OpenFlow State of the Art

SNE COLLOQUIUM 7 MAY 2013

Ronald van der Pol <Ronald.vanderPol@SURFnet.nl>

Outline

Why was OpenFlow developed?

How does OpenFlow work?

OpenFlow Standardisation (ONF)

Some examples of OpenFlow usage
OpenStack Cloud Computing Platform
Google Data Network
Protocol Implementation

Some of the OpenFlow Players

Wrapup

Outline

Why was OpenFlow developed?

How does OpenFlow work?

OpenFlow Standardisation (ONF)

Some examples of OpenFlow usage

OpenStack Cloud Computing Platform
Google Data Network
Protocol Implementation

Some of the OpenFlow Players

Wrapup

Why OpenFlow?

Enable network innovation (again)

Reducing operational costs (OPEX)

Alternative for “protocol soup”

Applying computing model to networking

Enable Network Innovation

OpenFlow was developed at Stanford University as part of Clean Slate program

Today’s university networks need to have 24x7 availability

Potential disruptive network tests are not possible anymore

OpenFlow enables slicing the network in production and experimental part

OPEX in Networking

Adding routers and switches to your network increases the operational cost

Each new device needs to be configured manually via the CLI and its neighbours need to be configured too

Firmware updates on routers and switches with slow CPUs takes a long time

Changes usually involves configuration actions on all devices

OPEX in Computing

Operational tasks in computing scale much better

Adding servers to a computer grid or cloud cluster does not increase the operational cost (automated bulk upgrades and configurations)

Middleware software with centralized policy (OpenStack, etc) controls the servers

Configure the policy once and push the button to apply the changes to all servers

OPEX with OpenFlow

Run networks similar to what is done with computing grids and clouds

Individual CLI configuration moved to centralised OpenFlow controller configuration

Application defines policy, translates it to forwarding entries which are sent via the OpenFlow protocol to the OpenFlow switches

“Protocol Soup”

Current way to handle new functionality in networking is to define a new protocol

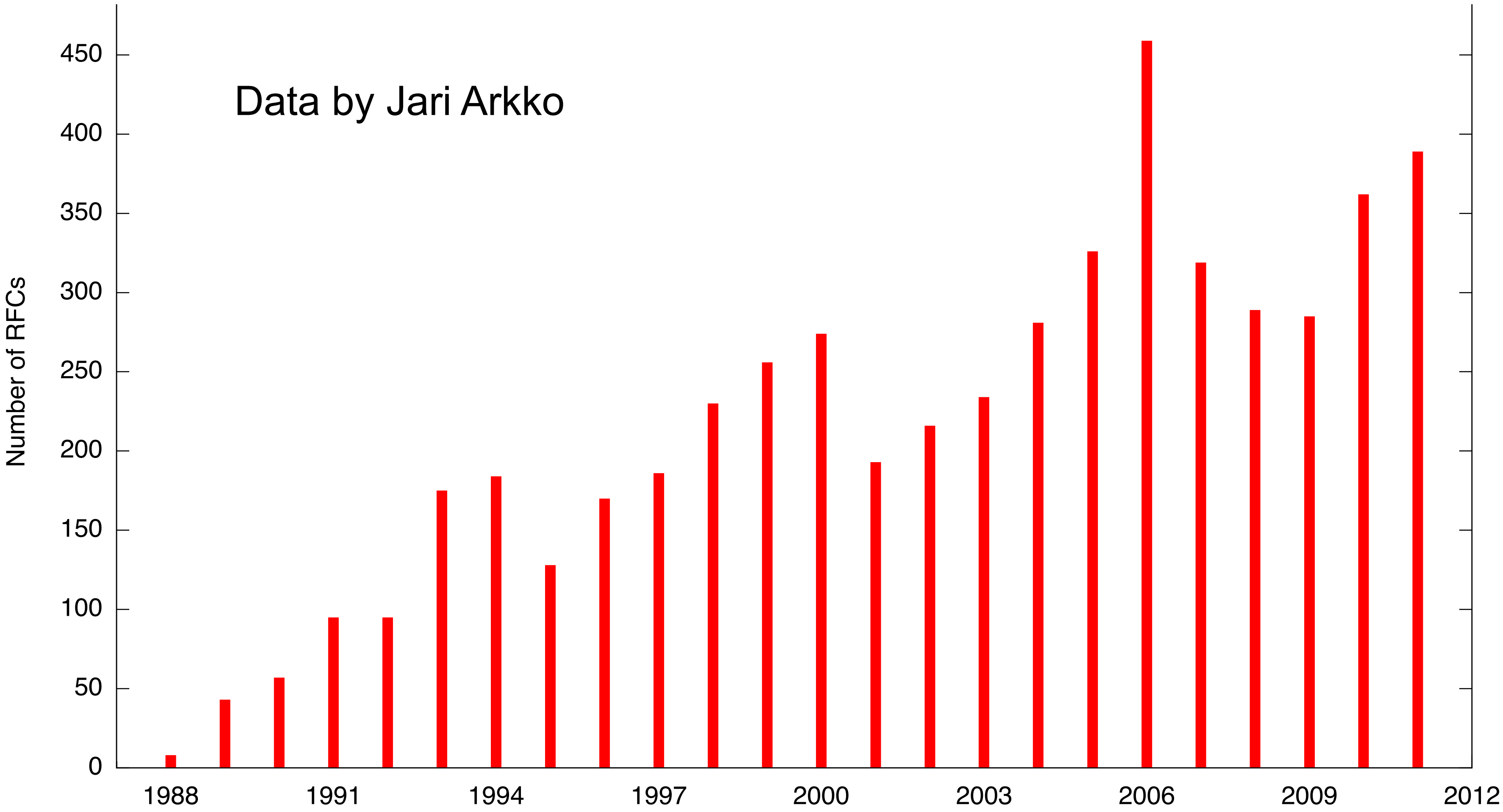
Exponential growth in network protocol standards

Standards seem to become larger and more complex

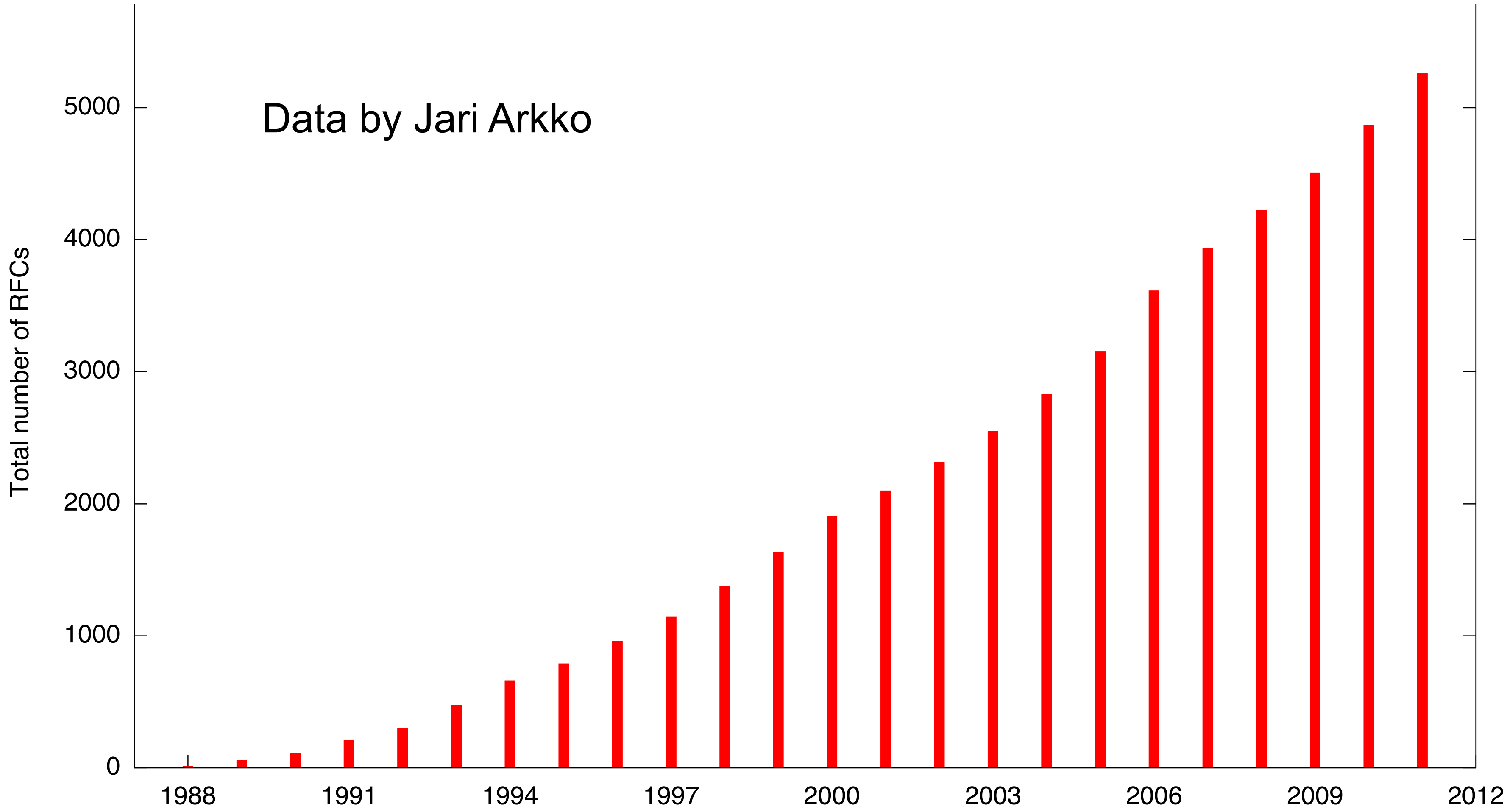
Vendors implement all standards, which increases costs and decreases stability

Do you need all those standards?

IETF RFC Publication Rate



Total Number of RFCs Published



IEEE 802.1Q

Simple VLAN standard?

