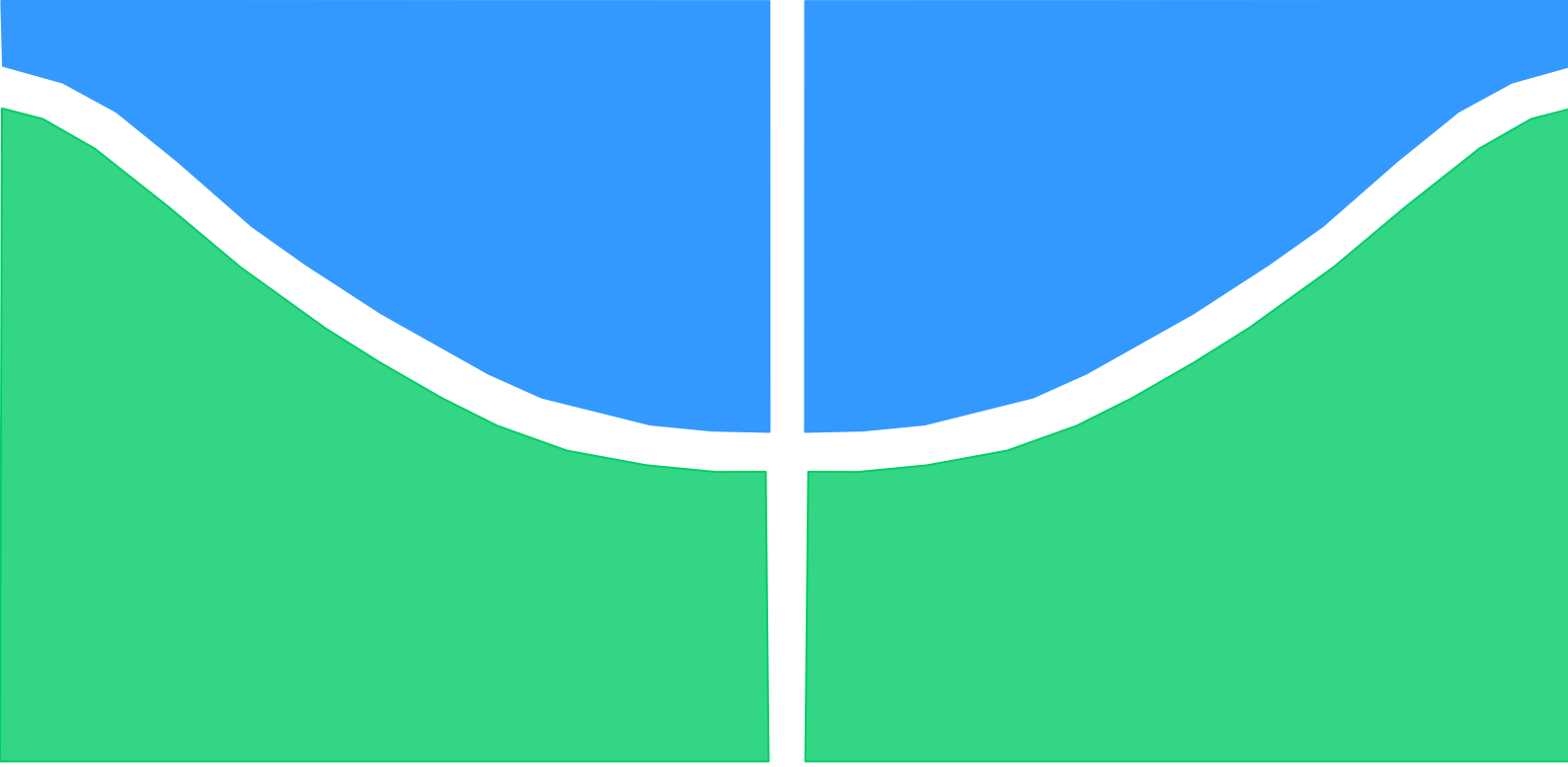


TRABALHO DE GRADUAÇÃO

A PROPOSAL TO AUTOMATE SERVICE PROVISIONING IN SERVICE PROVIDERS BASED ON SOFTWARE DEFINED NETWORKING

Flávio Augusto de Castro Júnior

Brasília, Julho de 2014

UNIVERSIDADE DE BRASILIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA

Faculdade de Tecnologia

TRABALHO DE GRADUAÇÃO

A PROPOSAL TO AUTOMATE SERVICE PROVISIONING IN SERVICE PROVIDERS BASED ON SOFTWARE DEFINED NETWORKING

Flávio Augusto de Castro Júnior

Relatório submetido ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Engenheiro de Redes de Comunicação

Banca Examinadora

Prof. William Ferrreira Giozza - ENE/UnB	_____
Prof. Georges Daniel Amvame-Nze – ENE/UnB	_____
Prof. Valério Aymoré Martins – ENE/UnB	_____
Me. Fernando López Rodriguez – ENE/UnB	_____

RESUMO

1 INTRODUÇÃO

As redes de hoje são estáticas e difícil de modificar, enquanto o tráfego de dados cresce devido a mudanças nos padrões de consumo e crescente disponibilidade de tecnologias de acesso móveis. O volume de dados vem aumentando; a tolerância a falhas dos usuários vem diminuindo; e as receitas não crescem tão rapidamente como tráfego. Provedores de serviço sofrem pressão crescente para inovar, reduzindo custos operacionais e diminuindo o tempo de entrega de novos serviços para, então, gerar novas fontes de receita.

Contudo, o provisionamento de serviço ainda é extremamente acoplado à infraestrutura de rede. De forma que, dispositivos de redes são, comumente, configurados manualmente e individualmente acarretando em ineficiências operacionais. Hoje, é necessário, basicamente, *re-projetar* toda a rede cada vez que precisar entregar um novo serviço. Uma solução de rede definida por software pode automatizar o provisionamento de serviço e permitir novos níveis de inovação possibilitando a ampliação das estratégias de monetização de seus serviços.

O objetivo desse trabalho é apresentar um modelo provisionamento de serviços baseado em redes definidas por software para automatizar a ativação, desativação e reativação de serviços; e ainda permitir a entrega de serviços diferenciados para cada usuário, por meio do controle ativo da quantidade e velocidade de tráfego permitido na rede para cada usuário.

