Software Defined Networking

Lecture#1
Introduction
Introduction

- Traditional IP networks are *Complex and hard to manage* [1]
- Network operator need to configure each individual network device separately using low-level and often vendor-specific commands
- Networks are also *vertically integrated*.
- the control plane and the data plane are bundled inside the networking devices. Reducing flexibility and hindering innovation and evolution of networking infrastructure.
  - Example: the transition from IPV4 to IPV6 started more than a decade ago and still largely incomplete. (IPV^ represented a protocol update.
  - A new routing protocol can take 5 to 10 years to be fully designed, evaluated and deployed.
  - What about changing the internet architecture!!! Simply not feasible in practice.[2],[3]
Software-Defined Networking (SDN) is an emerging networking paradigm that gives hope to change the limitation of current network infrastructures.[4],[5].

- First, it breaks the vertical integration by separating the network’s control logic (the control plane) from the underlying routers and switches that forward the traffic (the data plane).
- Second, with the separation of the control and data planes, network switches become simple forwarding devices and the control logic is implemented in a logically centralized controller (or network operating system), simplifying policy enforcement and network (re)configuration and evolution [6].
Simplified view of an SDN architecture

- It is important to emphasize that a logically centralized programmatic model does not postulate a physically centralized system [7].
- Instead, production-level SDN network designs resort to physically distributed control planes [7], [8].
Introduction

• The separation of the control plane and the data plane can be realized by means of a well-defined programming interface between the switches and the SDN controller. The controller exercises direct control over the state in the data plane elements via this well-defined application programming interface (API), as depicted in Figure 1.

• The most notable example of such an API is OpenFlow [9], [10]. An OpenFlow switch has one or more tables of packet-handling rules (flow table).

• Each rule matches a subset of the traffic and performs certain actions (dropping, forwarding, modifying, etc.) on the traffic. Depending on the rules installed by a controller application, an OpenFlow switch can – instructed by the controller – behave like a router, switch, firewall, or perform other roles (e.g., load balancer, traffic shaper, and in general those of a middlebox).
Introduction

• An important consequence of the software-defined networking principles is the separation of concerns introduced between the definition of network policies, their implementation in switching hardware, and the forwarding of traffic.

• This separation is key to the desired flexibility, breaking the network control problem into tractable pieces, and making it easier to create and introduce new abstractions in networking, simplifying network management and facilitating network evolution and innovation.
Introduction

• Although SDN and OpenFlow started as academic experiments [9], they gained significant traction in the industry over the past few years. Most vendors of commercial switches now include support of the OpenFlow API in their equipment.

• The SDN momentum was strong enough to make Google, Facebook, Yahoo, Microsoft, Verizon, and Deutsche Telekom fund Open Networking Foundation (ONF) [10] with the main goal of promotion and adoption of SDN through open standards development. As the initial concerns with SDN scalability were addressed [11] – in particular the myth that logical centralization implied a physically centralized controller, an issue we will return to later on – SDN ideas have matured and evolved from an academic exercise to a commercial success. Google, for example, has deployed a software-defined network to interconnect its data centers across the globe.
Introduction

• This production network has been in deployment for 3 years, helping the company to improve operational efficiency and significantly reduce costs [8].

• VMware’s network virtualization platform, NSX [12], is another example. NSX is a commercial solution that delivers a fully functional network in software, provisioned independent of the underlying networking devices, entirely based around SDN principles. As a final example, the world’s largest IT companies (from carriers and equipment manufacturers to cloud providers and financial-services companies) have recently joined SDN consortia such as the ONF and the Open Daylight initiative [13], another indication of the importance of SDN from an industrial perspective.
Introduction

