Xantaro
SERVICE INTEGRATION

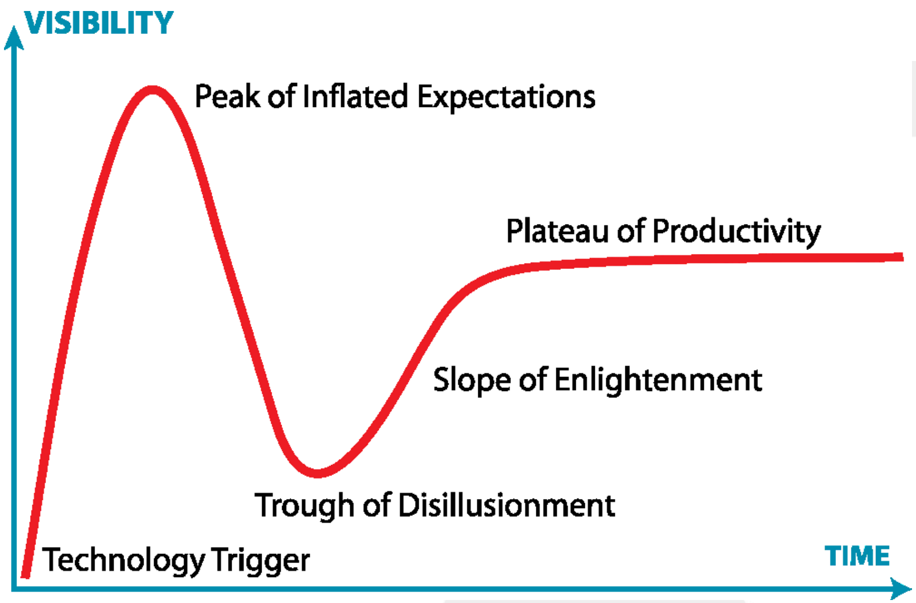
SOFTWARE DEFINED NETWORKS
REALITY CHECK

DENOG5, Darmstadt, 14/11/2013

Carsten Michel

Software Defined Networks (SDN)

- Why Software Defined Networking?
  - There's a hype in the industry!
- Dispelling some myths
  - SDN does not save CAPEX
  - SDN does not save OPEX
  - SDN is not a provisioning system or configuration management tool
  - SDN is not a new protocol
- You can't buy SDN!!
  - It's an architectural approach



(Source: Wikipedia, Hype Cycle)

Problem Statement

- Application Awareness
  - Applications implemented as over-the-top for speed, agility and avoidance of network interaction
  - Missing localization (especially true for CDN-traffic)
- Service differentiation
  - Difficult to introduce new functionality and services into the network; most often new services require additional boxes
- Flexibility in Forwarding
  - Routing based on business logical rather than shortest-path hard to implement; finest granularity in standard routing is the prefix
- …

Problem Solution by Abstraction

- How do you solve a problem?
  - Divide-and-conquer, i.e. breaking the complex problem into small solvable problems
  - Abstraction, i.e. building a model with information which is relevant to the problem
- Short primer on Abstraction: How did programming get simple?
  - Machine language has no abstraction, i.e. have to deal with low-level details
  - High-level programming languages have useful abstraction (e.g. file systems, virtual memory, abstract data types, etc.)
  - Development frameworks and state-of-the-art programming languages add even more abstraction (e.g. object orientation, garbage collection, etc.)
- Why is abstraction useful?
  - Well-defined interfaces used to instantiate abstraction
  - Freedom of implementation on both sides of the interface
  - Introduction of modular programming structure
- Complexity has not magically disappeared, but is merely hidden

Do we have Abstraction in Today‘s Networks?

