



# Control Plane Scalability in Software Defined Networking

**Eugen Borcoci**

University POLITEHNICA Bucharest,

*Eugen.Borcoci@elcom.pub.ro*



# Control Plane Scalability in Software Defined Networking



## Acknowledgement

The **overview (State of the Art) part is compiled**, based on several public original documents and other studies, belonging to different authors' and groups work: SDN Future Internet conferences public material, research papers and projects, overviews, tutorials, etc.: (see Reference list).

The **ALICANTE –project** examples of content/media –oriented architecture having SDN similarities is presented with permission of the ALICANTE Consortium.

This work has been partially supported by **ALICANTE FP7 IP Project 248652: “Media Ecosystem Deployment Through Ubiquitous Content-Aware Network Environments”**, 2010-2013.



# Control Plane Scalability in Software Defined Networking



## Motivation of this talk

- **Future Internet challenges** -> need to solve the current Internet limitation and ossification- as to support global integration of various forms of communications
  - **Evolutionary approach**
  - **Clean slate approach**
  - **Combined solutions**
  - **Novel significant trends**
    - Software Defined Networking (SDN), Software Defined Internet Architectures (SDIA)**
    - Cloud computing (can use SDN approach)**



# Control Plane Scalability in Software Defined Networking



## Motivation of this talk (cont'd)

- **Software Defined Networking (SDN) architecture**
- **SDN major features:**
  - Separation of the control plane from the data plane.
  - A centralized control and view of the network
    - underlying network infrastructure is abstracted from the applications.
  - Open interfaces between the control plane (controllers) and in data plane elements.
  - Programmability of the network by external applications- including network management and control
- *OpenFlow* : typical (vertical) protocol for communication between DPI and CPI
- **Open issue: Centralization, CPI/DPI decoupling, etc. → scalability problems- target of this talk**



# CONTENTS

---




1. Software Defined Networking Architecture (Summary)
2. SDN-OpenFlow
3. Control Plane Scalability in SDN
4. SDN-like architecture example : ALICANTE Project
5. Conclusions



# CONTENTS



1.  **Software Defined Networking Architecture (Summary)**
2. SDN-OpenFlow
3. SDN Control Plane Scalability
4. SDN-like architecture example : ALICANTE Project
5. Conclusions



# 1. Software Defined Networking Architecture



- **1.1 Introduction**
- Recent industry/research effort resulted in new approaches:
  - **Software- Defined Networking (SDN)** –new networking architecture
  - **Open Networking Foundation** (ONF- non-profit industry consortium ) → several OpenFlow I/F specifications for SDN
- **Promises for enterprises and carriers :**
  - higher programmability opportunities, automation, and network control
  - enabling them to build highly scalable, flexible networks
  - fast adapt to changing business needs
- *Source: Software-Defined Networking: The New Norm for Networks ONF White Paper April 13, 2012*
- **Note:**
  - ***traditional TCP/IP networking control : fully distributed***
  - ***SDN : more centralized (at least logical)***



# 1. Software Defined Networking Architecture



## ■ 1.1 Introduction

### ■ SDN + OpenFlow I/F (first standard) advantages:

- *high-performance, granular traffic control* across multiple vendors' network devices
- *centralized management and control* of networking devices improving automation and management
- *common APIs abstracting the underlying networking details* from the orchestration and provisioning systems and applications;
- *flexibility*: new network capabilities and services with no need to configure individual devices or wait for vendor releases
- *programmability* by operators, enterprises, independent software vendors, and users (not just equipment manufacturers) using common programming environments
- *Increased network reliability* and *security* as a result of centralized and automated management of network devices, uniform policy enforcement, and fewer configuration errors





# 1. Software Defined Networking Architecture



- **1.1 Introduction**
- **SDN + OpenFlow advantages (cont'd):**
  - *more granular network control* with the ability to apply comprehensive and wide-ranging policies at the session, user, device, and application levels
  - *better end-user experience* as applications exploit centralized network state information to seamlessly adapt network behavior to user needs
  - *protects existing investments* while future-proofing the network
  - **With SDN, today's static network can evolve into an extensible service delivery platform capable of responding rapidly to changing business, end-user, and market needs.**



# 1. Software Defined Networking Architecture



- **1.2 Early SDN products**
- Early SDN **products and activities examples**
  - 2008: Software-Defined Networking (SDN) : NOX Network Operating System [Nicira]; OpenFlow switch interface [Stanford/Nicira]
- **Open Networking Foundation (2011)**
  - <https://www.opennetworking.org/>
- **2013 status and activities**
  - **Membership surpassed 100 companies**
  - Published *OpenFlow* Switch Specification 1.3.2
  - Published *OpenFlow Configuration and Management Protocol* 1.1.1 (OF-Config 1.1.1)
  - Approved OpenFlow Switch Specification 1.4
  - Near completion of OpenFlow Switch Specification 1.5
  - Launched the OpenFlow Software Driver Competition
  - Created the ONF Chipmakers Advisory Board (CAB)
  - Created the Research Associates Program
  - Launched the OpenFlow Conformance Testing Program



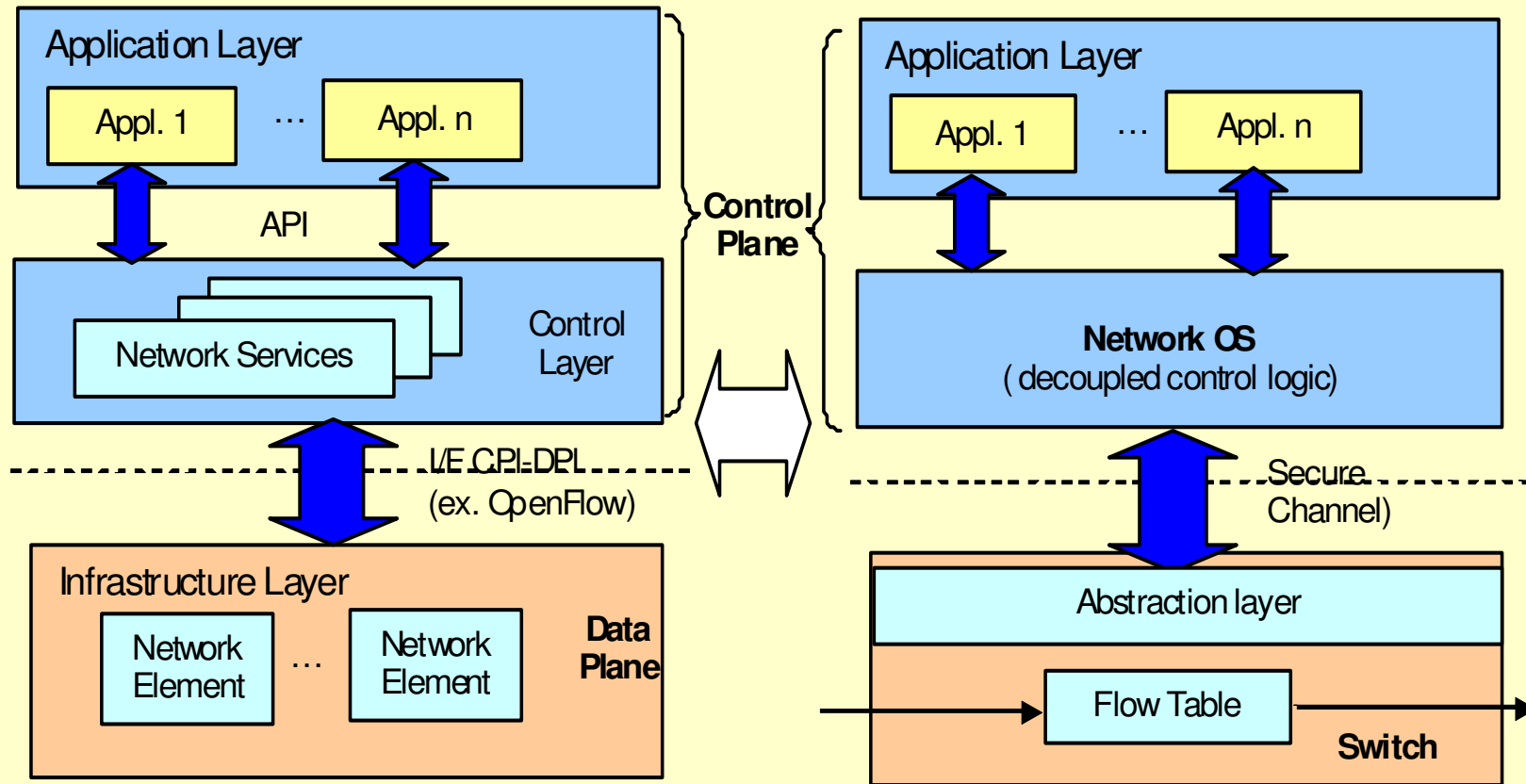
# 1. Software Defined Networking Architecture



## ■ 1.3 SDN Basic Architecture

- **Evolutionary**
- **CPI and DPI are separated**
- Network intelligence is (logically) centralized in SW-based SDN controllers, which maintain a global view of the network.
- Execute CPI SW on general purpose HW
  - Decoupled from specific networking HW
  - CPI can use commodity servers
- Data Plane (DPI ) is programmable
- Maintain, control and program data plane state from a central entity
- **The architecture defines the control for a network** (and not for a network device) The network appears to the applications and policy engines as a single, logical switch
- This simplified network abstraction can be efficiently programmed

## 1.3 SDN Basic Architecture



**SDN Generic Architecture**



# 1. Software Defined Networking Architecture



- **1.3 SDN Basic Architecture**
- **Control Plane**
  - **Control Applications/Program**
    - operates on view of network :
    - performs different functions ( routing, traffic engineering, QoS, security, etc.)
    - **Input:** global network view (graph/database)
    - **Output:** configuration of each network device
    - Control program –seen as whole could be not a distributed system  
Abstraction hides details of distributed state
  - **Network OS:** distributed system creating a consistent, global and up-to-date network view
    - In SDN it runs can on controllers (servers) in the network
    - It creates the “lower layer” of the Control Plane
    - Examples: NOX, ONIX, Trema, Beacon, Maestro, ...
- **Data Plane** : forwarders/switches ( **Forwarding elements -FE**)
  - NOS uses some abstraction to:
    - Get state information from FE
    - Give control directives to FE



# 1. Software Defined Networking Architecture



- **1.3 SDN Basic Architecture**
- **Advantages**
- **Centralization allows:**
  - To **alter network behavior in real-time** and faster deploy new applications and network services (hours, days not weeks or months as today).
  - **flexibility** to configure, manage, secure, and optimize network resources via dynamic, automated SDN programs ( not waiting for vendors) .
- **APIs facilitate implementation of:**
  - **common network services:** routing, multicast, security, access control, bandwidth management, QoS, traffic engineering, processor and storage optimization, energy usage
  - **policy management**, custom tailored to meet business objectives
    - Easy to define and enforce consistent policies across both wired and wireless connections on a campus
- Manage the entire network : intelligent orchestration and provisioning systems



# 1. Software Defined Networking Architecture



## ■ 1.3 SDN Basic Architecture

### ■ Advantages (cont'd)

- ONF studies **open APIs** to promote **multi-vendor management**:
  - possibility for **on-demand resource allocation, self-service provisioning**, truly virtualized networking, and secure cloud services.
- SDN control and applications layers, business apps can operate on an **abstraction of the network**, leveraging network services and capabilities without being tied to the details of their implementation.

### ■ **Open SDN issues/problems**

- Balance between distribution – centralization ( physical/logical)
- **Scalability**
  - Controller scalability (w.r.t. processing power)
  - Communication Control Plane- Data plane
  - Multiple controllers
    - How many controllers
    - CPI topology, controller location, inter-controller communication
    - Consistency, Synchronization
- Reliability (single points of failures?)

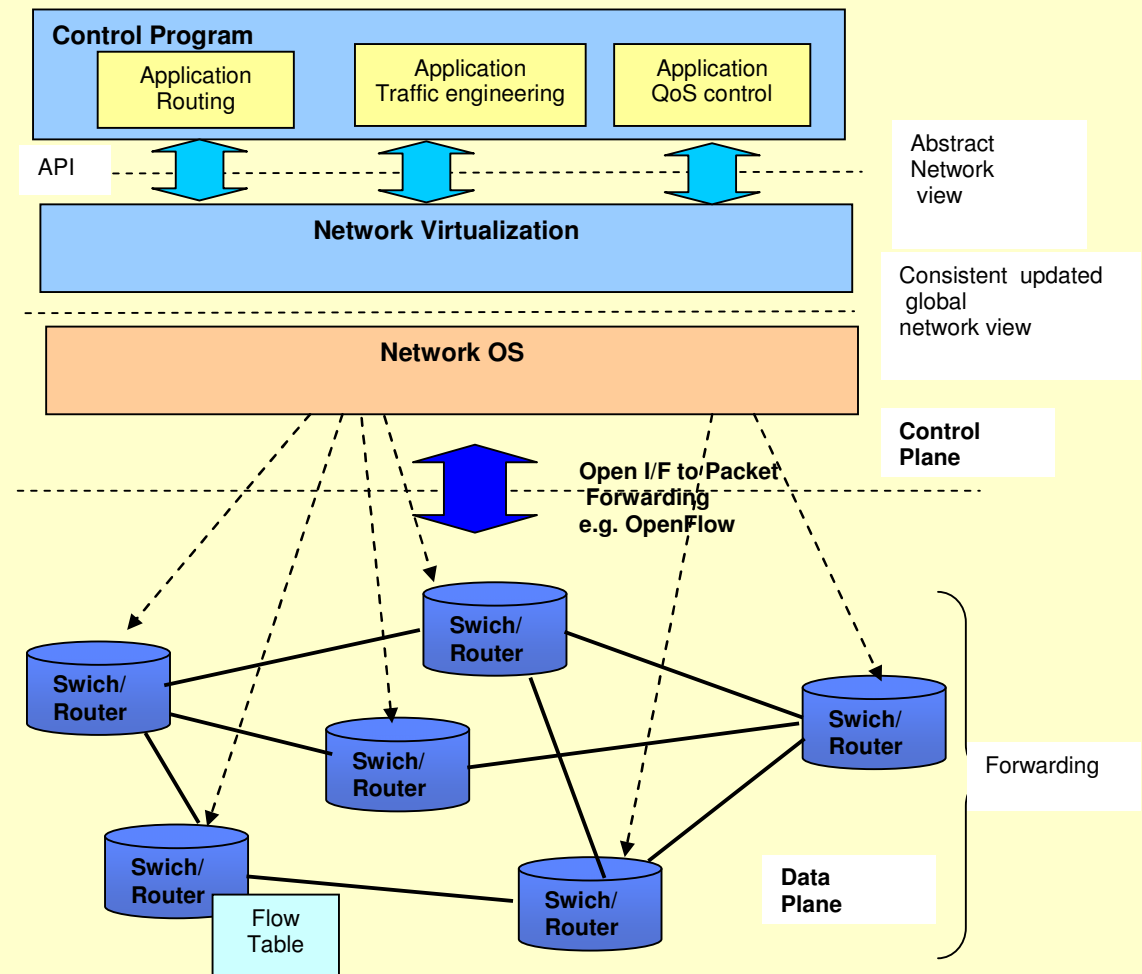
## 1.3 SDN Basic Architecture

### Network OS:

- Distributed system that creates a consistent, updated network view
- Executed on servers (controllers) in the network
- Examples: NOX, ONIX, HyperFlow, Floodlight, Trema, Kandoo, Beacon, Maestro,..

### Uses forwarding abstraction in order to:

- Collect state information from forwarding nodes
- Generate commands to forwarding nodes








# CONTENTS



1. Software Defined Networks Architecture (Summary)
2.  SDN-OpenFlow
3. Control Plane Scalability SDN
4. SDN-like architecture example : ALICANTE Project
5. Conclusions



## 2.SDN - OpenFlow



### ■ OpenFlow Summary

- the **first SDN standard** communications: CPI-DPI I/F
- allows direct access to the Fwd. Plane of network devices (switches and routers), both physical and virtual (hypervisor-based)
- allows to move network control out of the networking switches to logically centralized control software
- can be compared to the instruction set of a CPU
- specifies basic primitives to be used by an external SW application to program the FwdPI (~ instruction set of a CPU would program a computer system)



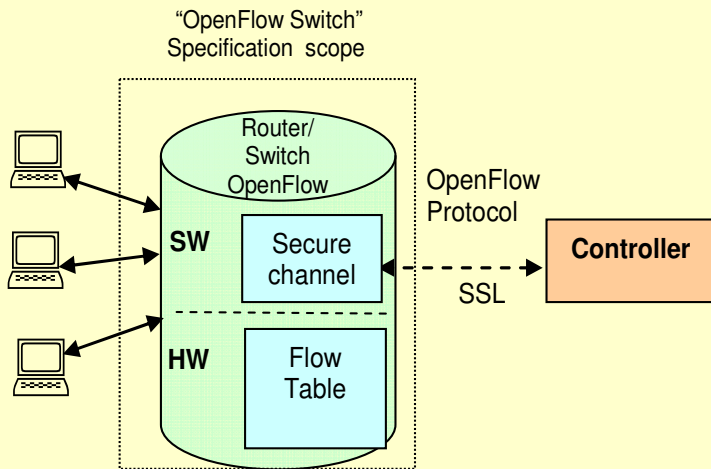
## 2. SDN- OpenFlow



### ■ OpenFlow Summary

- uses the *concept of flows* to identify network traffic based on pre-defined match rules that can be statically or dynamically programmed by the SDN control SW
- allows IT admin to define how traffic should flow through network devices based on parameters such as usage patterns, applications, and cloud resources
- allows the network to be programmed on a per-flow basis ( provides – if wanted- extremely granular control), enabling the network to respond to real-time changes at the application, user, and session levels

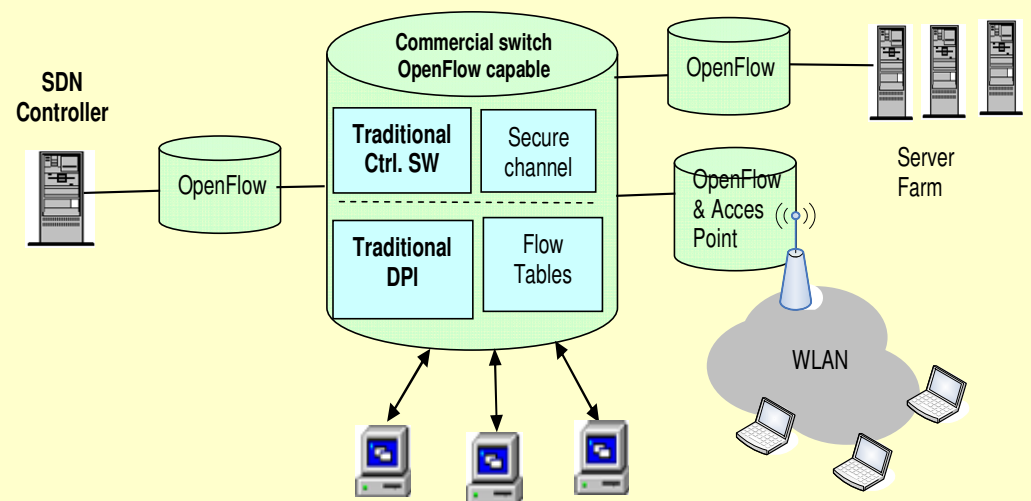
- **Open Flow Summary**
- **Source :** *“OpenFlow: Enabling Innovation in Campus Networks”- N.McKeown, T.Anderson, H.Balakrishnan, G.Parulkar, L.Peterson, J.Rexford, S.Shenker, J.Turner*



Ref1: Figure 1: Idealized OpenFlow Switch. The Flow Table is controlled by a remote controller via the Secure Channel.