• A few recent papers have surveyed specific architectural aspects of SDN [14], [15], [16].
• An overview of OpenFlow and a short literature review can be found in [14] and [15]. These OpenFlow-oriented surveys present a relatively simplified three-layer stack composed of high-level network services, controllers, and the controller/switch interface.
• In [16], the authors go a step further by proposing a taxonomy for SDN. However, similarly to the previous works, the survey is limited in terms of scope and it does not provide an in-depth treatment of fundamental aspects of SDN. In essence, existing surveys lack a thorough discussion of the essential building blocks of an SDN such as the network operating systems, programming languages, and interfaces. They also fall short on the analysis of cross-layer issues such as scalability, security, and dependability. A more complete overview of ongoing research efforts, challenges, and related standardization activities is also missing.
Section V: Ongoing research efforts and challenges

Section IV: Comprehensive survey: Bottom-up approach

Cross-layer issues (debugging, testing & simulation)

Network Applications
- Traffic engineering
- Mobility & Wireless
- Measurement & Monitoring
- Security & Dependability
- Data Center Networking

Programming languages

Language-based virtualization

Northbound Interfaces

Network Operating Systems (SDN Controllers)

Network Hypervisors

Southbound Interfaces

Infrastructure (data plane – forwarding devices)

Terminology
Definitions
Standardization
History

Section III: What is Software-Defined Networking?

Section II: State of quo in “computer networking” and motivation for SDN

Section I: Introduction
Computer networks can be divided in three planes of functionality: the data, control and management planes.

- **The data plane** corresponds to the networking devices, which are responsible for (efficiently) forwarding data.

- **The control plane** represents the protocols used to populate the forwarding tables of the data plane elements.

- **The management plane** includes the software services, such as SNMP-based tools [18], used to remotely monitor and configure the control functionality.

Network policy is defined in the management plane, the control plane enforces the policy, and the data plane executes it by forwarding data accordingly.
II STATE OF QUO IN NETWORKING...

• In traditional IP networks, the control and data planes are tightly coupled, embedded in the same networking devices, and the whole structure is highly decentralized.

• However, the outcome is a very complex and relatively static architecture, as has been often reported in the networking literature (e.g., [1], [3], [2], [6], [19]). It is also the fundamental reason why traditional networks are rigid, and complex to manage and control. These two characteristics are largely responsible for a vertically-integrated industry where innovation is difficult.
Network misconfigurations and related errors are extremely common in today’s networks. For instance, more than 1000 configuration errors have been observed in BGP routers [20]. From a single misconfigured device may result very undesired network behavior (including, among others, packet losses, forwarding loops, setting up of unintended paths, or service contract violations). Indeed, while rare, a single misconfigured router is able to compromise the correct operation of the whole Internet for hours [21], [22].

To support network management, a small number of vendors offer proprietary solutions of specialized hardware, operating systems, and control programs (network applications). Network operators have to acquire and maintain different management solutions and the corresponding specialized teams. The capital and operational cost of building and maintaining a networking infrastructure is significant, with long return on investment cycles, which hamper innovation and addition of new features and services (for instance access control, load balancing, energy efficiency, traffic engineering). To alleviate the lack of in-path functionalities within the network, a myriad of specialized components and middleboxes, such as firewalls, intrusion detection systems and deep packet inspection engines, proliferate in current networks. A recent survey of 57 enterprise networks shows that the number of middleboxes is already on par with the number of routers in current networks [23]. Despite helping in-path functionalities, the net effect of middleboxes has been increased complexity of network design and its operation.
WHAT IS SOFTWARE-DEFINED NETWORKING?

• The term SDN (Software-Defined Networking) was originally coined to represent the ideas and work around OpenFlow at Stanford University [24]. As originally defined, SDN refers to a network architecture where the forwarding state in the data plane is managed by a remote control plane decoupled from the former. The networking industry has on many occasions shifted from this original view of SDN, by referring to anything that involves software as being SDN. We therefore attempt, in this section, to provide a much less ambiguous definition of software-defined networking.
III. WHAT IS SOFTWARE-DEFINED NETWORKING?

We define an SDN as a network architecture with four pillars:

1. The control and data planes are *decoupled*. Control functionality is removed from network devices that will become simple (packet) forwarding elements.

