Agence Nationale de la Recherche (ANR) – Appel INFRA

Project N^{o}: ANR-13-INFR-013

DISCO

WP1

D1.1 Preliminary DISCO Architecture

Editor:	TCS
Contributors:	TCS, ENSL, INRIA, 6WIND
Status -Version:	Final version 1.0
Delivery Date:	15 September 2014

Disclaimer

The DISCO Consortium consists of the following companies:

No	Participant name	Participant short name	Role	Country
1	Thales Communications & Security	TCS	Coordinator	France
2	Ecole Normale Supérieure de Lyon	ENSL	Contractor	France
3	Institut national de recherche en informatique et en automatique	INRIA	Contractor	France
4	6WIND	6WIND	Contractor	France

Document Revision History

Date	**Issue**	**Author/Editor/Contributor**	**Summary of main changes**
29/06/2014	0.1	TCS	Initialization
29/07/2014	0.2	ENSL	Research challenge description
30/07/2014	0.3	INRIA	State of the art and research challenge
12/08/2014	0.4	TCS	Preliminary version
02/09/2014	0.5	ENSL	Document revision
08/09/2014	0.6	TCS	Document revision
10/09/2014	0.7	INRIA	Document revision
11/09/2014	0.8	6WIND	System architecture, challenges and document revision
12/09/2014	0.9	TCS	Document revision
15/09/2014	1.0	TCS	Finalization of the document

Table of contents

Table of figures

Acronyms

API	Application Programming Interface
CAPEX	CApital Expenditure
DPDK	Data Plane Development Kit
IaaS	Infrastructure as a Service
IDE	Integrated Development Environment
IETF	Internet Engineering Task Force
IRTF	Internet Research Task Force
ISP	Internet Service Provider
ITU	International Telecommunications Union
MBAC	Measurement-Based Admission Control
MNO	Mobile Network Operator
NaaS	Network as a Service
NSP	Network Service Provider
OF	OpenFlow
OSGi	Open Services Gateway initiative
REST	REpresentational State Transfer
SDN	Software-Defined Networking
SOA	Service Oriented Architecture

1 Executive Abstract

SDN (Software Defined Networking) represents a disruptive change in the way networks are architected, built, and operated. SDN opens up traditionally closed, single-vendor networks and brings competition and innovation to bear on some historically unsolvable networking problems. SDN is a new promising networking paradigm where network control is decoupled from forwarding and is directly programmable. This migration of control, formerly tightly bound in individual network devices, into accessible computing devices enables the underlying infrastructure to be abstracted for applications and network services, which can treat the network as a logical or virtual entity. SDN thus enables to automatically and flexibly manage network resources.

DISCO (DIstributed SDN COntrollers for rich and elastic network services) leans on the complementary technical expertise of the partners to bring flexibility, scalability and resiliency in SDN architectures, but also, richness and elasticity for the deployment of network services, for example for content delivery.

This deliverable first presents the context of the project. Then it highlights the research projects, standardization initiatives and open source initiatives that are linked with DISCO’s objectives. The system architecture is then detailed with the scenarios, the identification of the stakeholders and roles, the definition of the building blocks and the system requirements that DISCO should satisfy to meet the objectives. Finally, this document develops the research challenges that the project will address to complete the proof of concept study that DISCO aims at achieving.

2 DISCO’s proposition

This section first presents the technologic and economic contexts in which DISCO takes place.

2.1 Current economic and technologic ecosystem

The operators’ networks have significantly evolved over the last few years. Derived from the cloud computing paradigm, new network virtualization technologies are emerging. These technologies bring agility in the management of infrastructures and services while rationalizing expenditures. They also enable network operators to improve the value-added of their networks and datacenters by offering new services.

The major trends impacting the data center and campus environments are:

- **Data center consolidation.** Enterprise organizations have been actively consolidating and centralizing facilities, applications, and infrastructure for a number of years. It equates to a multitude of devices, complex cabling, and unprecedented network traffic loads.

- **Server virtualization.** While most organizations have already deployed server virtualization, it continues to penetrate and become pervasive across the entire enterprise. This will require an increasing number of virtual servers—and the virtual access networks they connect to—to be tightly integrated into the physical data center network infrastructure.

- **New application architectures.** As organizations deploy more SOA and web-applications and transition to service-based delivery models, their data centers will become increasingly dynamic and complex, and most legacy networks are a mismatch for this.

- **Cloud computing.** As organizations strive to become more agile and responsive to the needs of the business, more are looking to adopt rapidly scalable computing models based on public and private cloud technologies.

The Software-Defined Networking (SDN) paradigm has emerged from the need to overcome the primary limitations of today's networks: complexity, lack of scalability and vendor dependence. It is based on three main principles: separation of software and physical layers, centralized control of information and network programmability. The ability to program networks, to interact with network elements and to manage a unified multi-vendor multi-technology environment enables service providers and network operators to innovate faster and to reduce operational and capital expenditures. SDN was first massively used in datacenters, network management being thus integrated to cloud platforms. It is now envisioned for multi-datacenter environments and multi-domain networks.

The global SDN market was estimated to be valued at USD 326.5 million in 2013, which is expected to reach USD 4,909.8 million by 2020, growing at a Compound Annual Growth Rate (CAGR) of 44.2% from 2014 to 2020. Mobile Network Operators (MNOs) have been garnering increasing subscriber base, since majority of the population globally uses mobile devices. This trend is particularly prevalent in emerging nations, which have witnessed high demand for mobile phones, tablets, and phablets among others. MNOs need to ensure that their network is capable of handling the burgeoning number of users, which is made possible by SDN. Therefore, growing need and demand for mobility is a key driving force for the market. Additionally, SDN facilitates the establishment of an efficient network and ensures virtualization, cloud services, agility, etc., which is expected to propel market growth.