Not really, original version amended by at least 14 additional standards

802.1Q-1998 had 211 pages

802.1Q-2011 has 1365 pages, and includes:
802.1u, 802.1v, 802.1s (multiple spanning trees), 802.1ad (provider bridging), 802.1ak (MRP, MVRP, MMRP), 802.1ag (CFM), 802.1ah (PBB), 802.1ap (VLAN bridges MIB), 802.1Qaw, 802.1Qay (PBB-TE), 802.1aj, 802.1Qav, 802.1Qau (congestion management), 802.1Qat (SRP)

Specs of a Modern Ethernet Switch
(random example, but they are all the same)

Area Networks
- IEEE 802.3ad Static load sharing configuration and LACP based dynamic configuration
- Software Redundant Ports
- IEEE 802.1AB – LLDP Link Layer Discovery Protocol
- LLDP Media Endpoint Discovery (LLDP-MED), ANSI/TIA-1057, draft 08
- Extreme Discovery Protocol (EDP)
- Extreme Loop Recovery Protocol (ELRP)
- Extreme Link State Monitoring (ELSM)
- IEEE 802.1ag L2 Ping and traceroute, Connectivity Fault Management
- ITU-T Y.1731 Frame delay measurements

Management and Traffic Analysis

- RFC 2030 SNTP, Simple Network Time Protocol v4
- RFC 854 Telnet client and server
- RFC 783 TFTP Protocol (revision 2)
- RFC 951, 1542 BootP
- RFC 2131 BOOTP/DHCP relay agent and DHCP server
- RFC 1591 DNS (client operation)
- RFC 1155 Structure of Management Information (SMIv1)
- RFC 1157 SNMPv1
- RFC 1212, RFC 1213, RFC 1215 MIB-II, Ethernet-Like MIB & TRAPs

- XML APIs over Telnet/SSH and HTTP/HTTPS
- Web-based device management interface – ExtremeXOS ScreenPlay™
- IP Route Compression

Security, Switch and Network Protection

- Secure Shell (SSH-2), Secure C and SFTP client/server with enc authentication (requires export encryption module)
- SNMPv3 user based security, w encryption/authentication (see
- RFC 1492 TACACS+
- RFC 2138 RADIUS Authenticati
- RFC 2139 RADIUS Accounting
- RFC 3579 RADIUS EAP support
- RADIUS Per-command Authentic
- Access Profiles on All Routing P
- Access Policies for Telnet/SSH-
- Network Login – 802.1x, Web a MAC-based mechanisms
- IEEE 802.1x – 2001 Port-Based Access Control for Network Log
- Multiple supplicants with multip Network Login (all modes)
- Fallback to local authentication (MAC and Web-based methods)

Sping, Ascend, Stream, Land, Octopus

Security, Router Protection

- RFC 1587 OSPF NSSA Option
- RFC 1765 OSPF Database Overflow
- RFC 2370 OSPF Opaque LSA Option
- RFC 3623 OSPF Graceful Restart
- RFC 1850 OSPFv2 MIB
- RFC 2362 PIM-SM (Edge-mode)
- RFC 2934 PIM MIB
- RFC 3569, draft-ietf-ssm-arch-06.txt PIM-SSM PIM Source Specific Multicast
- draft-ietf-pim-mib-v2-o1.txt
- Mtrace, a "traceroute" facility for IP Multicast: draft-ietf-idmr-traceroute-ipm-07
- Mrinfo, the multicast router information tool based on Appendix-B of draft-ietf-idmr-dvmrp v3-11

IPv6 Host Services

- RFC 3587, Global Unicast Address Format
- Ping over IPv6 transport
- Traceroute over IPv6 transport
- RFC 5095, Internet Protocol, Version 6 (IPv6) Specification
- RFC 4861, Neighbor Discovery for IP Version 6, (IPv6)
- RFC 2463, Internet Control Message Protocol (ICMPv6) for the IPv6 Specification
- RFC 2464, Transmission of IPv6 Packets over Ethernet Networks
- RFC 2465, IPv6 MIB, General Group and Textual Conventions
- RFC 2466, MIB for ICMPv6
- RFC 2462, IPv6 Stateless Address Auto Configuration – Host Requirements
- RFC 1981, Path MTU Discovery for IPv6, August 1996 – Host Requirements
- RFC 3513, Internet Protocol Version 6 (IPv6) Addressing Architecture
- Telnet server over IPv6 transport
- SSH-2 server over IPv6 transport

IPv6 Interworking and Migration

- RFC 2893, Configured Tunnels
- RFC 3056, 6to4

IPv6 Router Services

- RFC 2462, IPv6 Stateless Address Auto Configuration – Router Requirements
- RFC 1981, Path MTU Discovery for IPv6, August 1996 – Router Requirements
- RFC 2710, IPv6 Multicast Listener Discovery v1 (MLDv1) Protocol
- Static Unicast routes for IPv6
- RFC 2080, RIPng

- RFC 1771 Border Gateway Protocol 4
- RFC 1965 Autonomous System Confederations for BGP
- RFC 2796 BGP Route Reflection (supersedes RFC 1966)
- RFC 1997 BGP Communities Attribute
- RFC 1745 BGP4/IDRP for IP-OSPF Interaction
- RFC 2385 TCP MD5 Authentication for BGPv4
- RFC 2439 BGP Route Flap Damping
- RFC 2918 Route Refresh Capability for BGP-4
- RFC 3392 Capabilities Advertisement with BGP-4
- RFC 4360 BGP Extended Communities Attribute
- RFC 4486 Subcodes for BGP Cease Notification message
- draft-ietf-idr-restart-10.txt Graceful Restart Mechanism for BGP
- RFC 4760 Multiprotocol extensions for BGP-4
- RFC 1657 BGP-4 MIB
- RFC 4893 BGP Support for Four-Octet AS Number Space
- Draft-ietf-idr-bgp4-mibv2-02.txt – Enhanced BGP-4 MIB
- RFC 1195 Use of OSI IS-IS for Routing in TCP/IP and Dual Environments (TCP/IP transport only)
- RFC 2763 Dynamic Hostname Exchange Mechanism for IS-IS
- RFC 2966 Domain-wide Prefix Distribution with Two-Level IS-IS
- RFC 2973 IS-IS Mesh Groups
- RFC 3373 Three-way Handshake for IS-IS Point-to-Point Adjacencies
- Draft-ietf-isis-restart-02 Restart Signaling for IS-IS
- Draft-ietf-isis-ipv6-06 Routing IPv6 with IS-IS
- Draft-ietf-isis-wg-multi-topology-11 Multi Topology (MT) Routing in IS-IS

QoS and VLAN Services

Quality of Service and Policies
- IEEE 802.1D – 1998 (802.1p) Packet Priority
- RFC 2474 DiffServ Precedence, including 8 queues/port
- RFC 2598 DiffServ Expedited Forwarding (EF)
- RFC 2597 DiffServ Assured Forwarding (AF)
- RFC 2475 DiffServ Core and Edge Router Functions

Traffic Engineering
- RFC 3784 IS-IS Externs for Traffic Engineering (wide metrics only)

VLAN Services: VLANs, vMANs
- IEEE 802.1Q VLAN Tagging
- IEEE 802.1v: VLAN classification by Protocol and Port

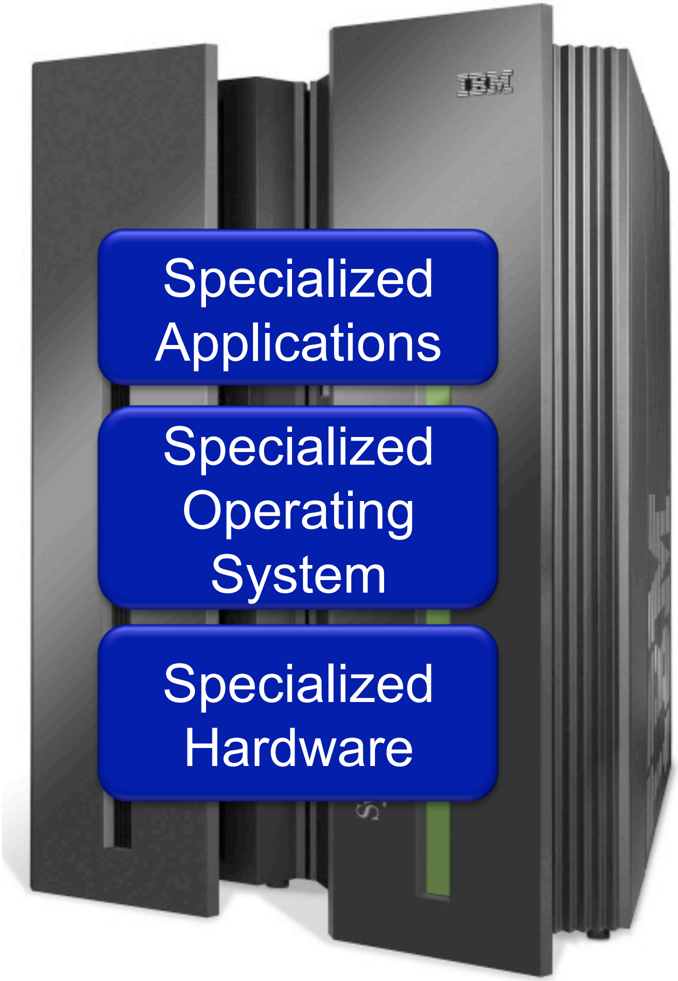
Advanced VLAN Services, MAC-in-MAC
- VLAN Translation in vMAN environments
- vMAN Translation
- IEEE 802.1ah/D1.2 Provider Backbone Bridges (PBB)/MAC-in-MAC

MPLS and VPN Services

Multi-Protocol Label Switching (MPLS)
Requires MPLS Layer 2 Feature Pack License
- RFC 2961 RSVP Refresh Overhead Reduction Extensions
- RFC 3031 Multiprotocol Label Switching Architecture
- RFC 3032 MPLS Label Stack Encoding
- RFC 3036 Label Distribution Protocol (LDP)
- RFC 3209 RSVP-TE: Extensions to RSVP for LSP Tunnels
- RFC 3630 Traffic Engineering Extensions to OSPFv2
- RFC 3784 IS-IS extensions for traffic engineering only (wide metrics only)
- RFC 3811 Definitions of Textual Conventions (TCs) for Multiprotocol Label Switching (MPLS) Management
- RFC 3812 Multiprotocol Label Switching (MPLS) Traffic Engineering (TE) Management Information Base (MIB)
- RFC 3813 Multiprotocol Label Switching (MPLS) Label Switching Router (LSR) Management Information Base (MIB)
- RFC 3815 Definitions of Managed Objects for the Multiprotocol Label Switching (MPLS), Label Distribution Protocol (LDP)
- RFC 4090 Fast Re-route Extensions to RSVP-TE for LSP (Detour Paths)
- RFC 4379 Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures (LSP Ping)
- draft-ietf-bfd-base-09.txt Bidirectional Forwarding Detection