2. Forwarding decisions are flow-based, instead of destination-based. A flow is broadly defined by a set of packet field values acting as a match (filter) criterion and a set of actions (instructions). In the SDN/OpenFlow context, a flow is a sequence of packets between a source and a destination. All packets of a flow receive identical service policies at the forwarding devices [25], [26]. The flow abstraction allows unifying the behavior of different types of network devices, including routers, switches, firewalls, and middleboxes [27]. Flow programming enables unprecedented flexibility, limited only to the capabilities of the implemented flow tables [9].

3. Control logic is moved to an external entity, the so-called SDN controller or Network Operating System (NOS). The NOS is a software platform that runs on commodity server technology and provides the essential resources and abstractions to facilitate the programming of forwarding devices based on a logically centralized, abstract network view. Its purpose is therefore similar to that of a traditional operating system.

4. The network is *programmable* through software applications running on top of the NOS that interacts with the underlying data plane devices. This is a fundamental characteristic of SDN, considered as its main value proposition.
III. WHAT IS SOFTWARE-DEFINED NETWORKING?

• Following the SDN concept introduced in [5], an SDN can be defined by three fundamental abstractions: (i) forwarding, (ii) distribution, and (iii) specification. In fact, abstractions are essential tools of research in computer science and information technology, being already an ubiquitous feature of many computer architectures and systems [28].
III. WHAT IS SOFTWARE-DEFINED NETWORKING?

• Ideally, the *forwarding abstraction* should allow any forwarding behavior desired by the network application (the control program) while hiding details of the underlying hardware. OpenFlow is one realization of such abstraction, which can be seen as the equivalent to a “device driver” in an operating system.
III. WHAT IS SOFTWARE-DEFINED NETWORKING?

- The *distribution abstraction* should shield SDN applications from the vagaries of distributed state, making the distributed control problem a logically centralized one. Its realization requires a common distribution layer, which in SDN resides in the NOS. This layer has two essential functions. First, it is responsible for installing the control commands on the forwarding devices. Second, it collects status information about the forwarding layer (network devices and links), to offer a global network view to network applications.

- The last abstraction is *specification*, which should allow a network application to express the desired network behavior without being responsible for implementing that behavior itself. This can be achieved through virtualization solutions, as well as network programming languages. These approaches map the abstract configurations that the applications express based on a simplified, abstract model of the network, into a physical configuration for the global network view exposed by the SDN controller. *Figure 4* depicts the SDN architecture, concepts and building blocks.
Fig. 4. SDN architecture and its fundamental abstractions.
III. WHAT IS SOFTWARE-DEFINED NETWORKING?

• As previously mentioned, the strong coupling between control and data planes has made it difficult to add new functionality to traditional networks, a fact illustrated in Figure 5. The coupling of the control and data planes (and its physical embedding in the network elements) makes the development and deployment of new networking features (e.g., routing algorithms) very hard since it would imply a modification of the control plane of all network devices – through the installation of new firmware and, in some cases, hardware upgrades. Hence, the new networking features are commonly introduced via expensive, specialized and hard-to-configure equipment (aka middleboxes) such as load balancers, intrusion detection systems (IDS), and firewalls, among others. These middleboxes need to be placed strategically in the network, making it even harder to later change the network topology, configuration, and functionality.
Fig. 5. Traditional networking versus Software-Defined Networking (SDN). With SDN, management becomes simpler and middleboxes services can be delivered as SDN controller applications.
III. WHAT IS SOFTWARE-DEFINED NETWORKING?

• In contrast, SDN decouples the control plane from the network devices and becomes an external entity: the network operating system or SDN controller. This approach has several advantages:
  – It becomes easier to program these applications since the abstractions provided by the control platform and/or the network programming languages can be shared.
  – All applications can take advantage of the same network information (the global network view), leading (arguably) to more consistent and effective policy decisions while re-using control plane software modules.
  – These applications can take actions (i.e., reconfigure forwarding devices) from any part of the network. There is therefore no need to devise a precise strategy about the location of the new functionality.
  – The integration of different applications becomes more straightforward [29]. For instance, load balancing and routing applications can be combined sequentially, with load balancing decisions having precedence over routing policies.
A. Terminology

- To identify the different elements of an SDN as unequivocally as possible, we now present the essential terminology used throughout this work.