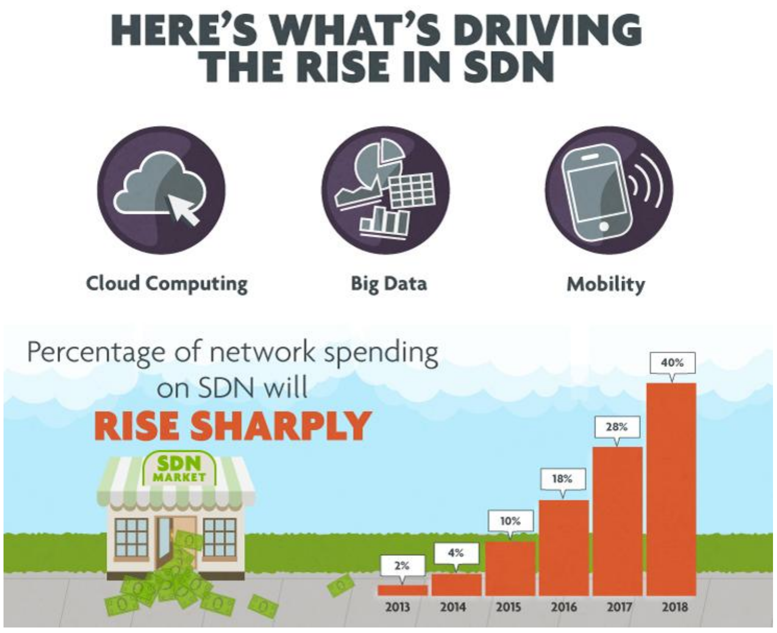

Figure 1: Network spending on SDN

The key factors fueling SDN adoption in the enterprise space include enhancing overall infrastructure flexibility, along with need to reduce the time-to-market of various applications and services. In order for SDN to be deployed across the enterprise, it is crucial to ensure complete knowledge of the implementation. Telecommunications service providers are expected to be the fastest growing segment, at an estimated CAGR of 45.6% from 2014 to 2020. SDN enables network resilience, as well as reduction in capital and operating expenditure in the telecom sector.

2.2 Overview of the technical proposition

With regards to the economic and technologic contexts previously exposed, DISCO's proposal focuses on bringing flexibility, scalability and resiliency in SDN architectures but also richness and elasticity for the deployment of network services. These features are of primal importance to foster the adoption of SDN by telecommunications service providers and inside enterprises.

The expected impacts of this project are in more details:

- A multi-controller software platform, built upon existing, widely-accepted SDN technologies, which supports robustness, scalability and flexibility of the control plane and extended capabilities for monitoring and configuration of virtualized network functions and devices.

- Innovative SDN data plane optimization mechanisms, controllable through a standard SDN interface and based on extensions of current technologies to dynamically accelerate the flow processing while consolidating other workloads on a single commoditized multi-core platform.

- NaaS (Network-as-a-Service) API that provides a mean for applications to get an instantaneous abstract view of network resources to determine the amount of resources they can use and compute their optimal placement and to request network services.

- To strength the French research community with state-of-the-art advances on SDN and industrial take up.

3 State of the art

This section presents a number of research projects, standardization bodies and open source initiatives related to DISCO.

3.1 *Research projects*

- **FP7 OFELIA [1]**: OFELIA created an OpenFlow-based testbed where IT and network facilities providing OpenFlow (OF) capabilities to experimenters, researchers and projects. OFELIA facility had ten interconnected islands, whose heterogeneous network substrate is composed of L1/L2 OF-enabled equipment.

- **FP7 SPARC [2]**: The SPARC project aimed at implementing a new split in the architecture of Internet components. It also studies carrier grade extensions to split architectures including OAM, restoration and reliability, network virtualization, and resource isolation to opening carrier networks towards the benefits of split architectures.

- **FP7 NetIDE [3]**: This project aims at aiding SDN developers through an Eclipse-like IDE and associated tools to support the whole development lifecycle of network controller programs in a vendor-independent fashion.

- **FP7 ALIEN [4]**: This project proposes a novel concept of hardware abstraction layer to develop an OpenFlow based programmable network architecture over non-OpenFlow capable hardware (such as traditional L2 switches, optical switches, EPON devices, FPGA cards, network processor, etc.).

- **FP7 FIRE OpenLab project [5]:** “Extending FIRE testbeds and tools”. OpenLab brings together the essential ingredients for an open, general purpose and sustainable large scale shared experimental facility, providing advances to the early and successful prototypes serving the demands of Future Internet Research and Experimentation. OpenLab extends the facilities with advanced capabilities in the area of mobility, wireless, monitoring, domain interconnections and introduces new technologies such as OpenFlow.

- **FP7 SAIL [27]:** This project aims at designing technology that takes advantage of the fact that information and applications are mobile, and can be found in many places in the network. For this, end-users need to be able to address content directly, rather than addressing servers to get the closest copy, application providers need to be able to move applications and content around in the network quickly and automatically to fulfill the varying demand, and finally the network needs to adapt rapidly to connect applications and end-users, and take advantage of all available resources.

- **EIT ICT Labs KIC project on Software-Defined Networking (SDN) [6]:** The objective of this activity is to explore software-defined networking at different positions on the axis between basic flow-level processing (using OpenFlow for end-to-end flows) in controlled fixed networks and cooperation between mobile end nodes in the open wireless Internet (using opportunistic networking for resources communicated hop-by-hop).

- **ANR Equipex FIT (2011-2018) [7]:** FIT (Future Internet of Things) aims to develop an experimental facility, a federated and competitive infrastructure with international visibility and a broad panel of customers. It will provide this facility with a set of complementary components that enable experimentation on innovative services for academic and industrial users. FIT is one of 52 winning projects from the first wave of the French Ministry of Higher Education and Research's "Équipements d'Excellence" (Equipex) research grant programme. Other partners are UPMC, IT, Strasbourg University, INRIA and CNRS.

- **FASTPASS MIT [28]:** it provides a new paradigm of resource allocations into the datacenters so the applications can continue to scale while avoiding creating overheads at some networking nodes. Instead of using the usual high latency resource planning logic of an OpenFlow controller, the instantiation of the datapath is done at runtime of the applications while the TCP/UDP sockets are opened and closed. It requires an ultra-low latency high rate of request SDN orchestrator that aggregates visibility over the network in order to establish some runtimes L2 and L3 datapath.

3.2 *Standardization initiatives*

- The IETF's **Forwarding and Control Element Separation** (ForCES) Working Group focuses on standardizing mechanisms, interfaces, and protocols aiming at the centralization of network control and abstraction of network infrastructure [8].

- The **Open Network Foundation (ONF)** aims to standardize the OpenFlow protocol [9]. As the control plane abstracts network applications from underlying hardware infrastructure, they focus on standardizing the interfaces between: (1) network applications and the controller (i.e. northbound interface) and (2) the controller and the switching infrastructure (i.e., southbound interface) which defines the OpenFlow protocol itself.

- The **Software- Defined Networking Research Group (SDNRG)** at IRTF focuses on SDN under various perspectives with the goal of identifying new approaches that can be defined and deployed, as well as identifying future research challenges [10].

- Some of the Study Groups (SGs) of **ITU Telecommunication Standardization Sector** (ITU-T) are currently working towards discussing requirements and creating recommendations for SDNs under different perspectives [11].

- The IETF **Interface to the Routing System** (I2RS) Working Group facilitates real-time or event driven interaction with the routing system through a collection of protocol-based control or management interfaces [12]. These allow information, policies, and operational parameters to be injected into and retrieved (as read or by notification) from the routing system while retaining data consistency and coherency across the routers and routing infrastructure, and among multiple interactions with the routing system.