In Port	VLAN ID	Ethernet			IP			TCP	
		SA	DA	Type	SA	DA	Proto	Src	Dst

Table 1: The header fields matched in a “Type 0” OpenFlow switch.



Ref1: Figure 2: Example of a network of OpenFlow-enabled commercial switches and routers.



## 2. SDN-OpenFlow



- **OpenFlow Summary**
- Available Software Switch Platforms
- SDN software switches
  - can be used to run a SDN testbed or when developing services over SDN.

Software Switch	Implementation	Overview	Version
Open vSwitch [14]	C/Python	Open source software switch that aims to implement a switch platform in virtualized server environments. Supports standard management interfaces and enables programmatic extension and control of the forwarding functions. Can be ported into ASIC switches.	v1.0
Pantou/OpenWRT [15]	C	Turns a commercial wireless router or Access Point into an OpenFlow-enabled switch.	v1.0
ofsoftswitch13 [11]	C/C++	OpenFlow 1.3 compatible user-space software switch implementation.	v1.3
Indigo [6]	C	Open source OpenFlow implementation that runs on physical switches and uses the hardware features of Ethernet switch ASICs to run OpenFlow.	v1.0

Current software switch examples compliant with the OpenFlow standard

Source: M.Mendonca, et. al., *A Survey of SDN: Past, Present, and Future of Programmable Networks* [http://hal.inria.fr/docs/00/83/50/14/PDF/bare\\_jrnl.pdf](http://hal.inria.fr/docs/00/83/50/14/PDF/bare_jrnl.pdf)



## 2. SDN-OpenFlow



### Examples of native SDN switches compliant with the OpenFlow standard

Provider	Switch Model	Version
HP	8200zl, 6600, 6200zl, v1.0 5400zl, and 3500/3500yl	v1.0
Brocade	NetIron CES 2000 Series	v1.0
IBM	RackSwitch G8264	v1.0
NEC	PF5240 PF5820	v1.0
Pronto	3290 and 3780	v1.0
Juniper	Junos MX-Series	v1.0
Pica8	P-3290, P-3295, P-3780 and P-3920	v1.2

Source: M.Mendonca, et. al., *A Survey of SDN: Past, Present, and Future of Programmable Networks* [http://hal.inria.fr/docs/00/83/50/14/PDF/bare\\_jrnl.pdf](http://hal.inria.fr/docs/00/83/50/14/PDF/bare_jrnl.pdf)



## 2. SDN-OpenFlow



### ■ Controller Implementation Examples

Source: M. Mendonca, et al., *A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks* [http://hal.inria.fr/docs/00/83/50/14/PDF/bare\\_jrnl.pdf](http://hal.inria.fr/docs/00/83/50/14/PDF/bare_jrnl.pdf)

Controller name	Implem.	Open Source	Developer	Characteristics
NOX	Python/C++	y	Nicira	General, first SDN controller
POX	Python	Y	Nicira	General
MUL	C	Y	Kulcloud	Multi-threaded infrastructure, multi-level north-bound I/F
Beacon	Java	Y	Stanford	Cross-platform, modular, event-based and threaded operation
Trema	Ruby/C	Y	NEC	Framework for developing OpenFlow Ctrl.
Maestro	Java	Y	Rice University	NOS, provide I/F to develop modular network control
Jaxon	Java	Y	Independent	Based on NOX
Floodlight	Java	Y	BigSwitch	Based on the Beacon; works with PHY/V OF switches.



## 2. SDN-OpenFlow



### ■ Controller Implementation Examples

Source: M. Mendonca, et al., *A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks* [http://hal.inria.fr/docs/00/83/50/14/PDF/bare\\_jrnl.pdf](http://hal.inria.fr/docs/00/83/50/14/PDF/bare_jrnl.pdf)

Controller name	Implem.	Open Source	Developer	Characteristics
SNAC	C++	No	Nicira	Based on NOX; web-based, user-friendly policy manager: manage configure, monitoring
ovs-controller	C	Y	Independent	Simple OF ctrl. Ref. implem. with Open vSwitch ; manages any number of switches through the OF protocol;
Ryu	Python	Y	NTT, OSRG group	SDN OS ; provide logically centralized control and APIs to create new network M&C applications. Supports OpenFlow v1.0, v1.2, v1.3, and the Nicira Extensions.
NodeFlow	JavaScript	Yes	Independent	Written in JavaScript for Node.JS
Flowvisor	C	Y	Stanford/Nicira	Transparent proxy between OF switches and multiple OF controllers; can create network slices and delegate control of each slice to a different controller; isolation between slices.
RouteFlow	C++	Y	CPQD	Provide virtualized IP routing over OF capable hardware. It is composed by an OF Ctrl. Appl., an independent server, and a VNet environment reproducing the connectivity of a PHY infrastructure; it runs IP routing engines.





## 2.SDN- OpenFlow



- **SDN versus Network Functions Virtualization (NFV)**
  - **NFV : ETSI Industry Specification Group initiative** - to virtualize network functions previously performed by proprietary dedicated hardware
  - goal : reduce the telecom network infrastructure cost
    - by allowing the appropriate functions to run on a common, commodity platform hosting the necessary virtualized environments.
  - **NFV in the market today includes:**
    - **Virtual Switching** – physical ports are connected to virtual ports on virtual servers with virtual routers using virtualized IPsec and SSL VPN gateways.
    - **Virtualized Network Appliances** –dedicated functional boxes can be replaced with a virtual appliance. ( e.g. firewalls and gateways, Broadband Remote Access Servers (BRAS) , LTE Evolved Packet Core (EPC)).
    - **Virtualized Network Services** –e.g. network management applications such as traffic analysis, network monitoring tools, load balancers and accelerators.
    - **Virtualized Applications** – almost any application ( e.g. . cloud applications: virtualized storage and photo imaging services, to support the explosion in media communications)



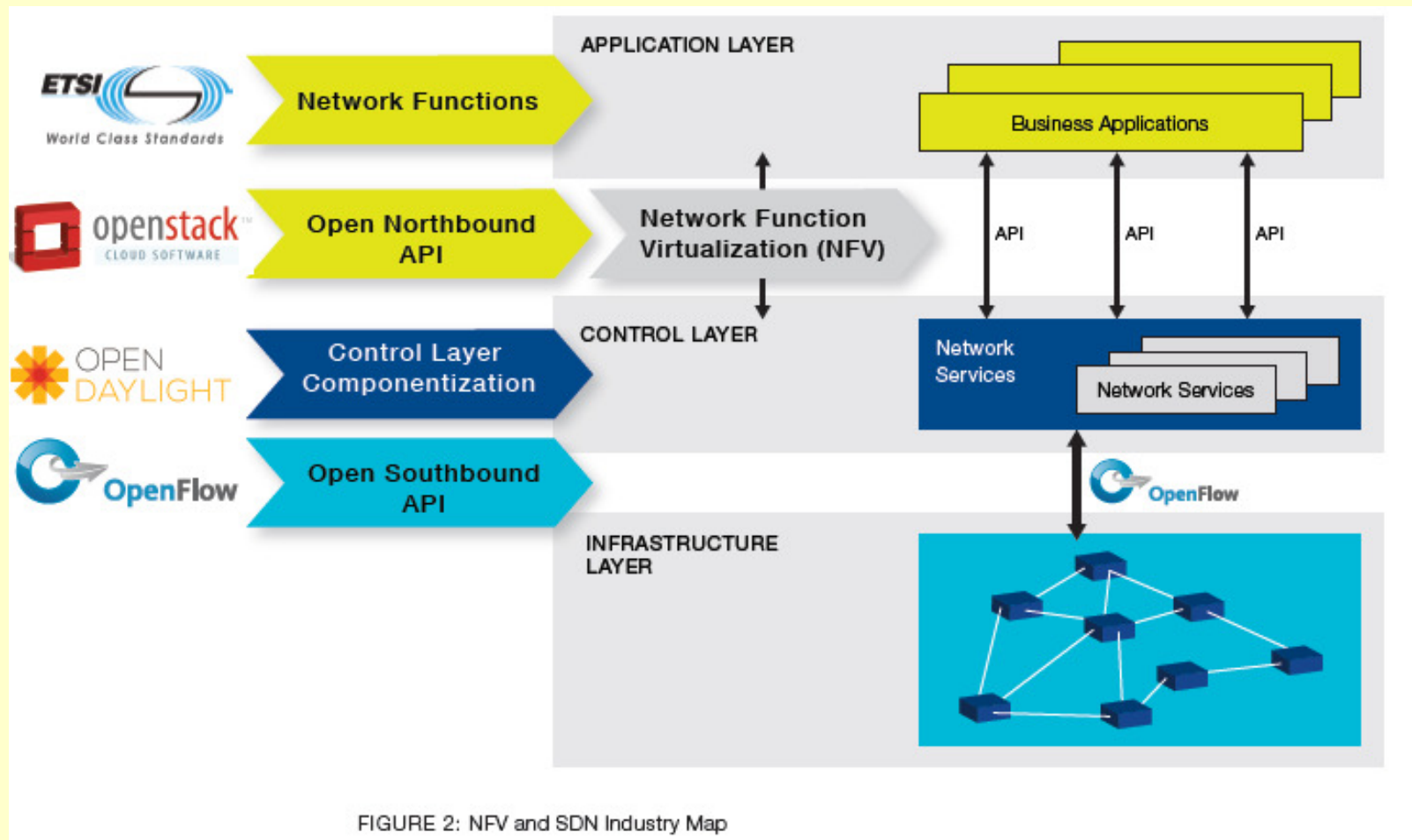
## 2. SDN- OpenFlow



- **SDN versus Network Functions Virtualization (NFV)- cont'd**
- *Source: ONF: OpenFlow-enabled SDN and Network Functions Virtualization, Feb. 2014*
  - By enabling NFV with OpenFlow-enabled SDN, network operators can realize even greater benefits from this promising new use of cloud technology.
  - OpenFlow-based SDN can accelerate NFV deployment by offering a scalable, elastic, and on-demand architecture
    - well suited to the dynamic NFV communications requirements for both virtual and physical networking infrastructures.

## 2. SDN- OpenFlow

- SDN versus Network Functions Virtualization (NFV)- cont'd
- ONF: NFV and SDN – industry view on architecture





# CONTENTS



1. Software Defined Networks Architecture (Summary)
2. SDN-OpenFlow
3. **👉 Control Plane Scalability in SDN**
4. SDN-like architecture example : ALICANTE Project
5. Conclusions



## 3. Control Plane Scalability in SDN



- **Scalability-related measures (ability of a distributed system):**
  - **Administrative** : for an increasing number of organizations or users to share a single distributed system.
  - **Functional** : to be enhanced by adding new functionality at minimal effort
  - **Geographic** : to maintain performance, usefulness, or usability regardless of expansion from concentration in a local area to a more distributed geographic pattern
  - **Load scalability**: to easily expand and contract its resource pool to accommodate heavier or lighter loads or number of inputs
    - the ease with which a system/component can be modified, added, or removed, to accommodate changing load
  - **Generation scalability** to scale up by using new generations of components.
  
- **Horizontal and vertical scaling**
  - to **scale horizontally (or scale out)** : to add more nodes to a system, such as adding a new computer to a distributed software application
  - To **scale vertically (or scale up)** : to add resources to a single node in a system, (e.g. addition of CPUs or memory to a single node)



## 3. Control Plane Scalability in SDN



- **SDN Scalability issues**
- **Why SDN scalability-related concerns ”? Main reasons:**
  - **Centralized control plane**
    - signaling overhead ( forwarders <--> controllers)
    - central controller limitations: will not scale for larger networks (no. of switches, flows, bandwidth, etc.)
    - lack of control communication between forwarders → the power of a distributed system is partially lost
    - Single point of failure (if a single controller)
  - **CPI/Dpl decoupling issues**
    - need standard API between Cpl/DPI - to allow their independent evolutions - not so simple
    - switch manufacturers should adopt the same APIs ( compatibility reasons)
    - moving control far away from switches/routers → may create additional signaling overhead ( both directions)
  - **Switch/forwarder limitations**
  - **Opinions: no unanimous w.r.t. SDN scalability problems**
    - **Optimistic**
    - **Pessimistic**
  - Note: OpenFlow is a protocol ~HTTP: not too much sense to talk about scalability of the protocol only but about the whole SDN solution scalability



## 3. Control Plane Scalability in SDN



- Positive/optimistic opinions:
  - E.g. Source: *S. H. Yeganeh, A. Tootoonchian, Y. Ganjali, On Scalability of Software-Defined Networking, IEEE Comm. Magazine, February 2013.*
    - The current research on SDN scalability shows that:
      - Scalability concerns are not fundamentally unique to SDN
      - Many issues can be addressed while keeping the SDN advantages
      - Current SDN deployments support this argument.
      - SDN adds much flexibility that can accommodate network programming and management at scale, while traditional networks have historically failed in this respect
      - Control/Data Planes decoupling – they can evolve independently, and offer advantages: high flexibility, vendor-agnostic features, programmability, possibility of realizing a centralized network view
      - However it is recognized that many challenges remain to be solved



### 3. Control Plane Scalability in SDN



- **(Rather) Pessimistic opinions**  
<http://highscalability.com/blog/2012/6/4/openflowsdn-is-not-a-silver-bullet-for-network-scalability.html>
  - “OpenFlow/SDN is Not a Silver Bullet for Network Scalability”
  
- **OpenFlow/SDN -related - problems:**
  - compatibility with existing chipsets to incomplete and fast-changing specs (and related compatibility issues) -> some hard scalability limits
  - If use OpenFlow to program a number of independent forwarders that don't interact with each other (at control level), then
    - these cases compare with a scale-out application with no shared state (in networking devices) and a back-end DB (OpenFlow controller)
    - scalability depends primarily on the back-end DB.





## 3. Control Plane Scalability in SDN



- (Rather) Pessimistic opinions (cont'd)
  - The number of flows a physical device can handle
    - in HW is limited
    - and the SW -based devices are still too slow
    - It's difficult to implement very granular E2E traffic control
  - E.g. NOX (the first SDN controller), could process max. 30,000 flow initiations per sec, [see NOX Refs] if less than 10 ms install time per flow is wanted
  - A device using HW -based packet forwarding can install only a limited no. of flows in a time period (usually less than thousand flows per sec.- data from manufacturers and users of high-end routers)
    - However The above limitations could be solved with the next-generation chipsets



## 3. Control Plane Scalability in SDN



- **(Rather) Pessimistic opinions (cont'd)**
  - Large-scale networks with distributed intelligence (control plane) perform inherently better than systems with centralized control
    - That is why IP routers with distributed routing protocols became so prevalent in the last two decades
    - While SONET/SDH, Frame Relay or ATM rather failed (hint: all three relied on centralized virtual circuit setup).
    - It is difficult to recover from a node or link failure in 50 ms (e.g. a typical requirement voice traffic) : time to get a reply from the central controller.
    - Other problems: network devices losing connectivity with the central controller if the primary uplink fails.



## 3. Control Plane Scalability in SDN



- **(Rather) Pessimistic opinions (cont'd)**
- Solutions: Offloading some of the intelligence and/or installing precomputed alternate paths into the network nodes
  - Traditional WAN (e.g. : SONET or SDH) relied on redundant circuits to implement fast failover times (because the controllers couldn't be relied upon to find and install an alternate path in time)
  - However , the price is wasting bandwidth
    - One reason why SP frequently prefer – when possible IP+MPLS-based –distributed solutions-networks over traditional optical networks.



## 3. Control Plane Scalability in SDN



- **(Rather) Pessimistic opinions (cont'd)**
- Google recognized the challenges – they use OpenFlow in their G-scale network, but only within a data center, where a cluster of OpenFlow controllers manages local devices.
  - They use traditional routing protocols (BGP+IS-IS) between sites and further manage traffic flows with proprietary TE technology similar (in functionality) to MPLS-TE.
  - NEC also hit limits of real-time control with their ProgrammableFlow product.
  - A single OpenFlow controller can control only a few dozens of top-of-rack switches without supporting “linecard protocols” like LACP or BFD, or running routing protocols or spanning tree protocol (STP) with the external devices.
  - A network built with their OpenFlow controller interacts with the outside world through static routes and static link aggregation groups (LAG)..