Layer 2 VPNs
Requires MPLS Layer 2 Feature Pack License
- RFC 4447 Pseudowire Setup and Maintenance using the Label Distribution Protocol (LDP)
- RFC 4448 Encapsulation Methods for Transport of Ethernet over MPLS Networks
- RFC 4762 Virtual Private LAN Services (VPLS) using Label Distribution Protocol (LDP) Signaling
- RFC 5085 Pseudowire Virtual Circuit Connectivity Verification (VCCV)
- RFC 5542 Definitions of Textual Conventions for Pseudowire (PW) Management
- RFC 5601 Pseudowire (PW) Management

(slide by Nick McKeown, Stanford University)



App | App | App | App | App | App | App | App | App | App | App

Open Interface

Windows (OS) or Linux or Mac OS

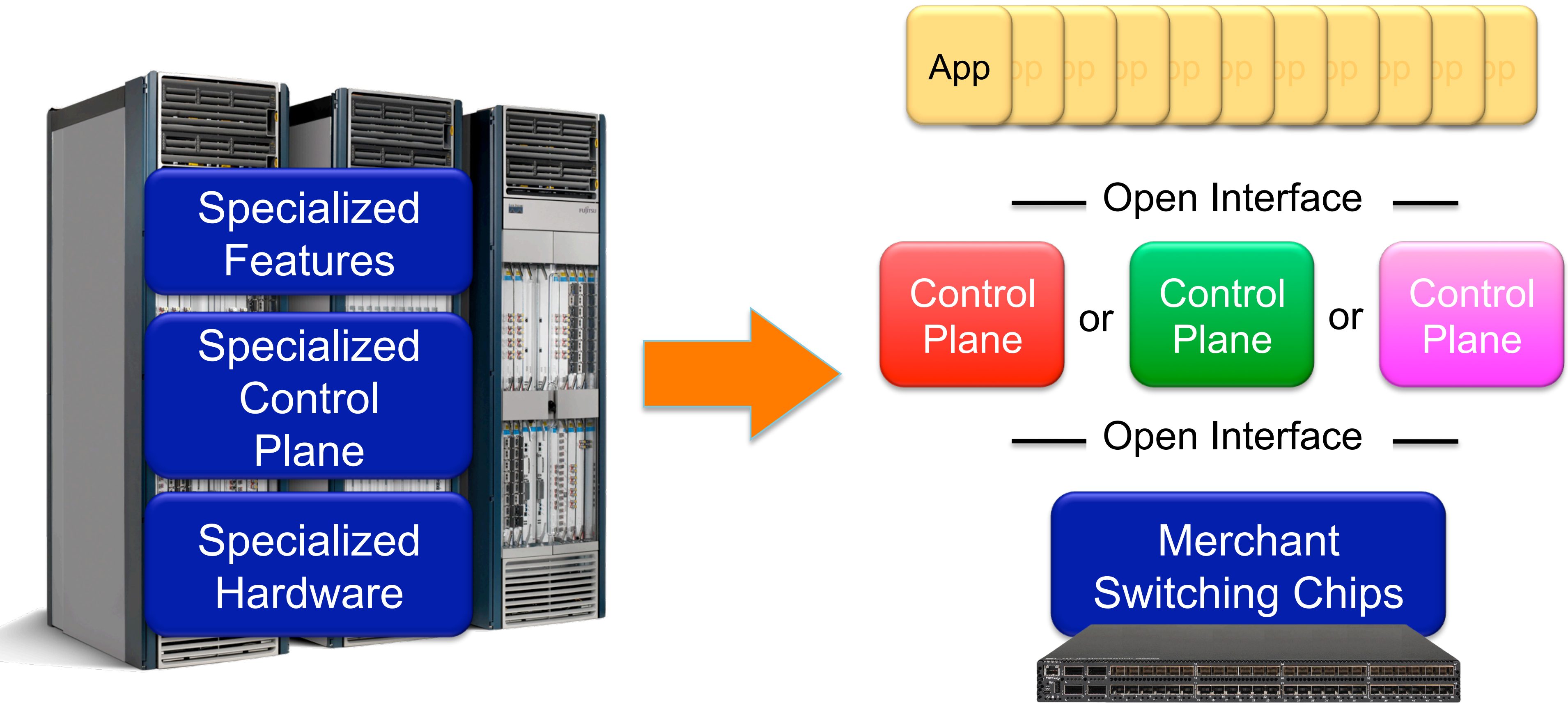
Open Interface

Microprocessor

Vertically integrated
Closed, proprietary
Slow innovation
Small industry

Horizontal
Open interfaces
Rapid innovation
Huge industry

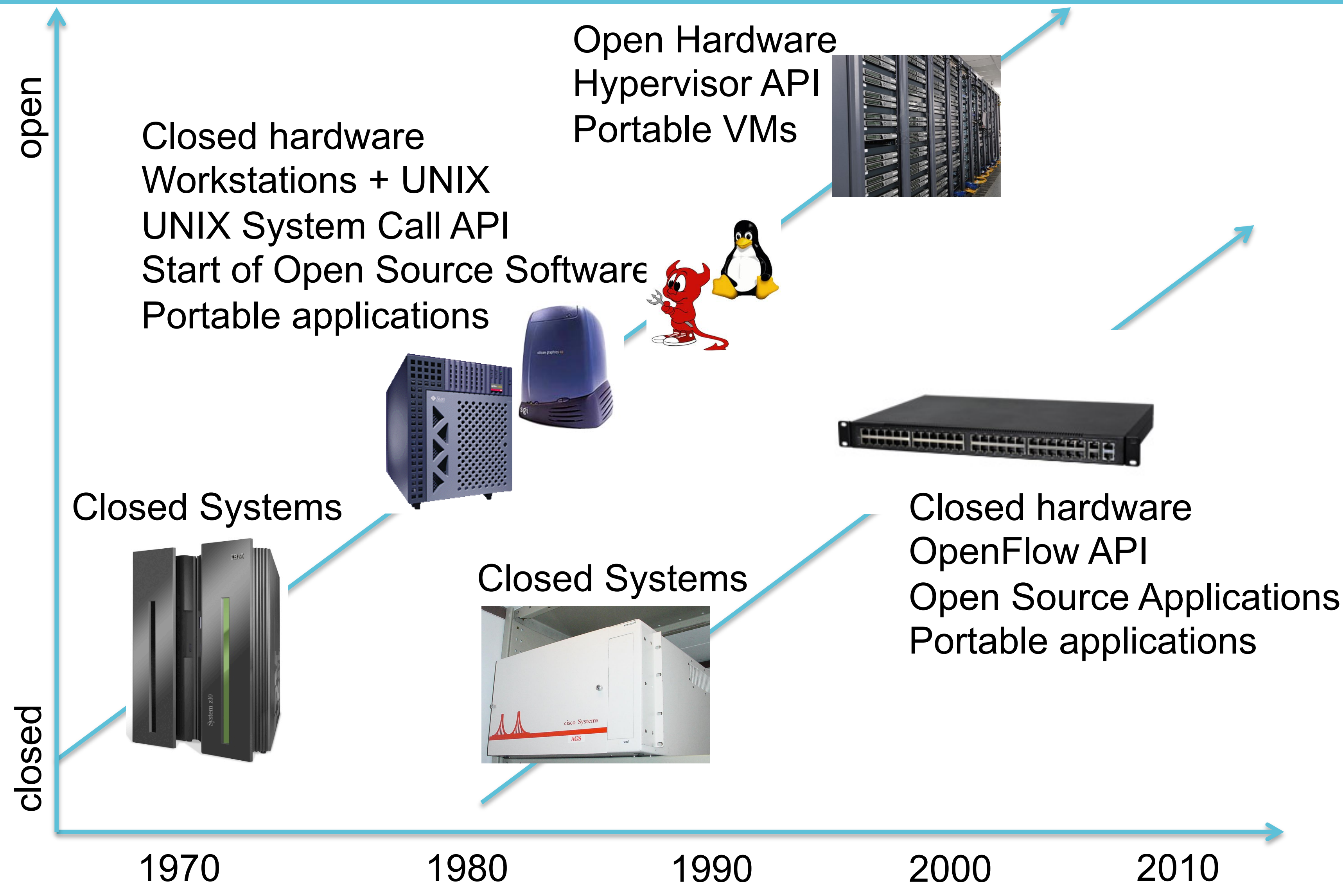
(slide by Nick McKeown, Stanford University)



Vertically integrated
Closed, proprietary
Slow innovation

Horizontal
Open interfaces
Rapid innovation

Computing vs Networking



Outline

Why was OpenFlow developed?

How does OpenFlow work?

OpenFlow Standardisation (ONF)

Some examples of OpenFlow usage

OpenStack Cloud Computing Platform
Google Data Network
Protocol Implementation

Some of the OpenFlow Players

Wrapup

How Does OpenFlow Work?