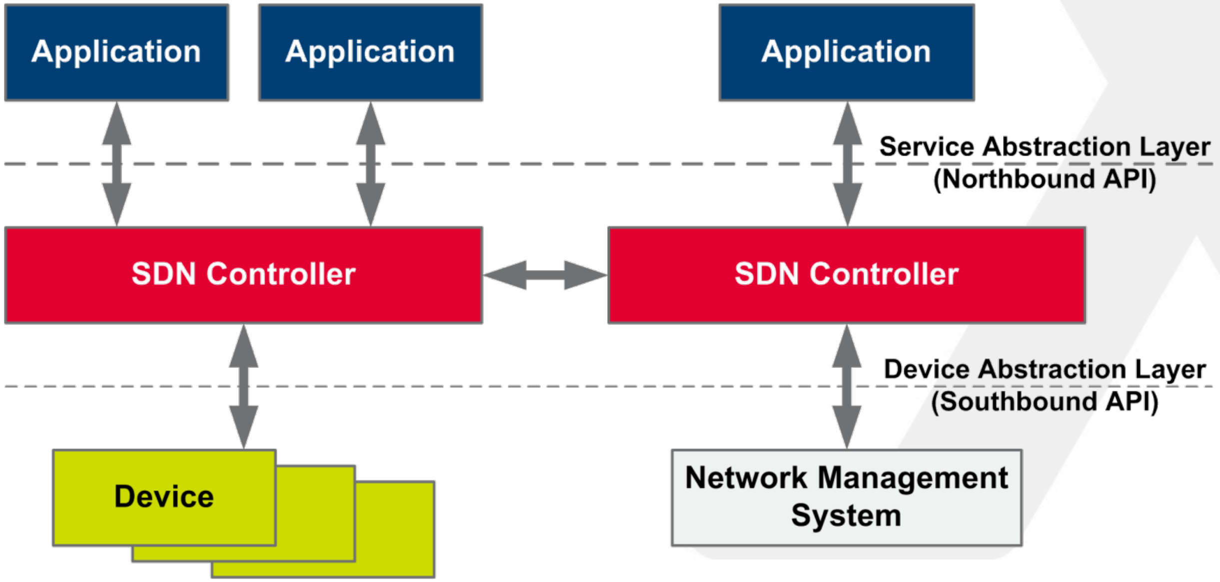
- Data plane abstraction by network layer models (physical, IP, TCP, …)
  - Good abstraction by layered model
  - Individual layer can evolve independently as long as the interface as not changed (e.g. move from 10GE to 100GE on physical layer or from IPv4 to IPv6 on layer-3)
  - Bad interfaces which violate principles of modularity
- Control plane abstraction does not exist!
  - Constantly adding complexity to get what we want
  - No clear building blocks, i.e. with every new problem we start from the scratch by defining new protocols
  - Variety of totally different mechanisms like distributed algorithms (e.g. routing protocols), isolation technology (e.g. ACLs, firewalls, …) and traffic engineering
  - Manual operator configuration on individual devices

SDN Approach to the Control Plane

- Remember, control plane is all about computing forwarding state including
  - Detecting how the network looks like globally, i.e. network topology discovery
  - Determine what has to be done to get desired functionality
  - Distribute forwarding state
- Separation of control plane and forwarding plane (not new; done before!)
  - Create data model to describe forwarding states
  - Use standard protocols and open interfaces with small set of primitives (forwarding instructions)
- Introduce Control Plane Abstraction and Programmability
  - Integration with routing, signaling and policy logic
  - Open standard-based APIs to allow multivendor setup
  - Global network view providing virtualization of network infrastructure and simplification of provisioning
- Logically centralize the Control Plane

SDN Reference Architecture

- SDN Controller is a policy-based abstraction between network and application and makes use of re-usable components
  - Northbound API provides interface for SDN application
  - Southbound API provides instructions to the network devices and collects network information (e.g. topology, statistics, etc.)

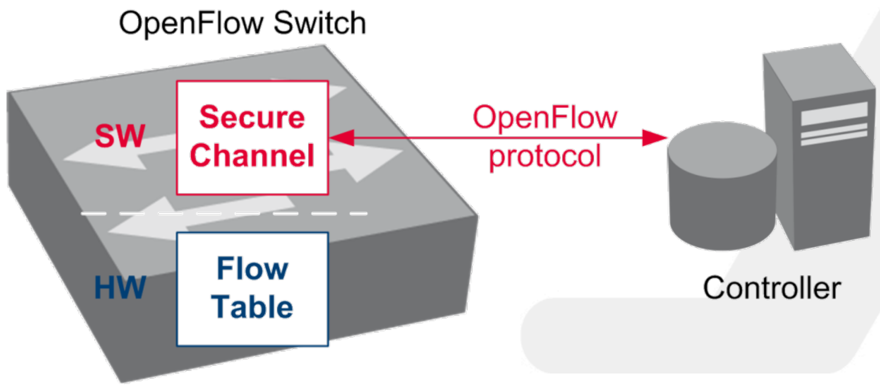


SDN Application

- SDN application is responsible for the calculation of the forwarding state
  - Might be IP prefixes for routers or NAT entries for firewall
- Primary goal: Network operators define what they want to do with the network, not how!
  - Implementation details are hidden inside the SDN controller
  - Distributed routing algorithms running locally on a box in the past evolve into graph algorithms running centrally within the control application
- Drawback: Application must respond to topology changes
  - Use network virtualization technologies