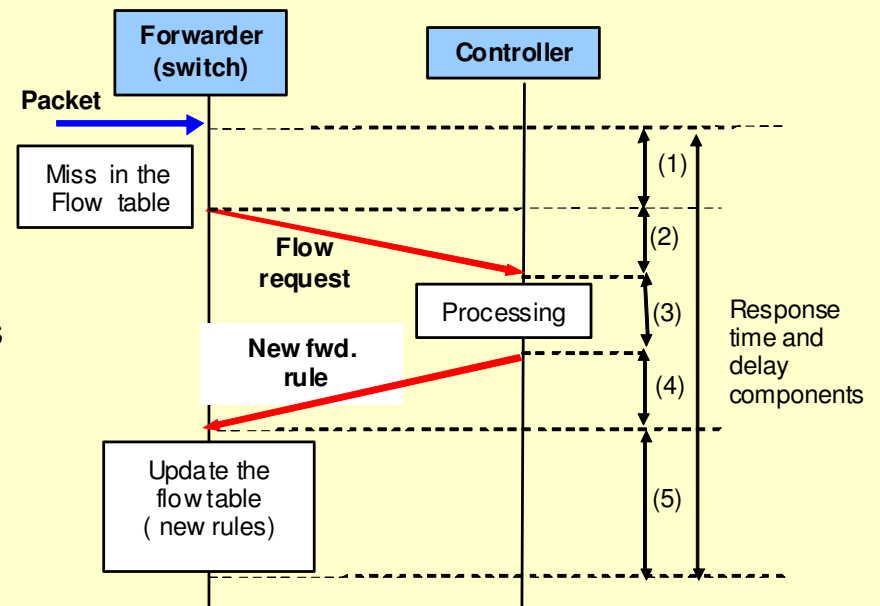
### 3. Control Plane Scalability in SDN



- **(Rather) Pessimistic opinions (cont'd)**
- **Conclusions ( of the study mentioned above)**
  - **OpenFlow/SDN solutions cannot overcome limitations inherent in asynchronous distributed systems with unreliable communication paths.**
    - Some of them focused on controlling a large number of independent edge devices,
    - others decided to use the advantages of OpenFlow while retaining the distributed nature of the system that gives large-scale IP networks (example: the Internet) their resilience.
  - Google's G-scale network is a good example of optimal SDN architecture in a large-scale WAN environment.
    - They use OpenFlow solely to control of a cluster of local devices and combine local controllers with centralized path computation and traditional time-proven technologies (routing protocols).
  - One can expect major OpenFlow/SDN-based advances in the network edge
    - individual devices don't interact with each other
    - minor impact of OpenFlow in the network core (see SDIA)

- **SDN Problem: new flows setup - response time**

- Signaling overhead components
  - Switch (CPU, mem, ..)- (1), (5)
  - Controller (CPU, mem, ..) - (3)
  - Transport through network – (2), (4)
- Transport: (2), (4) → place controller closer to the switch
- Switch
  - OpenVSwitch: install tens of ( $10^{**3}$ ) flows/s with < 1 ms latency
  - HW switches: install few ( $10^{**3}$ ) with 10ms latency
    - Weak mgmt CPU
    - Low speed communication CPU-switching chipset
  - **Hope for faster switches**
- Frequent events can stress
  - Controller resources
  - Control channel
  - Switch



On demand flow setup



## 3. Control Plane Scalability in SDN



- **Note: A control program installing reactively an E2E path on a per-flow basis does not scale:**
  - the per switch memory is fixed
  - while the number of forwarding entries in the data path grows with the number of active flows
  - Delay of going through the controller
  - Switch complexity
  - Misbehaving hosts
  - **Example:**
    - Source: *Li Erran Li, Software Defined Networking COMS 6998-8, Fall 2013*
    - <http://www.cs.columbia.edu/~lierranli/coms6998-8SDNFall2013/>
    - Switch with finite BW between data / control plane, (overheads between ASIC and CPU)
      - Setup capability: **275~300 flows/sec**
      - While in a data center: mean interarrival 30 ms; Rack with 40 servers  $\Rightarrow$  1300 flows/sec



## 3. Control Plane Scalability in SDN



- **Solutions:**
  - **Direct solutions**
    - Increase controller processing power
    - Increase switch processing power
  - **Aggregation of rules**
  - Proactive installation of rules
    - Problems: no host mobility support, not enough memory in switches
  - **Delegate more responsibilities to the data plane**
    - to switch control plane [e.g. Diffane, DevoFlow]
  - **Distributed controllers**
    - Flat structure multiple controllers [e.g. ONIX]
    - Recursive controller design [e.g. Xbar]
    - Hierarchical controller design [e.g. Kandoo]





### 3. Control Plane Scalability in SDN



- **SDN Scalability-related solutions**
- **Increase the controller processing power**
  
- *Source: A. Tootoonchian et al., "On Controller Performance in Software-Defined Networks," Proc. USENIX Hot-ICE '12, 2012, pp. 10–10.*
  - Multicore systems for higher level of parallelism
  
  - And improved IO performance
    - Simple modifications to the NOX controller → performance increase more than 10 times an order of magnitude on a single core. ( w.r.t 30000 flows/sec)
  
    - A single controller can support larger networks, ( if the controller channel has enough bandwidth and acceptable latency)



## 3. Control Plane Scalability in SDN



- ***Solution proposals examples:***
  - ( **DIFANE** - **D**istributed **F**low **A**rchitecture for **N**etworked **E**nterprises) :
  - *Source: M. Yu et al., “Scalable Flow-Based Networking with DIFANE,” Proc. ACM SIGCOMM 2010 Conf., 2010, pp. 351–62.*
    - Scalable solution **keeping all traffic in the data plane** by selectively directing packets through intermediate switches that store the necessary rules
    - It relegates the controller to the simpler task of partitioning these rules over the switches.
    - implemented with commodity switch HW , since all data-plane functions can be expressed in terms of wildcard rules –performing simple actions on matching packets
    - Experiments: prototype on modular router “**Click**”-based **OpenFlow switches**
  - Flexible Policies in Enterprises
    - Access control : Drop packets from malicious hosts
    - Customized routing: Direct VoIP calls on a low-latency path
    - Measurement: Collect detailed HTTP traffic statistics



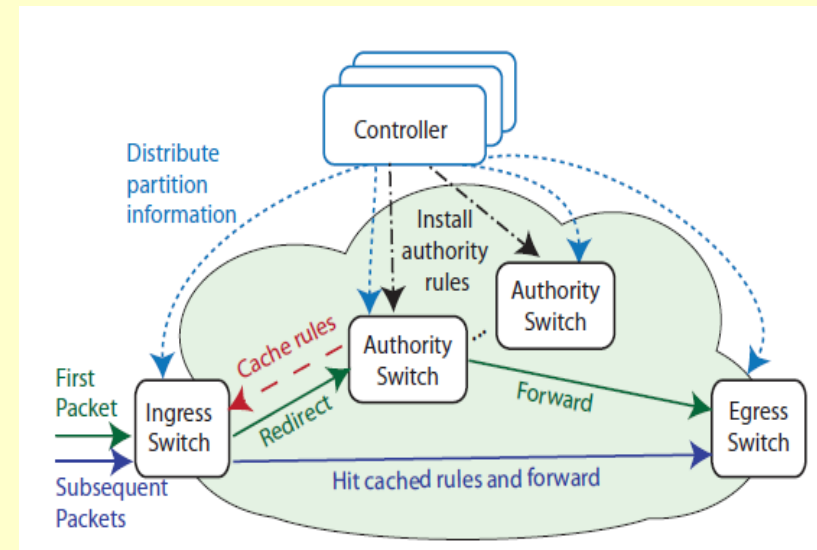
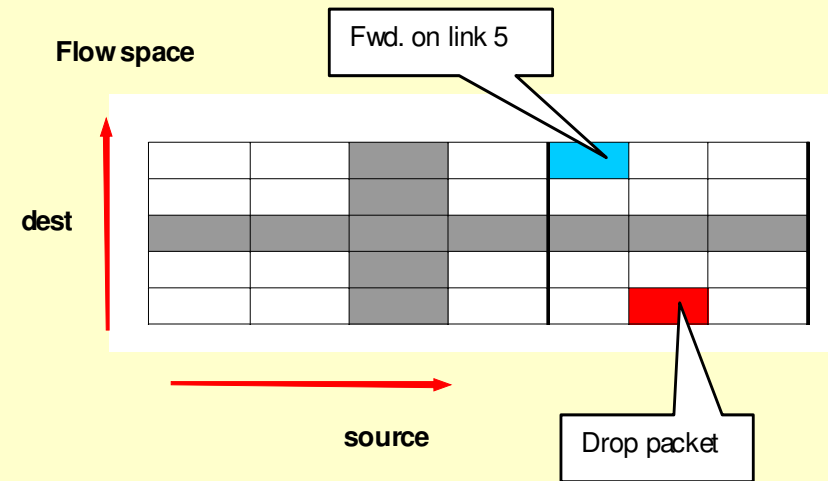
## 3. Control Plane Scalability in SDN



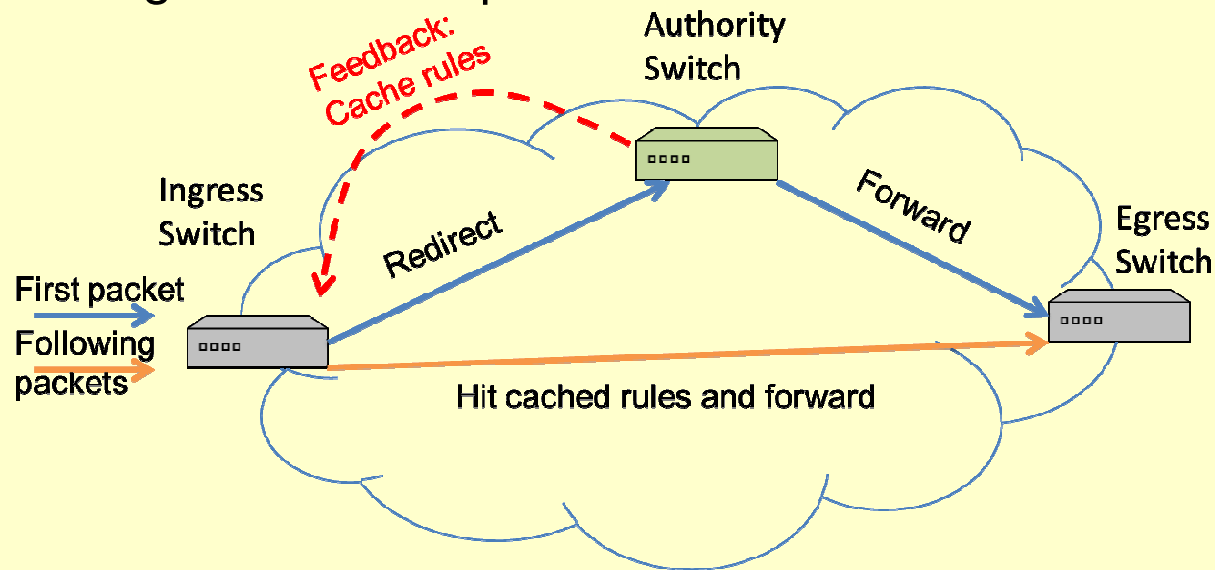
- ***Solution proposals examples:***
  - **DIFANE (cont'd)**
  - The **controller distributes the rules** across (a subset of) the switches, called “authority switches- ASw,” to scale to large topologies with many rules.
  - The **controller runs a partitioning algorithm** that divides the rules evenly and minimizes fragmentation of the rules across multiple ASw-s
  - The switches handle **all packets in the data plane** (*i.e.*, TCAM), diverting packets through authority switches as needed to access the appropriate rules.
  - The “rules” for diverting packets are themselves expressed as TCAM entries.
  - **Locate Authority Switches**
    - Partition information in ingress switches
      - Using a small set of coarse-grained wildcard rules
      - ... to locate the authority switch for each packet
    - Distributed directory service but not DHT
      - Hashing does *not* work for wildcards
      - Keys can have wildcards in arbitrary bit positions

- **Solution proposals examples:**
- **DIFANE (cont'd)**
  - Flow-based Switches
    - Install rules in flow-based switches: Store rules in high speed memory (TCAM)
    - Perform simple actions based on rules
      - Rules: Match on bits in the packet header
      - Actions: Drop, forward, count

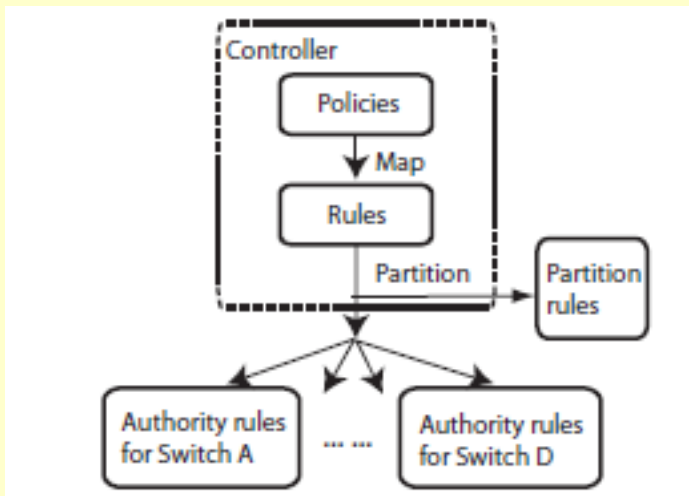
DIFANE Flow Management Architecture



- **Solution proposals examples**
  - **DIFANE Phase 1:** The controller *proactively* generates the rules and distributes them to authority switches
    - Controller distribute different rules to Authorities switches :A, B, C,..
    - And also to Ingress Switches, Egress switches
  - **DIFANE Phase 2:** The authority switches keep packets always in the data plane and *reactively* cache rules
    - Note: A slightly longer path in the data plane is faster than going through the control plane



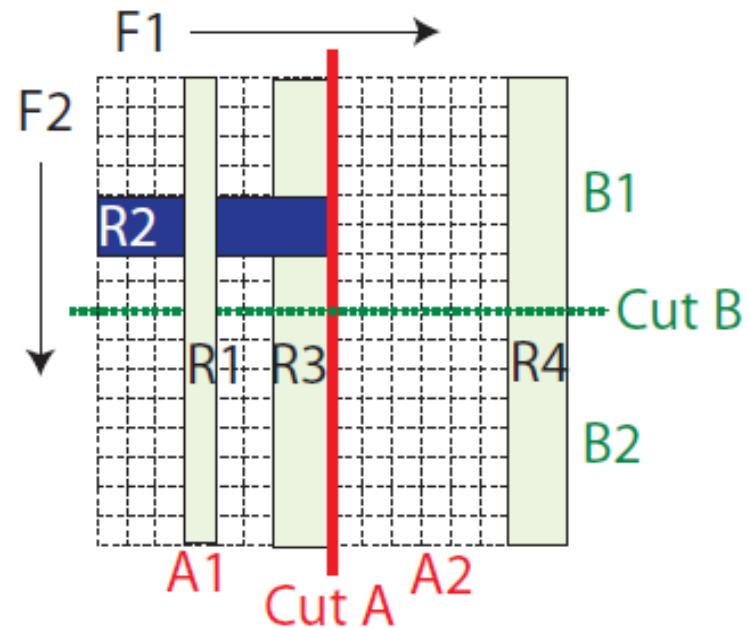
- *Solution proposals - examples*
  - DIFANE (cont'd)



Rule operations in the controller.

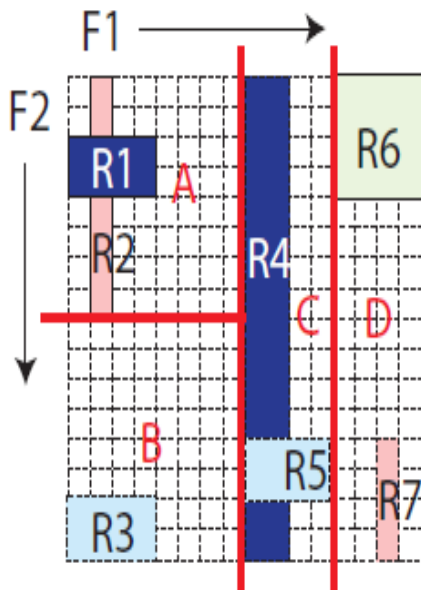
Rule	F <sub>1</sub>	F <sub>2</sub>	Action
R <sub>1</sub>	4	0-15	Accept
R <sub>2</sub>	0-7	5-6	Drop
R <sub>3</sub>	6-7	0-15	Accept
R <sub>4</sub>	14-15	0-15	Accept

(a) Wildcard rules listed in the decreasing order of priority. ( $R_1 > R_2 > R_3 > R_4$ )



Example of wildcard rules

- *Solution proposals -examples*
- DIFANE (cont'd)



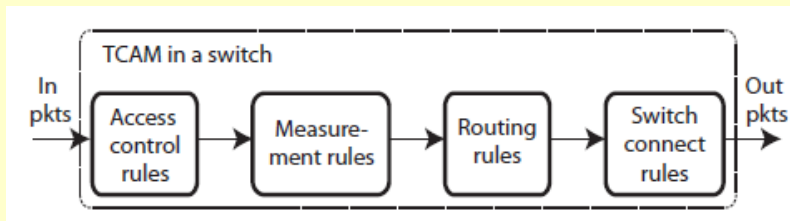
(a) Low-level rules and the partition

Type	Priority	F <sub>1</sub>	F <sub>2</sub>	Action	Timeout	Note
Cache rules	210	00**	111*	Encap, forward to B	10 sec	R <sub>3</sub>
	209	1110	11**	Drop	10 sec	R <sub>7</sub>
	...	...	...	... ..	...	...
Authority rules	110	00**	001*	Encap, forward to D, trigger ctrl. plane func.	∞	R <sub>1</sub>
	109	0001	0***	Drop, trigger ctrl. plane func.	∞	R <sub>2</sub>
Partition rules	15	0***	000*	Encap, redirect to B	∞	Primary
	14	0***	1***	Encap, redirect to C	∞	
	13	11**	****	Encap, redirect to D	∞	
	5	0****	000*	Encap, redirect to B'	∞	Backup
	...	...	...	... ..	...	...