- **The Linux Foundation** is driving the open source initiatives that should instantiate the interconnect APIs of OpenDaylight and OpenNFV (OPNFV) [29].

3.3 Open source initiatives

3.3.1 SDN Data Plane

DPDK was initiated by Intel and quickly adopted by 6WIND since 6WIND was the first commercial company using it. Then, 6WIND turned this Intel DPDK to an open DPDK open source vendor neutral project [30]. It is aimed at providing open innovation for any open and commercial dataplanes (see below the vSwitch examples).

DPDK is a set of libraries (librte) that provides fast IO using kernel bypass technologies. It includes three set of libraries:

- the PCI IO librte which are named PMD (Poll Mode drivers)
- the memory management optimized to CPU architectures and including an optimal packet buffers (rte_mbuf)
- some examples and libraries of starting kits for those who need to be educated on how to achieve high bandwidth (60Mpps, 120Mpps) on legacy service architectures.

It does not include any networking stacks, but many commercial and open source projects are now embracing it.

It is mostly used on Linux OS even if other OSes can be used. It includes: support of 40G Mellanox, Intel 10G/40G, Emulex, which means that the fastest 40G NIC boards which are available on the markets are supported by the DPDK. Most NICs which are not supported by DPDK are due to a lack of performance that does not justify the effort of running with a stressful environment that requires DPDK (remember 40G means 60Mpps for 64 bytes or 30Mpps for 128 bytes packets).

It is already available into Fedora 20 and it should be available into commercial Redhat Linux by 2016.

Open vSwitch (ovs) is an open source implementation of a distributed virtual multilayer switch [13] , it had been the core of Nicira's assets. The main purpose of Open vSwitch is to provide a switching stack for hardware virtualization environments mostly using OpenFlow standard, while supporting multiple protocols and standards used in computer networks. Project's source code is distributed under the Apache License 2.0. It is now widely used in datacenters as a key component of Openstack based deployment of virtual networking either thru a SDN controller (OpenDaylight, Alcatel Nuage) or barely from Neutron's orchestration rules. OVS is based on three dataplanes: a kernel one (Linux kernel module), a userland one (available for Linux, Windows and BSD systems) and then since last June 2014, Nicira did include the netdev-DPDK support which allows performance improvement. It is already widely commercialized into commercial Linux distributions (Redhat Linux, Debian, Ubuntu, HP/Helion, Oracle Linux) and it is the only vSwitch that is mainstream into Openstack releases.

DPDK vSwitch (OVDK) is a fork of Open vSwitch [14]. Intel's team in Ireland re-created the kernel forwarding module (data plane) by building the switching logic on top of the Intel DPDK library to significantly boost packet switching throughput. The Forwarding engine incorporates Intel DPDK Huge Page Tables. The forwarding module runs in Linux user space with BSD license rights. Intel DPDK vSwitch implements a subset of the switching functionality of Open vSwitch. It is commercialized by Wind River. OVDK and netdev-DPDK from Nicira are two open source project in competitions. It is likely that OVDK will be terminated since its designed was rejected by the open source community and the Nicira's stakeholders. We can notice that OVDK brings an open innovation of vSwitching to the open source and research community while commercial companies like 6WIND, Cisco or Nuage were leading the commercial NFVIs.

Snabbswitch is a new design of vSwitching from a Swiss startup [31]. It is based on an innovative programming logic based of LUA+JIT. It can hook on DPDK in order to get the packets as most of applications using DPDK librte. Its dataplane use cases are driven by the new flat infrastructure from Deutsch Telecom Terastream that will optimize the OPEX and CAPEX of deploying end to end high 40G to 100G connectivity. It was blueprinted to be into Openstack Juno releases, it was supported by the Openstack-nfv group, but because of cross dependencies with Nova VIF drivers and Neutron, it has been postponed by the core dev team of Neutron and Nova. Its acceptance by the open source community may be an issue since Redhat has made the choice of OVS so the success of Snabbswitch would interfere with Redhat's productization process of OVS.

3.3.2 SDN Control Plane

Floodlight is a fork of the Beacon controller [15]. This OpenFlow controller is actively developed and has the support of Big Switch Networks' engineers who are actively testing and fixing bugs and building additional tools, plugins, and features. The Open Source community is well developed and many plug-ins and fixes were added by external developers in the project's repository. It can easily be modified and is able to support any size of network with consistent performances. Major improvement can be possible by the use of a more powerful and optimized hardware. We can notice that it seems to become a deprecated project. Now, most activities is switching to Ryu and OpenDaylight.

Ryu is a component-based SDN framework [16]. It provides software components with well-defined API that make it easy for developers to create new network management and control applications. Ryu supports various protocols for managing network devices, such as OpenFlow, Netconf, OF-config, etc. About OpenFlow, Ryu supports fully 1.0, 1.2, 1.3, 1.4 and Nicira Extensions. All of the code is freely available under the Apache 2.0 license. Ryu is fully written in Python. Ryu is currently loosing open source momentum.

OpenDaylight is a collaborative open source project hosted by The Linux with a modular, pluggable, and flexible controller platform at its core [17]. This controller is implemented strictly in software and is contained within its own Java Virtual Machine (JVM). As such, it can be deployed on any hardware and operating system platform that supports Java.

The controller exposes open northbound APIs which are used by applications. OpenDaylight supports the OSGi framework and bidirectional REST for the northbound API. The OSGi framework is used for applications that will run in the same address space as the controller while the REST (web based) API is used for applications that do not run in the same address space (or even necessarily on the same machine) as the controller. The business logic and algorithms reside in the applications. These applications use the controller to gather network intelligence, run algorithms to perform analytics, and then use the controller to orchestrate the new rules, if any, throughout the network.

The controller platform itself contains a collection of dynamically pluggable modules to perform needed network tasks. There are a series of base network services for such tasks as understanding what devices are contained within the network and the capabilities of each, statistics gathering, etc. In addition, platform oriented services and other extensions can also be inserted into the controller platform for enhanced SDN functionality.

The southbound interface is capable of supporting multiple protocols (as separate plugins), e.g. OpenFlow 1.0, OpenFlow 1.3, BGP-LS, etc. These modules are dynamically linked into a Service Abstraction Layer (SAL). The SAL exposes device services to which the modules north of it are written. The SAL determines how to fulfill the requested service irrespective of the underlying protocol used between the controller and the network devices.

Calico provides a new L3 and L2 over L3 approach to virtual networking based on IP scalability instead of L2 scalability [32]. Its goals is to provide an alternative to VxLAN, GRE and other technologies leveraging a new SDN control plane for the networks and leveraging the scale out of IP networking.