2.1 INOVAÇÃO EM PROVEDORES DE SERVIÇO

O modelo de negócios utilizado por provedores de serviço vem mostrando dificuldades de escalabilidade conforme o volume de dados cresce. *O tráfego oferecido a provedores de serviço cresce de forma quase exponencial enquanto a receita proveniente de serviços de rede tende à estagnação. Dessa forma*, "o gap entre receita e investimentos necessários para suprir as expectativas de qualidade de serviço dos usuários está crescendo cada vez mais, como ilustrado na Fig. 2.4. Por causa disso, o modelo de negócios atual pode se tornar insustentável no futuro.

A situação enfrentada por provedores é desafiante, principalmente, por causa dessas quatro características combinadas:

- Decrescente receita média por usuário.
- Competição intensa a partir de serviços de camadas superiores, como Netflix.
- Modelo de negócio baseado em quantidade de assinantes
- Saturação do mercado de conexão com a internet

É necessário inovar para reduzir custos operacionais ou monetizar as estruturas de rede existentes.

Ao mesmo tempo, o atual paradigma arquitetural de redes de comunicação vem se demonstrando obsoletos frente aos desafios modernos. Provedores de serviço têm capacidade limitada de lançar novos serviços ou adaptar a rede para suportar as necessidades de novos serviços. O atual modelo arquitetural de redes não foi projetado para ser *evoluível* ou entregar novos serviços de forma rápida.

Operadores precisam aprimorar os serviços correntes com informação inteligente da rede; e fornecer novos serviços de forma rápida utilizando ferramentas simples de gerenciamento de forma a reduzir custos e o tempo de implantação dessas operações.

2.2 REDES DEFINIDAS POR SOFTWARE EM PROVEDORES DE SERVIÇO

A popularização do paradigma de redes definidas por software (software defined network - SDN) é oportuna frente aos desafios enfrentados por provedores de serviço. SDN basicamente significa a extração das funções de controle de rede da estrutura física e centralização da inteligência de rede em um ponto de controle único e potencialmente virtualizado como ilustrado na Fig. 2.6. Existem diversas abordagens para implementação de SDNs como ilustrado na Fig. 2.5.

Novas possibilidades de monetização de serviço para provedores pode ser implementada por meio do link direto entre aplicações e o controlador de rede, presente na arquitetura SDN. Por exemplo, é possível automatizar a alocação de recursos de rede de forma a prover serviços de forma dinâmica e otimizada para demandas específicas provenientes dos usuários da rede. O modelo SDN pode diminuir custos operacionais, agilizar a ativação de serviços, e portanto, enriquecer serviços existentes.

Uma possibilidade de inovação em provedores de serviço é a criação de um serviço de *Bandwitdh on demand*. Os padrões de tráfego em rede estão mudando rapidamente. Tecnologias como big data estão criando picos de tráfego extremamente altos e curtos. Neste cenário, a contratação de uma taxa constante de transmissão se torna muito cara, ou não suficiente. Um serviço que permita a contratação de banda de transmissão que permita redimensionar ou reativar a conectividade de rede dinamicamente é muito mais interessante. Esse tipo de tecnologia iria reduzir custos para esse tipo de cliente, pois eles pagariam somente pelo que consumissem.

Uma estratégia de provisionamento de serviços baseada em SDN pode permitir o controle automático da rede. Essa estratégia pode também proporcionar uma visão em tempo real da rede. Esses duas funcionalidades combinadas permitem a alocação inteligente e automática de recursos de rede de forma a suprir diferentes requisitos de nível de serviço para cada usuário.

3 MODELO ARQUITETURAL

Nesse trabalho, propomos um modelo de rede baseado na arquitetura de redes definidas por software que pode agilizar a ativação de serviços e facilitar a entrega de serviços customizados, portanto, permitindo a criação e utilização de novos padrões de nível de serviço. Para fazer isso, aplicamos uma arquitetura de rede definida por software baseada em OpenFlow, ilustrada na Fig. 3.1, para controlar a rede e reforçar as políticas de utilização de rede e limitações de volume de dados.

O comportamento da rede é controlado pelas aplicações de rede implementadas no controlador: ativação de serviço; controle de velocidade e monitoração de volume de dados. Políticas de rede externas são reforçadas de acordo com uma base dado externa, por meio das aplicações de rede. A Figura 3.2 ilustra essa arquitetura.

O sistema funciona da seguinte forma: cada vez que um pacote chega a um switch, esse switch procura por uma entrada correspondente a esse pacote em sua tabela de fluxos. Se um pacote não tem entradas correspondentes, então o switch direciona esse pacote ao controlador da rede. Em seguida, o controlador identifica o usuário que gerou o pacote baseado no seu endereço IP e sua localização física e procura autorização de serviço para aquele usuário. Se confirmado, então o tráfego de rede para aquele usuário será permitido e limitado de acordo com as políticas de rede, utilizando a funcionalidade *meter* do OpenFlow 1.3 para limitar a taxa de transmissão daquele usuário. Simultaneamente, o controlador requisita todos os dispositivos de rede, repetidamente, informações sobre estatísticas de “metering” e, em seguida, atualizar as estatísticas de utilização. Posteriormente, de acordo com essas atualizações, o controlador decidirá sobre a desativação, reativação ou aprimoramento. Se aquele usuário não for permitido na rede de acordo com a base de dados de políticas de rede, então esses fluxos serão instruídos a descartar os pacotes como ação correspondente.

4 ESTUDO DE CASO

Para demonstrar a aplicabilidade da solução proposta, emulamos uma rede simples no Mininet e a conectamos ao controlador RYU integrado às aplicações de rede desenvolvidas: ativação de serviços, controle de velocidades, monitoramento de volume de dados. Para demonstrar a efetividade da solução proposta, resumimos o estudo de caso a 5 cenários:

- START: teste de ativação de serviços automática, limitada às políticas de limitação de velocidades.
- STOP: teste de desativação de serviços automática, limitada às políticas de volume de dados.
- RESTART: teste de reativação de serviços automática.
- STOP AGAIN: teste de desativação de serviços
- UPGRADE: teste de reativação de serviços de acordo com novas políticas de rede.

As etapas de teste e seus resultados são detalhados no trabalho completo.

5 CONCLUSÃO

No estudo de caso desenvolvido projetamos e desenvolvemos uma solução que implementa provisionamento de serviços rápido e automático; e permite controle de utilização da rede de forma personalizada. E portanto permite operadores de redes a implementar novas estratégias de monetização baseadas na entrega automática de níveis de serviço customizados a diferentes usuários.

