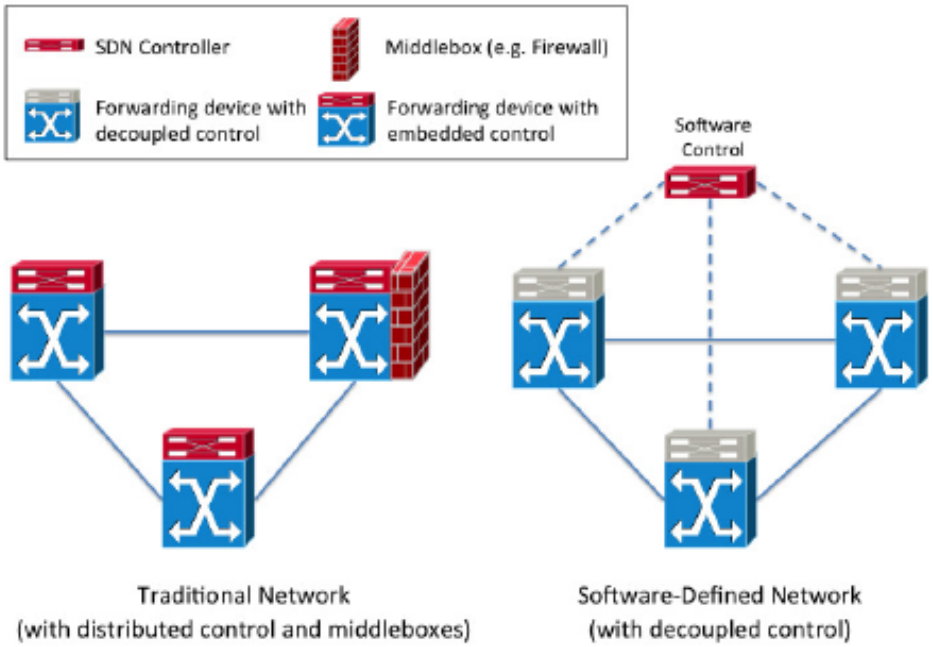
DIstributed **S**DN **CO**ntrollers for rich and elastic services

ANR-13-INFR-013

M1: New SDN paradigms analysis and survey of recent contributions

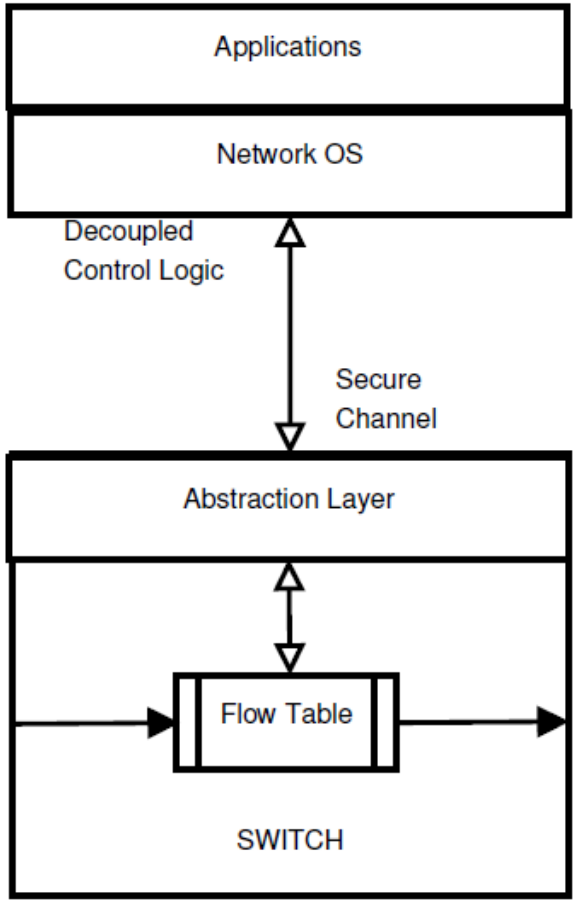
Software Defined Networks (SDN)

- Separation between Control and Data
- Communication between Control- and Data-plane (E.g. via OpenFlow protocol)