OpenStack Neutron is a cloud networking controller and a networking-as-a-service (NaaS) project within the OpenStack cloud computing initiative [18]. Neutron includes a set of application program interfaces (APIs), plug-ins and authentication/authorization control software that enable interoperability and orchestration of network devices and technologies within infrastructure-as-a-service (IaaS) environments.

Neutron was introduced as a core part of OpenStack with the initiative's Folsom release. Prior to the Folsom release, networking functionality was hard-coded in the Nova compute module of OpenStack, which required developers to modify both compute and network features of OpenStack together. With Neutron, networking is a more modular element of OpenStack that can evolve independently.

The core Neutron API includes support for Layer 2 networking and IP address management (IPAM), as well as an extension for a Layer 3 router construct that enables routing between Layer 2 networks and gateways to external networks. Neutron includes a growing list of plugins that enable interoperability with various commercial and open source network technologies, including routers, switches, virtual switches and SDN controllers.

Neutron can be used locally on compute and networking nodes of an IaaS and it can be used to interoperate with an OpenDaylight architecture that centralizes the SDN policies. It means that it includes some plugins to interwork with Ryu, OpenDaylight, etc...

4 System architecture

This section presents the scenarios, the stakeholders, the building blocks, the requirements that DISCO should satisfy and finally the DISCO architecture.

4.1 Scenarios

4.1.1 Content delivery scenario

Figure 2 presents the first scenario. In this scenario, a network operator aims at offering a content distribution service, for example medical data or VoD, this service being deployed on an infrastructure that can span from campus to country and continent and that is shared with other network services/tenants.

SDN enables operators to centrally control services, for instance through efficient load balancing that allows pre-caching of frequent content, but also to reduce network utilization as requests are served locally to increase QoE as latency and larger bandwidth can be provided.

In the SDN context, the operator must be able to have a consistent view of the available network resources and benefit from the same API to trigger the on-demand provisioning of network services. The software-defined networks must be able to optimize the allocation of the resources taking into account the properties of the content distribution services.

The end-users should have a one-time access to videos/data and pay per view/access. This involves authentication, accounting and on-the-fly encryption mechanisms. Transcoding to adapt to the device, mobility support and caching mechanism to avoid overloading the network with popular content should be functionalities provided by the network.

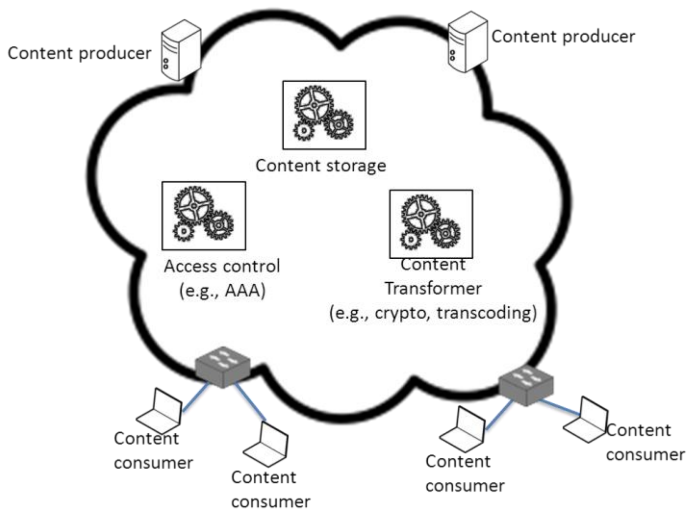

Figure 2: Content delivery scenario

4.1.2 Mission-critical overlay

Figure 3 presents the second scenario. Mission-critical networks now interconnect datacenters, enterprise, customer sites and mobile entities. They thus must be resilient, adaptable and easily extensible, especially when they support critical applications (e.g. from banks, energy plants, transportation…). They also have to adopt the cloud computing business model, which is on demand network services in multi-tenant environments, in order to monetize infrastructures and rationalize CAPEX and OPEX and move towards the "as a service" approach.

The emergence of SDN protocols, which enables to decouple the control plane from the data plane, opens up new ways to architect mission-critical networks. It allows fine-grained per-flow policies, faster innovation and network update with the software approach, tighter integration with network orchestrators and accelerate cloud-network convergence.

In this scenario, a network operator leverages network services provided by third telcos to create an overlay network interconnecting its campuses and datacenters. This overlay network, because of its centrality, must be robust and cope with attacks and failures. It must also provide on-demand network services taking into account the criticality of the requesting application/tenant but also the available resources.

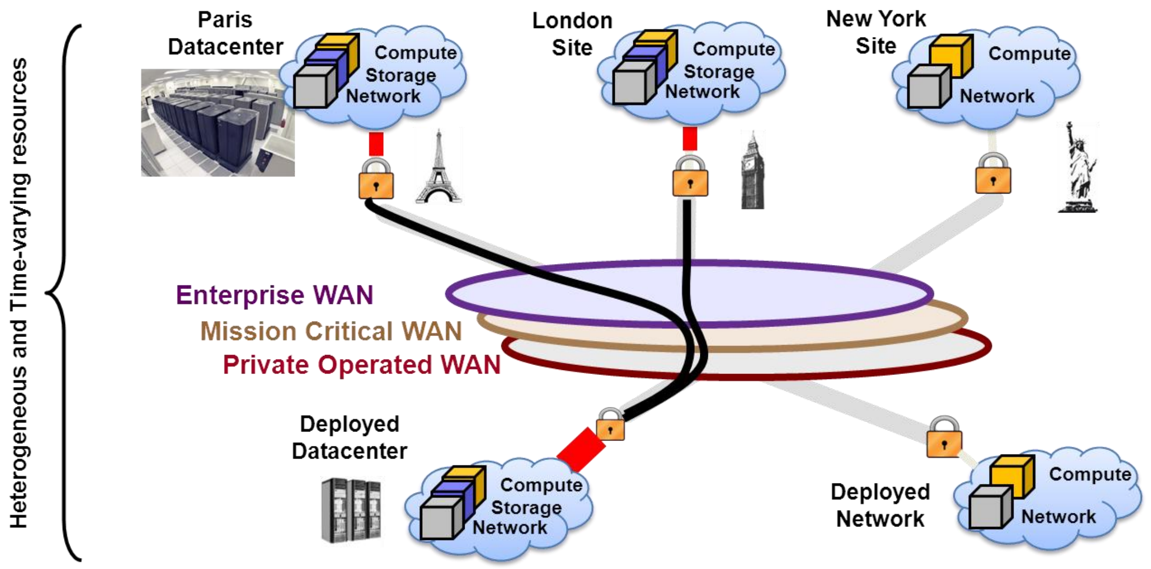

Figure 3: Mission-critical overlay scenario

4.2 Stakeholders and roles

Considering the new SDN technologies that DISCO is targeting, the following stakeholders are intended to be interested in or concerned by the DISCO architecture:

- **Network software vendor**: A network software vendor is a company that develops and sells software tools to network device manufacturers and system builders/integrators. For instance, 6WIND provides its high performance packet processing software suite

(6WINDGate™) to large telecom, network equipment providers and service provides or datacenter operators that build their vendor neutral IaaS.