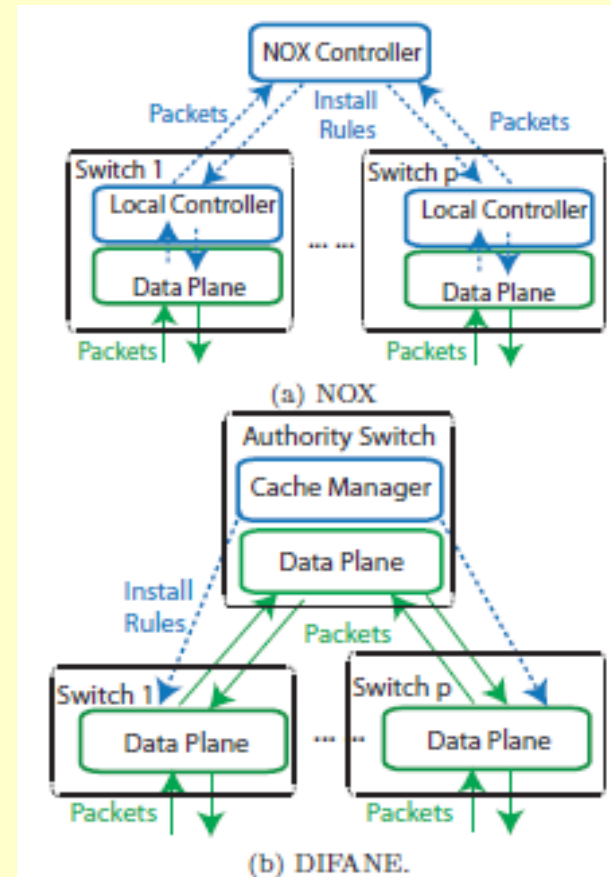
(b) Wildcard rules in the TCAM of Switch A

**Wildcard rules in DIFANE (A-D are authority switches)**

- *Solution proposals examples*
- **DIFFANE** (cont'd)



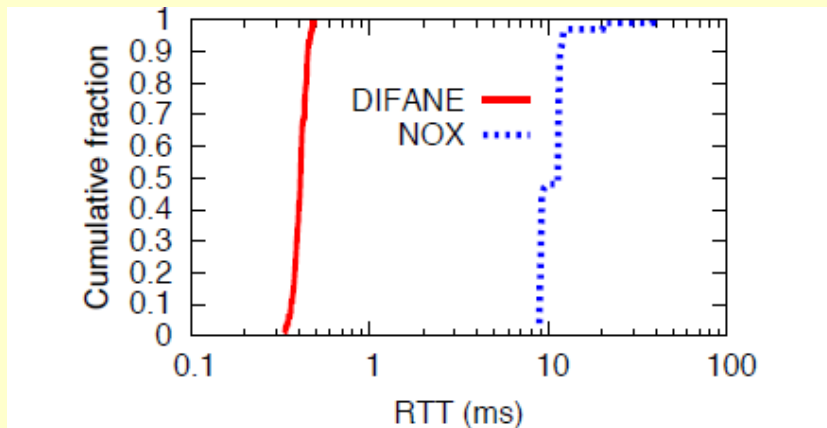
Rules for various management modules



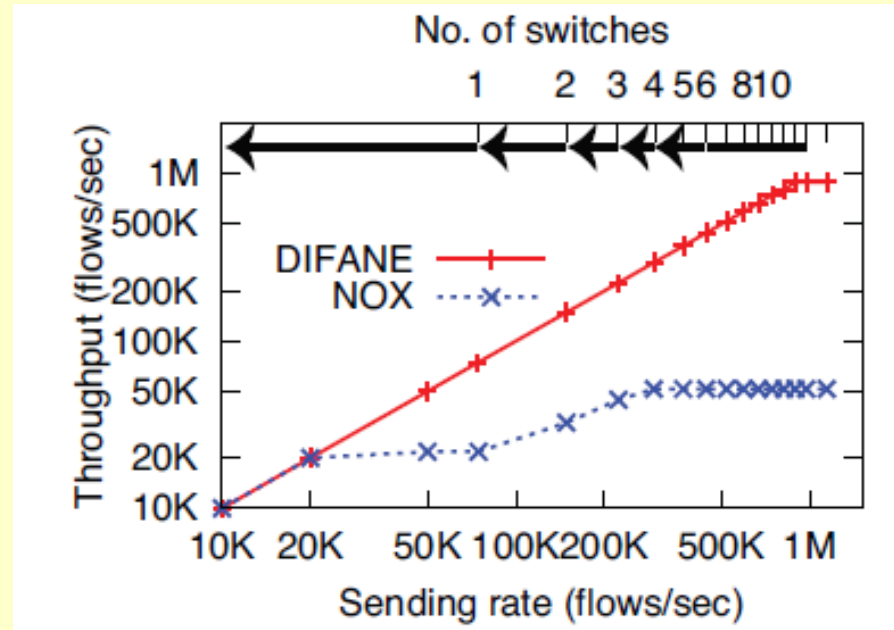
Setup in NOX and DIFFANE



- *Solution proposals examples*
- **DIFFANE (cont'd)**



**Delay comparison of DIFANE and NOX**

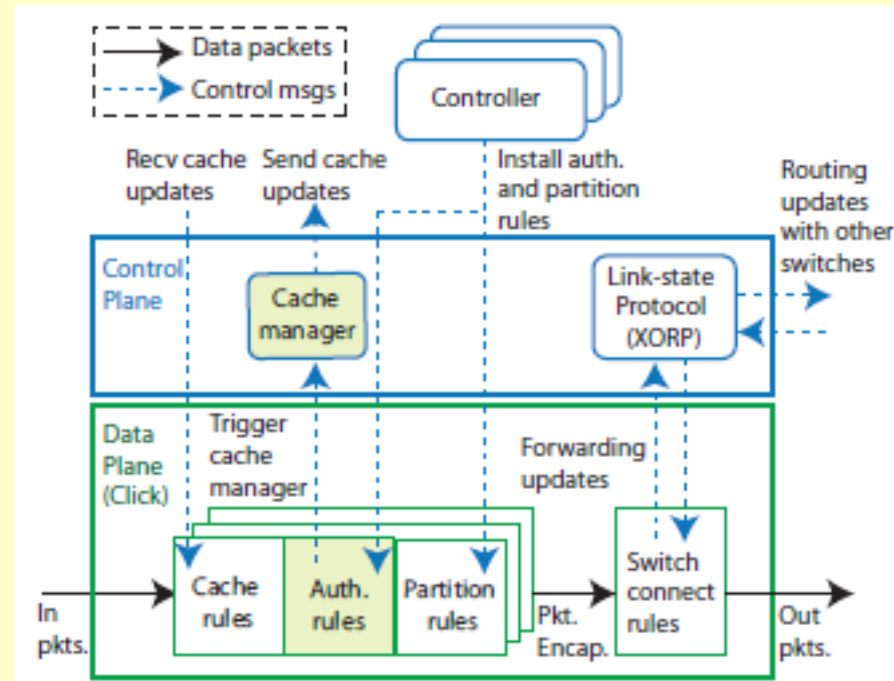


**Throughput comparison of DIFANE and NOX**

- *Solution proposals examples:*

- **DIFFANE** (cont'd)

**DIFANE** prototype implementation. (Cache manager and authority rules (shaded boxes) only exist in authority switches)



Criticism: DIFANE does not address the issue of global visibility of flow states and statistics.



## 3. Control Plane Scalability in SDN



- ***Solution proposals examples:***
- **DevoFlow:** *Source: A. R. Curtis et al., "DevoFlow: Scaling Flow Management for High-Performance Networks," Proc. ACM SIGCOMM '11, 2011, pp. 254–65.*
- **Observation in experiments:**
  - Switches have finite bandwidths between their data- and control-planes, and finite compute capacity limiting the rate of flow setup
  - Cannot provide fast flow statistics for traffic mgmt. tasks (e.g. load balancing).
- **Solution: Decreasing the number of interactions between switches and controller**
- **Main DevoFlow principles**
  - **Keep flows in the data-plane** as much as possible to reduce communication overhead DPI- CPI
  - **Maintain enough visibility over network flows** for effective centralized flow management, but otherwise provide only aggregated flow statistics
  - **Simplify the design and implementation** of fast switches while retaining network programmability.



## 3. Control Plane Scalability in SDN



- ***Solution proposals examples***
- **DevoFlow Principles (cont'd)**
  - Devolve control of most flows back to the switches;
    - the controller controls only targeted significant flows and has visibility over only these flows and packet samples.
  - Aggressive use of wild-carded OpenFlow rules thus reducing the number of switch-controller interactions and the no. of TCAM entries
    - new mechanisms to efficiently detect significant flows, by waiting until they actually become significant.
  - New *mechanisms* to allow switches to make local routing decisions, which forward flows that do not require asking the controller.
    - **Mechanisms:**
      - Control: Rule cloning; Local actions
      - Statistics-gathering: Sampling, Triggers and reports, Approximate counters



## 3. Control Plane Scalability in SDN



- ***Solution proposals examples:***
- **DevoFlow:** (cont'd)
- DevoFlow attempts to resolve two dilemmas :
  - ***Control dilemma:***
    - Per/flow Invoking the OpenFlow controller for flow setup :
      - provides good start-of-flow visibility
      - but : much load on the CPI and too much setup delay to latency-sensitive traffic
    - Aggressive use of OpenFlow flow-match wildcards or hash-based routing (such as ECMP)
      - reduces CPI load
      - but prevents the controller from effectively managing traffic.
  - ***Statistics-gathering dilemma:***
    - Collecting OpenFlow counters on lots of flows, via the pull-based Read-State mechanism:
      - too much CPI load
    - Aggregating counters over multiple flows via the wild-card mechanism
      - may undermine the controller's ability to manage specific elephant flows.



## 3. Control Plane Scalability in SDN



- **Solution proposals examples:**
- **DevoFlow:** (cont'd)
- **Mechanisms for devolving control**
- **1. Rule cloning:**
  - **Standard OpenFlow** mechanism for wildcard rules say:
    - *all packets matching a given rule are treated as one flow.*
  - So, using a wildcard to avoid invoking the controller on each microflow arrival  $\Rightarrow$  routing all matching microflows over the same path,
  - aggregating all statistics for these microflows into a single set of counters.
  
  - **DevoFlow** augments the “action” part of a wildcard rule with a boolean CLONE flag.
    - $F=0 \Rightarrow$  the switch follows the standard wildcard behavior.
    - $F=1 \Rightarrow$  the switch locally “clones” the wildcard rule
      - to create a new rule in which all of the wildcard fields are replaced by values matching this microflow, and all other aspects of the original rule are inherited.
    - Subsequent packets for the microflow match the microflow-specific rule, and thus contribute to microflow-specific counters.
  - Also, this rule goes into the exact-match lookup table, reducing the use of the TCAM, and so avoiding most of the high TCAM power cost .



## 3. Control Plane Scalability in SDN



- **Solution proposals examples:**
- **DevoFlow:** (cont'd)
  
- **Mechanisms for devolving control**
- **2. Local actions:**
  - Certain flow-setup need decisions intermediate between the
    - heavyweight “invoke the controller”
    - and the lightweight “forward via this specific port”
  - This are choices offered by standard OpenFlow.
  
- **DevoFlow:** rules augmented with a small set of possible “local routing actions” that a switch can take without invoking the controller.
  - If a switch does not support an action, it defaults to invoking the controller, so as to preserve the desired semantics.
  - Examples of local actions include
    - multipath support
    - rapid re-routing



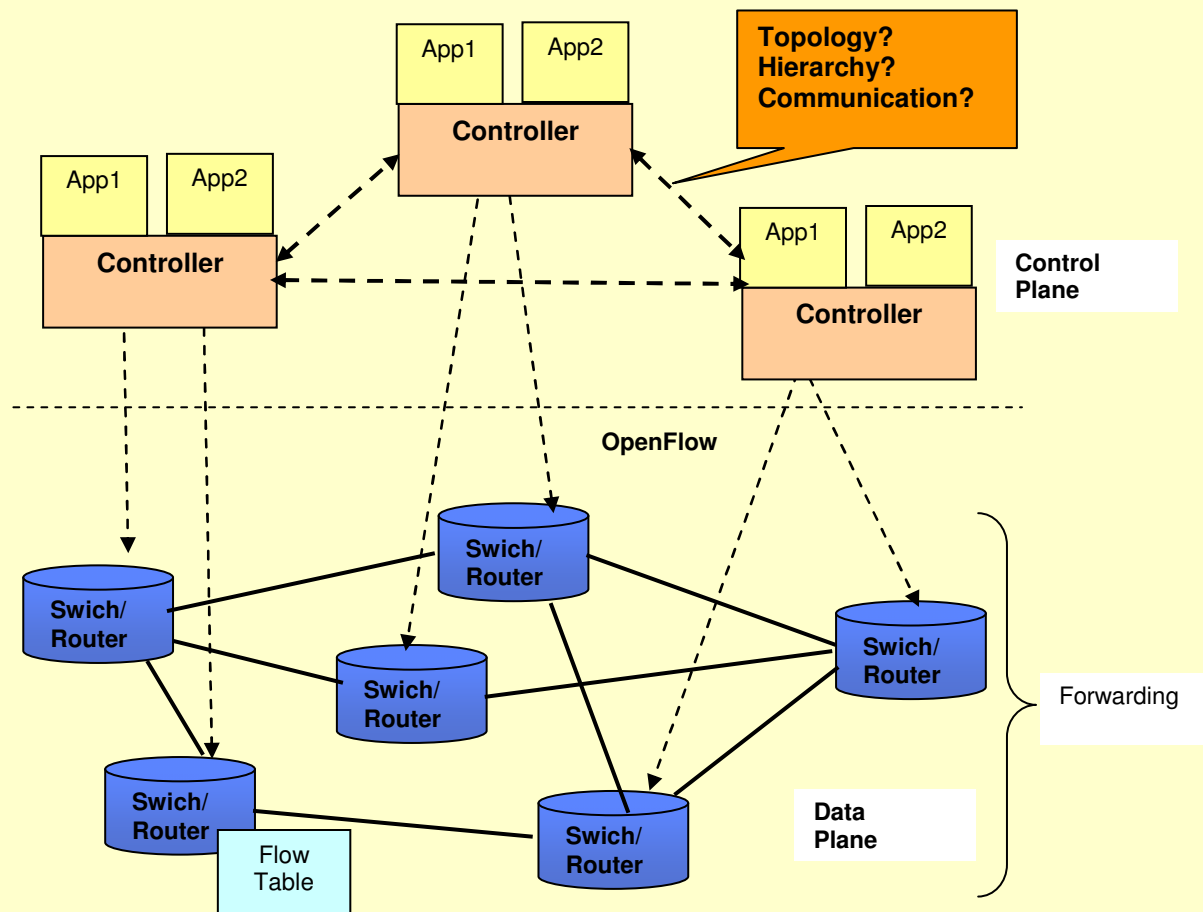
## 3. Control Plane Scalability in SDN



- **Multiple controllers- solution for large networks**
  - **Several controllers ( distributed DPI)**
  - However maintaining the unified view on the network ( to benefit from SDN advantages)
  - Need to maintain consistency between them
    - Full/strong consistency is difficult to achieve ( affects the control plane response time)
    - Define a convenient consistency level (while maintaining availability and partition tolerance)
    - The necessary degree of consistency between several controllers depends on the type of control applications
    - Under research and std. : protocols for communication/synchronization between controllers



- **SDN Scalability issues**
  - **Multiple controller solution**
  - Control Plane and Data Plane – independent; they can have different topologies





## 3. Control Plane Scalability in SDN



- **Multiple controllers- still open issues**
  - Control plane topology
  - Flat or hierarchical organization
  - Global consistency assurance
  - Geo-localisation of controllers
  - Inter-controller communication protocols
  - Reliability
  - Security



## 3. Control Plane Scalability in SDN



- **Multiple controllers- (cont'd)**
- **Approaches:**
  - **Flat organization**
    - Distribution of the CPI while maintaining a logically centralized using a distributed file system, a distributed hash table and a pre-computation of all possible combinations respectively.
    - however they impose a strong requirement: a consistent network-wide view in all the controllers.
    - Examples: HyperFlow, Onix and Devolved controllers
  - **Hierarchical organization**
    - Hierarchical distribution of the controllers based on two layers:
    - (i) the bottom layer, a group of controllers with no interconnection, and no knowledge of the network-wide state,
    - (ii) the top layer, a logically centralized controller that maintains the network wide state.
    - Example: Kandoo



## 3. Control Plane Scalability in SDN



- **Solution proposals examples:**

- **Onix :**

- *Source:*

- *T. Koponen et al., “Onix: A Distributed Control Platform for Large-Scale Production Networks,” Proc. 9th USENIX, OSDI Conf., 2010*
- *Li Erran Li , Software Defined Networking COMS 6998-8, Fall 2013; SDN Scalability*
  - *<http://www.cs.columbia.edu/~lierranli/coms6998-8SDNFall2013/>*

- **Onix:**

- distributed control platform implementing a **distributed CPI**
- provides general APIs for control appl. to access network state (NIB), which is distributed over Onix instances.

- **Basic Onix functionalities:**

- State distribution primitives between controllers and network elements
- Virtualized network elements

- Note: Onix is not a complete novel solution; it continues ideas a the work of 4D project RCP, SANE, Ethane and NOX – see references.