A característica automática do sistema foi evidenciada por meio dos testes na seção 4 e é baseada na utilização de uma base de dados de controle de rede como interface com sistemas finais. Essa estratégia permite várias possibilidades para automação como a criação de sistemas de ativação de serviço que forneçam uma interface direta a clientes de provedores de serviço. Nesse cenário, o sistema pode ser projetado de forma que os sistemas de cobrança também sejam automaticamente conectados ao modelo de provisionamento de rede. E portanto permite aos provedores a implementação de uma estratégia de *Bandwidth on Demand*.

Nesse trabalho, apresentamos uma solução de provisionamento de serviço com tempo de resposta menor que 200 milissegundos e tempo de reativação menores que 1 segundo. Nós também implementamos controle de taxa de transmissão e monitoração de volume de dados trafegado com pequenas imprecisões de 10% e 20%, respectivamente. Pudemos verificar que a funcionalidade de *metering* do OpenFlow 1.3 pode ser aplicada para prover níveis de serviços customizados e pode ser usada para controlar a velocidade de tráfego UDP e TCP.

Palavras-chaves: Redes definidas por software, redes de computadores, Openflow.

ABSTRACT

This work proposes a useful network model to automate service provisioning in service providers based on software-defined networking. We performed a study case based on the proposed model and simulated it over an emulated network in Mininet to verify the proposed features. In the study case, we designed and developed a solution to implement fast and automatic service provisioning; and control network utilization in a customized approach. Therefore, we provided a solution that allows network operators to implement new monetization strategies by delivering customized service levels automatically to different users.

Keywords: Computer Networking, Software-defined networking, Openflow.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SYMBOLS AND ABREVIATIONS

QoS	Quality of Service
ISP	Internet Service Provider
TCP	Transmission Control Protocol
IP	Internet Protocol
VPN	Virtual Private Network
SLA	Service Level Agreement
QoE	Quality of Experience
SDN	Software defined networking
ONF	Open Networking foundation
WAN	Wide area network
VoIP	Voice over IP
API	Application programing interface
UDP	User Datagram protocol
RTT	Round-trip time
Bps	Bits per second

1 INTRODUCTION

This chapter describes the context and motivations for this work. We also introduce the approached problem and the proposed solution and, briefly, go through the work organization structure.

1.1 CONTEXT AND MOTIVATIONS

Today networks are static and hard to modify and evolve, whereas data traffic grows due to changes in usage patterns and the growing pervasiveness of mobile data access technologies. The volume of data is getting bigger, users' tolerance to failure is getting lower and revenue is far from increasing as faster as traffic. Service providers are suffering increasing pressure to innovate, reduce its operational costs and decrease delivery time for new services to generate new sources of revenues.

The main motivation for this work was to apply innovative networking technologies to real-world problems widely faced in the industry, specifically service provisioning in a service provider environment. The popularization of the software defined networking paradigm is opportune to solve the network operators' challenges. In this work, we want to propose a solution evidencing that the software defined networking paradigm can fasten service delivery and ease network operation.

1.2 PROBLEM

Nowadays, service provisioning is extremely tied to the physical network infrastructure. Moreover, network devices are often manually and individually configured increasing operational costs. Right now, we basically have to re-engineer the whole network each time we want to deliver a new service. A software defined networking solution can automate service provisioning and promote a higher level of innovation allowing providers to embrace new monetization strategies, such as pay-for-QoS, or bandwidth on demand.

1.3 OBJECTIVES

The goals of this work are to present and implement service provisioning based on a software defined networking architecture to automate service activation, reactivation and deactivation, and permit service customization on a per-user basis, by automatically controlling the network speed and volume of data permitted in the network.

1.4 WORK PRESENTATION

This work is divided in the following way. Chapter 2 presents some of the challenges faced by service providers, discusses the necessary innovations in the industry, gives an overview of software defined networking architectures and finally describes some innovation opportunities for software defined networking in service providers.

Chapter 3 presents the architectural model of our study case and how our propose aims to automate, ease and fasten service provisioning. Chapter 4 describes our simulation outcomes and analyze those outcomes according expected results. Chapter 5 presents our conclusions and suggests related works to amplify the effect of the service provisioning architecture proposed.

2 THEORETICAL BACKGROUND

This chapter defines the main concepts necessary to the understanding of the presented study case. We briefly describe some challenges faced by service providers and how software-defined networking architectures can be applied to solve these problems.

2.1 SERVICE PROVIDER NETWORKS

An extraordinary technologic development has been experienced in the last decades. Some people assign this development to two principal components: first, the growing pervasiveness of computation and second, the increasing access availability to information [5]. Part of these technologic advances are due to the development in the communication networks field, such as the construction and popularization of the Internet. Internet access availability is increasing, and, in parallel, access to information is becoming easier and cheaper. This situation wrongly, gives the mistaken impression that internet connectivity provisioning is a trivial operation. This operation suffers a lot from scalability problems, and it is becoming a great challenge as both increase: the number of users connected to the internet and the average, per user, data volume [5].

By definition, the internet is an interconnection between networks provided by the TCP-IP network layer model. Internet access is normally obtained by purchase of an internet connectivity service from an Internet Service Provider (ISP). A Communication Service Provider (CSP) delivers connectivity between, residential or mobile end-users, enterprises, other providers and the internet. The ISP is an example of service provider which provides internet connectivity using its network. Figure 2.1 illustrates the internet from an end-user's point of view: the user buys a connectivity from an ISP that forwards its traffic to the rest of the internet through its private network. [1]