- **Network equipment manufacturer**: A network device manufacturer is a company that manufactures and/or integrates different pieces of hardware and software to sell network equipment.

- **System builder/integrator**: A system builder/integrator is a company that integrates network equipment, management systems and pieces of software, potentially from different vendors, for the internet/network service providers and the IT department of enterprises.

- **Internet and Network service providers**: An ISP (Internet Service Provider) is a company that provides individuals and other companies access to the Internet and other related services such as Web site building and virtual hosting. A network service provider (NSP) is a company that provides backbone services to an ISP.

From the system's point of view, one main role can be identified:

- **Network administrator**: People in charge of the administration/management (supervision, monitoring and maintenance) of a network. Large enterprises with large networks as well as large network service providers typically have a network operations center, a room containing visualizations of the network or networks that are being monitored, workstations at which the detailed status of the network can be seen, and the necessary software to manage the networks.

4.3 Building blocks

4.3.1 SDN data plane

The data plane (sometimes also known as the user plane, forwarding plane, carrier plane or bearer plane) is the part of a network that carries user traffic. The data plane, the control plane and the management plane are the three basic components of a telecommunication architecture. The control plane and management plane serve the data plane, which bears the traffic that the network exists to carry.

The data plane enables data transfer to and from clients, handling multiple conversations through multiple protocols, and manages conversations with remote peers. Data plane traffic travels through routers, switches and vRouter/vSwitch of the NFV/compute nodes (NFVI), rather than to or from them.

In conventional networking, all three planes are implemented either the firmware of routers, of switches or as companion software of the hypervisors. SDN decouples the data and control planes and implements the control plane in software instead. This enables programmatic access, for instance via the OpenFlow protocol, to make network administration much more flexible.

Moving the control plane to software allows dynamic access and administration. A network administrator can shape traffic from a centralized control console without having to touch individual switches or update networking settings of the hypervisors. The administrator can change any network switch's rules when necessary - prioritizing, de-prioritizing or even blocking specific types of packets with a very granular level of control.

In addition, SDN leverages the implementation of the data plane in pure software or the open APIs of HW dataplanes, which enables to instantiate and tune virtual network topologies. This approach allows deploying virtual switches on commodity hardware, for instance on an IT server, thus reducing CAPEX.

4.3.2 SDN controller

A SDN controller is an application in SDN that implements flow control to enable intelligent networking. SDN controllers are based on protocols, such as OpenFlow, that allow servers (control entity) to tell switches where to send packets.

The controller is the core of an SDN network. It lies between network devices at one end and (end-users) applications at the other end. Any communications between applications and devices have to be assessed by the controller. The controller also uses protocols such as OpenFlow to configure network devices and chooses the optimal network path for application traffic.

In effect, the SDN controller serves as a sort of operating system (OS) for the network. By taking the control plane off the network hardware and running it as software instead, the controller facilitates automated network management and makes it easier to integrate and administer business applications.

4.3.3 SDN applications

SDN applications are programs that explicitly, directly, and programmatically communicate their network requirements and desired network behavior to SDN controllers via a northbound interface (NBI).

In addition they may consume an abstracted view of the network for their internal decision making purposes. SDN applications may themselves expose another layer of abstracted network control, for instance to provide Network as a Service (NaaS) or distribute content.

4.4 Requirements

4.4.1 SDN data plane

There are three sets of SDN data plane:

1. Infrastructure data plane running on DWDM/L2/L3 switches with some SDN APIs to be programmable. They are mostly implemented using ASIC or NPU,

2. Hypervisor stack dataplanes which are mostly software stacks compliant with ETSI NFV architectures

3. VNF stack dataplanes which can overlap with the infrastructure dataplane because it is a case by case design choice either to run the features (Firewalling, load balancing, L3 routing, etc.) as a VNF guest or as a dedicated infrastructure equipment. Both models are allowed.

The VNF data planes can be vBRAS (L2TP/PPPoE/accouting), vCPE, vFirewall, vIPsec, vNAS, vEPC (LTE), vNetworkAnalytics, etc. These VNFs shall provide the same services than the physical appliances. The amount of compute required by these packet processing nodes (physical or virtual – VNF) is correlated with the number of CPU cycles and the packet rate. In order to support elastic use cases, the amount compute required by these dataplanes can vary along with the requirements in throughput of a given service. For instance, a 1Tbps IPsec can be decreased to a 10G IPsec overtime that will have impacts on resources: using fewer cores on the compute nodes and using fewer nodes of the IaaS. The IPsec capacity planning can be either executed at the (1) infrastructure data plane level, at the (2) hypervisor level that is executing the NFVI dataplane or either at the (3) VNF level. As a matter of fact, for these 3 levels, the same dynamicity is required: turning up/down nodes, turning up/down cores on compute nodes.

4.4.2 SDN control plane

SDN relies on a logically centralized control plane. This logical centralization is needed to consistently control flows and enable intelligent networking on the network. Furthermore it facilitates the integration of SDN applications. However, several key features call for the logical but also physical distribution of the SDN control plane and thus for distributed controllers.

The distribution of SDN controllers precludes the presence of a single point of failure in the network architecture, especially for this central building block, which makes the glue between equipment and applications. Controller failures and disruptions can originate from hardware failures, bugs from the piece of software that implements the control logic and specific cyber-attacks on this entity.

The distribution of SDN controllers also enables to make the system scalable. While the size of network (traffic, devices, etc.) increases, additional controllers can be instantiated to take over some of the load at the control level. This allows having a flexible control plane, evolving according to the load.

Having multiple controllers requires that they are able to communicate one with the others to ensure consistent network-wide flow control, access control, monitoring, etc. A controller-to-controller API and communication channel must thus be defined. It should be able to convey large amounts of control traffic in case of tight synchronization between controllers (e.g. in data centers) and sparse control traffic in case of loose state sharing - eventually consistent – between distributed controllers (e.g. for access and deployable networks). The controllers should also be able to adapt the traffic they send/request to/from other controllers according the available resources (bandwidth, CPU etc.) at the data plane level.

Finally, the software architecture of an SDN controller shall be modular in order to integrate specific functions, potentially developed by third-party network software developers, such as access control, policy-based routing, monitoring etc.