## 3. Control Plane Scalability in SDN



- **Solution proposals examples:**
- **Onix (cont'd)**
  - **General design requirements**
    - **Scalable:** any scaling limitations should be due to the inherent problems of state management, not the implementation of the control platform
    - **General:** offer APIs to support a wide range of network management applications
    - **Reliable:** to gracefully handle various failures
    - **Simplify** the framework for Network management applications
    - **Control Plane good performance :** ONIX establishes a tradeoff between solution generality and performance, trying to optimize the former while still satisfying the latter



## 3. Control Plane Scalability in SDN

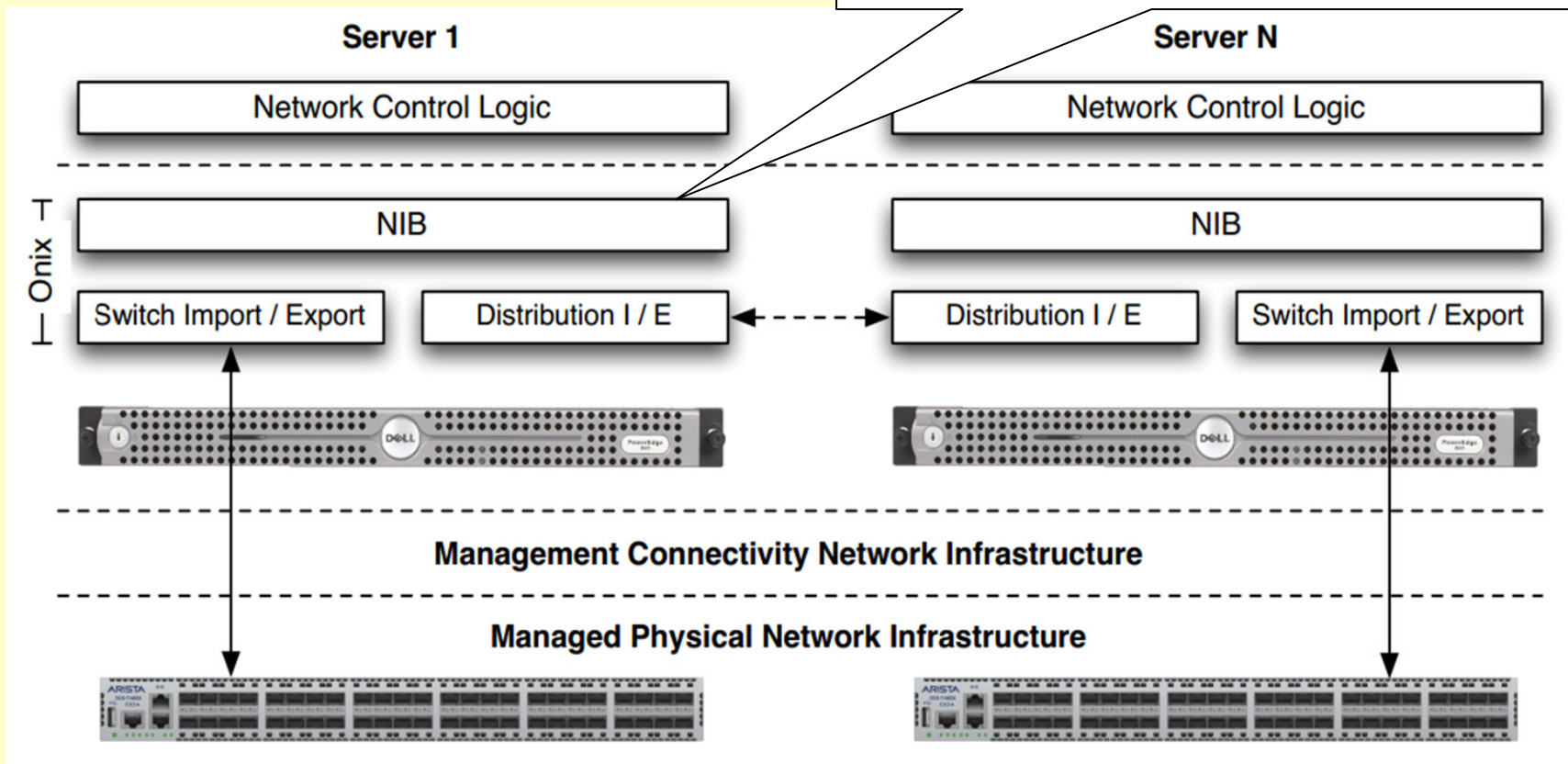


- **Solution proposals examples:**
- **Onix (cont'd)**
  - **Main Onix functional components**
    - **Physical infrastructure:** switches, routers, load balancers, etc.
      - Having Onix capable I/Fs To read/write the state controlling the element's behavior (e.g. Fwd. table entries)
      - They need not run any SW other than that required to support this I/F and achieve basic connectivity.
    - **Connectivity infrastructure:** Channels for DPI/CPI control messages
      - In-band: the control traffic shares the same forwarding elements as the data traffic on the network
      - Out of band: separate physical network handles the control traffic
    - **Onix:** A distributed system running a cluster of one or more physical servers, each of which may run multiple Onix instances.
    - **Control logic:** Network management applications on top of Onix
      - The control logic determines the desired network behavior
      - Onix merely provides the primitives needed to access the appropriate network state

# 3. Control Plane Scalability in SDN

- **Solution proposals examples:**
- **Onix (cont'd)**
- **Architecture**

NIB : major sub-system  
 - State for applications to access  
 - External state changes imported into it  
 - Local state changes exported from it





## 3. Control Plane Scalability in SDN



- **Solution proposals examples:**
- **Onix (cont'd)**
- **Abstracted vision upon the network**
  - **Global View:** Centralized network view observed and controlled including all physical network elements.
  - **Flow:** a sets of packets (the first and subsequent packets) -having some common fields in their headers - which are treated in a similar way.
  - **Switch:** <header: counters, actions>
  - **Event-based operation:** The controller operations are triggered by routers or applications
- **ONIX API**
  - Developers program on a network graph
  - Nodes represent physical network entities
  - Onix's API consists of a **data model representing the network** infrastructure, with each network element corresponding to one or more data objects
  - **The control logic can:**
    - read the current state associated with that object;
    - alter the network state by operating on these objects;
    - register for notifications of state changes to these objects
  - The platform allows the CPI to customize the data model and have control over the placement and consistency of each component of the network state.





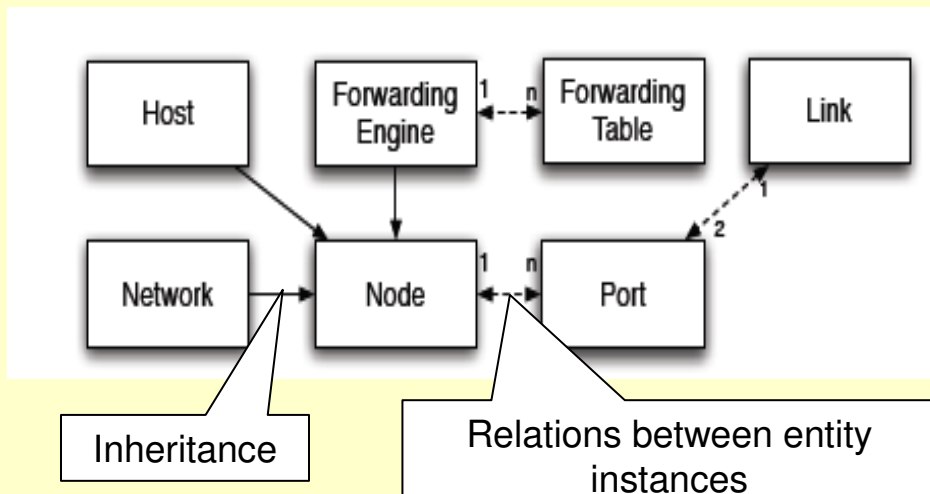
# 3. Control Plane Scalability in SDN



- **Solution proposals examples:**
- **Onix (cont'd)**
- **Network Information Base**
  - Nodes, ports and links constitute the network topology.
  - All entity classes inherit the same base class providing generic key-value pair access.

The default network entity classes provided by Onix's API

Functions provided by the Onix NIB API.



Category	Purpose
Query	Find entities.
Create, destroy	Create and remove entities.
Access attributes	Inspect and modify entities.
Notifications	Receive updates about changes.
Synchronize	Wait for updates being exported to network elements and controllers.
Configuration	Configure how state is imported to and exported from the NIB.
Pull	Ask for entities to be imported on-demand.



## 3. Control Plane Scalability in SDN



- **Solution proposals examples:**
- **Onix (cont'd)**
- **Other scalability- related features**
  - **Partitioning**
    - A particular controller instance keeps only a subset of the NIB in memory and up-to-date.
    - One Onix instance may have connections to a subset of the network elements
  - **Aggregation**
    - One Onix instance can expose a subset of the elements in its NIB as an aggregate element to another Onix instance
    - Example:
      - large network;
      - each sub-network is managed by an Onix controller
      - exposing all sub-network as a single aggregate node to a global Onix instance which performs TE for the whole network



## 3. Control Plane Scalability in SDN



- **Solution proposals examples:**
- **Onix : Other scalability- related features (cont'd)**
  - **Consistency and durability**
    - The CPI dictates the consistency requirements for the network state it manages.
      - by implementing any of the required distributed locking and consistency algorithms for state requiring strong consistency
      - providing conflict detection and resolution for state not guaranteed to be consistent by use of these algorithms.
  - Onix : offers two data stores that an application can use
    - For *state applications* that favor durability and stronger consistency: replicated transactional database and,
    - for *volatile state* (more tolerant of inconsistencies) a memory-based one-hop DHT.



## 3. Control Plane Scalability in SDN



- **Solution proposals examples:**
- **ONIX (cont'd)**
- **Reliability**
  - Network Element & Link Failures: Applications' responsibility
  - Connectivity Infrastructure Failures: Assumed reliable
  - Onix Failures: Onix provides distributed coordination facilities provided for app failover
- **Implementation [ ]:**
  - ~ 150 000 lines of C++ and integrates a number of third party libraries
    - Onix contains logic for communicating with the network elements
    - aggregating information into the NIB
    - providing a framework in which application programmers can write a management application
- **Onix Applications**
  - Ethane
  - Distributed Virtual Switch (DVS)
  - Multi-tenant virtualized data centers.
  - Scale-out carrier-grade IP router
  - ....



## 3. Control Plane Scalability in SDN



- **Solution proposals examples**
- **HyperFlow**
  - *Source: A. Tootoonchian et.al., "Hyperflow: A Distributed Control Plane for OpenFlow," Proc. 2010 INMConf., 2010.*
  - Among the first distributed (event-based) Control Plane for OpenFlow
  - **Logically centralized**
    - All the controllers share the same consistent network-wide view and locally serve requests without actively contacting any remote node, thus minimizing the flow setup times.
  - **Physically distributed**: scalable while keeping the network control centralization benefits
  - By passively synchronizing network-wide views of OpenFlow controllers, it localizes decision making to individual controllers, thus minimizing the CPI response time to data plane requests
  - It is resilient to network partitioning and component failures
  - **It enables interconnecting independently managed OpenFlow networks** - - an essential feature



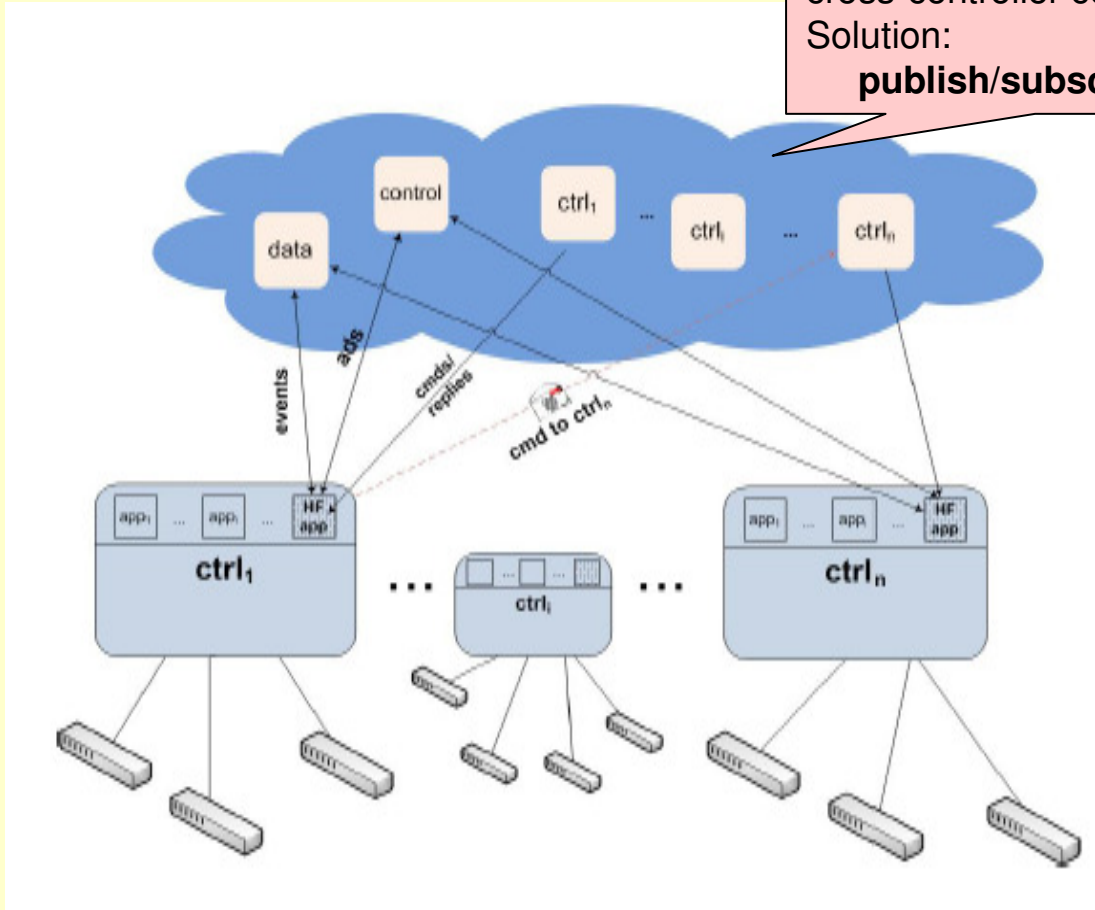
## 3. Control Plane Scalability in SDN



- **Solution proposals examples**
  - **HyperFlow (cont'd)**
  - *HyperFlow* is partially similar to *FlowVisor* but difference exist
    - ***FlowVisor* enables multiple controllers in an OpenFlow network by slicing network resources and delegating the control of each slice to a single controller.**
    - **Alternative design** : keep the controller state in a distributed data store (e.g., a DHT) and enable local caching on individual controllers
      - Even though a decision (e.g., flow path setup) can be made for many flows by just consulting the local cache, inevitably some flows require state retrieval from remote controllers → spikes in the CPI service time.
      - This design requires modifications to applications to store state in the distributed data store
    - In contrast,
      - **HyperFlow proactively pushes state to other controllers**, thereby enabling individual controllers to locally serve all flows.
      - **HyperFlow's operation is transparent** to the control applications

- Solution proposals examples
- HyperFlow (cont'd)
  - High level view

Event propagation system for cross-controller communication  
 Solution:  
**publish/subscribe system**





## 3. Control Plane Scalability in SDN



- Solution proposals examples
- HyperFlow (cont'd)
  - Design Principles
    - A HyperFlow (HF) -based network is composed by
      - **OpenFlow switches/forwarders**
      - **NOX controllers** (decision elements) each running an instance of the HF controller application
      - **Event propagation system** for cross-controller communication.
    - All the controllers have a consistent network-wide view and run as if they are controlling the whole network.
    - They **all run the exact same controller software and set of applications.**
    - Each switch is connected to the closest controller
    - Controller failure → affected switches must be reconfigured to connect to an active nearby controller
    - **Each controller**
      - directly manages a subset of switches connected to it
      - indirectly programs or queries the rest (via comm.. with other controllers).





## 3. Control Plane Scalability in SDN



- **Solution proposals examples**
- **HyperFlow**
  - **Design Principles (cont'd)**
    - **To achieve a consistent network-wide view among controllers,**
      - the HF controller appl. instance in each controller selectively publishes the events (that change the state of the system) through a **publish/subscribe system**.
      - Other controllers replay all the published events to reconstruct the state.
    - **Design choices motivation**
    - (a) **Network events → changes to the network-wide view of controllers**
      - A single event may affect the state of several applications, → control traffic (for direct state synchronization) grows with the number of applications, but is bounded to a small number of events in HyperFlow solution



## 3. Control Plane Scalability in SDN



- **Solution proposals examples**
- **HyperFlow**
  - **Design Principles (cont'd)**
    - (b) **Only a low percentage of network events → changes to the network-wide view** (~ tens of events per sec. for networks of  $10^{**3}$  of hosts ).
      - The majority of network events (i.e., packet in events) only request service (e.g., routing).
    - (c) **The temporal ordering of events, except those targeting the same switch, does not change the network-wide view**
    - (d) **HF does require a minimal appl. modification** : only need to dynamically identify the events which affect their state (unlike direct state synchronization which requires each application to directly implement state synchronization and conflict resolution)



## 3. Control Plane Scalability in SDN



- **Solution proposals examples**
- **HyperFlow**
  - **Event Propagation (cont'd)**
    - HF uses publish/subscribe (P/S) messaging paradigm.
    - It provides
      - **persistent storage** of published events (to provide guaranteed event delivery)
      - **keep the ordering of events** published by the same controller,
      - **resiliency** against network partitioning (i.e., each partition continue its operation independently and upon reconnection, partitions must synchronize).
    - The P/S minimizes the cross-site traffic required to propagate events
      - i.e., controllers in a site should get most of the updates of other sites from nearby controllers to avoid congesting the cross-region links.
    - Finally, the system enforces access control to ensure authorized access



## 3. Control Plane Scalability in SDN



- **Solution proposals examples**
- **HyperFlow**
  - **Event Propagation (cont'd)**
  - The HF P/S uses **WheelFS** - a distributed file system offering flexible wide-area storage for distributed applications
    - WheelFS → appl-s may control: consistency, durability, and data placement according to their requirements via semantic *cues* (embedded in the pathnames) to change the FS behaviour
    - In WheelFS, *channels* are represented with *directories* and *messages* with *files*.
  - **Each controller subscribes to three channels: *data channel, control channel, and its own channel.***
    - **All the controllers may to publish to all channels and subscribe to the three channels mentioned.**
    - The HF application publishes to the *data channel* selected local network and application events (general interest ones)
    - Events and OF commands targeted to a specific controller are published in the respective controller's channel.
    - Each controller periodically advertise itself in the *control channel* to facilitate controller discovery and failure detection.
    - Access control for these channels are enforced by the P/S system.