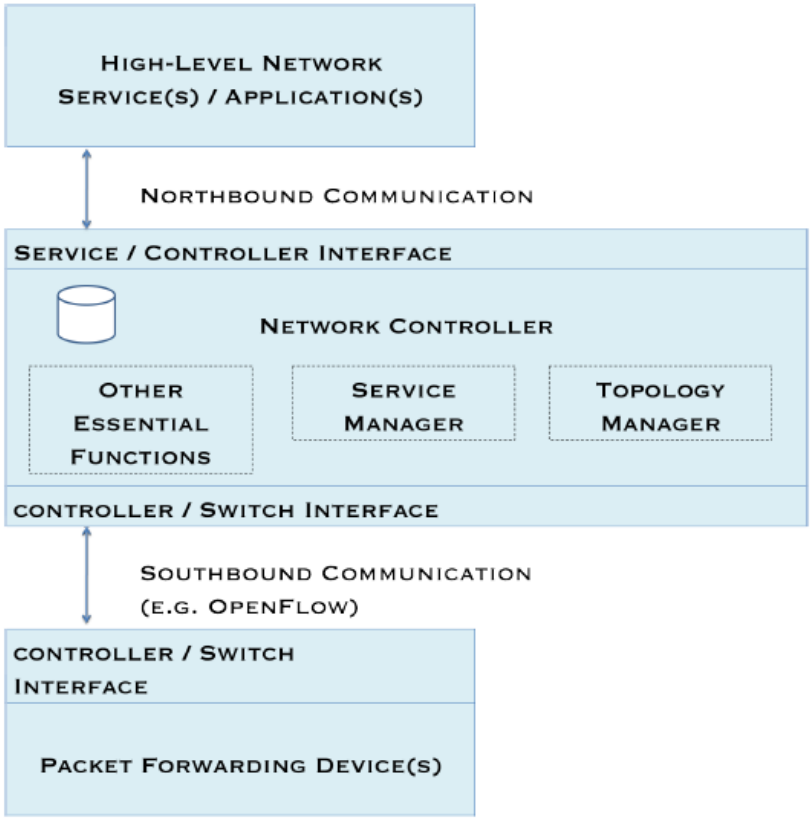
Network Operating System - NOS

- The separated control logic can be viewed as a network operating system (NOS).
- Applications can be built to "program" the network



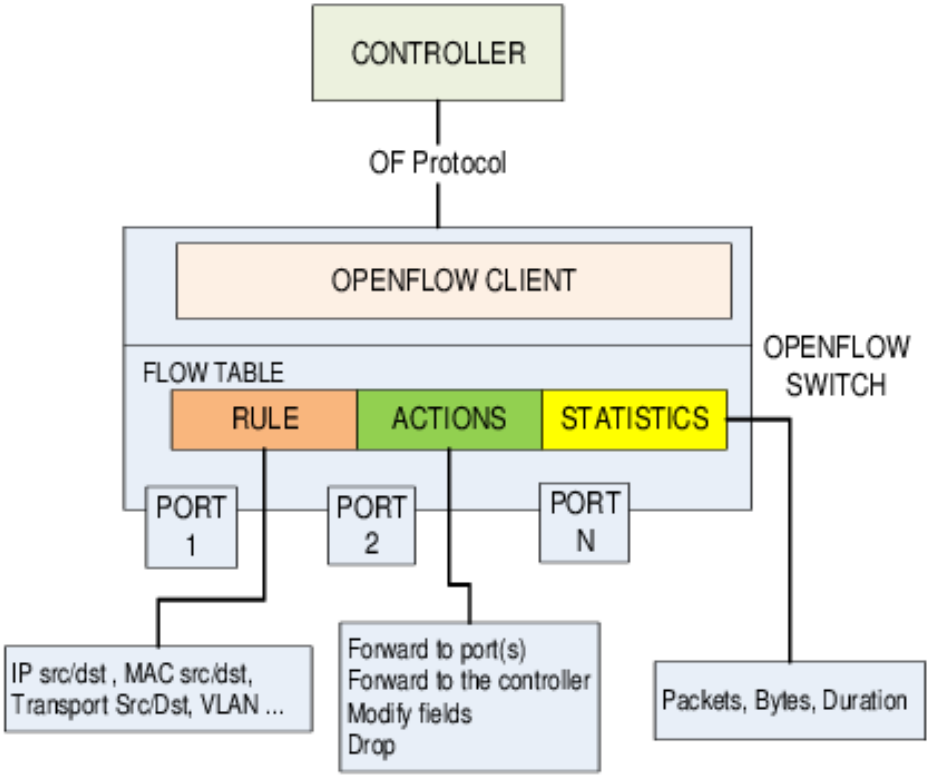
Network Operating System - NOS

- **North-Bound Interface**
  - e.g., Procera, Frenetic, FML, Nettle.
- **South-Bound Interface**
  - E.g. OpenFlow, ForCES.