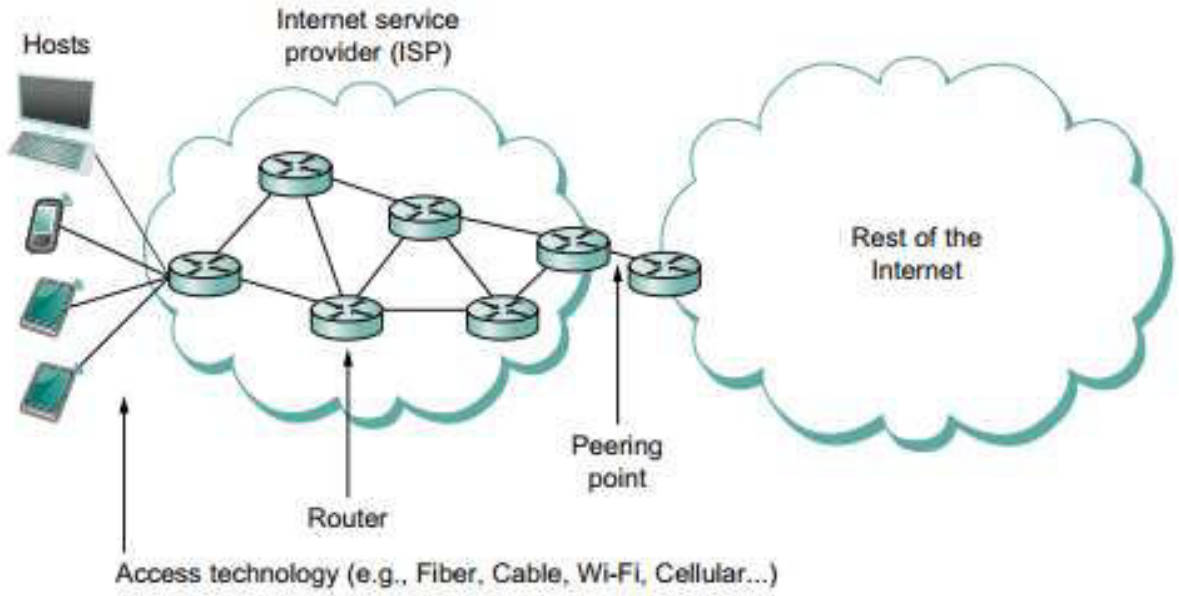


Figure 2. 1: An end-user's view of the internet [1].

However, Figure 2.1 represents an oversimplification of the real world. The service providers capable of providing a global access to every other ISP and, hence, the internet are considered Tier-1 ISPs. Tier-1 ISPs have global reachability to the internet; that means that these handful of ISPs have routes to all reachable Internet prefixes. The remaining providers are classified as Tier 2 or 3. Tier-3 ISPs are small providers that have a small number of end-customers, normally geographically centralized. Tier-2 ISPs normally have a regional scope or larger geographical reach but still cannot deliver global connectivity.

Tier 2 and 3 ISPs have limited access to the internet and have to purchase transit services from Tier 1 ISPs, in this case, they are referred to as Consumer ISPs. This scenario is exemplified in Fig. 2.2. [1][2][16]

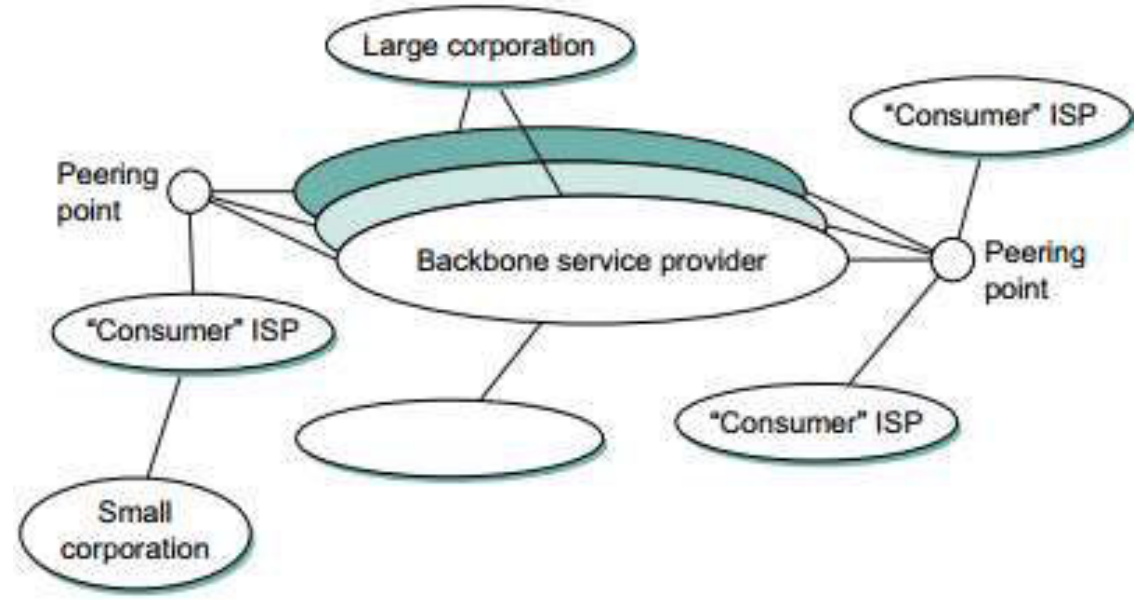


Figure 2. 2: A Simple multi-provider Internet [1].

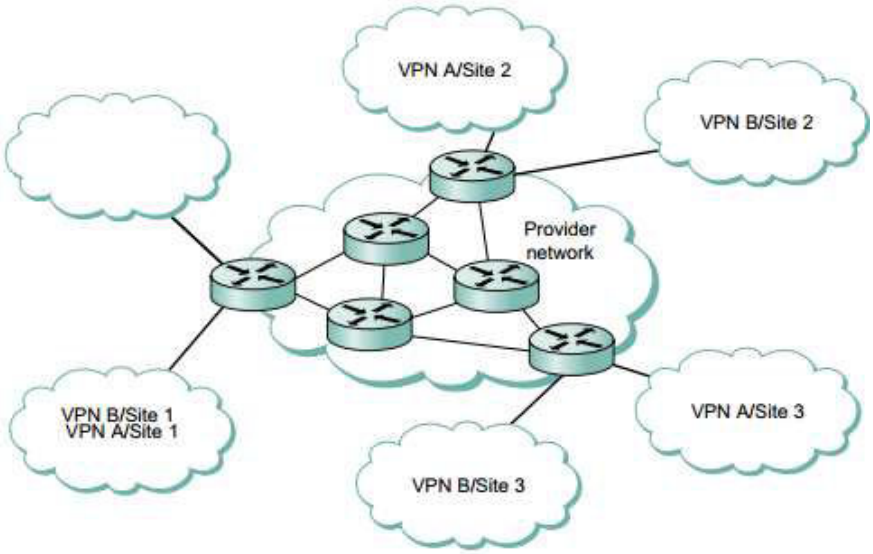


Figure 2. 3: Illustration of Layer 3 VPN connectivity service. Customers A and B each obtain a virtually private IP service. [1]

Several types of services are delivered by service providers, such as, data or voice connectivity between distant branches of a company. In Figure 2.3 a VPN connectivity service is exemplified. [1]

2.1.1 SERVICE LEVEL AGREEMENT (SLA)

In order to define a formal agreement between service providers and their customers, the services delivered are defined by means of a Service Level Agreement (SLA) between provider and customer. The SLA will describe the provided services in detail and it will be a common point between customer expectations of service delivery and the actual service delivery by the provider, in fact, we want them to be the same thing. When a provider is capable of delivering services according to its SLA definitions, it is said that it can guarantee Quality of Service (QoS). Some providers publish their SLAs publicly on

the internet. For instance, Verizon, a large network provider in the USA, has several of its Service Level Agreements published on the internet. [3]

The creation of new types of SLAs is not prohibited, since it is basically a business agreement. In fact, the creation of new types of SLA creates room for innovation in the industry. Additionally, The SLA defines how service level is measured, and possible penalties due to failure to comply with the SLAs.