## 3. Control Plane Scalability in SDN



- **Solution proposals examples**
  - **HyperFlow (cont'd)**
  - **Controller Application- Original Implementation ( see refs)**
  - HF application is a C++ NOX appl. ensuring all the controllers have a consistent network-wide view.
    - Each controller runs an instance of the HF appl.
    - Minor changes are required for the core controller code, mainly, to provide appropriate hooks to intercepts commands and serialize events.
  - **Requirements on Controller Applications**
    - Event reordering
    - Correctness
    - Bounded number of possibly effective events
    - Measurement applications
    - Interconnecting HyperFlow-based OpenFlow networks
  - **Major Functions:**
    - Initialization
    - Publishing events
    - Replaying events
    - Redirecting commands targeted to a non-local switch
    - Proxying OpenFlow messages and replies
    - Health checking



## 3. Control Plane Scalability in SDN



- **Solution proposals examples**

- **HyperFlow (cont'd)**

- **Evaluation :**

- Assumption: sufficient control bandwidth, to bound the window of inconsistency among controllers by a factor of the delay between the farthest controllers,
- The network changes must occur at a rate lower than 1000 events per second across the network.



### 3. Control Plane Scalability in SDN



- **Solution proposals examples:**

- **Kandoo**

- *Source: S. H. Yeganeh and Y. Ganjali, "Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications," Proc. HotSDN '12 Wksp., 2012*

- **A hierarchical framework for preserving scalability without changing switches.**
- **Two layers of controllers:**
  - (i) **top layer** - logically centralized controller that maintains the network-wide state
  - (ii) **bottom layer** - group of controllers with no interconnection, and no knowledge of the network-wide state,
    - Controllers at the bottom layer run only local control applications (i.e., applications that can function using the state of a single switch) near datapaths.
    - They handle most of the frequent events and effectively shield the top layer.
- Kandoo's design enables network operators to replicate local controllers on demand and relieve the load on the top layer, which is the only potential bottleneck in terms of scalability.
- Evaluations : Kandoo allows ~10 times lower control channel consumption compared to normal OpenFlow networks.



# 3. Control Plane Scalability in SDN

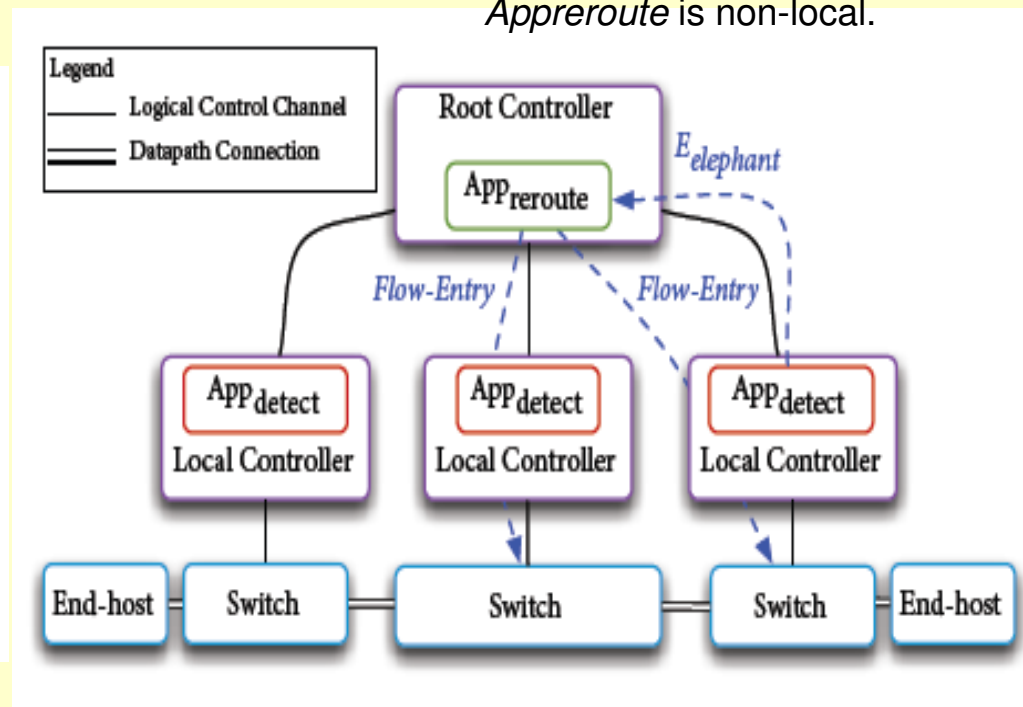
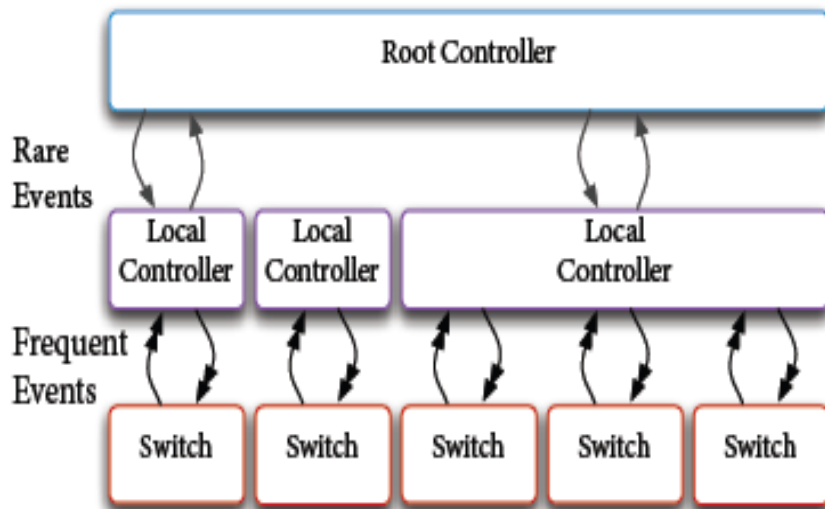


- Solution proposals examples:
- Kandoo (cont'd)

## Two Levels of Controllers

Local controllers handle frequent events, while a logically centralized root controller handles rare events.

Simple design - example  
Each switch - one local controller  
The root controller controls the local ones  
Two control applications:  
*Appdetect* is a local control  
*Appreroute* is non-local.





# 3. Control Plane Scalability in SDN

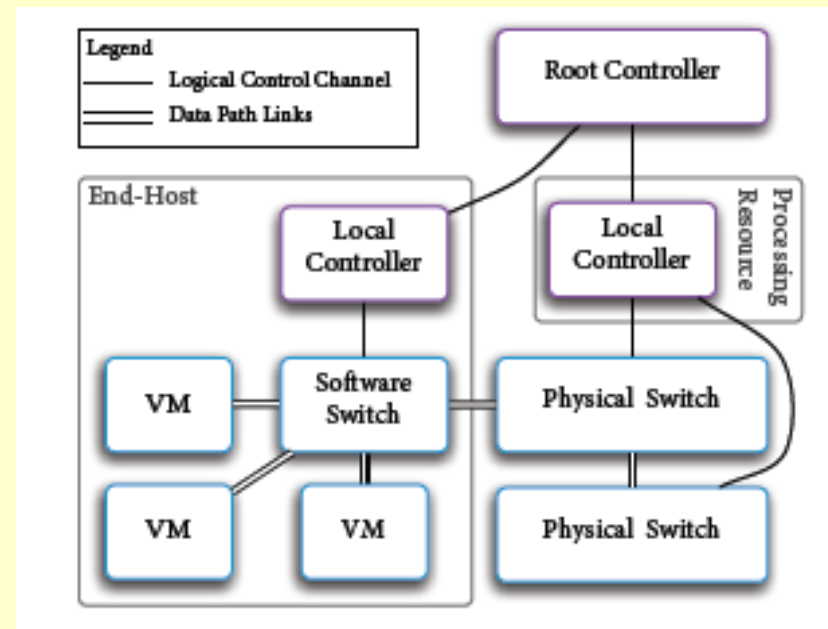
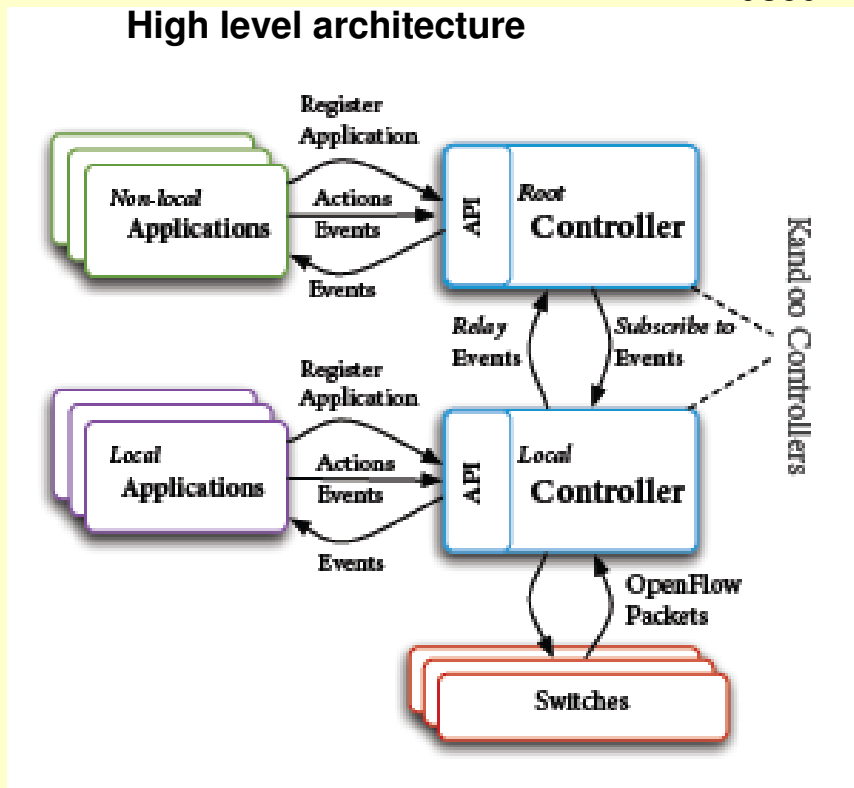
- Solution proposals examples:
- Kandoo (cont'd)

### Kandoo in a virtualized environment

For SW switches the same end-host for local controllers can be used

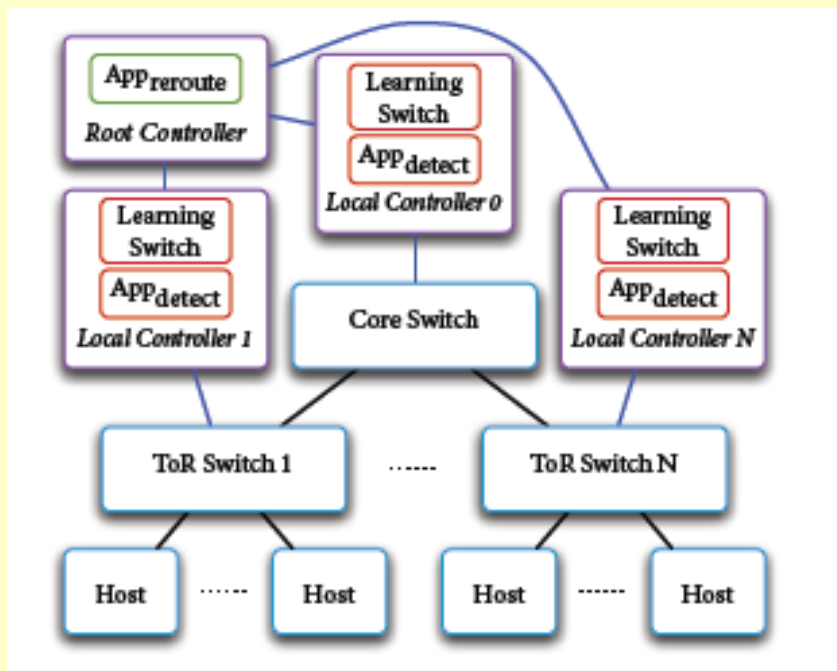
For physical switches separate processing resources are used

### High level architecture

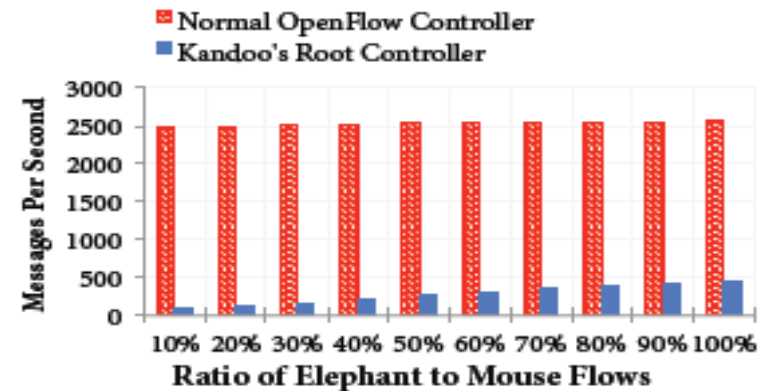


### 3. Control Plane Scalability in SDN

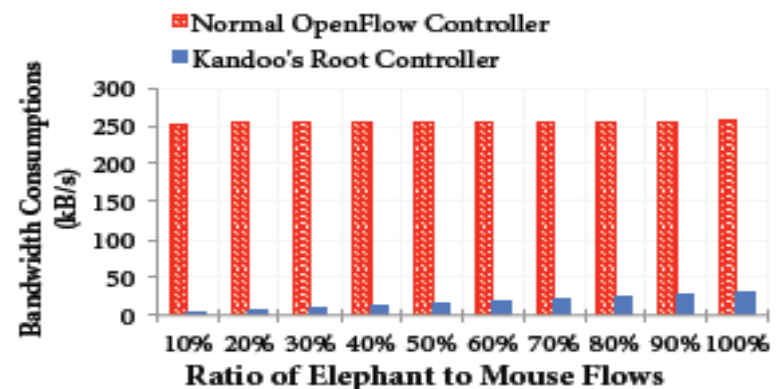
- Solution proposals examples:
- **Kandoo (cont'd)**
  - Experiment example for evaluation:
    - tree topology



CPI load for the Elephant Flow Detection Scenario. The load is based on the number of elephant flows in the network.

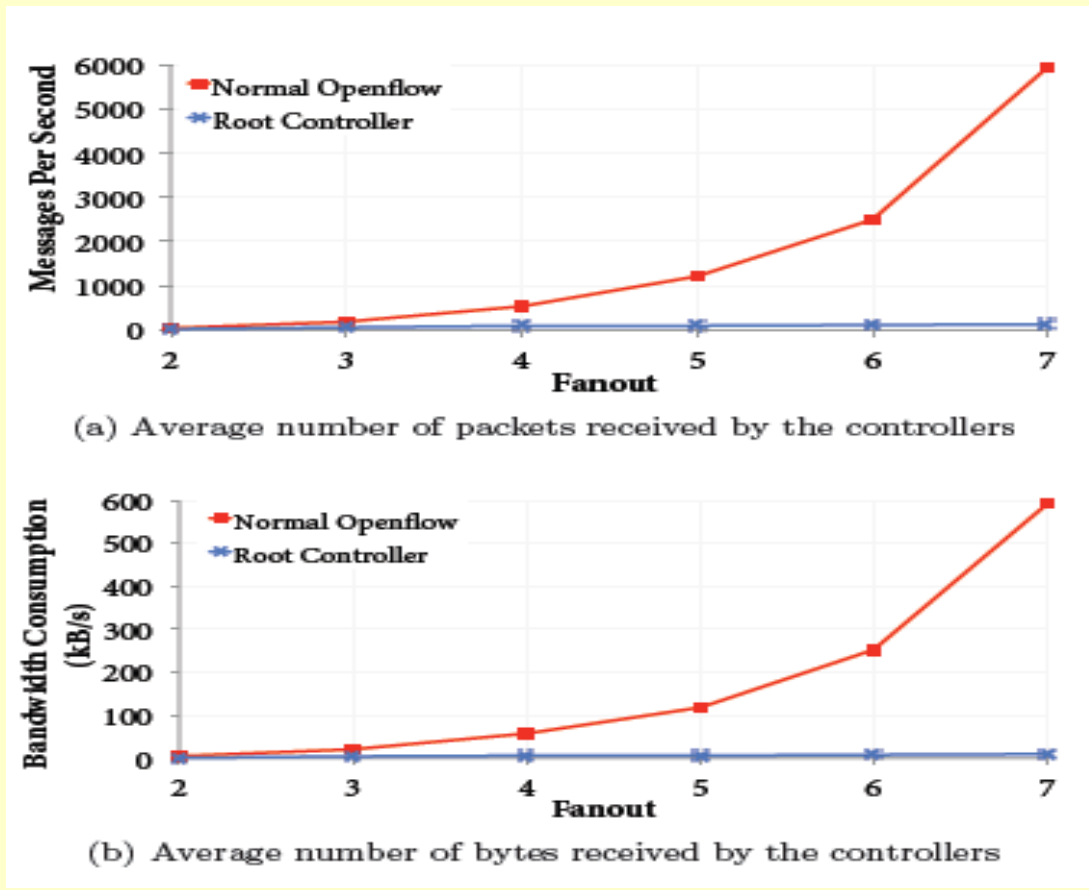


(a) Average number of messages received by the controllers



(b) Average number of bytes received by the controllers

- Solution proposals examples:
- Kandoo (cont'd)
  - CPI Load for the Elephant Flow Detection Scenario. The load is based on the number of nodes in the network.