**Forwarding Devices (FD):** Hardware- or software-based data plane devices that perform a set of elementary operations. The forwarding devices have well-defined instruction sets (e.g., flow rules) used to take actions on the incoming packets (e.g., forward to specific ports, drop, forward to the controller, rewrite some header). These instructions are defined by south-bound interfaces (e.g., OpenFlow [9], ForCES [30], Protocol- Oblivious Forwarding (POF) [31]) and are installed in the forwarding devices by the SDN controllers implementing the southbound protocols.

**Data Plane (DP):** Forwarding devices are interconnected through wireless radio channels or wired cables. The network infrastructure comprises the interconnected forwarding devices, which represent the data plane.


A. Terminology

- **Southbound Interface (SI)**: The instruction set of the forwarding devices is defined by the southbound API, which is part of the southbound interface. Furthermore, the SI also defines the communication protocol between forwarding devices and control plane elements. This protocol formalizes the way the control and data plane elements interact.

- **Control Plane (CP)**: Forwarding devices are programmed by control plane elements through well-defined SI embodiments. The control plane can therefore be seen as the “network brain”. All control logic rests in the applications and controllers, which form the control plane.

- **Northbound Interface (NI)**: The network operating system can offer an API to application developers. This API represents a northbound interface, i.e., a common interface for developing applications. Typically, a northbound interface abstracts the low level instruction sets used by southbound interfaces to program forwarding devices.

- **Management Plane (MP)**: The management plane is the set of applications that leverage the functions offered by the NI to implement network control and operation logic. This includes applications such as routing, firewalls, load balancers, monitoring, and so forth. Essentially, a management application defines the policies, which are ultimately translated to southbound-specific instructions that program the behavior of the forwarding devices.
B. Alternative and Broadening Definitions

Since its inception in 2010 [24], the original OpenFlow-centered SDN term has seen its scope broadened beyond architectures with a cleanly decoupled control plane interface. The definition of SDN will likely continue to broaden, driven by the industry business-oriented views on SDN – irrespective of the decoupling of the control plane. In this survey, we focus on the original, “canonical” SDN definition based on the aforementioned key pillars and the concept of layered abstractions. However, for the sake of completeness and clarity, we acknowledge alternative SDN definitions [32], including:

- **Control Plane / Broker SDN**: A networking approach that retains existing distributed control planes but offers new APIs that allow applications to interact (bidirectionally) with the network. An SDN controller – often called orchestration platform – acts as a broker between the applications and the network elements. This approach effectively presents control plane data to the application and allows a certain degree of network programmability by means of “plug-ins” between the orchestrator function and network protocols. This API-driven approach corresponds to a hybrid model of SDN, since it enables the broker to manipulate and directly interact with the control planes of devices such as routers and switches. Examples of this view on SDN include recent standardization efforts at IETF (see Section III-C) and the design philosophy behind the OpenDaylight project [13] that goes beyond the OpenFlow split control mode.

- **Overlay SDN**: A networking approach where the (software- or hardware-based) network edge is dynamically programmed to manage tunnels between hypervisors and/or network switches, introducing an overlay network. In this hybrid networking approach, the distributed control plane providing the underlay remains untouched. The centralized control plane provides a logical overlay that utilizes the underlay as a transport network. This flavor of SDN follows a proactive model to install the overlay tunnels. The overlay tunnels usually terminate inside virtual switches within hypervisors or in physical devices acting as gateways to the existing network. This approach is very popular in recent data center network virtualization [33], and are based on a variety of tunneling technologies (e.g., STT [34], VXLAN [35], NVGRE [36], LISP [37], [38], GENEVE [39]) [40].
Recently, other attempts to define SDN in a layered approach have appeared [41], [16]. From a practical perspective and trying to keep backward compatibility with existing network management approaches, one initiative at IRTF SD-NRG [41] proposes a management plane at the same level of the control plane, i.e., it classifies solutions in two categories: control logic (with control plane southbound interfaces) and management logic (with management plane southbound interfaces). In other words, the management plane can be seen as a control platform that accommodates traditional network management services and protocols, such as SNMP [18], BGP [42], PCEP [43], and NETCONF [44].