There are several types of common SLAs, such as internet connectivity SLAs, VoIP traffic SLAs, VPN connectivity SLAs, WAN connectivity SLAs and etc. The metrics for definition of SLAs include but are not limited to [3]:

- End-to-End Delay
- End-to-End Jitter
- Network packet delivery rate
- Network availability
- Average connection speed
- Mean Time to Repair
- Mean-Opinion-Score for VoIP traffic.

2.2 NETWORK INNOVATION FOR SERVICE PROVIDERS

The technological development faced in the last decades has enabled us to reach a progress speed that is surprising even to the most optimistic persons. The high availability of information is much correlated with innovation speed. Business competition and the need to evolve current business models increase as the innovation pace increases too. This is also a challenge for service providers and telecommunication operators. [5]

The pervasiveness of internet access; the increasing proliferation of mobile devices; combined with the progressive expansion of mobile applications consuming data services are providing a substantial emergent increase in the traffic offered to providers. This data traffic is growing at an almost exponential pace, whereas the revenue prevenient from network services is leaning towards stagnation. Figure 2.4 illustrates the rising gap between revenues and necessary investments to supply users' quality of experience (QoE) expectations. [5]

Due to the provider's rising gap between costs and revenue, the actual business model of network operators of building infrastructure then selling services will probably come to be unsustainable in the future, occasioning on higher prices or reduced supply of services. The situation faced by network operators is challenging, mainly, because of these 4 characteristics combined: [5]

- A decreasing average revenue per user;
- Intense competition for over the top services, such as Netflix;
- A business model based on the volume of subscribers;
- The saturation of the internet connectivity market.

The operators' ability to produce income from existing infrastructure can define the success or failure of the industry as it is today. This condition creates a lot of pressure over operators towards new monetization strategies. The service providers are being forced to develop new business models, capable of adding differentiated value to customers by means of new user's experiences, taking advantage of new service levels. Certainly, service providers need to evolve.

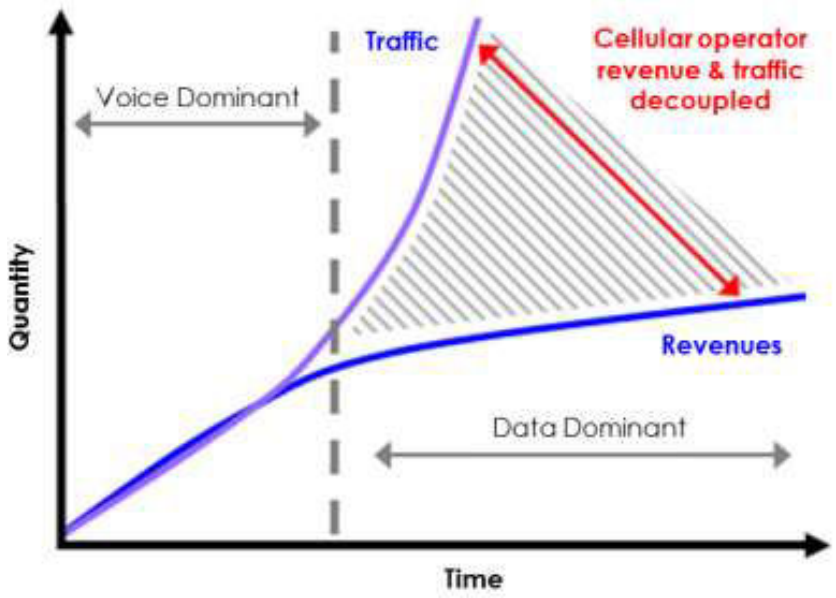


Figure 2. 4: Traffic Volume and Revenue Decoupling [5]

However, the actual architectural network paradigms are proving themselves obsolete towards the challenges faced by network operators. Now, the services offered by the network are extremely tied to the physical network infrastructure. Operators often need to configure devices individually and manually causing operational inefficiency. The improvement process in the network is slow and very expensive to providers and operators, mainly, because the traditional network paradigms are inflexible, static and, thus, closed to innovation. The current network architecture was not made to be evolvable or to deliver new services easily. [5][13]

We believe that today networks do not offer enough tools and information to allow us to provide, for example, a pay-as-you-go billing strategy to service providers. It is hard both: to account and to control the network. A degraded user experience and inefficient utilization of network resources are the consequences of both: the lack of real-time visibility of the network state; and the lack of controlling capabilities between applications and the network. [5][13]

Service Providers urgently need to find ways to drive value out of their network by making their services more relevant to application developers, and their users and peers. They want to be able to, quickly and easily, create and launch new services. While the network infrastructure remains rigid and hard to be modified, digital services developers are taking full benefit out of highly flexible technologies to innovate rapidly. Operators must obtain more value out of the network infrastructure investment and match the pace innovation in networks to innovation speed of digital services. Therefore, they must improve current classes of service (SLAs) to create new sources of revenue. For example, one possible way to do that is to offer a faster implementation and easier management than what is being practiced in the industry now. [5][13]

This modern, and dynamic, scenario is presenting new requisites of scalability, performance and user experience to providers. The network providers' ability to respond efficiently to these requests is inadequate, due to the inflexibility of current network design patterns.

Two main problems: the lack of traffic demand visibility and the lack of feedback between applications and network are pushing operators to either, over-provision the network or resign to a best-effort service delivery. These specificities affect both: the operators' ability to add differentiated value to the network and the ability to obtain revenue prevenient from advanced and customized SLA-based services. [5][13]

On the other side, providers are expected to improve their services with intelligent information from the network and be able to provide new services in a fast manner using simple management tools in order to reduce the costs and time of these operations. Finally, the main issues can be summed to increasing the capacity of creating and extracting value out of the network, as well as reducing operational costs. [13]

2.3 SOFTWARE DEFINED NETWORKING

As stated before, the speed of innovation in Communication Networks does not match the speed of innovation in digital services and intelligent systems [5]. The software-defined networking paradigm is a possible solution to solve these issues by centralizing the network intelligence and abstracting the network controlling functions to a central management point, as defended by the Open Networking Foundation.