## 3. Control Plane Scalability in SDN



### ■ Solution proposals examples

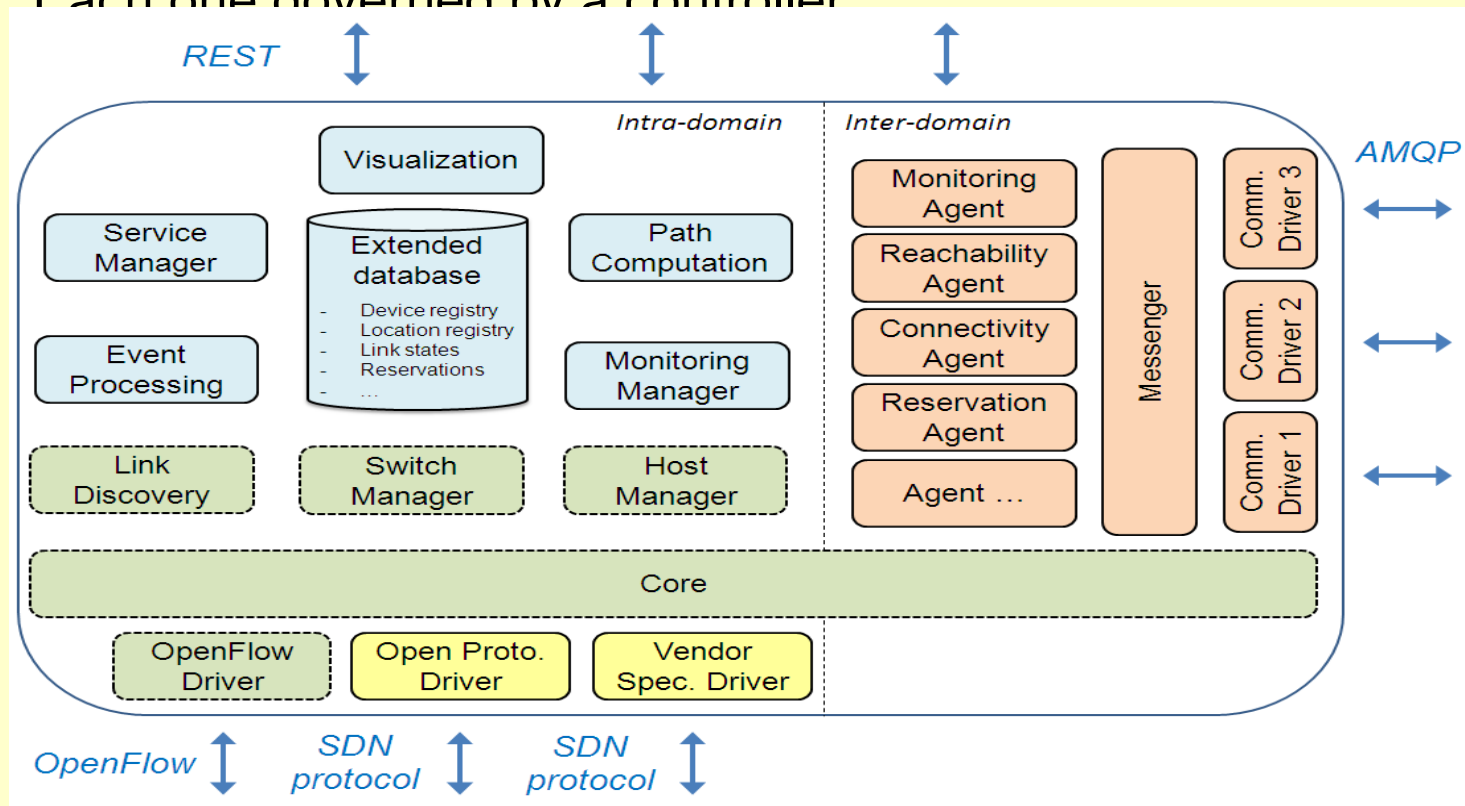
#### ■ DISCO

- *K.Phemius, et.al., DISCO: Distributed Multi-domain SDN Controllers, 2013, <http://arxiv.org/pdf/1308.6138.pdf>*

- DISCO considers distributed and heterogeneous nature of modern overlay networks and WANs
- **Each DISCO controller manage its own network domain; but they inter-communicate to provide E2E network services.**
- **The communication is based on a unique lightweight and highly manageable control channel used by agents to self-adaptively share aggregated network-wide information.**
- Implemented on top of the Floodlight OpenFlow controller
- Using AMQP (*Advanced Message Queuing Protocol – OASIS Standard*)
- It was shown that DISCO's CPI can
  - dynamically adapt to heterogeneous network topologies
  - while being resilient enough to survive to disruptions and attacks
  - and providing classic functionalities such as end-point migration and network-wide TE

### 3. Control Plane Scalability in SDN

- Solution proposals examples
- DISCO Architecture
  - Several SDN domains : A,B,C, ...linked via WAN
  - Each one governed by a controller





## 3. Control Plane Scalability in SDN



- **Solution proposals examples:**
- **IETF standardization**
  - Early stage yet; Some drafts have been elaborated but not voted
  - Example: Summary SDNi proposal
    - **SDNi : IETF draft proposal (2012) of a protocol for interfacing SDN domains.**
    - SDNi: A Message Exchange Protocol for Software Defined Networks (SDNi) across Multiple Domains: [draft-yin-sdn-sdni-00.txt](#)
    - It supports coordinating behaviors between SDN controllers
    - SDNi should be implemented by the NOS
    - SDNi protocol should be able to coordinate flow setup originated by applications, transporting information such as path requirement, QoS, SLA etc. across multiple SDN domains; exchange reachability information to facilitate inter-SDN routing.
    - Applications/SDN controllers that run in the NOS can use this protocol (i.e. what is the API to use it) for various purposes- outside the protocol specifications.
    - Possible way to implement SDNi : BGP extension, SIP over SCTP



## 3. Control Plane Scalability in SDN



- **Solution proposals examples:**
- **IETF standardization**
  - **Messages types examples exchanged via SDNi:**
    - **Reachability update**
    - **Flow setup/tear-down/update request** (including application capability requirement such as QoS, BW, latency etc.)
    - **Capabilities update** (including network related capabilities : BW, QoS etc. and system and software capabilities available inside the domain).



## 3. Control Plane Scalability in SDN



### ■ Software Defined Internet Architecture

- **SDN is a promise for enhancing the networking flexibility and performance**
- **Going further: define a flexible and scalable architecture “software defined”**
- Recent proposal:
- (\*) *Source: B. Raghavan, T. Koponen, A. Ghodsi, M. Casado, S. Ratnasamy, S. Shenker Software-Defined Internet Architecture: Decoupling Architecture from Infrastructure, Hotnets '12, October 29–30, 2012, Seattle, WA, USA.*
- **General comment: attempts to solve incrementally the Internet deficiencies, including “clean slate” ones – had limited success**
- **Main SDIA ideas:**
  - Make the architectural evolution more flexible and scalable through software
  - By decoupling the architecture w.r.t infrastructure
  - Authors (\*) claim that even after recent advances (including “clean slate-ICN/CCN, etc. and SDN) the *architecture* remained *coupled* with *infrastructure*
    - Architecture: IP protocols and packet handling rules
    - Infrastructure: PHY equipments
  - Coupling means that changes at IP level will need some changes in the routers (e.g. because lack of ASIC flexibility)





## 3. Control Plane Scalability in SDN



- **Software Defined Internet Architecture (cont'd)**
  - OpenFlow has increased the flexibility but still does not solve the decoupling;architecture/infrastructure
  - to support a wide range of architectures, the forwarders should support very general set of matching rules and fwd. actions.
    - Big header size, cost
- Proposal in (\*) considers useful features of several technologies and tries combine them in an intelligent way as to realize that decoupling:
  - **MPLS** : (distinction :edge/core, partial separation DPI/CPI )
  - **SDN**: separation CPL/DPI, I/F through which the CPI can program the forwarders
  - **Middleboxes** (perform tasks beyond IP fwd.)
  - **SW forwarding** (based on fast processors)- the other extreme is ASIC based routers ( highest ratio cost/perf)



## 3. Control Plane Scalability in SDN



- **Software Defined Internet Architecture (cont'd)**
  - **Data Plane (DPI) splitted in**
    - **Core network ( its own addressing scheme)**
    - **Edge network**
  - **Architectural dependencies – placed at the edges**
  - **SW forwarding in the edge (assure flexibility)**
  - **Control Plane (CPI) uses SDN-like control to edge routers (can be OpenFlow- based but not mandatory)**
  - **Each core network domain has its own design**
  - **The approach allows a top-down perspective**
    - Still SDN style of control is proposed
    - Openflow or equivalent is needed to be standardized
    - However – no need to specify beforehand the behavior of each box- because the controller assures interoperation



## 3. Control Plane Scalability in SDN



- **Software Defined Internet Architecture (cont'd)**
- Top-down perspective
  - **Tasks ( to get E2E connectivity):**
    - *Interdomain: Domain A-Domain B*
    - *intradomain transit*
    - *intradomain delivery ( from domain edges to/from hosts or between hosts)*
  - **Main suggestion: separation between intradomain and interdomain addressing**
  - **interdomain addressing :**
    - some form of domain identifiers, to support interdomain task
    - no ref. to any intradomain addresses (each domain can choose its own internal addressing scheme)
    - this important choice can be solved in “clean-slate” style or some specific solutions can be applied ( e.g. using the IPv6 flow ID as the interdomain Id.)
  - **each domain is represented by a single logical server in the algorithm to compute interdomain routes**
    - the server may be replicated for reliability, but a single logical entity represents the domain for interdomain routing algorithm
      - Routing alg.: BGP-like or *any other new algorithm*



## 3. Control Plane Scalability in SDN



- **Software Defined Internet Architecture (cont'd)**
- **Intradomain tasks: *edge-to-edge transit, edge-to-H delivery, and H-to-H delivery***
  - -implemented independently w.r.t. interdomain task
  - different domains can use different implementations for intradomain tasks ( e.g MPLS)
  - the core can use *any internal fwd and control plane* ( SDN.... traditional protocols)
  - each domain's core can use their own internal addressing scheme.
- **The edge uses SW fwd.**
  - commodity processors managed by an SDN edge controller
  - SDN edge controller knows the core requirements to insert the appropriate packet headers to achieve internal or edge delivery.
- Result: highly modularity and scalability



## 3. Control Plane Scalability in SDN



- **Software Defined Internet Architecture (cont'd)**
  - **Advantages:**
    - Only the edge routers need to understand interdomain addressing
    - Core routers need to understand intradomain addressing in their domain only
    - Only the edge-controller participate in the interdomain route computation
    - Only the core Cpl needs to determine the internal routes
    - The only components needed to forward packets based on interdomain addresses are edge routers, which use software forwarding.
  - **Result: high architectural freedom**
  - Question: SW fwd- is it realistic in this context?
    - apparently yes , encouraging results [ ]: longest-prefix match forwarding on minimum-sized packets, including checksum verification and TTL adjustment, can be done at 6.7Gbps on a single 3.3Ghz core.

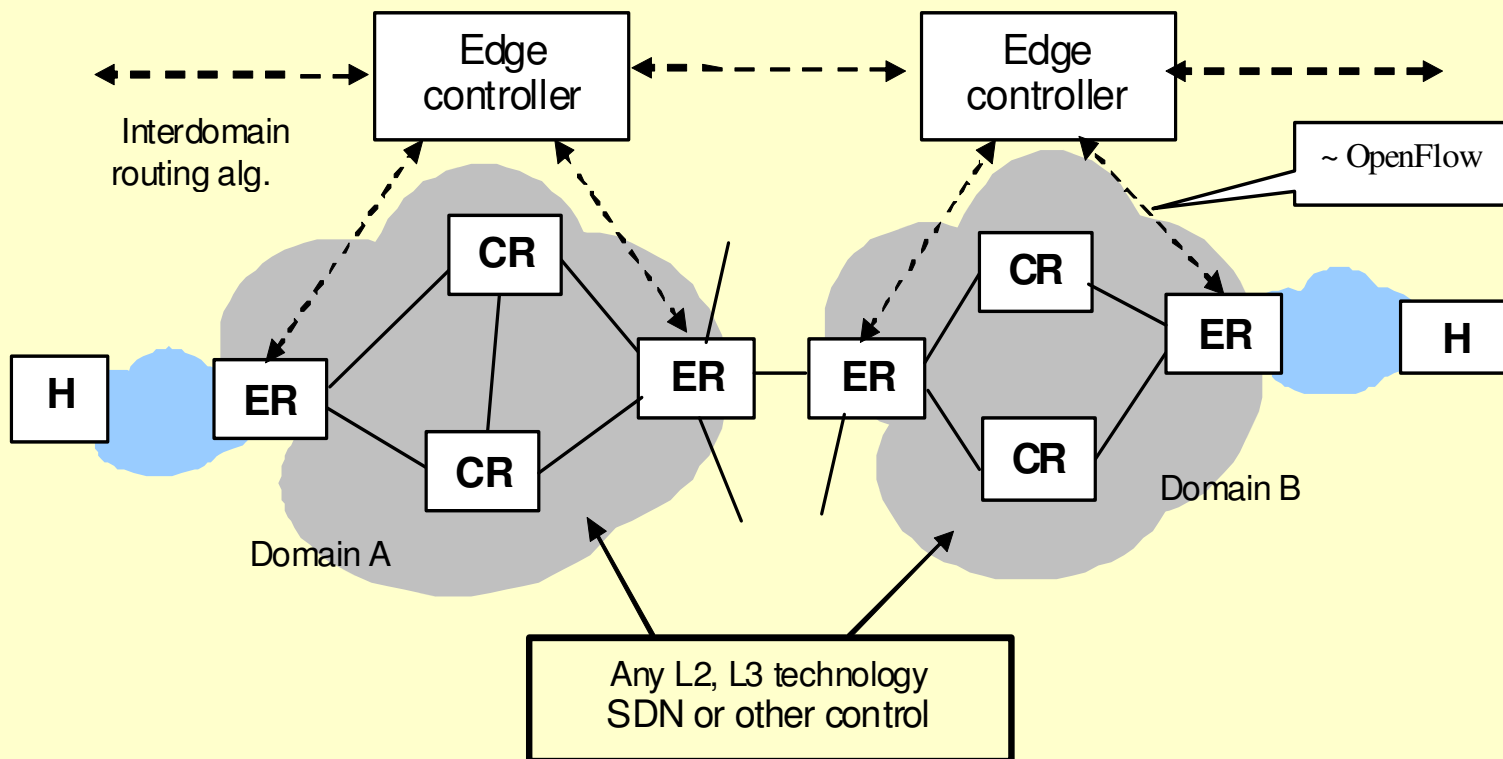


## 3. Control Plane Scalability in SDN



- **Software Defined Internet Architecture (cont'd)**
- **SDIA defines an Interdomain Service Model (ISM) [ ]:**
  - only edge controllers (one per domain) are involved in interdomain task
  - and edge routers (controlled by the edge controller)
- **Implications:**
  - Interdomain routing changes ( e.g. BGP to other) only involve changing SW in the edge controllers
  - Changing how domains are addressed → a change only to the controller SW
  - Changing how hosts are addressed, (e.g. IP to IPv6), is done per domain.
- **ISM in SDIA main requirements:**
  - A distributed interdomain algorithm between *the edge controllers* that computes whatever state the controllers need to implement the service model; (e.g. BGP)
  - A set of forwarding actions to be sent to the edge routers by the edge-controllers.
  - Allow incremental/partial deployment; need a basic unicast packet-delivery ISM (such as supplied by IP and BGP), so that non-peering domains can set up tunnels with each other
  - a discovery mechanism : domains participating in an ISM are aware of each other.

- **Software Defined Internet Architecture (cont'd)**
  - Illustration of the ISM principles





## 3. Control Plane Scalability in SDN



- **Scalability in different network types**
- **Data Centers (DC)**
  - A typical DC network has  $\geq 10000$  switching elements
  - High no. of control events  $\rightarrow$  overload any centralized controller.
  - Solutions: proactively install rules on switches, effectively eliminating most control requests before they enter the control plane.
  - Cost : loss of precision and reactivity in the controller.
  
  - When an application requires accurate flow statistics and/or reactivity, one can deploy the application close to the switches.
    - E.g. frequent events can be delegated to processes running on end hosts as long as access to global state is minimized.
  
  - Availability of processing resources in data centers  $\rightarrow$  e.g. Kandoo can be used to reach arbitrary scalability levels.
  - Distributed controllers (e.g., HyperFlow or Onix) can also be reasonable solutions
  
  - Low latency in DC nets  $\rightarrow$  synchronization of state and flow setup latencies would be minimal and acceptable for most applications.