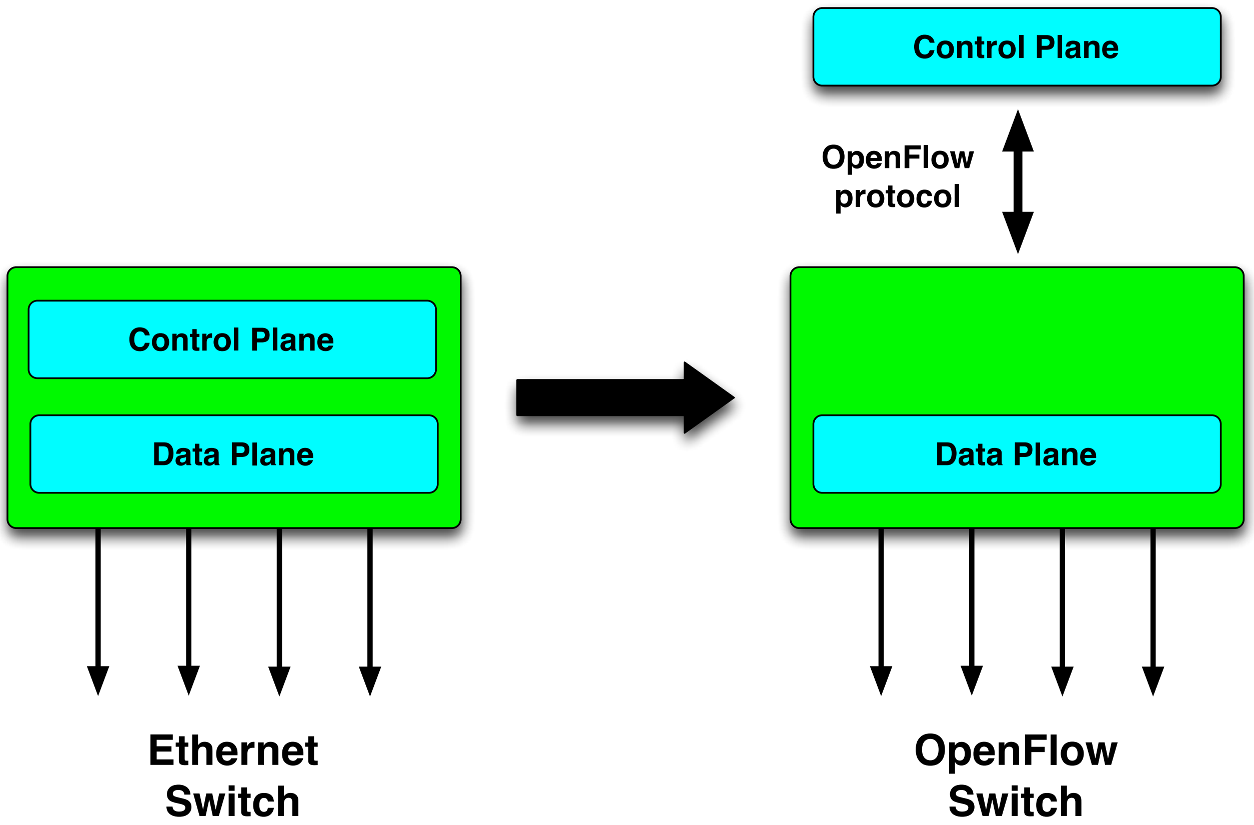
Control Plane moved out of the switch

Standardised protocol between Data Plane and Control Plane → OpenFlow

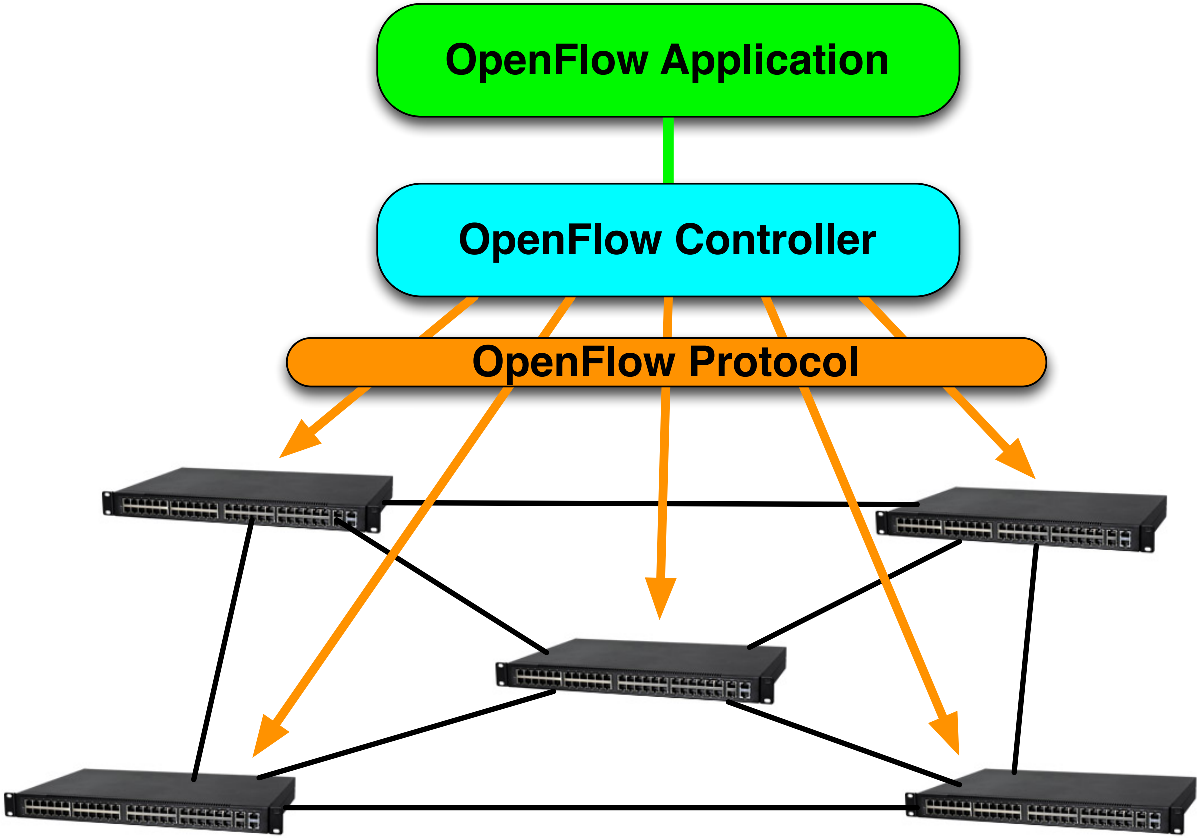
OpenFlow controller typically connects to many switches

Centralised view of the whole network

Data and Control Plane Separation



OpenFlow Controlled Network



OpenFlow Protocol

Insert flow forwarding entries in switches

Send packets to OpenFlow switch data path

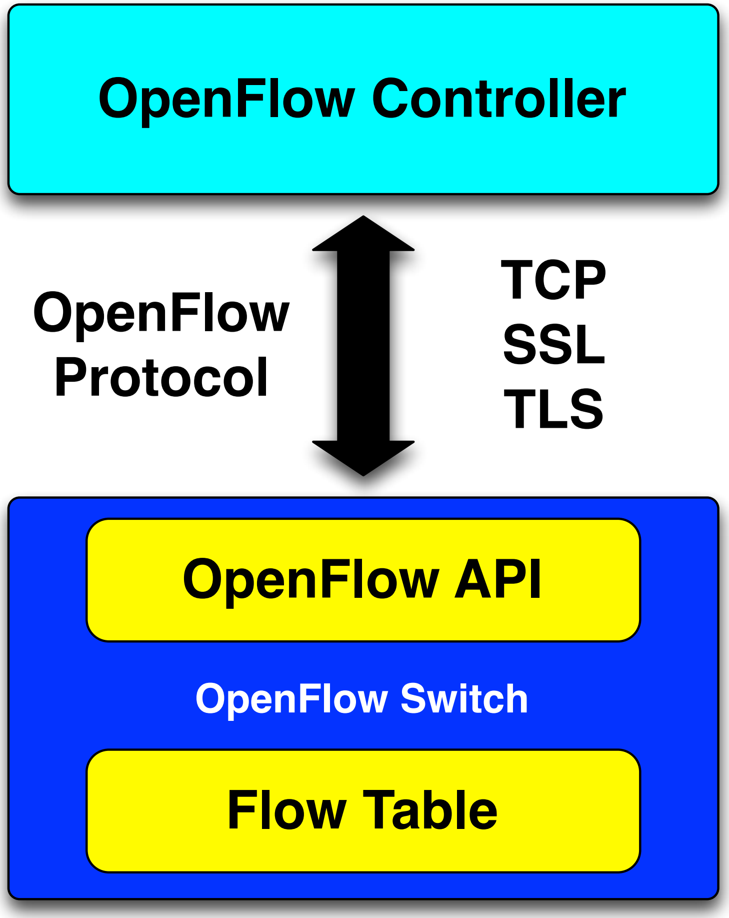
Receive packets from OpenFlow switch data path

Receive data path traffic statistics from OpenFlow switch

Retrieve flow tables from OpenFlow switch

Retrieve parameters from OpenFlow switch
E.g. number of ports

OpenFlow Components



Flow Table

Header Fields	Counters	Actions

Header Fields: match against packets
Counters: count matching packets
Actions: Actions to take when packet matches

Header Matching (OF 1.0)

Ingress port
Ethernet source/destination address
Ethernet type
VLAN ID
VLAN priority
IPv4 source/destination address
IPv4 protocol number
IPv4 type of service
TCP/UDP source/destination port
ICMP type/code

Counters (1/3)

Per table:

Active entries (32 bits)

Packet lookups (64bits)

Packet matches (64 bits)

Per flow:

Received packets (64 bits)

Received bytes (64 bits)

Duration <seconds> (32 bits)

Duration <nanoseconds> (32 bits)

Counters (2/3)

Per port:

Received/Transmitted packets (64 bits)
Received/Transmitted bytes (64 bits)
Receive/Transmit drops (64 bits)
Receive/Transmit errors (64 bits)
Receive frame alignment errors (64 bits)
Receive overrun errors (64 bits)
Receive CRC errors (64 bits)
Collisions

Counters (3/3)

Per queue:

Transmit packets (64 bits)
Transmit bytes (64 bits)
Transmit overrun errors (64 bits)

Actions

Forward
Required: All, Controller, Local, Table, IN_PORT
Optional: Normal, Flood

Enqueue (Optional)

Drop (Required)

Modify Field (Optional)

Required Forward Actions

All

Sent packet out on all interfaces, not including incoming interface

Controller

Encapsulate and send the packet to the controller

Local

Send the packet to the switch local network stack

Table

Perform action in flow table (for packet_out)

IN_PORT

Send the packet out to the input port

Optional Forward Actions

Normal

Process the packet using the traditional forwarding path supported by the switch

Flood

Flood the packet along the spanning tree, not including the incoming interface

Optional Modify Field Action

Set VLAN ID (or add VLAN tag)
Set VLAN priority
Strip VLAN header
Modify Ethernet source/destination address
Modify IPv4 source/destination address
Modify IPv4 type of service bits
Modify TCP/UDP source/destination port

Flow Insertion

Proactive
Flow entries are inserted in the OpenFlow switches before packets arrive

Reactive
Packets arriving at an OpenFlow switch without a matching flow entry are sent to OpenFlow controller. They examined by the controller after which flow entries are inserted in the switches

Example of Proactive Flow Entries

Forward all packets between port 1 and 2

```
ovs-ofctl add-flow br0 in_port=1,actions=output:2
ovs-ofctl add-flow br0 in_port=2,actions=output:1
```

Forward all packets between access port 4 and trunk port 6 using VLAN ID 42

```
ovs-ofctl add-flow br0 in_port=4,actions=output:6,mod_vlan_id:42
ovs-ofctl add-flow br0 in_port=6,actions=output:4,strip_vlan
```

Outline

Why was OpenFlow developed?