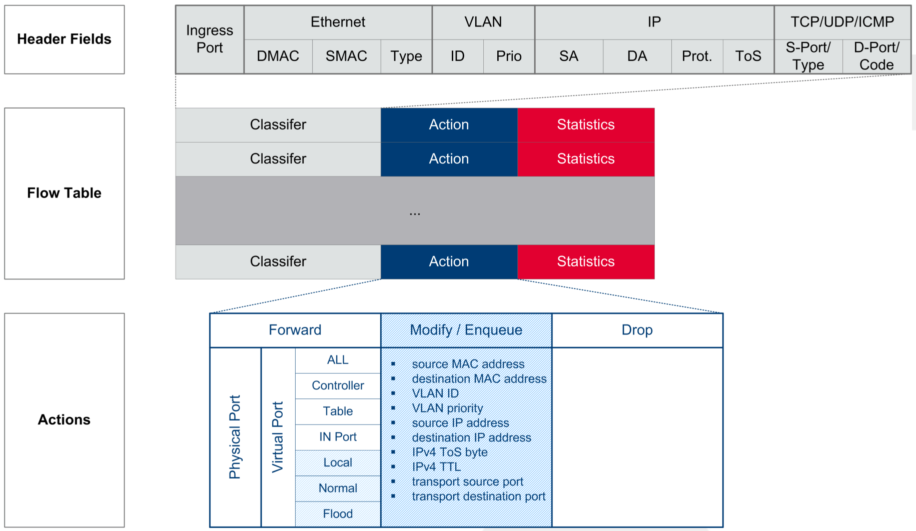
Implementing the Southbound API with OpenFlow

- In traditional networking devices, the control processes and forwarding functionality resides on the network device
- In OpenFlow architecture, an interface is created on the network device through which an external control process is able to program the packet matching and forwarding operations of the network device
  - A standardized API and communication method between external OpenFlow controller and OpenFlow process on the networking device
  - Flow tables held by the networking device which are populated by the external controller



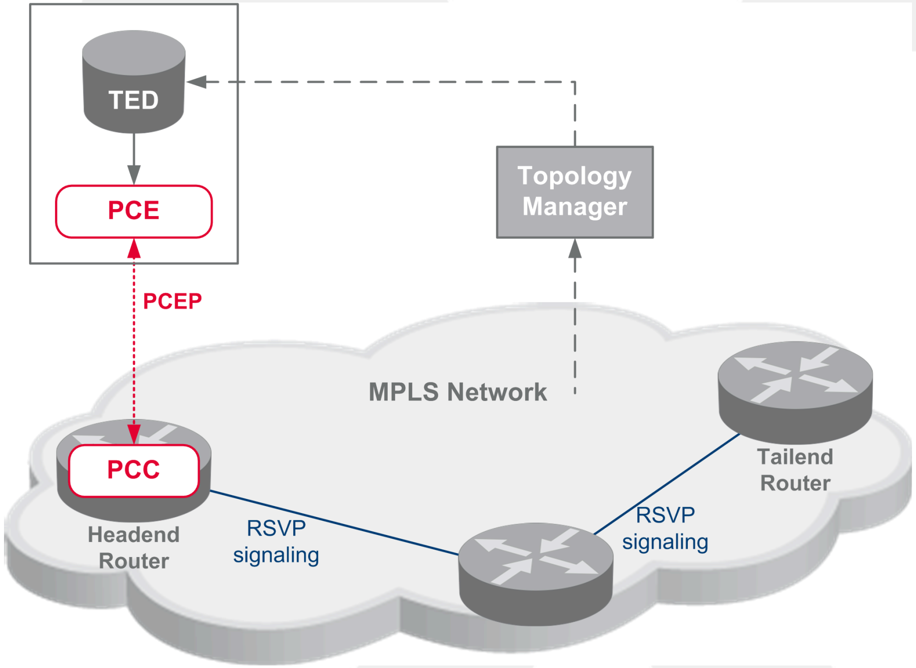
Implementing the Southbound API with OpenFlow (2)