4.4.3 Content distribution

It is nowadays complex and inefficient to distribute contents due to the increasingly stringent bandwidth and delay requirements of applications. Indeed, routing tables are constructed using fully distributed algorithms that prevent from fine-grained granularity of decisions as they would incur large signaling overhead and convergence time. Thankfully the introduction of SDN in network permits to use a logically centralized approach where a central entity can take consistent routing and forwarding decisions with a very fine level of granularity (e.g., potentially down to the packet level [19]). Such granularity helps to better utilize network capacity as it increases path diversity as compared to a traditional shortest path routing approach [20], [21]. Nevertheless, SDN is a networking solution focusing strictly on routing and forwarding and is thus not aware of contents. To help operators to implement efficient content distribution in their network, it is necessary to provide a content distribution API solution, that we call *NaaS API*, which is able to take into account at the same time network related matters (e.g.., links, router, failures, latency, bandwidth) and content related issues (e.g., accountability, catalogue provisioning, storage). In addition to that, in virtualized networks, operators must be able to provision without any service interruption the virtual appliances involved in the distribution of contents (e.g., virtual server, storage capacity, content customization elements).

The NaaS API has two functional requirements. On the one hand it must provide *monitoring* capabilities (i.e., what is happening in the network?). On the other hand it must supply *management* functionalities (i.e., influence the behavior of the network). The NaaS API also has the meta requirements of being *responsive* such that it can react fast to sudden events (e.g., flash crowd) and *modular* such that it can add functionalities in the network without having to change the API.

The monitoring part of the NaaS API can be refined in the following sub-classes:

- **network utilization**: determines how network resources are utilized (e.g., to detect congestion);
- **content utilization**: determines how content are utilized (e.g., content popularity);
- **appliance utilization**: determine how virtualized appliances are utilized (e.g., CPU utilization).

The management part of the NaaS API can be refined in the following sub-classes:

- **network management**: influence forwarding and routing decisions;
- **content management**: influence content placement and distribution;
- **appliance management**: influence appliance provisioning.

The network utilization is essential to be sure that forwarding decisions may not cause the network to collapse (e.g., increase of router's queues suggests the emergence of congestion events). Content utilization gives at the same time business-level information/accountability and essential technical information. As a matter of fact, it is not necessary to cache unpopular content in the network but it may be profitable to cache popular ones so to reduce server load and latency. Also, a content may be popular in a particular place (e.g., French movies in France) and seldom consulted in other places (e.g., French movies in USA). Finally appliance utilization is required to reduce costs and improve content delivery. On the one hand, if two appliances are under-utilized, they might be merged. On the contrary, if an appliance is over utilized, additional appliances should second it.

Network management consists in allowing the operator (by the mean of a SDN controller) to change forwarding decisions with a fine grained granularity so as to ensure each packet follows exactly the path that high level policies recommend. The content management is related to the possibility to enforce the storage of content in appliances decided by the high level policies but also to prevent their access to some class of users or for some period of time. Finally the appliance management is related to the capacity of increasing or reducing resources allocated to the appliance without service interruption and to allow seamless mobility of appliance between physical locations.

It is worth noting that constructing the best content distribution system is a joint optimization combining at the same time the (physical) network infrastructure, the (virtual) appliance placement, and the content catalogue usage. This optimization can only be done with the conduction of the three monitoring components and the three management components of the NaaS API.

4.5 *Putting things together – DISCO architecture*

The above stakeholders, roles, building blocks and requirements constitute the core elements that structure the DISCO architecture presented in Figure 4.

The SDN application leverages the NaaS API (a northbound API) exposed by the control plane to manage the distribution of content. The control plane is composed of distributed SDN controllers. Each controller manages several modules for network equipment and intelligent flow control, among them: discovery, monitoring, path computation and admission control. It allows choosing the method or algorithm needed for each functionality, for instance an advanced measurement-based admission control instead of a basic one. The controllers communicate among them through a controller-to-controller API. Mechanisms enable to adapt the quantity of control traffic they send/request according to the resources available at the data plane level, but also mechanisms to react to failures and disruptions both at the control plane and data plane levels. Finally the data plane is composed of commodity hardware on which run virtual switches and potentially other network functions. It can integrate specific pieces of software to optimize the performance, especially to automatically allocate CPUs according to the traffic load. The SDN controllers communicate with the network equipment through a standard SDN API, which can be extended, for instance the OpenFlow protocol.