How does OpenFlow work?

OpenFlow Standardisation (ONF)

Some examples of OpenFlow usage

OpenStack Cloud Computing Platform
Google Data Network
Protocol Implementation

Some of the OpenFlow Players

Wrapup

OpenFlow Standardisation

Open Networking Foundation (ONF)

Non-Profit consortium

Founded in March 2011 by Deutsche Telecom, Facebook, Google, Microsoft, Verizon and Yahoo!

Mission: promotion of Software Defined Networking (SDN)

OpenFlow Protocol Standards

OpenFlow 1.0.0 (December 2009)
Most widely used version
OpenFlow 1.1.0 (February 2011)
OpenFlow 1.2 (December 2011)
IPv6 support, extensible matches
OpenFlow 1.3.0 (June 2012)
Flexible table miss, per flow meters, PBB support
OpenFlow 1.3.0 (June 2012)

OF-Config 1.0 (December 2011)
OF-Config 1.1 (January 2012)
OF-Config 1.1.1 (March 2013)

OpenFlow Test
Interoperability Event technical papers

Outline

Why was OpenFlow developed?

How does OpenFlow work?

OpenFlow Standardisation (ONF)

Some examples of OpenFlow usage
OpenStack Cloud Computing Platform
Google Data Network
Protocol Implementation

Some of the OpenFlow Players

Wrapup

OpenStack and OpenFlow

OpenStack is an open source cloud computing platform

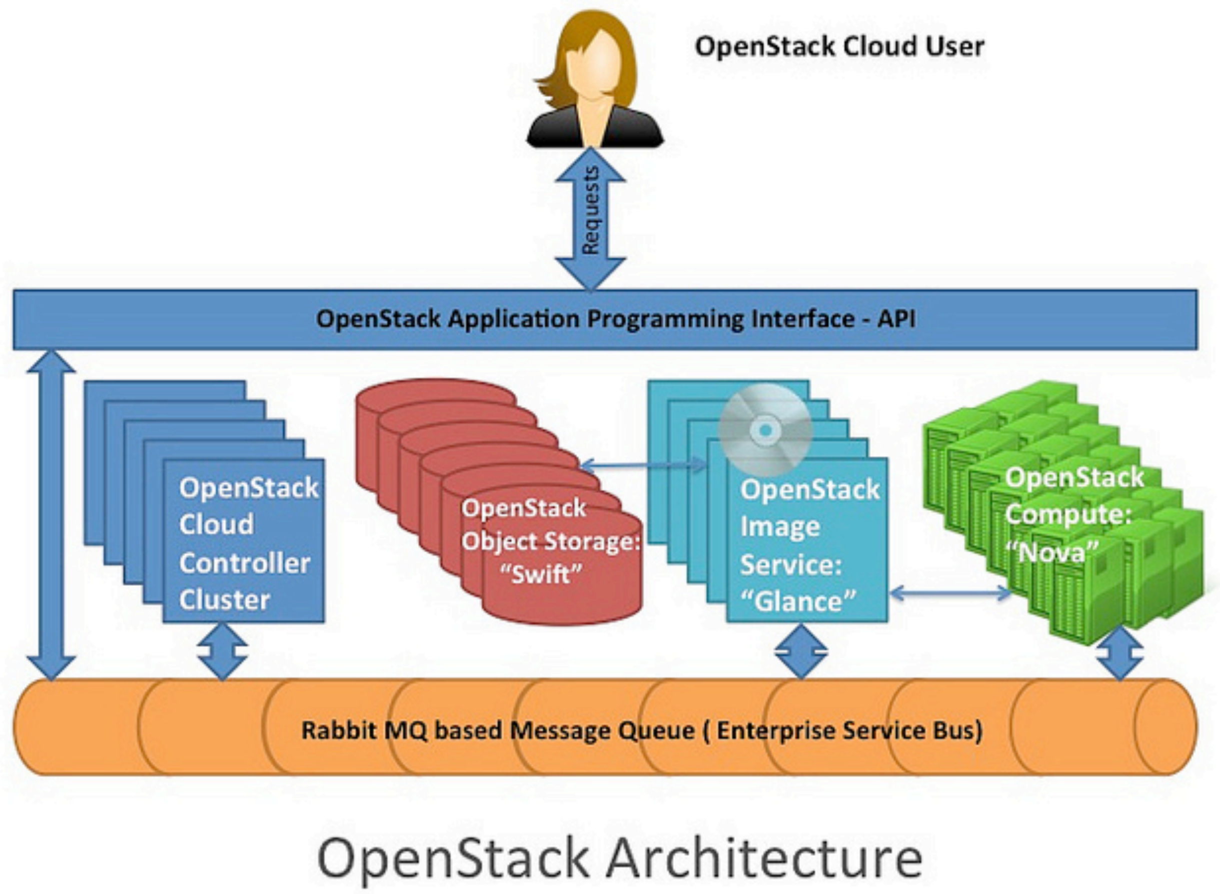
Computing
OpenStack Compute (Nova)
OpenStack Image service (Glance)

Networking
OpenStack Networking (Quantum)

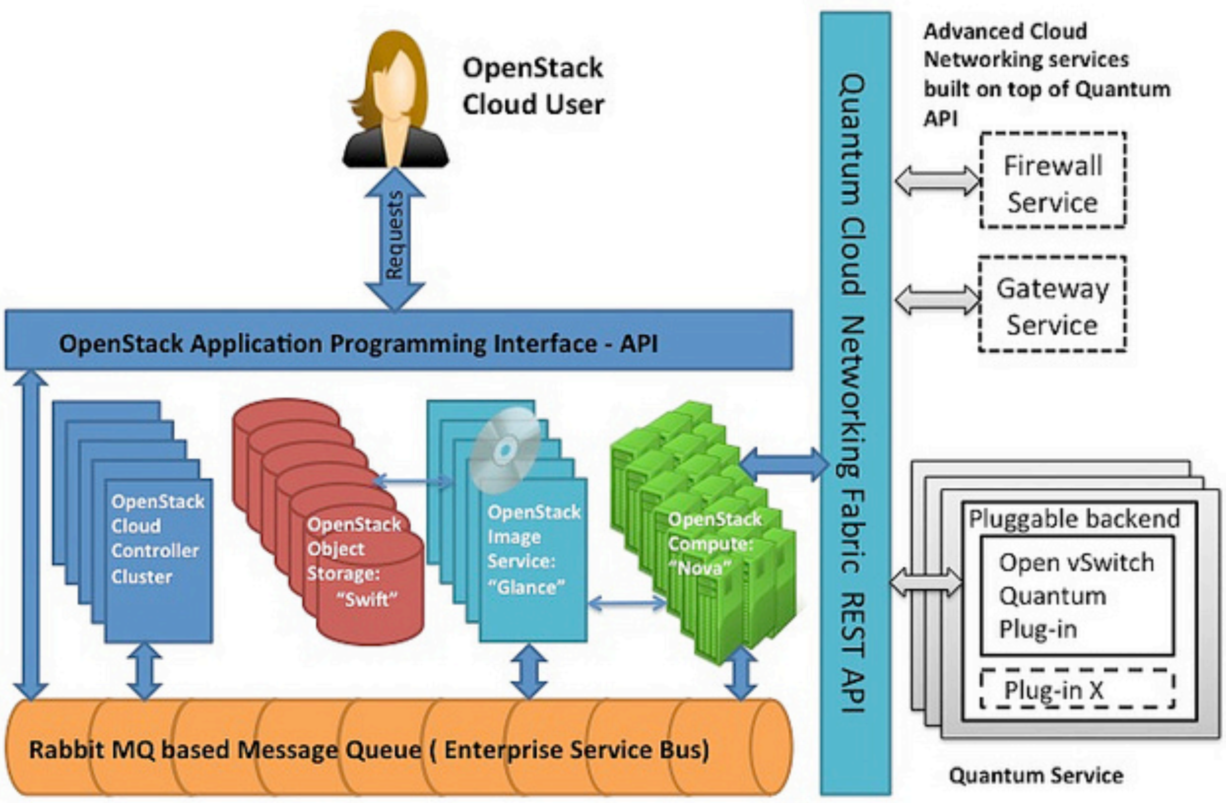
Storing
OpenStack Object Storage (Swift)
OpenStack Block Storage (Cinder)

http://www.openstack.org/

OpenStack Architecture



OpenStack Networking: Quantum



OpenStack + Quantum Integration Architecture

Open vSwitch

Open vSwitch is a software OpenFlow switch

Runs on Linux (kernel 3.3) and FreeBSD
Also used in various hardware OpenFlow switches
Implements OpenFlow version 1.0 with extensions

Developed by Nicira

Nicira was founded in 2007 by Martin Casado (Stanford), Nick McKeown (Stanford) and Scott Shenker (UC Berkeley)
Vmware bought Nicira in 2012 for 1.2 Billion US Dollar

Outline

Why was OpenFlow developed?

How does OpenFlow work?