- OpenFlow tables contain flow entries consisting of
  - Match fields to classify the packet (e.g. ingress port, packet header, or metadata)
  - Priorities defining the precedence of matching
  - Counters for statistics reporting capabilities
  - Actions defining how a matched packet should be handled (including drop/forward, enqueue packet, push/pop header, etc.)
  - Timeouts
  - Cookies which might by used by the controller



Implementing the Southbound API with Path Computation Elements (PCE)

- Approach to solving the inter-domain problem
  - In this context domain also might be hierarchy level (e.g. GMPLS domain)
- Formalism of functional architecture
  - *"An entity (component, application, or network node) that is capable of computing a network path or route based on a network graph and applying computational constraints."* (RFC 4655)
  - The ability to perform path computation as a (remote) service
    - Stateless and stateful off-line computation based on TED
  - PCE can issue provisioning commands



xantaro

Implementing the Southbound API with Path Computation Elements (PCE) (2)

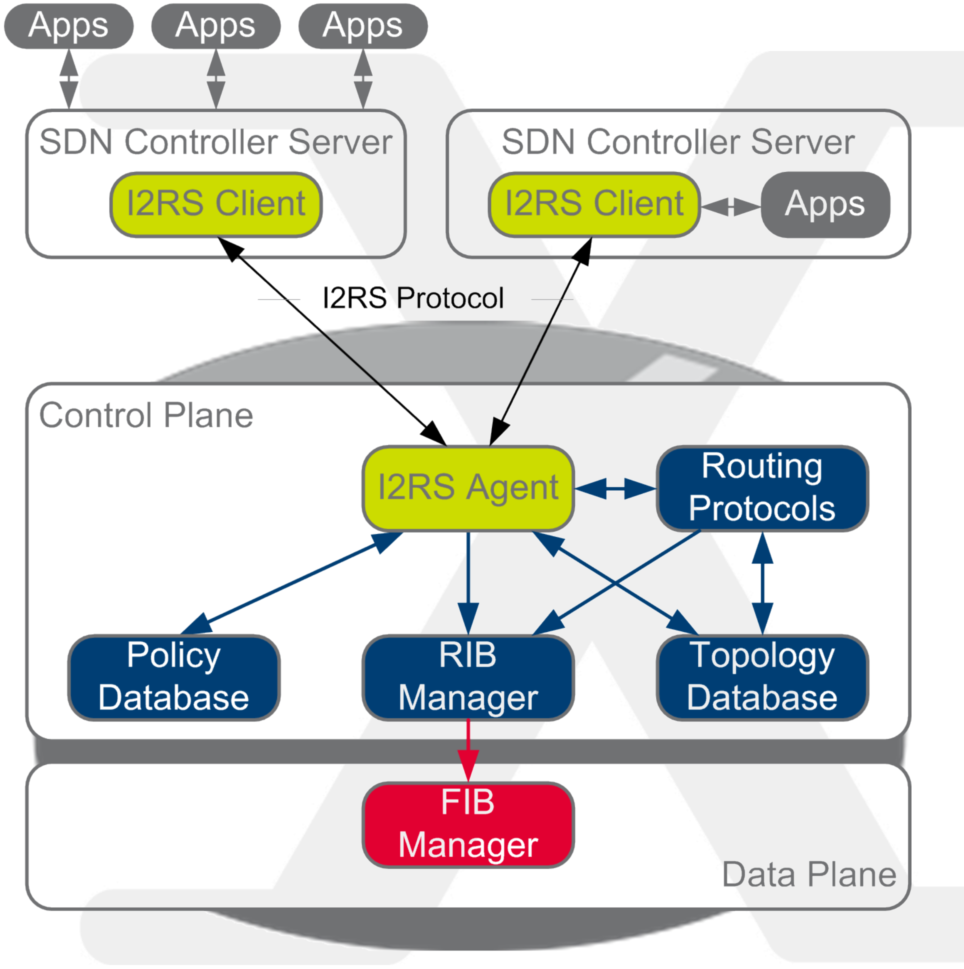
- PCE Communication Protocol (PCEP) runs between a Path Computation Element (PCE) and Path Computation Client (PCC)
  - PCEP operates over TCP to guarantee reliable messaging and flow control without further protocol work
  - PCC may have PCEP session with more than one PCE
- Stateful PCE uses strict synchronization to extract information of active paths and their reserved resources for its computations (e.g. traffic engineering database and existing LSPs)
  - Delegation mechanism available to change controller which is responsible
  - Stateful PCE might also retain information regarding LSPs under construction in order to reduce churn and resource contention.
  - Additional state allows the PCE to compute constrained paths while considering individual LSPs and their interactions