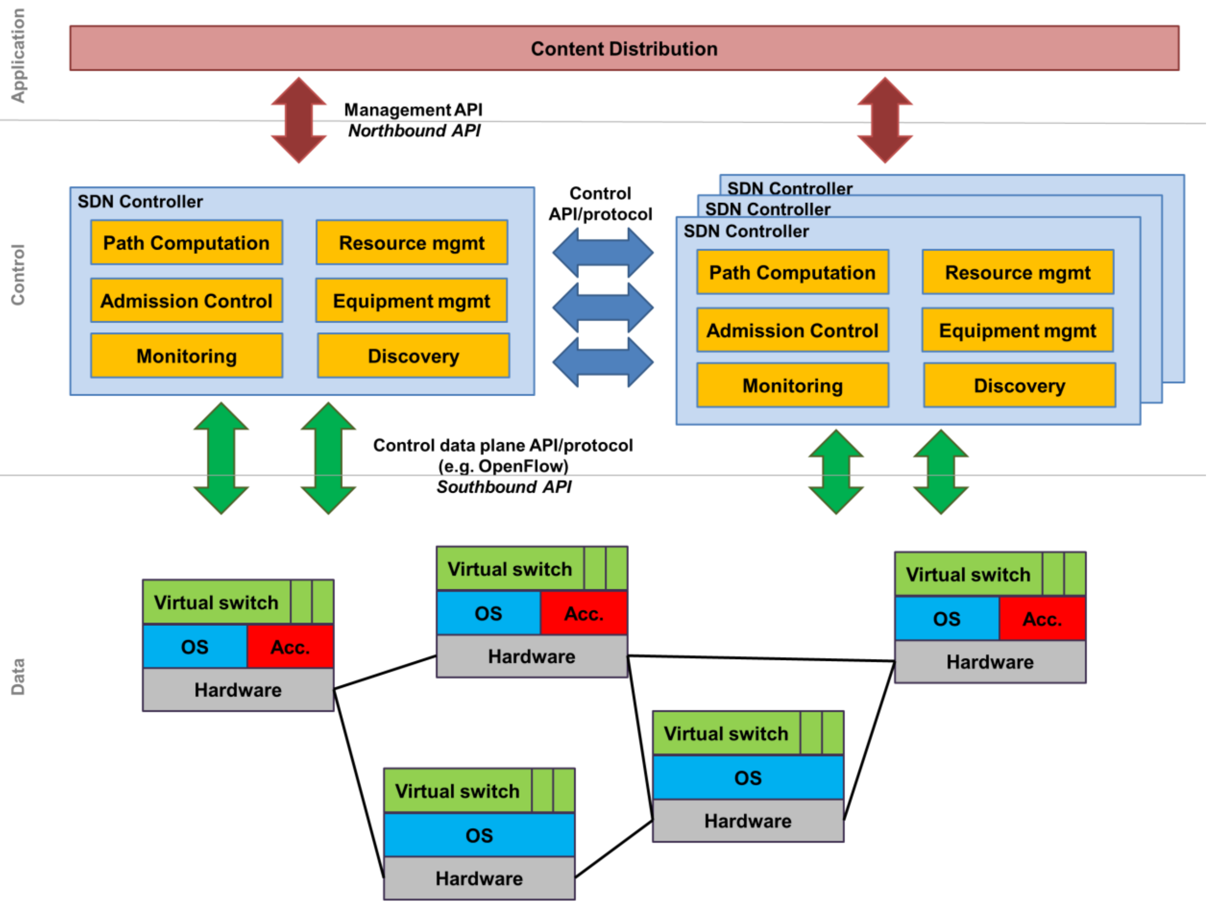

Figure 4: DISCO architecture

5 Research challenges

This section presents the research challenges derived from the DISCO architecture and the associated requirements.

5.1 Dynamic core allocation for data plane performance

The performance of a dataplane is tight to the number of clocks or CPU cycles which are required per packets. Once CPU cycles are missing, other cores from the same CPUs can be used (case 1). When these CPUs start missing resources (memory, bandwidth, cores, over-heating), other CPUs can be turned on (case 2).

The case (2) is about turning on/off some new nodes into the IaaS. It means that a new datapath needs to be spawned while the routing of the packets uses this new network capacity. 6WIND will assume that the SDN controller is able to define a new datapath.

The case (1) is about turning on/off some cores into a node of the IaaS.

6WIND will assumes that northbound SDN APIs exist to request for more or less capacities, then 6WIND needs to find solutions to scale up and scale down while doing non-stop-forwarding of processing. For instance, when an application is spawn based on DPDK, it has to be turned off/on every time the core masks need to be increase or decrease.

Even worst, the core allocations on a node for packet processing can be done at 2 levels: the vCPU of the cores allocated to the guests and the physical cores on the host side of the hypervisor. The hypervisor may need to add or remove some vCPU during run time in order to adjust the cost of the running virtual machines. While both guest and host core allocations can be updated, we need to study the optimal evolutions of matrix of interworking packets between the host and the guest sides.

On a NUMA system, it can be assumed that some physical nodes of the system can be dynamically added or removed (power on/off of CPU sockets). Every time you turn off a socket, you save:

- The watts of the CPU
- The watts of the memory connected to this CPU
- The watts of the PCI devices (network cards) connected to this CPU

We do not analyse the case when cores are turned off or on, since it'll be the same dynamic that expanding the data plane or a different number of cores.

In order to provide to the SDN controller some capabilities to make proper decisions, it is required to define and to provide APIs about the CPU usage of each cores allocated for packet processing.

5.2 Robust SDN control plane

The majority of current SDN architectures, OpenFlow-based or vendor-specific, relies on a single or master/slave controller(s), that is a physically centralized control plane. This centralization, adapted for datacenters, is not suitable for wide multi-technology multi-domain networks. In addition, the centralized SDN controller represents a Single Point Of Failure (SPOF), which makes SDN architectures highly vulnerable to disruptions and attacks [22].

Recently, proposals have been made to physically distribute the SDN control plane, either with a hierarchical organization [23] or with a flat organization [24]. These approaches avoid having a SPOF and enable to scale up sharing load among several controllers. However, these distributed SDN control planes have been designed for datacenters, where controller instances share huge amount of information to ensure fine-grained network-wide consistency.

In DISCO, we will address SDN for mission-critical networks, which can be used to interconnect datacenters, enterprise networks, customer sites and mobile entities (see scenario 2 described above). The distributed and heterogeneous nature of these deployments call for a distributed multi-domain network control plane which should be lightweight, adaptable to user or network requirements, and robust to failures. Current state of the art distributed SDN solutions are not suitable, as they do not provide a fine grained way to control and adapt inter-controller information exchanges.

In DISCO, we will also address SDN resilience studying and proposing mechanisms to cope with controller failure/disruption. If one controller fails, the switches that it piloted then become orphan. A failover mechanism has to enable to migrate the orphan switches to other SDN controllers already active in the network. Another approach could consist in instantiating a new controller on the same domain.

5.3 Network-wide distributed and multi-class admission control

In a packet switching communication, admission control (AC) may accept or refuse access to incoming flows depending on the availability of the network resources (typically the bandwidth in the case of a network). By regulating the number of on-going flows, AC aims at preventing the occurrences of overloading, congestion and performance collapses, so that, accepted flows experience a good Quality of Service (QoS), which is of utmost importance for delay-sensitive applications and resource- intensive applications.

The emergence of SDN to build dynamic networks with a programmable control plane will provide new tools to devise a distributed and scalable admission control. The control plane should enforce local policies for the network appliances that together aim at realizing a network-scale QoS objective.

Measurement-based admission control (MBAC), which exclusively requires on-line measurements to be run, represents an attractive class of solutions as compared to endpoint admission control and traffic descriptors based solutions. However, and despite the abundant literature devoted to this topic, virtually all existing MBAC solutions are limited to the case of a single link, instead of considering the case of a complete network. This limitation tends to reduce their attractiveness for the operators.

A possible option to apply a given MBAC solution to the case of a network consists in performing admission control (AC) at each link based on measurements collected at this link. However, we discard this solution as it appears hazardous how the several locally applied AC's should be combined so that their outcome results in a satisfying global admission control (i.e., at the network scale).

Hence, we chose an alternate option that rather aims at combining the local measurements collected at each link into global measurements pertaining to the network. Note that we assume that getting directly the global measurements is a too costly and complex process, e.g., in the case of the delay, its measurement requires the existence of a timely synchronized clock. Once this combination step is performed, we aim at relying on a refined MBAC approach to control the admission of incoming flows.

For instance, let us consider the case of an end-to-end delay along a network path (composed of several link hops). Its expected value is equal to the sum of the expected delays at each link composing the path from the source to the destination. However, we are looking to get a finer prediction that would also include an estimation of the actual standard deviation of the end-to-end delay. Under independence assumption, this standard deviation could be simply computed as the sum of the standard deviations measured at each link. However, this independent assumption does clearly not hold in this case.