OpenFlow

- **Flow entries typically consist of:**
  - (1)match fields, (2)counters,(3)set of instructions



Forwarding Device

- Underlying network infrastructure may involve:
  - FORWARDING DEVICE = routers, switches, virtual switches, wireless access points, etc.
- Challenges:
  - "Mice Flows" and "Elephant" Flows.
    - Ex: Devoflow
      - Handles mice flow on the switches.
      - Only contact the controller for Elephant flows.
  - Ternary Content-Addressable Memory (TCAM)
    - Expensive and power-hungry.
  - Optimizing memory usage.

Software Switches

- Software Switches Implementations compliant with OpenFlow Standard:

Software Switch	Implementation	Overview	Version
Open vSwitch [15]	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 [16]	C	Turns a commercial wireless router or Access Point into an OpenFlow-enabled switch.	v1.0
ofsoftswitch13 [12]	C/C++	OpenFlow 1.3 compatible user-space software switch implementation.	v1.3
Indigo [7]	C	Open source OpenFlow implementation that runs on physical switches and uses the hardware features of Ethernet switch ASICs to run OpenFlow.	v1.0

Control Plane - Challenges

- **Latency in the control link:**
  - "Why latency does matter". In Integrated Network Management (IFIP/IEEE IM 2013)
    - Bandwidth
      - arbitrates how many flows can be processed by the controller
    - Latency
      - major impact on the overall behavior of the network
  - "The controller placement problem", HotSDN '12
    - optimal number of controllers and their location in order to reduce latency.

Control Plane – Challenges (2)

- **Centralized vs. Distributed Control Plane**
  - Controller-to-controller communication is not defined in OpenFlow
    - Necessary for distribution and redundancy;
  - Physically centralized controller = single point of failure!
  - Ex: Onix and HyperFlow
    - logically centralized but physically distributed
    - Enable communication with local controllers
      - decreases the look-up overhead
      - Concern: Maintaining consistency between controllers!

Control Plane – Challenges (3)

- **Centralized vs. Distributed Control Plane**
  - Ex: Kandoo: Hybrid Approach!
    - Uses local controllers for local applications
    - Redirects decisions that require centralized network state to a global controller.
    - Advantages:
      - reduces the load on the global controller
      - Reduces latency for local applications.
  - Ex: DISCO: Logically decentralized
    - Inter domain e intra domain communication

Control Plane – Challenges (4)

- **Control Granularity**
  - Control can be further abstracted to an aggregated flow-match;
  - Flow aggregation may be based on:
    - source, destination, application, etc.

Control Plane – Challenges (5)

- **Reactive vs. Proactive Policies**
  - Reactive: (e.g. Ethane)
    - forwarding elements must consult a controller each time a decision must be made.
    - An issue specially for short lived flows and/or large networks!
  - Proactive: (e.g. DIFANE)
    - push policy rules from the controller to the switches.
    - Reduces control overhead and latency.

Controller Implementations

- Current Implementations:

Controller	Implementation	Open Source	Developer
POX [17]	Python	Yes	Nicira
NOX [54]	Python/C++	Yes	Nicira
MUL [9]	C	Yes	Kulcloud
Maestro [32]	Java	Yes	Rice University
Trema [21]	Ruby/C	Yes	NEC
Beacon [1]	Java	Yes	Stanford
Jaxon [8]	Java	Yes	Independent Developers
Helios [6]	C	No	NEC
Floodlight [5]	Java	Yes	BigSwitch
SNAC [20]	C++	No	Nicira
Ryu [18]	Python	Yes	NTT, OSRG group
NodeFlow [10]	JavaScript	Yes	Independent Developers
ovs-controller [15]	C	Yes	Independent Developers
Flowvisor [107]	C	Yes	Stanford/Nicira
RouteFlow [93]	C++	Yes	CPQD

References

B. Astuto, M. Mendonca, X.N. Nguyen, K. Obraczka, T. Turletti, ``A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks'', to appear in IEEE Communications Surveys & Tutorials, September 2014, http://hal.inria.fr/hal-00825087 .