Implementing the Southbound API with BGP – Link State (LS) / Traffic Engineering (TE)

- How does PCE/ALTO obtain the traffic engineering database (TED)?
  - Unspecified in the architecture
  - Early implementations participate in IGP
    - Updates may be too frequent
    - Implementations must support IS-IS and OSPF
- Most traffic engineered networks have a BGP-capable router
  - BGP nodes are designed to process routing policies
- BGP-LS is set of simple extensions to advertise topology info
  - Speaker is possibly route reflector using policy to determine what to advertise and when
  - SDN Server (e.g. PCE/ALTO) uses very lightweight BGP implementation

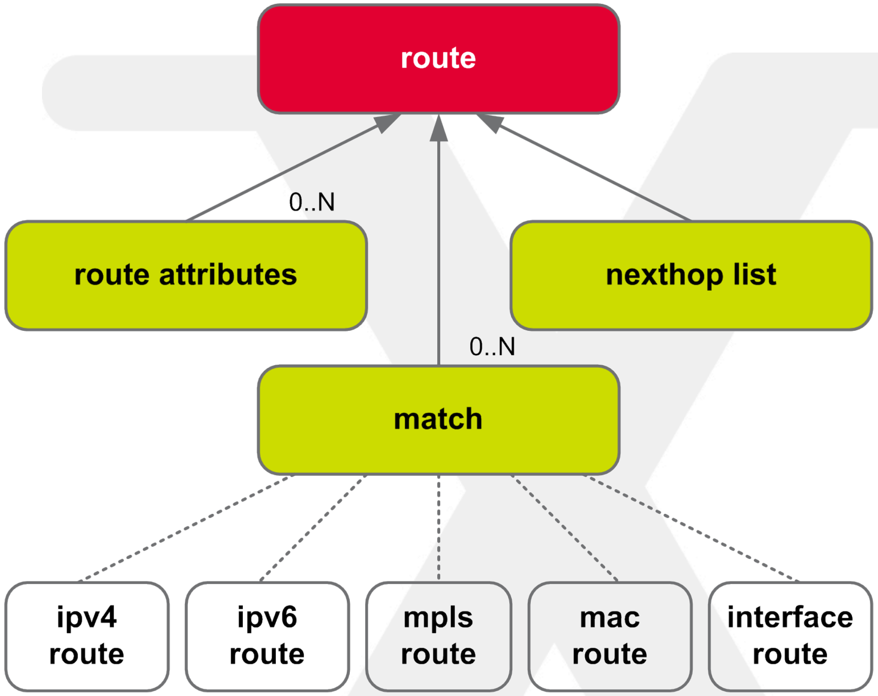
Implementing the Southbound API with Interface to the Routing System (I2RS)

- Framework for integrating external data into routing
  - Indirection, policy, loop-detection
- Filtered events for triggers, verification, and learning about changes to router state
- Data models for state
  - Topology model, interface, measurements, etc.
- Data model for routing
  - RIB layer (unicast/multicast RIBs, MPLS LFIBs, …)
  - Protocol layer (IS-IS, BGP, etc.)
- Device-level and network-level interfaces and protocols



Implementing the Southbound API with Interface to the Routing System (I2RS) (2)

- Data Encoding Language which is parsable, extensible, recursion, programmable (e.g. YANG)
- Non-blocking transactions, stateless, duplex, multi-channel Data Exchange Protocol
- Different types of operations
  - Read Data from RIB (routes/next-hops, routing tables, etc)
  - Write Data into unicast/multicast routes to RIB (*No direct programming of FIB!*)
    - RIB manager MAY do next-hop resolution
  - Notification sent by I2RS agent to controller if route changes or next-hop not resolved