OpenFlow Standardisation (ONF)

Some examples of OpenFlow usage

OpenStack Cloud Computing Platform
Google Data Network
Protocol Implementation

Some of the OpenFlow Players

Wrapup

Google Data Network

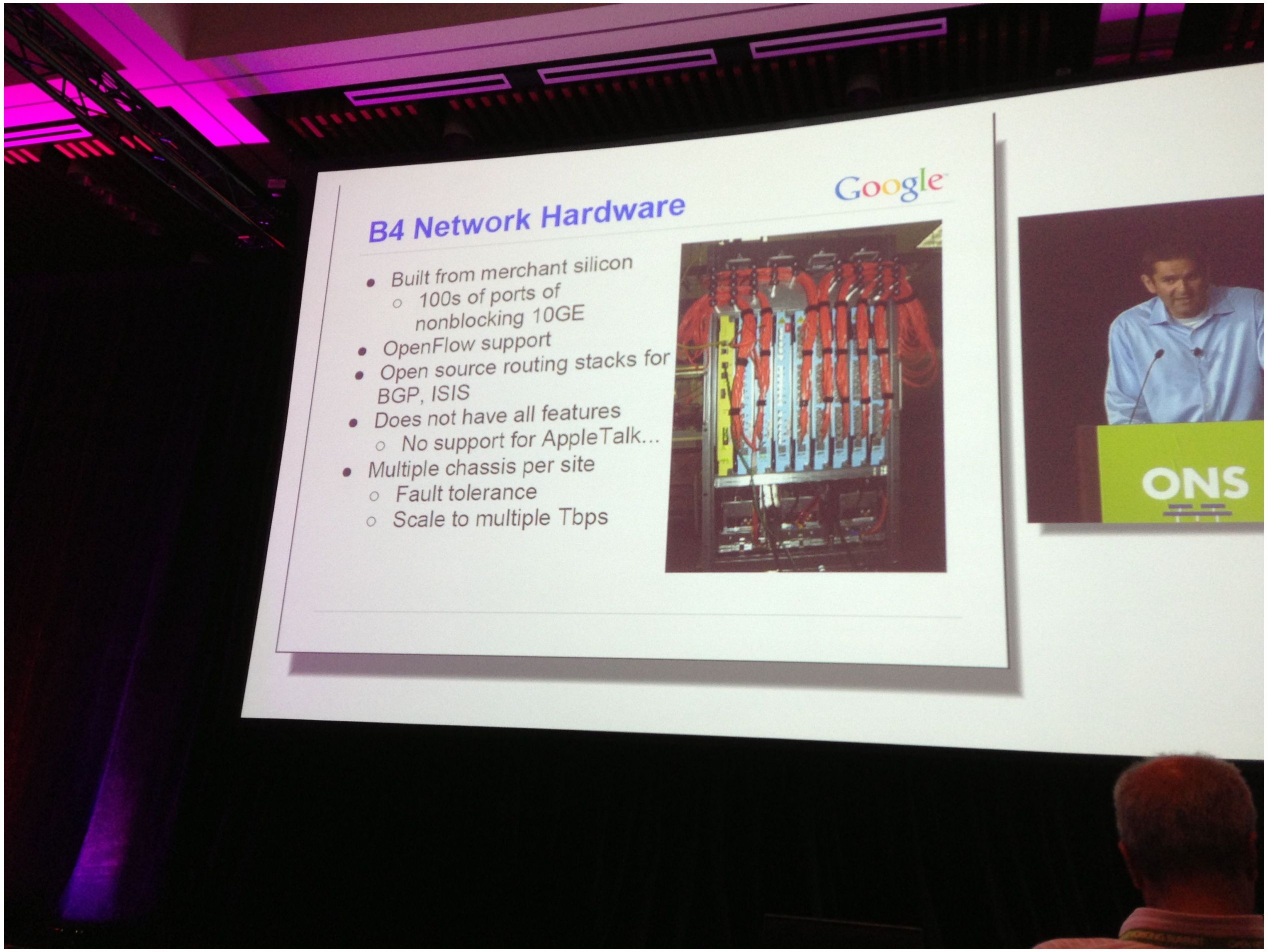
Google has two networks:

I-Scale: User facing services (search, YouTube, Gmail, etc), high SLA
G-Scale: Data centre traffic (intra and inter), lower SLA, perfect for OpenFlow testing

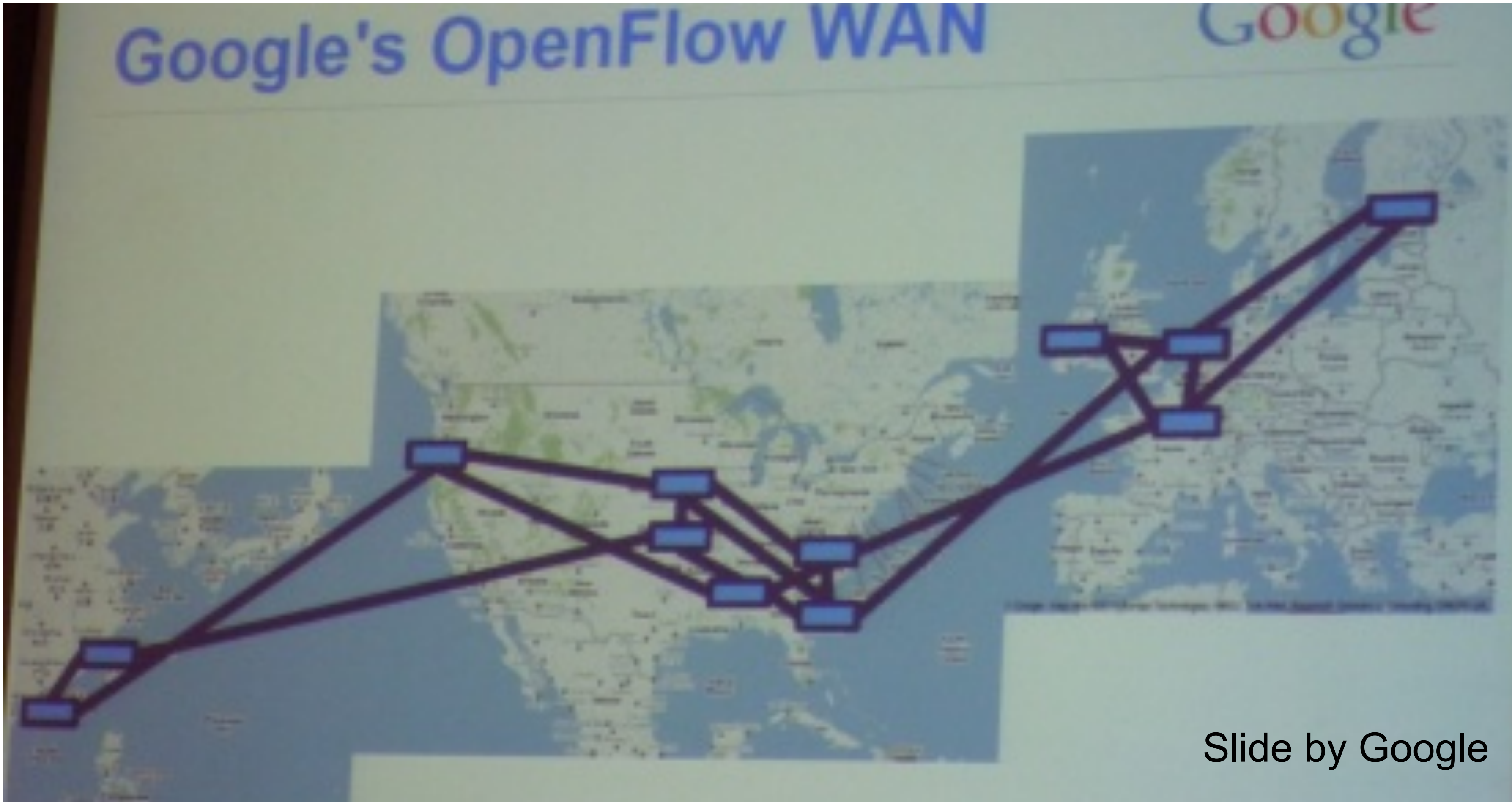
Google uses custom built switches with merchant chip sets (128 ports of 10GE)

Custom build just because such switches were not commercially available yet
Next (commercial) switch will probably have 1000+ ports of 40GE (2013)