5.4 *Network appliances placement*

In order to efficiently deliver on-demand network services to multi-tenant, the utilization of the low level resources (compute, storage, and network) available in the network has to be optimized. Commodity hardware, such as servers, can host several virtual machines embedding network functions (e.g., virtual router, Deep Packet Inspection engines, ciphers, load balancers, caches). However, compute, storage, and network resources have to be jointly managed. These capacities are indeed tightly dependent. For example, a content distribution service will require distributed storage resources and communication between them for synchronization or updates.

As mentioned previously, DISCO will tackle the performance issue at the network equipment level proposing dynamic core allocation schemes for the different network functions. But, DISCO will also investigate the performance issue at the network-wide level. Indeed, it is necessary to optimize the placement and the usage of virtualized network appliances to limit their energy consumption, but also their quantity, and thus the associated license fees. This issue becomes increasingly complex in the case of multi-tenant networks, where the physical networking infrastructures (equipment and links) are shared among multiple virtual networks, each tenant having its own virtual networks. In this context, optimizing the placement of the network appliances for each tenant will ensure a proper repartition of the resources while providing dedicated network services and extended monitoring and configuration capabilities to the SDN control plane.

In parallel to the optimization issue, the placement of distributed SDN controllers is another challenge for the management of appliances in order to have the minimal footprint on the physical layer while assuring high level services. Several initial works have been proposed, for example [25]. However, inspired by the virtual machine placement problem in datacenters, they do not take into account critical constraints such as the network footprint of the control traffic corresponding to the traffic generated by the controllers to share information in the control plane. The placement of a controller has also to take into account the underlying topology, especially the latencies with the virtual switches/routers in order to ensure a small response time. In DISCO, we will thus leverage the distribution of the SDN controllers to investigate this issue.

5.5 NaaS API exploitation for distributing content

As mentioned in Sec. 4.4.3., the NaaS API has functional requirements as well as meta requirements in order to fulfill the needs inherent to content distribution virtualized network infrastructures.

Developing monitoring capabilities represents a challenge when the size of the network or its workload (i.e., traffic and content catalogue size) becomes very large. The problem resides in the ideal need of obtaining monitoring information in real time. The challenge consists in finding an information abstraction that is precise enough to fulfill all operational needs and aggregated enough to avoid high signaling overhead without loss of essential information. More precisely, it is complex to determine *(i)* the most appropriate aggregation level and *(ii)* the frequency at which monitoring information is retrieved. In addition, as networks are distributed systems by nature, the same event (e.g., a download) can be observed from different points, potentially with different observations which may cause inconsistencies of decision when observations are not joint correctly. Finally, in some cases the information to monitor is not straightforward to obtain (e.g., instantaneous bandwidth or CPU utilization) and the simple fact of measuring it may alter the observation.

The management part of the API itself is less challenging as at the end it can be summarized as *add*, *copy*, and *remove* functions. However, in this project our target is to respect SLA that might be very strict in terms of packet loss or downtime. This strict requirement imposes to orchestrate the call to such functions in an appropriate way as atomic operations are impossible to produce network-wide. For example, the sequence followed to install forwarding rule in the network is essential as a wrong sequence may cause transient loops [26]. When it comes to virtual appliance mobility the problem is even harder as it requires moving active state without interruption and the simple fact of moving the state may cause state changes.

The latter problem resides in the responsiveness of the system and a trade-off must be found between speed of reaction and signaling overhead.

6 Conclusion

This deliverable has presented the key components of the DISCO platform as well as their associated requirements and research challenges that the projects aims at satisfying and targeting.

This deliverable is a first report that is expected to evolve during the project. The final version of it will consist in the deliverable D1.2.

7 References

[1] FP7 OFELIA - http://www.fp7-ofelia.eu/

[2] FP7 SPARC - http://www.fp7-sparc.eu/

[3] FP7 NetIDE – http://www.netide.eu/

[4] FP7 ALIEN – http://www.fp7-alien.eu/

[5] FP7 FIRE OpenLab – http://www.ict-openlab.eu/

[6] EIT ICT Labs KIC project on Software-Defined Networking – http://www.mobilesdn.org/

[7] ANR Equipex FIT – http://fit-equipex.fr/

[8] IETF FORCES – http://tools.ietf.org/html/rfc5810

[9] Open Networking Foundation (ONF) – https://www.opennetworking.org/

[10] IRTF SDNRG – http://irtf.org/sdnrg

[11] ITU-T SDN - http://www.itu.int/en/ITU T/sdn/Pages/default.aspx

[12] IETF I2RS - https://datatracker.ietf.org/wg/i2rs/documents/

[13] Open vSwitch – http://openvswitch.org/

[14] Intel DPDK vSwitch – https://01.org/packet-processing

[15] Floodlight – http://www.projectfloodlight.org/

[16] Ryu – http://osrg.github.io/ryu/

[17] OpenDaylight – http://www.opendaylight.org/

[18] OpenStack Neutron – https://wiki.openstack.org/wiki/Neutron

[19] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal, “Fastpass: A Centralized Zero-Queue Datacenter Network,” in ACM SIGCOMM, August 2014.

[20] S. e. a. Jain, “B4: Experience with a globally-deployed software defined WAN,” in ACM SIGCOMM, 2013.

[21] X. N. Nguyen, D. Saucez, C. Barakat, and T. Turletti, “Optimizing Rules Placement in OpenFlow Networks: Trading Routing for Better Efficiency,” in ACM HotSDN, Apr. 2014.

[22] D. Kreutz, F. Ramos, and P. Verissimo, “Towards secure and dependable software-defined networks,” in HotSDN, 2013.

[23] S. H. Yeganeh and Y. Ganjali, “Kandoo: a framework for efficient and scalable offloading of control applications,” in HotSDN, 2012.

[24] A. Tootoonchian and Y. Ganjali, “Hyperflow: a distributed control plane for OpenFlow,” in INM/WREN, 2010.

[25] Brandon Heller, Rob Sherwood, and Nick McKeown, “The controller placement problem”, in Proceedings of the first workshop on Hot topics in software defined networks (HotSDN '12).

[26] Laurent Vanbever, Stefano Vissicchio, Cristel Pelsser, Pierre Francois and Olivier Bonaventure. Seamless Network-Wide IGP Migrations. Proceedings of the 2011 ACM SIGCOMM Conference, Toronto, Canada, Aug. 2011. ACM.

[27] FP7 SAIL – http://www.sail-project.eu/about-sail/

[28] FASTPASS MIT - http://fastpass.mit.edu/

[29] The Linux Foundation - http://www.linuxfoundation.org/

[30] DPDK - http://dpdk.org/

[31] Snabbswitch - https://github.com/SnabbCo/snabbswitch

[32] Colico - https://github.com/Metaswitch/calico