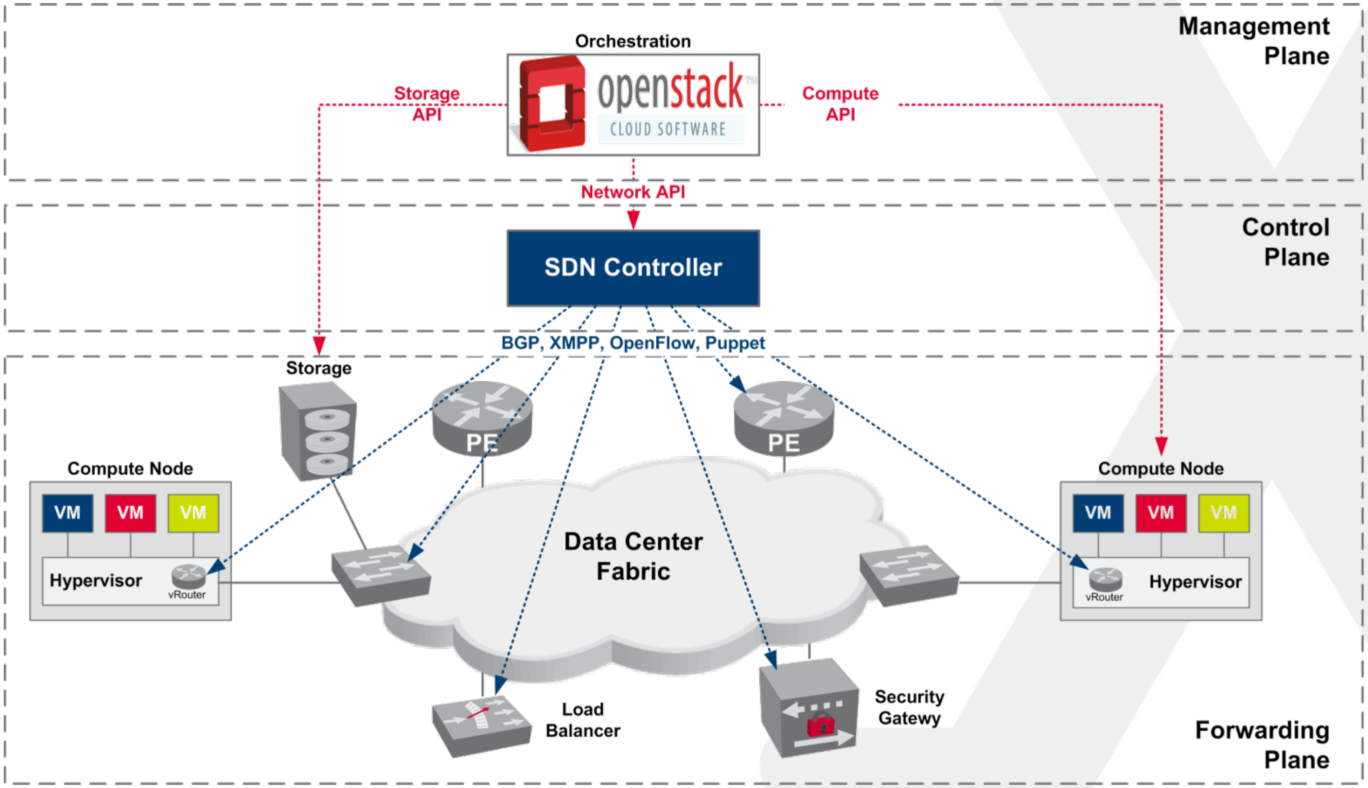


SDN Northbound Interface

- Northbound API allows control information of the network to be used by applications
  - Northbound interface allows to make the difference to traditional networking
  - Could be traditional network services, e.g. firewalls, load balancers, ...
  - Orchestration (e.g. OpenStack)
- Northbound API not constrained by hardware innovation because it‘s independent from underlying physical infrastructure
  - Allows more agile development
- As of today, no standard northbound API available
  - Up to now, proprietary vendor solutions

SDN Use Case for Data Center

- SDN Controller provides centralized touchpoint to enable network virtualization using individual APIs to implement box-specific states



Conclusion

- Abstraction is fundamental concept to solving real-world problems
- SDN provides abstraction, centralization and virtualization for the control plane
- References
  - Scott Shenker: “The Future of Networking and the Past of Protocols”, Open Networking Summit 2012
  - Kireeti Kompella: “SDN: OS or Compiler”, SDN Summit 2013