There are, essentially, four approaches to software defined networking: a traditional SDN architecture, proposed by the Open Networking Foundation; a hybrid architecture i.e. not fully software-defined architecture; a programmability-based architecture; and an overlay based architecture. Virtual overlays will not be described in this work, because it is mainly applied to network virtualization scenarios and, hence, is not correlated to the scope of this work. For information on the subject refer to [17]. Fig 2.5 illustrates the different architectures. In this section, we give further details on three different approaches to SDN.

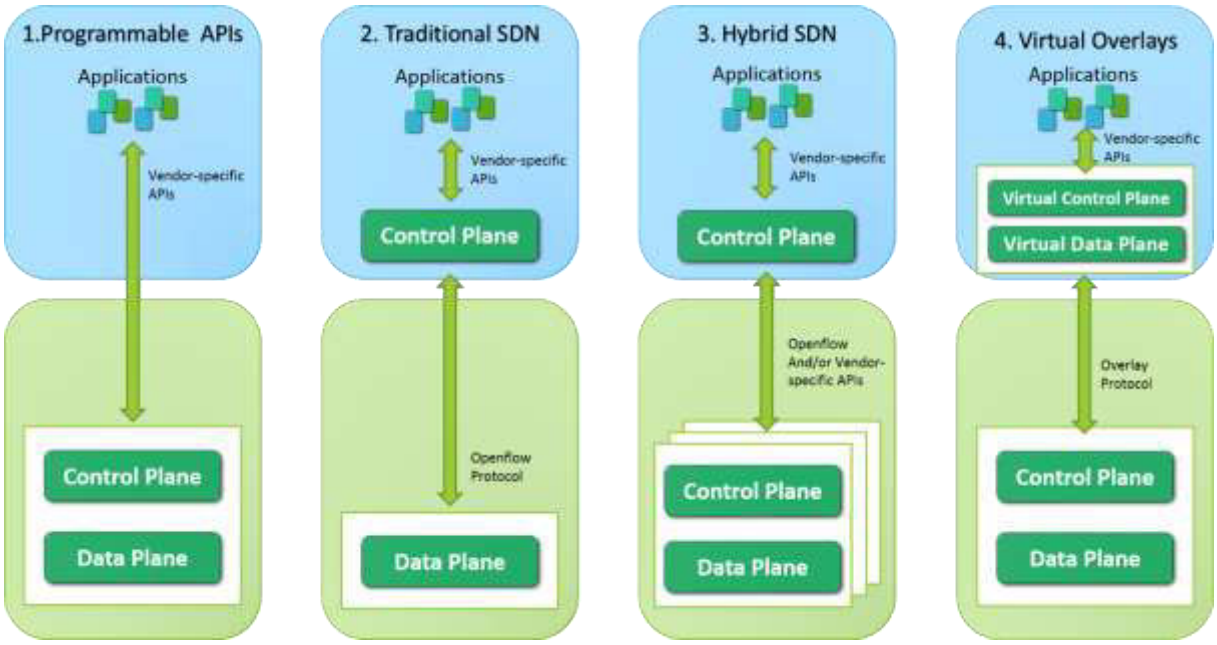


Figure 2. 5: Different approaches for Network Programmability

2.3.1 OPEN NETWORKING FOUNDATION SDN ARCHITECTURE

The Open Networking Foundation (ONF) defines *SDN* in the following way:

"In the SDN architecture, the control and data planes are decoupled, network intelligence and state are logically centralized, and the underlying network infrastructure is abstracted from the applications. As a result, enterprises and carriers gain unprecedented programmability, automation, and network control, enabling them to build highly scalable, flexible networks that readily adapt to changing business needs."[6]

The ONF defends the total centralization of network intelligence in a control management layer, decoupled of the network, with a centralized view of the network. Figure 2.6 contrasts the differences between a traditional network architecture and a SDN architecture with a centralized control layer extracted from the network. Additionally, to the ONF, SDN means the total separation between the data forwarding plane and the routing control plane as illustrated on Fig. 2.6.

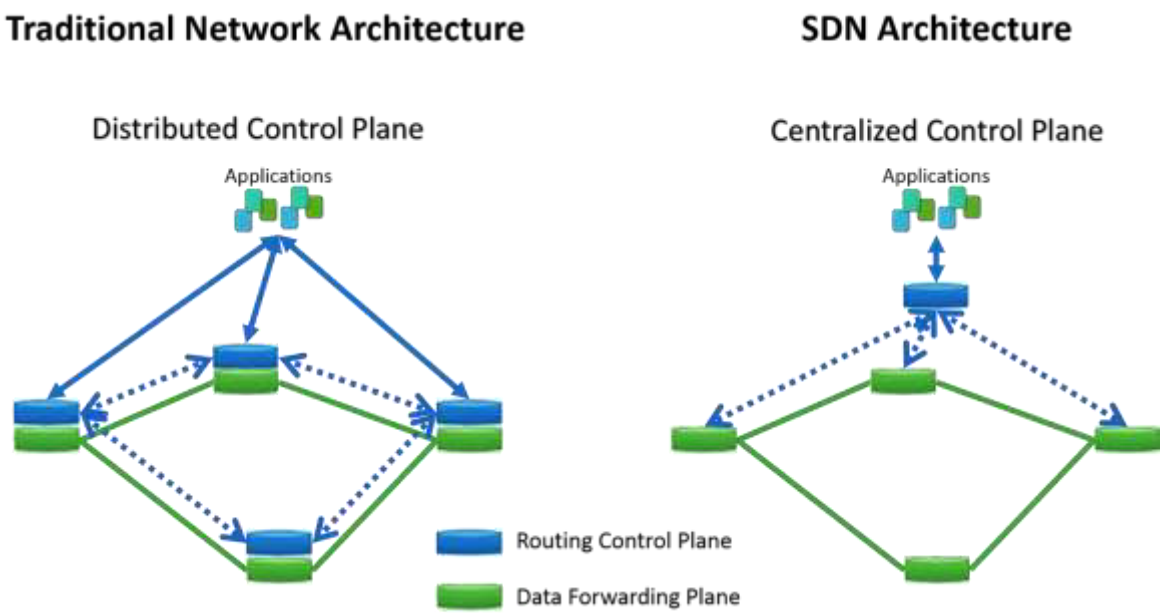


Figure 2. 6: Traditional Network Architecture vs Software Defined Architecture

A Southbound interface is defined as the interface that permits the control layer to control and communicate with the network. The ONF supports OpenFlow as the main southbound interface to a software-defined network. They also define an application layer that provides different network function which will be, later, a great tool for innovation in the networking industry. In addition, a northbound interface is defined as the interface that allows controlling and communication between the control layer and the application layer. [6]

Figure 2.7 illustrates the open networking foundation's propose of SDN architecture by means of the OpenFlow protocol as southbound interface, custom APIs as northbound interfaces to business applications that can control and manage the network.