Google OpenFlow Switch

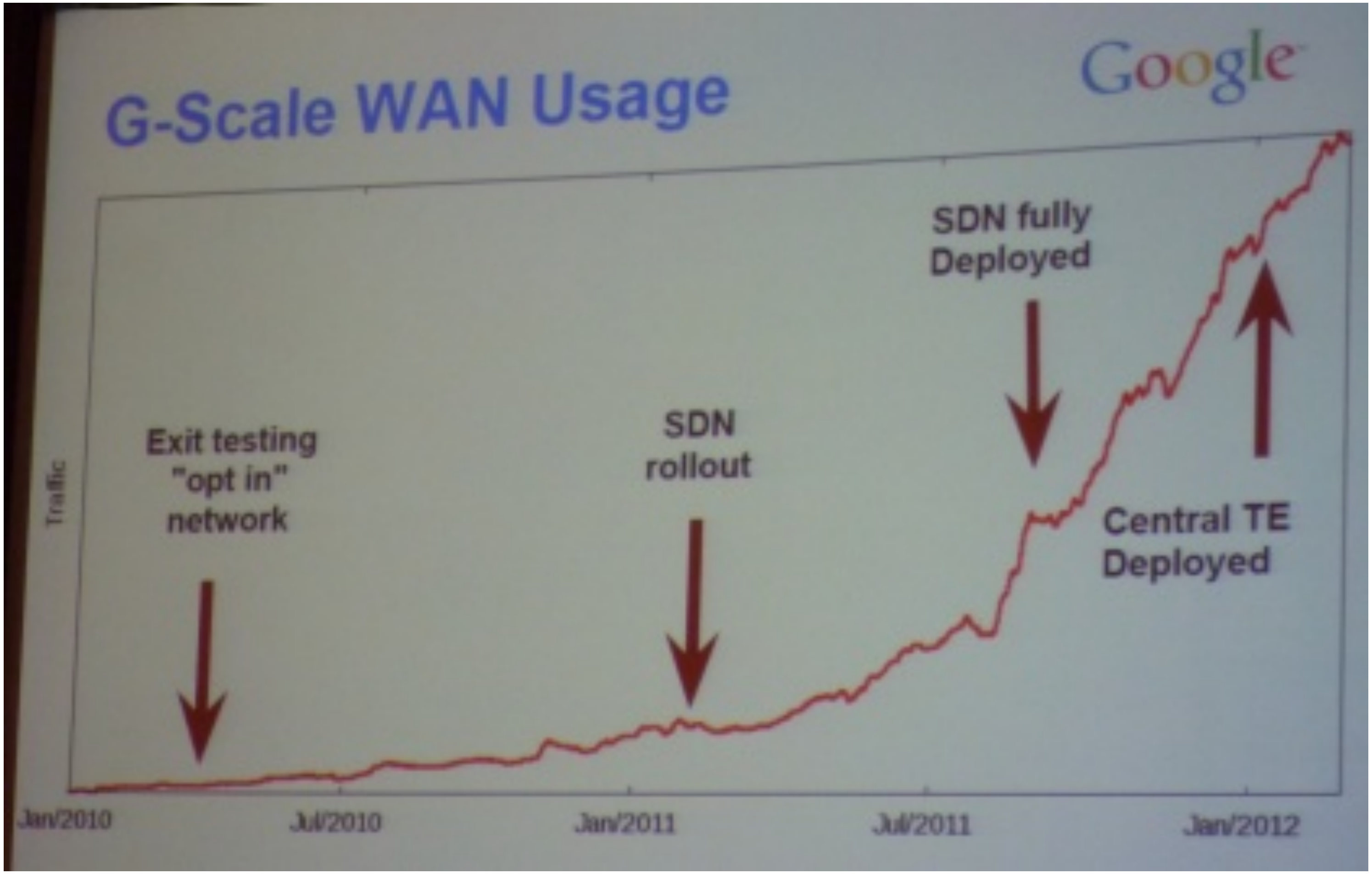


Google Data Network



Slide by Google

Google Data Network



Google Data Network

Multiple controllers

3, 5, 7 with Paxos majority voting

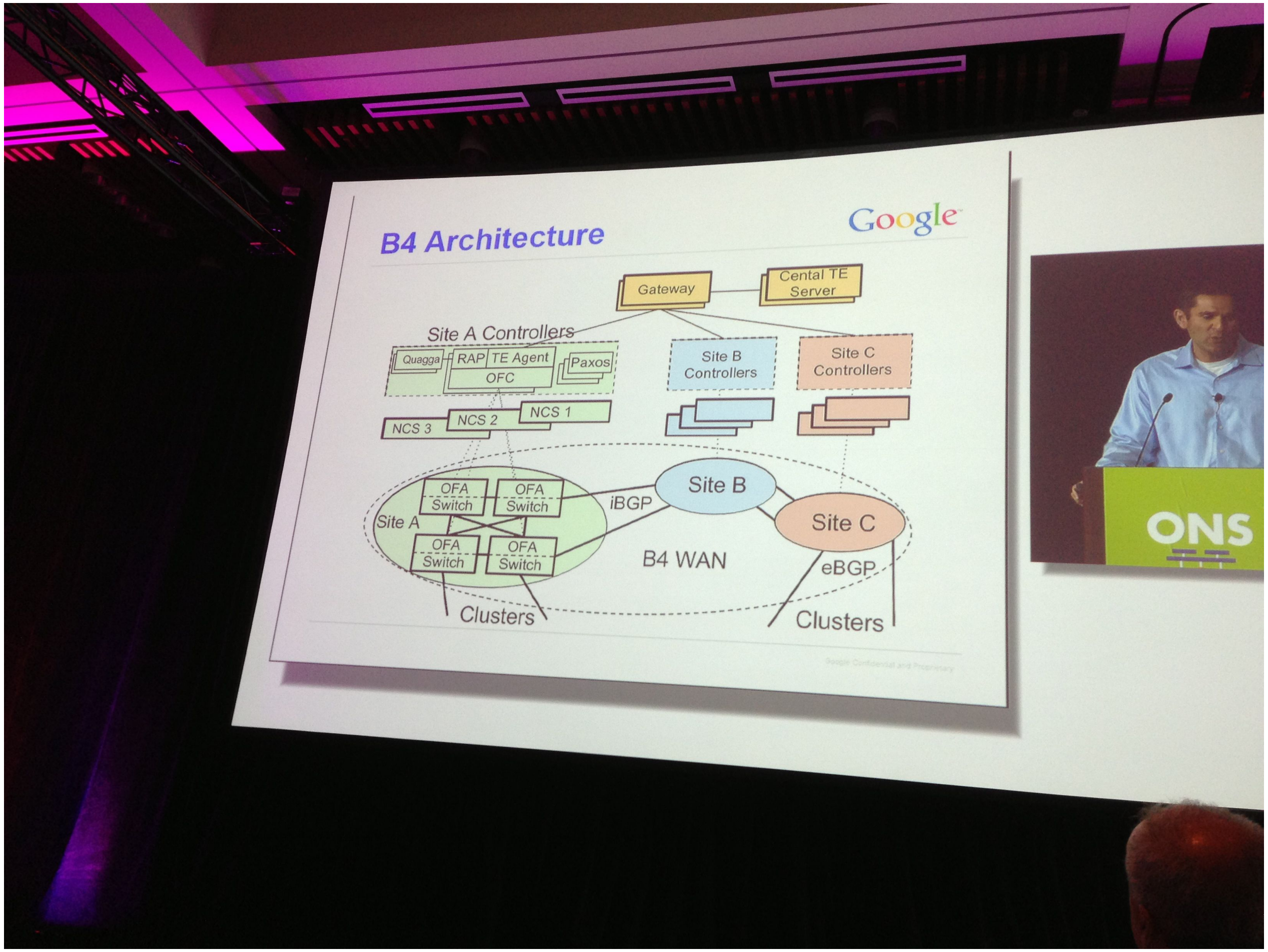
The whole network is emulated in a simulator

New software revisions can be tested in the simulator

Network events (e.g. link down) are sent to production servers + testbed

Testing in simulator but with real network events

Google OpenFlow Architecture



Google Data Network

Experience/benefits:

Software development for a high performance server with modern software tools (debuggers, etc) much easier and faster and produces higher quality software than development for an embedded system (router/switch) with slow CPU and little memory

Centralised Traffic Engineering much faster on a 32 core server (25-50 times as fast)

Almost 100% Link Utilization

Outline

Why was OpenFlow developed?

How does OpenFlow work?

OpenFlow Standardisation (ONF)

Some examples of OpenFlow usage
OpenStack Cloud Computing Platform
Google Data Network
Protocol Implementation

Some of the OpenFlow Players

Wrapup

Protocol Implementation in OpenFlow

Example of how to implement a network protocol in OpenFlow

Implementation of IEEE 802.1ag Ethernet OAM

Added as module to NOX OpenFlow controller

NOX controller sends and receives OAM frames via *packet_in* and *packet_out*

IEEE 802.1ag Ethernet OAM Protocol

Continuity Check (CC)

Detect loss of connectivity
Periodic hello messages from MEPs
Processed by MEPs
CC frames sent to multicast group, no replies are sent

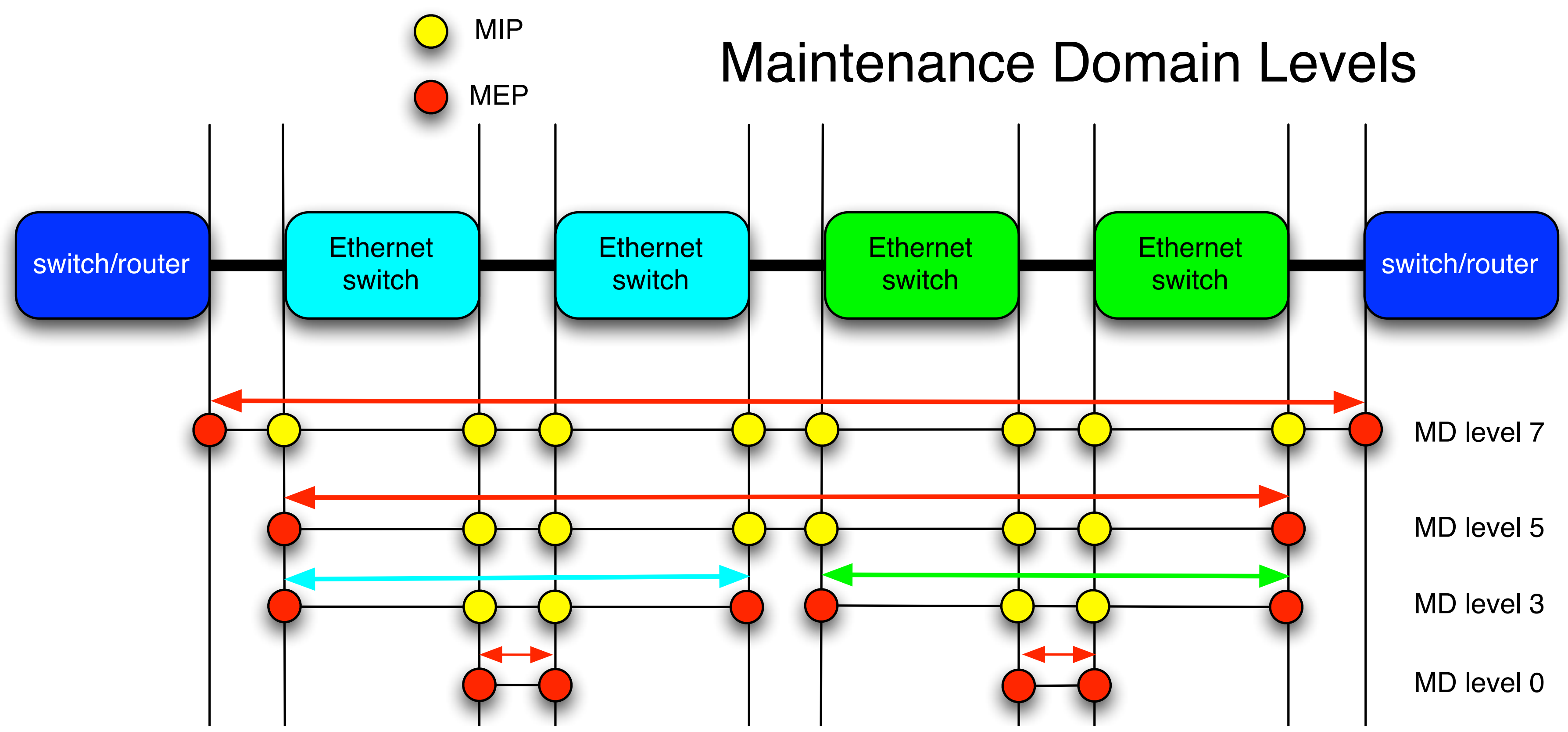
Loopback Message/Reply (LBM/LBR)