In addition the broadening definitions above, the term SDN is often used to define extensible network management planes (e.g., OpenStack [45]), whitebox / bare-metal switches with open operating systems (e.g., Cumulus Linux), open-source dataplanes (e.g., Pica8 Xorplus [46], Quagga [47]), specialized programmable hardware devices (e.g., NetFPGA [48]), virtualized software-based appliances (e.g., Open Platform for Network Functions Virtualization - OPNFV [49]), in spite of lacking a decoupled control and data plane or common interface along its API. Hybrid SDN models are further discussed in Section V-G.
C. Standardization Activities

- The standardization landscape in SDN (and SDN-related issues) is already wide and is expected to keep evolving over time. While some of the activities are being carried out in Standard Development Organizations (SDOs), other related efforts are ongoing at industrial or community consortia (e.g., OpenDaylight, OpenStack, OPNFV), delivering results often considered candidates for *de facto* standards. These results often come in the form of open source implementations that have become the common strategy towards accelerating SDN and related cloud and networking technologies [50]. The reason for this fragmentation is due to SDN concepts spanning different areas of IT and networking, both from a network segmentation point of view (from access to core) and from a technology perspective (from optical to wireless).
C. Standardization Activities

- Table I presents a summary of the main SDOs and organizations contributing to the standardization of SDN, as well as the main outcomes produced to date.