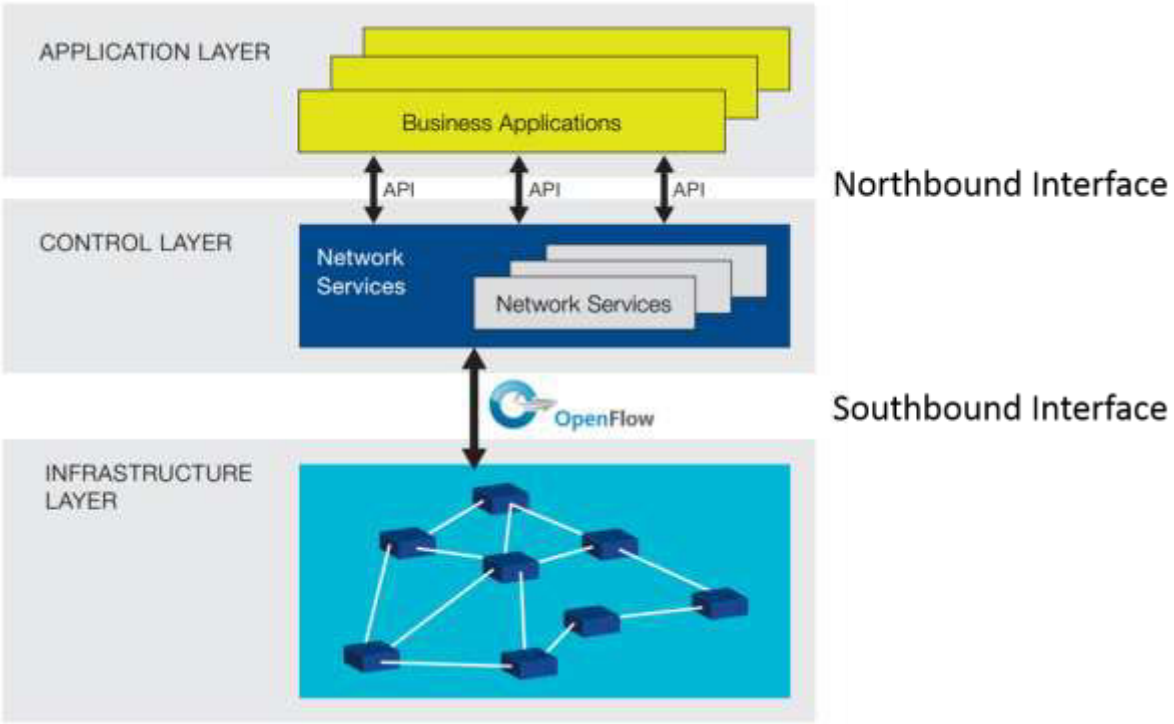


Figure 2. 7: SDN framework adapted from the Open Networking Foundation [6]

An OpenFlow-based SDN architecture is defined by the use of a logically centralized point of control, an OpenFlow network controller, to enable control of network behavior by means of OpenFlow enabled network devices. This behavior is dictated by the network services implemented in the controller. The OpenFlow main characteristics will be briefly described in the next section. [6][13]

The major point of this strategy relies on the extraction of network device complexity and centralization of this complexity in a, potentially virtualized, single control point. Note that decreasing the network device complexity implies in decreasing the costs of network devices, as well as it increases the flexibility of the network controlling and management capabilities. In this way, a network operator could change the function of a device from border router to a firewall with a simple installation or configuration of an application in the network controller. [6][13]

The weakness of this strategy is its revolutionary characteristic. This strategy would require heavy investments in change of equipment. It is necessary to take in consideration the fact that changing the architecture in a drastic way would affect, not only, the network operations as well as billing and monetization systems. It is reasonable to believe that network operators will avoid: overnight investments, the replacement of network infrastructure, the redesign of network architecture and possibly avoid the perturbation of operational standards. Taking this into consideration, the necessary transition effort from the traditional network model to the ideal SDN model proposed by ONF stills very costly. Therefore, the first software defined networks will probably abstract the network as an overlay or set of APIs above the existing network in a partial way, in other words, the network will not be totally transformed from day to night. [5]

2.3.2 SDN – NETWORK PROGRAMMABILITY

Some network vendors defend a different SDN approach. They defend a SDN approach based on direct network programmability by means of an API. An API-based implementation and controller-based SDN implementation follow the same principles. Both consist in providing a northbound open API for a superior application layer that has the possibility of controlling and extracting information out of the network. [17]

The main difference between them is that a controller-based architecture usually presents one extra level of abstraction, and thus, higher layer applications can be developed independently of the southbound interfaces implemented by the controller and the network. APIs that interfere in the network directly can have different innovation possibilities whereas a controller-based SDN solution presents higher flexibility of innovation possibilities. The controller can communicate and control the network in several different ways, therefore abstracting network complexity, and providing standardized APIs to the application layer. [5]

Software-defined networking basically means the extraction of network functions from its physical infrastructure. It is important to highlight that this point of view is slightly different from the ONF definition. Changing the physical network, today, is hard, slow and it often requires manual intervention in a fragmented way, whereas in a software-defined network this would be flexible and almost imperceptible. Software-defined networking will permit the management of the network as a single entity, orchestrating provisioning and providing a real-time visualization of the network performance and availability. [5]

This point of view, compared to the original one, proposed by the Open Networking Foundation, is much more flexible and much more realistic regarding the solutions proposed and implemented in the networking industry, including several types of possible approaches to the deployment of Software Defined Networks happening in the real market.

2.3.3 HYBRID SOFTWARE DEFINED NETWORKING ARCHITECTURE

Another SDN approach defended by some vendors is a hybrid software-defined architecture where devices still can detain certain part of network intelligence. In other words, it is defended that the total decoupling of control plane and data plane is not strictly necessary.

The main motivation for that way of thinking is the proposed extraction of well-defined and consolidated network functions, consequently, necessity of a certain amount of work to redesign the implementation of those functions in a new design, pure SDN architecture.

The best transition strategy to a SDN environment is the incremental implementation of justified new network functions using SDN solutions, for example: network optimization for video delivery, or customized routing optimization. Operators will gradually deploy programmability interfaces in their networks and, hence, the capacity to deliver faster innovation to their customers.

