 2013*
Contained Domains – Data Centers Are Easy

• Service providers are inspired by DC SDNs as proof of concept
  – 3 years of Google resources/investments + Nicira smarts solves operational problems of data center environment
  – SDNs work

• DC is simple contained domain… while service provider networks are much more complex
  – Much bigger challenge / Much bigger payoff

• SDNs will take time
Google’s SDN based OpenFlow WAN
Telco Operators SDN/NFV Design
WAN Services

- Long service lifecycle
- Manual service activation
- Infrequent changes
- Proprietary & hard to program

Cloud Services

- Short Service lifecycle
- Automated service activation
- Frequent changes (Elastic)
- Open and programmable
Cloud applications changing how WAN services are used
Carrier networks must evolve to support this new reality
Community Building is a Core Objective

It's only the Beginning
Thanks :) 
See you next year