- The Open Networking Foundation (ONF) was conceived as a member-driven organization to promote the adoption of SDN through the development of the OpenFlow protocol as an open standard to communicate control decisions to data plane devices. The ONF is structured in several working groups (WGs). Some WGs are focused on either defining extensions to the OpenFlow protocol in general, such as the Extensibility WG, or tailored to specific technological areas. Examples of the latter include the Optical Transport (OT) WG, the Wireless and Mobile (W&M) WG, and the Northbound Interfaces (NBI) WG. Other WGs center their activity in providing new protocol capabilities to enhance the protocol itself, such as the Architecture WG or the Forwarding Abstractions (FA) WG.
<table>
<thead>
<tr>
<th>SDO</th>
<th>Working Group</th>
<th>Focus</th>
<th>Outcomes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ONF</td>
<td>Architecture &amp; Framework</td>
<td>SDN architecture, defining architectural components and interfaces</td>
<td>SDN Architecture [51]</td>
</tr>
<tr>
<td></td>
<td>Northbound Interfaces</td>
<td>Definition of standard NBIs for SDN controllers</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Testing and Interoperability</td>
<td>Specification of OpenFlow conformance test suites</td>
<td>Conformance tests [52]</td>
</tr>
<tr>
<td></td>
<td>Extensibility</td>
<td>Development of extensions to OpenFlow protocol, producing specifications of the OpenFlow switch (OF-WIRE) protocol</td>
<td>OF-WIRE 1.4.0 [53]</td>
</tr>
<tr>
<td></td>
<td>Configuration &amp; Management</td>
<td>OAM (operation, administration, and management) capabilities for OF protocol, producing specifications of the OF Configuration and Management (OF-CONFIG) protocol</td>
<td>OF-CONFIG 1.2 [54] OpenFlow Notifications Framework [55]</td>
</tr>
<tr>
<td></td>
<td>Forwarding Abstractions</td>
<td>Development of hardware abstractions and simplification of behavioral descriptions mapping</td>
<td>OpenFlow Table Type Patterns [56]</td>
</tr>
<tr>
<td></td>
<td>Optical Transport</td>
<td>Specification of SDN and control capabilities for optical transport networks by means of OpenFlow</td>
<td>Use cases [57] Requirements [58]</td>
</tr>
<tr>
<td></td>
<td>Wireless &amp; Mobile</td>
<td>Specification of SDN and control capabilities for wireless and mobile networks by means of OpenFlow</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Migration</td>
<td>Methods to migrate from conventional networks to SDN-based networks based on OpenFlow</td>
<td>Use cases [59]</td>
</tr>
<tr>
<td></td>
<td>Market Education</td>
<td>Dissemination of ONF initiatives in SDN and OpenFlow by releasing White Papers and Solution Briefs</td>
<td>SDN White Paper [60]</td>
</tr>
<tr>
<td>IETF</td>
<td>Application-Layer Traffic Optim-</td>
<td>Provides applications with network state information</td>
<td>Architectures for the coexistence of SDN and ALTO [61]</td>
</tr>
<tr>
<td></td>
<td>ization (ALTO) Optimization</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Forwarding and Control Element Separation (ForCES)</td>
<td>Protocol specifications for the communication between control and forwarding elements.</td>
<td>Protocol specification [30]</td>
</tr>
<tr>
<td></td>
<td>Interface to the Routing System (I2RS)</td>
<td>Real-time or event driven interaction with the routing system in an IP routed network</td>
<td>Architecture [62]</td>
</tr>
<tr>
<td></td>
<td>Network Configuration (NET-CONF)</td>
<td>Protocol specification for transferring configuration data to and from a device</td>
<td>NETCONF protocol [63]</td>
</tr>
<tr>
<td></td>
<td>Network Virtualization Overlays (NVO3)</td>
<td>Overlay networks for supporting multi-tenancy in the context of data center communications (i.e., VM communication)</td>
<td>Control plane requirements [64]</td>
</tr>
<tr>
<td></td>
<td>Path Computation Element (PCE)</td>
<td>Path computation for traffic engineering and path selection based on constrains</td>
<td>ABNO framework [65] Cross stratum path computation [66]</td>
</tr>
<tr>
<td></td>
<td>Source Packet Routing in Networking (SPRING)</td>
<td>Specification of a forwarding path at the source of traffic</td>
<td>OpenFlow interworking [67] SDN controlled use cases [68]</td>
</tr>
<tr>
<td></td>
<td>Abstraction and Control of Transport Networks (ACTN) BoF</td>
<td>Facilitate a centralized virtual network operation</td>
<td>Virtual network controller framework [69]</td>
</tr>
<tr>
<td>IRTF</td>
<td>Software-Defined Networking Working Group (SDN-WG)</td>
<td>Prospection of SDN for the evolution of Internet</td>
<td>SDN operator perspective [70]</td>
</tr>
</tbody>
</table>
C. Standardization Activities

• Similar to how network programmability ideas have been considered by several Working Groups (WGs) of the Internet Engineering Task Force (IETF) in the past, the present SDN trend is also influencing a number of activities. A related body that focuses on research aspects for the evolution of the Internet, the Internet Research Task Force (IRTF), has created the Software Defined Networking Research Group (SDNRG). This group investigates SDN from various perspectives with the goal of identifying the approaches that can be defined, deployed and used in the near term, as well as identifying future research challenges.
C. Standardization Activities

• In the International Telecommunications Union’s Telecommunication sector (ITU-T), some Study Groups (SGs) have already started to develop recommendations for SDN, and a Joint Coordination Activity on SDN (JCA-SDN) has been established to coordinate the SDN standardization work.

• The Broadband Forum (BBF) is working on SDN top-ics through the Service Innovation & Market Requirements (SIMR) WG. The objective of the BBF is to release recommendations for supporting SDN in multi-service broadband networks, including hybrid environments where only some of the network equipment is SDN-enabled.
C. Standardization Activities

• The Metro Ethernet Forum (MEF) is approaching SDN with the aim of defining service orchestration with APIs for existing networks.