## 3. Control Plane Scalability in SDN




- **Scalability in different network types (cont'd)**
- ***Service Provider Networks***
  - Typically, SP networks less switches/routers w.r.t. data center networks
  - Nodes in such networks are usually geographically distributed.
  - The large network diameters → controller scalability concerns, flow setup and state convergence latencies, and consistency requirements
  - Physical network distribution → one can partition it into separate regions
    - Each partition can be controlled by an independent controller
    - Controllers can exchange only the required state changing events, effectively hiding most events from external controllers.
  - Delay in such networks → control applications should be latency tolerant and have weak consistency requirements
  - SP nets have large numbers of flows → data path resource limits are also of concern
    - Flows aggregation can be applied but with the cost of granularity in control
  - **However such concerns are also present in traditional networks, and are not unique to SDN**



# CONTENTS



1. Software Defined Networks Architecture (Summary)
2. SDN-OpenFlow
3. Control Plane Scalability SDN
4.  SDN-like architecture example : ALICANTE Project
5. Conclusions



## 4. ALICANTE Project



- **ALICANTE**, 2010-2013, FP7 Integrated Project (IP):  
MediA Ecosystem Deployment Through Ubiquitous  
Content-Aware Network Environment- *Future Internet  
oriented project*
  
- *<http://www.ict-alicante.eu/>*
  
- *19 European partners*
  - *Industry, SME*
  - *Operators*
  - *Universities*
  - *Research groups*



## 4. ALICANTE Project



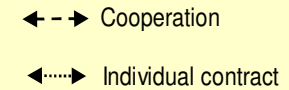
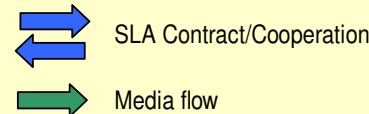
- **Research Area: Networked Media**
  - **Content Aware Networking (CAN) & Network Aware Application (NAA)**
  - **Evolutionary architecture** for networked media systems
    - Mid-way between traditional Internet solutions and full ICN
    - However the architecture was not intentionally to be full SDN
- **ALICANTE general objectives:**
  - **End users**
    - Flexible access to MM services, consume, share, generate A/V content
  - **Providers** (high level services, connectivity services)
    - extend their services range for large number of users
    - efficiently manage their services and /or resources
  - Flexible cooperation between actors
  - Media services and network resources management in multi-domain, multi-provider environment
- **Novel virtual (CAN) layer**
  - Content-Awareness delivered to Network Environment
  - Network- and User Context-Awareness to Service Environment
  - Different levels of QoS/QoE, security, etc. for media-oriented services
- **ALICANTE architecture : SDN style**

## Business Model

Business Actors:

- End-User (EU)
- Content Provider (CP)
- Service Provider (SP)
- Network Provider (NP)
- CAN Provider (CANP) (new)

Flexible Business Model : B2C, B2B, C2C and to consider new CAN features and service environment new capabilities



Cooperation, interaction:

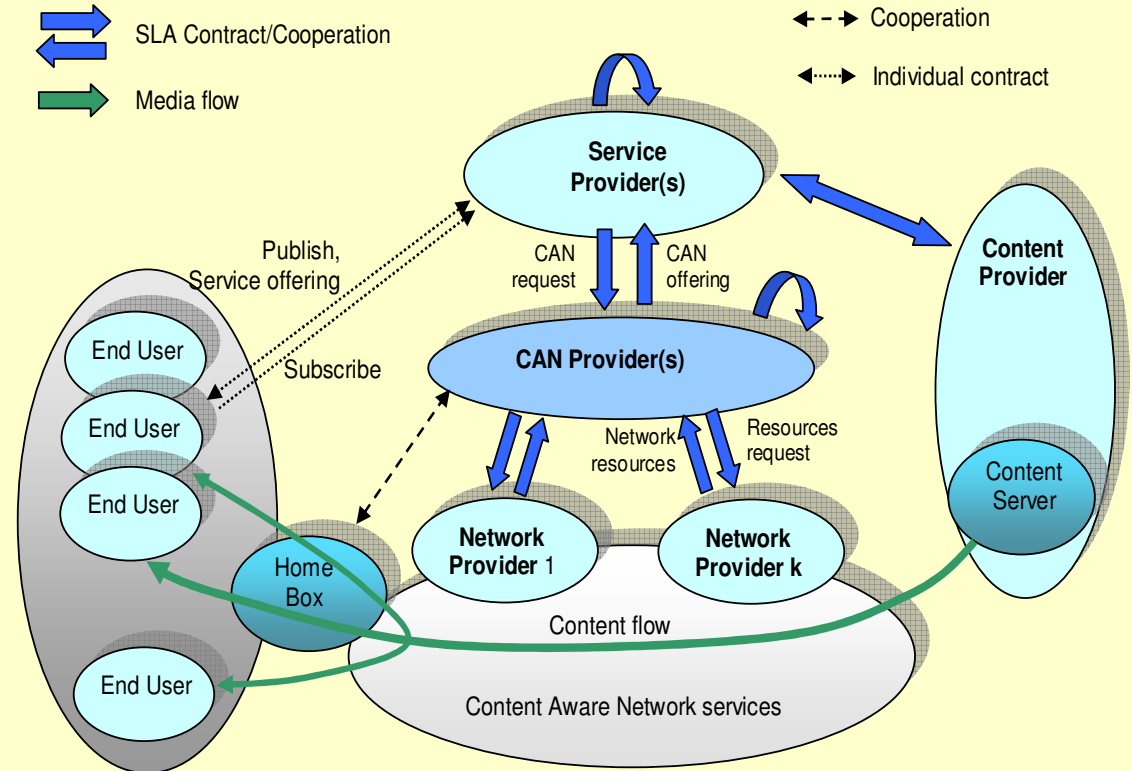
- Single/aggregated roles of SP, CP, NP, ANP, C/SCs,
- Cooperation, via static and/or dynamic SLAs
- Distributed management
- Independent resource management for each actors

**Services: Fully Managed (FM)**

**Partially managed (PM)**

**Unmanaged (UM)**

Services requirements: established by SLAs, or:  
CANP has some freedom to perform  
autonomic actions





## 4. ALICANTE Project



- **ALICANTE architecture**
- Two virtual layers,
  - **CAN layer** for virtual connectivity services on top of the the core IP network
    - Combine **resource provisioning** at CAN layer with per/flow **adaptation** solution for the multimedia flow delivery over multi-domains
    - On top of the traditional IP Network layer, **virtualising the network nodes**
  - **Home-Box** layer- content delivery
- **User Environment:** interaction of End Users with the underlying layers
- **Service Environment:** cooperation between SPs and End-Users (through their HBs)
- Hierarchical **Multi-layered monitoring** sub-system at all levels: User, Service, Home-Box, CAN, Underlying network



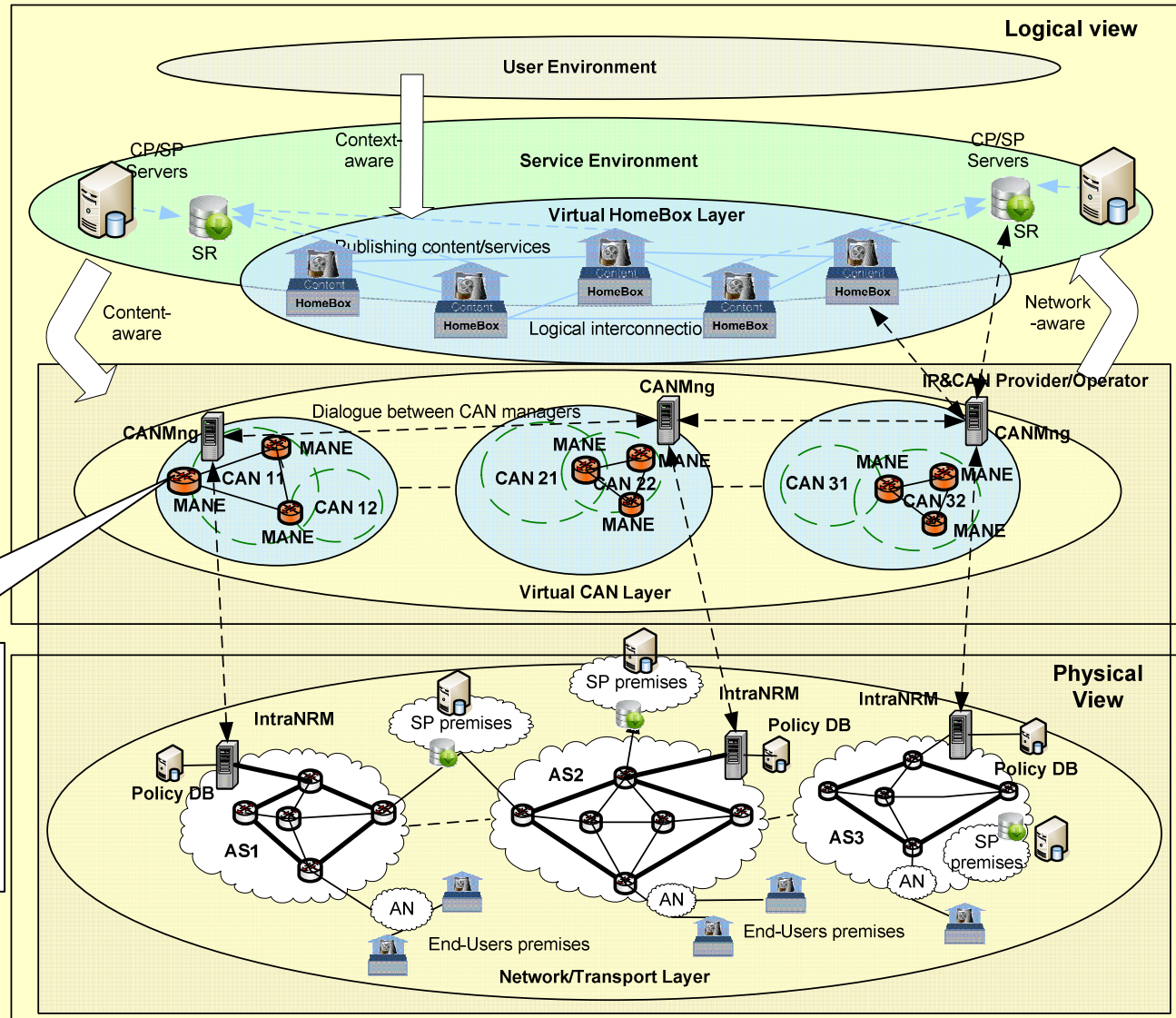
## 4. ALICANTE Project



- **ALICANTE Architecture (cont'd)**
- *mid-way architecture* : CAN/NAA logical coupling, extendable both at service level and network/ transport level
- support integration
  - *vertical* (based on CAN/NAA) of high level services and connectivity ones,
  - *horizontal integration* on top of single or multiple-domain IP networks.
- network virtualization techniques is applied
  - to create parallel *content-aware virtual planes*
    - enriched in terms of functionality (due to content –awareness)
    - represented by *Virtual Content Aware Networks (VCANs)*
      - *Constrained routing and forwarding depending on content type*
    - VCANs spanning single or multiple IP domains
- **Note: ALICANTE current architecture does not offer full network virtualization, but only in the Data Plane**

## Overall Architecture View

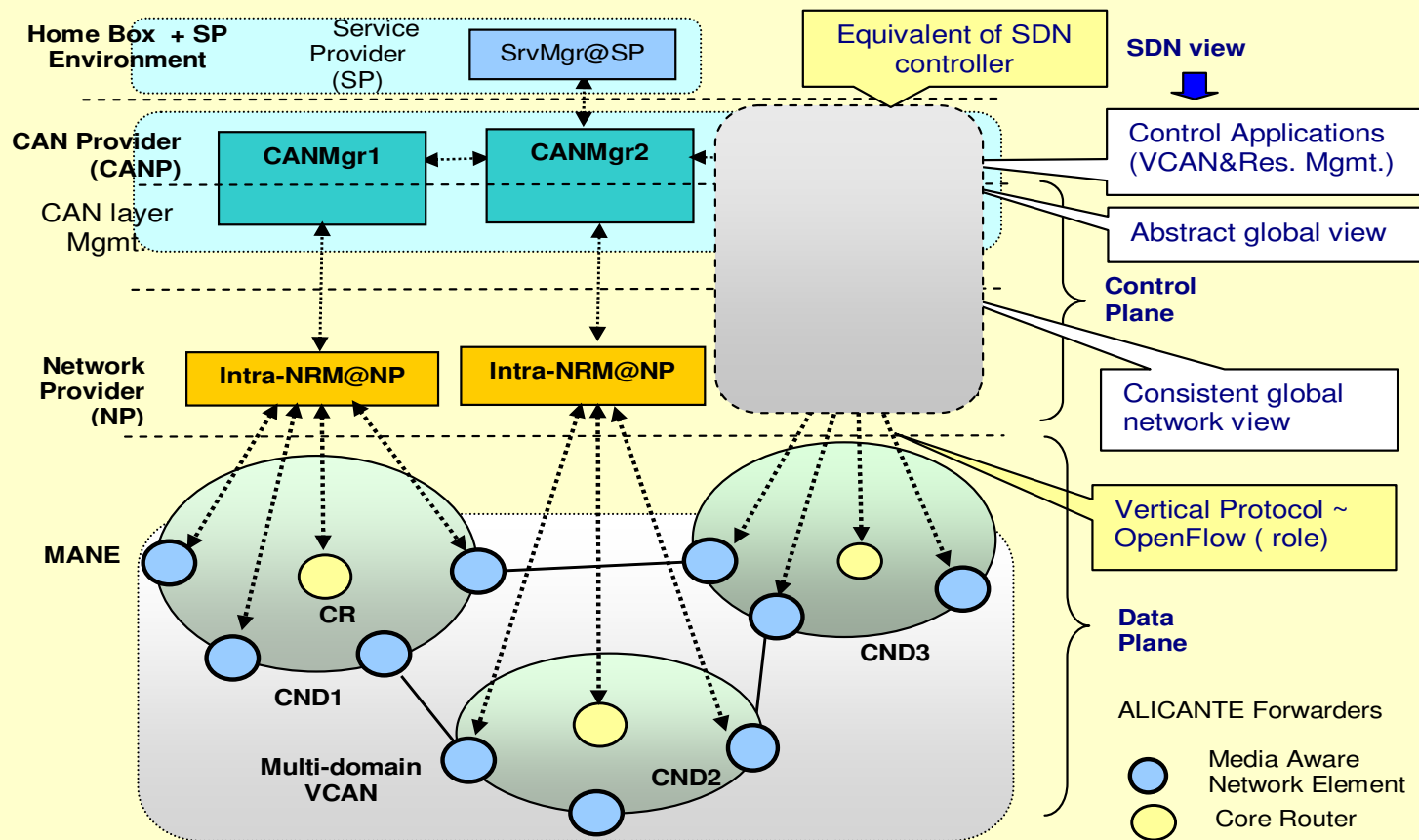
- User Env
- Service Env
- HB-layer
- Net Env
  - CAN layer
  - Infrastructure layer



MANE –  
Novel ALICANTE  
router-  
Media Aware  
Network Element




- ALICANTE Architecture- SDN mapping**
  - Multi-controller management approach assures solution scalability
  - Specific task solved by the controllers: management of Virtual Content Aware Networks





# CONTENTS



1. Software Defined Networks Architecture (Summary)
2. SDN-OpenFlow
3. Control Plane Scalability SDN
4.  SDN-like architecture example : ALICANTE Project
5. Conclusions



## 5. Conclusions



- **SDN – technology opening new perspective to networking**
  - Generally we agree with conclusions on SDN scalability published in:
    - *S. H. Yeganeh, et.al., On Scalability of Software-Defined Networking, IEEE Comm. Magazine, February 2013*
  - The scalability concerns are neither caused by nor fundamentally unique to SDN
  - Scalability issues can be addressed while still keeping the important SDN advantages
  - Current research and implementation shows that
  - However many open research issues and challenges remain



- Thank you !
- Questions?



## References-1



1. N.McKeown, T.Anderson, et. Al., OpenFlow: Enabling Innovation in Campus Networks, - <http://www.openflow.org/documents/openflow-wp-latest.pdf>.
2. M.Mendonca, et. al., A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks, <http://hal.inria.fr/hal-00825087/>
3. A. Greenberg, et. al., A clean slate approach to network control and management. ACM SIGCOMM Computer Communication Review, 35(5):41-54, 2005.
4. M. Casado, et al., Ethane: Taking control of the enterprise. ACM SIGCOMM Computer Communication Review, 37(4):1-12, 2007.
5. M. Casado, et.al., Fabric: a retrospective on evolving sdn, Proceedings of the first workshop on Hot topics in software defined networks, HotSDN '12, pages 85-90, New York, NY, USA, 2012. ACM.
6. Open networking foundation, <https://www.opennetworking.org/about>.
7. Open Networking Research Center (ONRC). , <http://onrc.net>.
8. Open vswitch and ovs-controller, <http://openvswitch.org/>.
9. Pantou: Openow 1.0 for openwrt, <http://www.openow.org/wk/index.php/>
10. OpenFlow switch specification v1.0. , <http://openflow.org/wp/documents/>



## References-2



11. OpenFlow Switch Specification, V 1.3.0 (Wire Protocol 0x04 ) June 25, 2012
12. HP SDN/Openflow Technology Solutions:  
<http://h17007.www1.hp.com/us/en/solutions/technology/openflow/index>
13. SDN Controller Product Fact Sheet:  
<http://h17007.www1.hp.com/docs/interopny/4AA4-3881ENW.PDF>
14. SDN for cloud providers and enterprises:  
<http://h17007.www1.hp.com/docs/interopny/4AA4-3872ENW.pdf>
15. SDN Technical White Paper  
<http://h17007.www1.hp.com/docs/interopny/4AA4-3871ENW.pdf>
16. Software-Defined Networking: The New Norm for Networks ONF White Paper April 13, 2012
17. SDN: the service provider perspective, Ericsson Review, February 21, 2013
18. S. H.Yeganeh, et.al., On Scalability of Software-Defined Networking, IEEE Comm. Magazine, February 2013
19. A. Tootoonchian et al., On Controller Performance in Software-Defined Networks, Proc. USENIX Hot-ICE '12, 2012, pp. 10–10
20. M. Yu et al., Scalable Flow-Based Networking with DIFANE, Proc. ACM SIGCOMM 2010 Conf., 2010, pp. 351–62.



## References-3



21. A. R. Curtis et al., DevoFlow: Scaling Flow Management for High-Performance Networks, Proc. ACM SIGCOMM '11, 2011, pp. 254–65.
22. T. Koponen et al., Onix: A Distributed Control Platform for Large-Scale Production Networks, Proc. 9th USENIX, OSDI Conf., 2010.
23. S. H. Yeganeh and Y. Ganjali, Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications, Proc. HotSDN '12 Wksp., 2012.
24. A. Tootoonchian et al., Hyperflow: A Distributed Control Plane for OpenFlow, Proc. 2010 INMConf., 2010.
25. B. Raghavan, T. et al., Software-Defined Internet Architecture: Decoupling Architecture from Infrastructure, Hotnets '12, October 29–30, 2012, Seattle, WA, USA
26. W. Liu, J. Ren, J. Wang, IETF, Draft-icn-implementation-sdn-00, A Unified Framework for Software-Defined Information-Centric Network, August 09, 2013
27. J. Choi, Jinyoung Han, E. Cho, Ted Kwon, and Y. Choi, A Survey on Content-Oriented Networking for Efficient Content Delivery, IEEE Communications Magazine • March 2011
28. D. Kutscher, B. Ahlgren, H. Karl, B. Ohlman, S. Oueslati, I. Solis, Information-Centric Networking— Dagstuhl Seminar — 2011
29. Operator Network Monetization Through OpenFlow-Enabled SDN, ONF Solution Brief, April 3, 2013, <https://www.opennetworking.org/images/stories/downloads/sdn-resources/solution-briefs/sb-network-monetization.pdf>
30. G. Pavlou, Information-Centric Networking: Overview, Current State and Key Challenges, IEEE ISCC 2011 Keynote <http://www.ee.ucl.ac.uk/~gpavlou/>



## References-4



31. H. Koumaras, et. al., Media Ecosystems: A Novel Approach for Content-Awareness in Future Networks, Future Internet: Achievements and Promising Technology, Springer Verlag, pp.369-380, May 2011
32. E. Borcoci, et. al., Resource Management in Multi-Domain Content-Aware Networks for Multimedia Applications, Int'l. Journal on Advances in Networks and Services", vol 5 no 1 & 2, year 2012, [http://www.iariajournals.org/networks\\_and\\_services/](http://www.iariajournals.org/networks_and_services/)