Check for reachability
Sent manually from MEPs via CLI
Processed by MIPs/MEPs
Unicast request, unicast reply

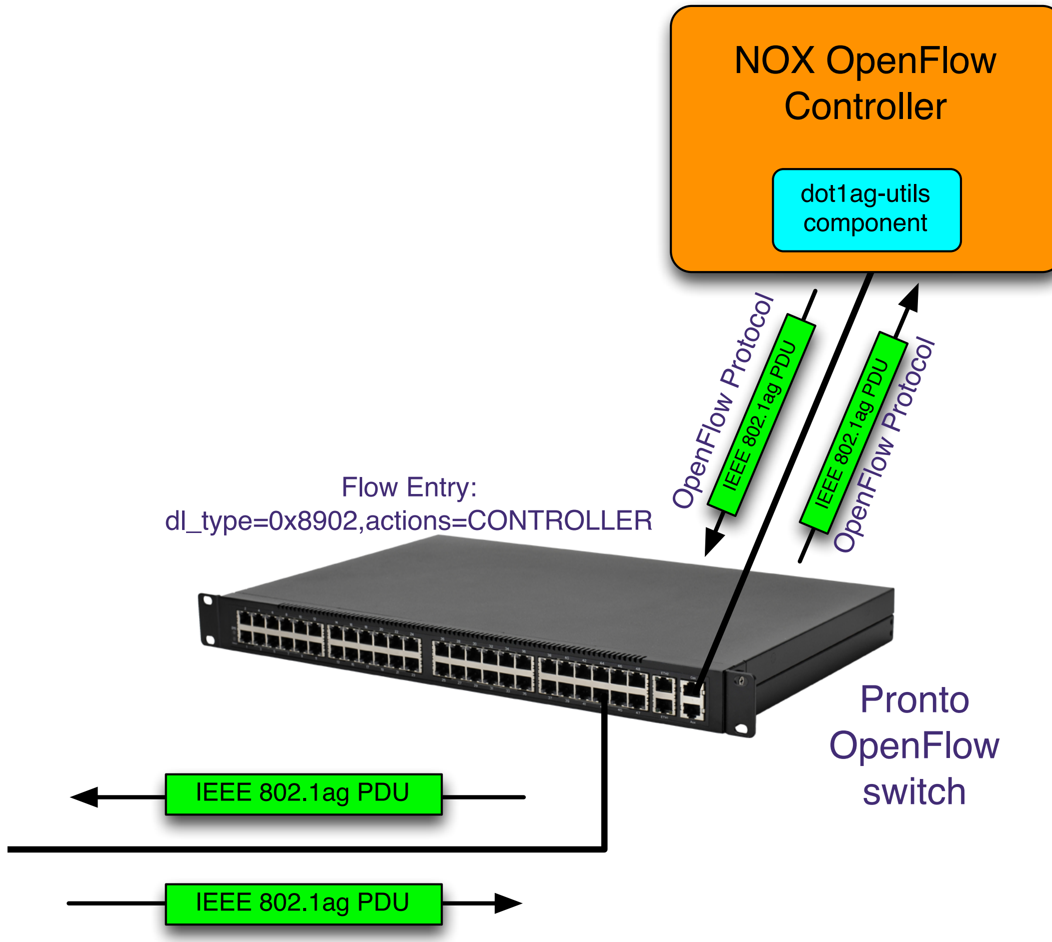
Link Trace Message/Reply (LTM/LTR)

Path information
Sent manually from MEPs via CLI
Processed by MIPs/MEPs in path
Multicast request including TTL, unicast replies

802.1ag Domains



Sending OAM frames via NOX



Outline

Why was OpenFlow developed?

How does OpenFlow work?

OpenFlow Standardisation (ONF)

Some examples of OpenFlow usage
OpenStack Cloud Computing Platform
Google Data Network
Protocol Implementation

Some of the OpenFlow Players

Wrapup

Some OpenFlow Players

Startups

Big Switch Networks
Nicira
Pica8

ONRC

ON.LAB

OpenDaylight consortium

Stanford University Startups

Big Switch Networks (http://www.bigswitch.com/)

FloodLight (open source OpenFlow controller), Big Switch, Big Tap

Guido Appenzeller (CEO/Co-Founder)

- Former head of Stanford University Clean Slate program

Kyle Forster (Co-Founder, ex-Cisco, ex-Joost)

Rob Sherwood (CTO)

Nicira (aquired by VMware) (http://nicira.com/)

Open vSwitch (open source software switch)

Steve Mullaney (CEO)

Martin Casado (CTO, Co-Founder)

- SDN was graduate work at Stanford University, supervised by Nick KcKeown & Scott Shenker

Nick McKeown (Co-Founder)

- Former faculty director of Stanford University Clean Slate program

Scott Shenker (Chief Scientest, Co-Founder)

- University of California at Berkeley

Pica8

Founded in 2008

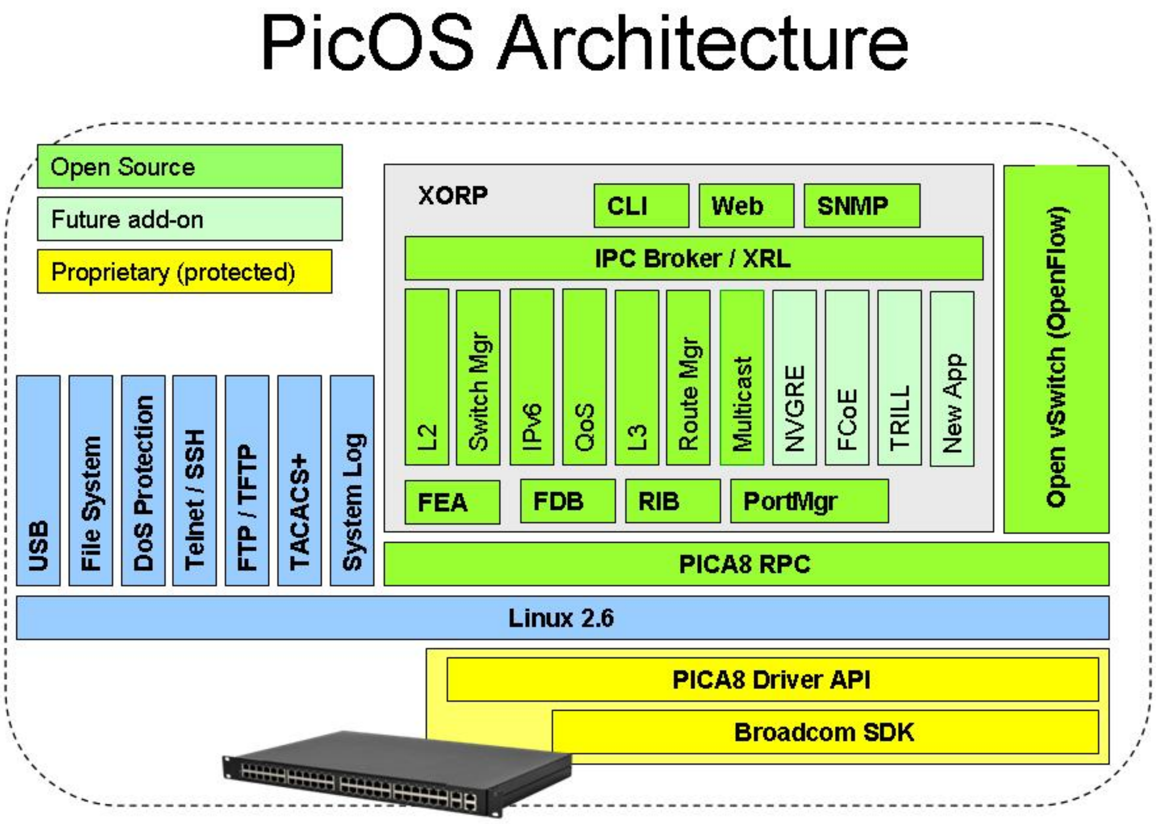
Open the switch and router platforms

High quality software with merchant silicon (Pronto)

PicOS based on:
XORP (open source routing project)
Open vSwitch (open source OpenFlow switch)

http://www.pica8.com/

Pica8 Switch Architecture



Pronto Switches

Pronto 3290

48x 10/100/1000 BASE-T RJ45 & 4x 10GE SFP+

USD 2,750

Pronto 3780

48x 10GE SFP+

USD 9,950

Pronto 3920

48x 10GE SFP+ & 4x 40GE QSFP

USD 13,500

Open Networking Research Center

Located at Stanford University & UC Berkeley

Sponsors: CableLabs, Cisco, Ericsson, Google, Hewlett Packard, Huawei, Intel, Juniper, NEC, NTT Docomo, Texas Instruments, Vmware

People:
Nick McKeown @ Stanford University
Scott Shenker @ UC Berkeley

http://onrc.stanford.edu/

ON.LAB

Headed by Guru Parulkar

Professor at Stanford University

Build open source OpenFlow tools and platforms

Beacon, NOX, FlowVisor, Mininet

http://onlab.us/

OpenDaylight

Announced during Open Networking Summit in April 2013

Community-led, industry-supported open source SDN framework

Software hosted by Linux Foundation

Supported by big names: Cisco, Juniper, IBM, Microsoft, Redhat

http://www.opendaylight.org/

Outline

Why was OpenFlow developed?

How does OpenFlow work?

OpenFlow Standardisation (ONF)

Some examples of OpenFlow usage
OpenStack Cloud Computing Platform
Google Data Network
Protocol Implementation

Some of the OpenFlow Players

Wrapup

Wrapup

OpenFlow has got a lot of attention in 2011/2012

Possible **disruptive (network) technology (time will tell)**
New networking paradigm
Could be the start of an open hardware/open software network ecosystem

OpenFlow is a protocol to interact with switch forwarding table

Used in OpenStack cloud computing and Google network
Network protocols can be implemented as OpenFlow application

Several successful startups

Open source

Ronal van der Pol
Ronald.vanderPol@SURFnet.nl

WHAT SURF CAN DO