• At the Institute of Electrical and Electronics Engineers (IEEE), the 802 LAN/MAN Standards Committee has recently started some activities to standardize SDN capabilities on access networks based on IEEE 802 infrastructure through the P802.1CF project, for both wired and wireless technologies to embrace new control interfaces.
C. Standardization Activities

• The Optical Internetworking Forum (OIF) Carrier WG released a set of requirements for Transport Software-Defined Networking. The initial activities have as main goal to describe the features and functionalities needed to support the deployment of SDN capabilities in carrier transport networks.

• The Open Data Center Alliance (ODCA) is an organization working on unifying data center in the migration to cloud computing environments through interoperable solutions. Through the documentation of usage models, specifically one for SDN, the ODCA is defining new requirements for cloud deployment.
C. Standardization Activities

- The Alliance for Telecommunication Industry Solutions (ATIS) created a Focus Group for analyzing operational issues and opportunities associated with the programmable capabilities of network infrastructure.
C. Standardization Activities

• At the European Telecommunication Standards Institute (ETSI), efforts are being devoted to Network Function Virtualization (NFV) through a newly defined Industry Specification Group (ISG). NFV and SDN concepts are considered complementary, sharing the goal of accelerating innovation inside the network by allowing programmability, and altogether changing the network operational model through automation and a real shift to software-based platforms.

• Finally, the mobile networking industry 3GPP consortium is studying the management of virtualized networks, an effort aligned with the ETSI NFV architecture and, as such, likely to leverage from SDN.
D. History of Software-Defined Networking

- Although a fairly recent concept, SDN leverages on networking ideas with a longer history [17]. In particular, it builds on work made on programmable networks, such as active networks [81], programmable ATM networks [82], [83], and on proposals for control and data plane separation, such as NCP [84] and RCP [85].
D. History of Software-Defined Networking

• In order to present an historical perspective, we summarize in Table II different instances of SDN-related work prior to SDN, splitting it into five categories. Along with the categories we defined, the second and third columns of the table mention past initiatives (pre-SDN, i.e., before the OpenFlow-based initiatives that sprung into the SDN concept), and recent developments that led to the definition of SDN.

• Data plane programmability has a long history. Active networks [81] represent one of the early attempts on building new network architectures based on this concept. The main idea behind active networks is for each node to have the capability to perform computations on, or modify the content of, packets. To this end, active networks propose two distinct approaches: programmable switches and capsules. The former does not imply changes in the existing packet or cell format. It assumes that switching devices support the downloading of programs with specific instructions on how to process packets. The second approach, on the other hand, suggests that packets should be replaced by tiny programs, which are encapsulated in transmission frames and executed at each node along their path.
## TABLE II
SUMMARIZED OVERVIEW OF THE HISTORY OF PROGRAMABLE NETWORKS

<table>
<thead>
<tr>
<th>Category</th>
<th>Pre-SDN initiatives</th>
<th>More recent SDN developments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data plane programmability</td>
<td>xbind [82], IEEE P1520 [86], smart packets [87], ANTS [88], SwitchWare [89], Calvert [90], high performance router [91], NetScript [92], Tennenhouse [93]</td>
<td>ForCES [30], OpenFlow [9], POF [31]</td>
</tr>
<tr>
<td>Control and data plane decoupling</td>
<td>NCP [84], GSMP [94], [95], Tempest [96], ForCES [30], RCP [85], Soft-Router [97], PCE [43], 4D [98], IRSCP [99]</td>
<td>SANE [100], Ethane [101], OpenFlow [9], NOX [26], POF [31]</td>
</tr>
<tr>
<td>Network virtualization</td>
<td>Tempest [96], MBone [102], 6Bone [103], RON [104], Planet Lab [105], Impasse [106], GENI [107], VINI [108]</td>
<td>Open vSwitch [109], Mininet [110], FlowVisor [111], NVP [112]</td>
</tr>
<tr>
<td>Network operating systems</td>
<td>Cisco IOS [113], JUNOS [114], ExtremeXOS [115], SR OS [116]</td>
<td>NOX [26], Onix [7], ONOS [117]</td>
</tr>
<tr>
<td>Technology pull initiatives</td>
<td>Open Signaling [118]</td>
<td>ONF [10]</td>
</tr>
</tbody>
</table>
D. History of Software-Defined Networking

- ForCES [30], OpenFlow [9] and POF [31] represent recent approaches for designing and deploying programmable data plane devices. In a manner different from active networks, these new proposals rely essentially on modifying forwarding devices to support flow tables, which can be dynamically configured by remote entities through simple operations such as adding, removing or updating flow rules, i.e., entries on the flow tables.
The earliest initiatives on separating data and control signaling date back to the 80s and 90s. The network control point (NCP) [84] is probably the first attempt to separate control and data plane signaling. NCPs were introduced by AT&T to improve the management and control of its telephone network. This change promoted a faster pace of innovation of the network and provided new means for improving its efficiency, by taking advantage of the global view of the network provided by NCPs. Similarly, other initiatives such as Tempest [96], ForCES [30], RCP [85], and PCE [43] proposed the separation of the control and data planes for improved management in ATM, Ethernet, BGP, and MPLS networks, respectively.
More recently, initiatives such as SANE [100], Ethane [101], OpenFlow [9], NOX [26] and POF [31] proposed the decoupling of the control and data planes for Ethernet networks. Interestingly, these recent solutions do not require significant modifications on the forwarding devices, making them attractive not only for the networking research community, but even more to the networking industry. OpenFlow-based devices [9], for instance, can easily co-exist with traditional Ethernet devices, enabling a progressive adoption (i.e., not requiring a disruptive change to existing networks).
Network virtualization has gained a new traction with the advent of SDN. Nevertheless, network virtualization also has its roots back in the 90s. The Tempest project [96] is one of the first initiatives to introduce network virtualization, by introducing the concept of switchlets in ATM networks. The core idea was to allow multiple switchlets on top of a single ATM switch, enabling multiple independent ATM networks to share the same physical resources. Similarly, MBone [102] was one of the early initiatives that targeted the creation of virtual network topologies on top of legacy networks, or overlay networks. This work was followed by several other projects such as Planet Lab [105], GENI [107] and VINI [108]. It is also worth mentioning FlowVisor [119] as one of the first recent initiatives to promote a hypervisor-like virtualization architecture for network infrastructures, resembling the hypervisor model common for compute and storage. More recently, Koponen et al. proposed a Network Virtualization Platform (NVP [112]) for multi-tenant datacenters using SDN as a base technology.
The concept of a network operating system was reborn with the introduction of OpenFlow-based network operating systems, such as NOX [26], Onix [7] and ONOS [117]. Indeed, network operating systems have been in existence for decades. One of the most widely known and deployed is the Cisco IOS [113], which was originally conceived back in the early 90s. Other network operating systems worth mentioning are JUNOS [114], ExtremeXOS [115] and SR OS [116]. Despite being more specialized network operating systems, targeting network devices such as high-performance core routers, these NOSs abstract the underlying hardware to the network operator, making it easier to control the network infrastructure as well as simplifying the development and deployment of new protocols and management applications.
Finally, it is also worth recalling initiatives that can be seen as “technology pull” drivers. Back in the 90s, a movement towards open signaling [118] started to happen. The main motivation was to promote the wider adoption of the ideas proposed by projects such as NCP [84] and Tempest [96]. The open signaling movement worked towards separating the control and data signaling, by proposing open and programmable interfaces. Curiously, a rather similar movement can be observed with the recent advent of OpenFlow and SDN, with the lead of the Open Networking Foundation (ONF) [10]. This type of movement is crucial to promote open technologies into the market, hopefully leading equipment manufacturers to support open standards and thus fostering interoperability, competition, and innovation.

For a more extensive intellectual history of programmable networks and SDN read ref [17].
SOFTWARE-DEFINED NETWORKS: BOTTOM-UP

Next time!!