As a new network management layer arises and becomes the single network touch point, it will be easier and faster to implement and provide new services, as well as the reconfiguration of new environments will become easier. The SDN management layer will also provide a real time representation of the network. [5]

2.3.4 SDN CONSIDERATIONS

According to the ONF, SDN can permit the fast implementation and development of new and intelligent network services by means of [6]:

- An abstracted view of network functions and network capacities
- Standardized programmability interfaces to enable innovation
- Network statistics to create new services or enhance existing applications
- A single contact point with the network, optimizing network management and accelerating the implementation of networked applications.

Caroline Chappell brings a SDN concept aligned with the original SDN concept proposed by the ONF, and synchronized with the market perspective [5]. This author defends software defined networking as an evolution of networks rather than a revolution. Moreover, the author states that all approaches are complementary, and operators can use one or all of them, depending on their level of investment, their business priorities, and their operational priorities. Again, the author also defends that SDN need to be implemented gradually, incrementally, to support new services.

2.4 SERVICE PROVIDER INNOVATION WITH SDN

According to the ONF, The abstraction of network functions out of the physical infrastructure and network intelligence centralization provided by SDN can transform the architectural network paradigm of wide area networks, permitting the development of flexible WAN solutions ready to adapt to the changing business environment. [13]

The challenging situation faced by providers now is very favorable to the emergence of SDN solutions. To extract value of their key active, the network, operators need to drive a transformation in the market trends. Moving from a commoditized communication services model to a model of service differentiation and customized plans to each subscriber. The network intelligence can be used to deliver innovation, creating new types of service in an agile manner and creating innovative business models. [13]

New possibilities for service monetization can be implemented by the direct link between the application layer and the network controller. For example, it is now possible to automate the resource allocation optimally and dynamically from specific application demands. This capability is necessary to allow service delivery with measurable and differentiated Quality of Services. The SDN framework can decrease operational costs and fasten service activation and, hence, enrich existing services. [13]

Four possible strategies to new service creation and therefore new network monetization strategies are exemplified by the open networking foundation: [13]

- Bandwidth on demand
- Bandwidth Exchange
- Pay for QoS

- Network features for pay

In the next sections, we explain the strategies of Bandwidth on Demand, e pay for QoS. The strategies of bandwidth exchange and network features for pay are slightly more complex than the others and are not included in the scope of this work. Nevertheless, they are very interesting possibilities for innovation in service providers.

2.4.1 BANDWIDTH ON DEMAND

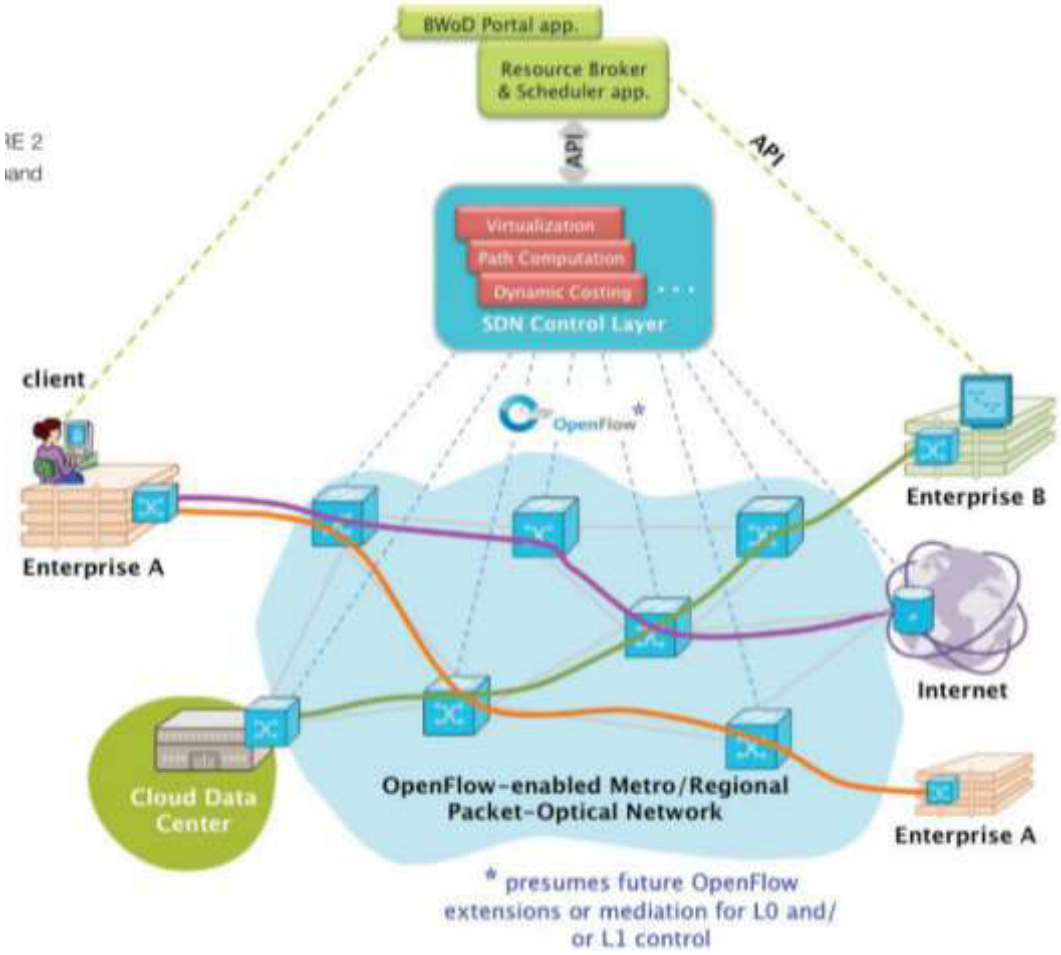


Figure 2. 8: Bandwidth on Demand Architecture [13]

The networking traffic patterns are changing very fast. The creation of new technologies such as big data and cloud networking are creating extremely high and short demand peaks. In this scenario, the purchase of a committed information rate is either very expensive or not effective enough. A service Bandwidth on Demand, with the capability to resize or reactivate network connectivity dynamically is far more interesting to the companies. This solution would reduce costs to these costumers because they could pay only for what they consumed, in terms of network bandwidth and traffic. [13]

Some providers already offer bandwidth on demand services. According to the ONF, the main difficulties faced by these providers are: [13]

- The lack of service provisioning automation capabilities
- The lack of a standardized control interface for operators. These capabilities are implemented by means of vendor-specific solutions.

A SDN-based service provisioning strategy can permit automatic network control. A SDN-based service provisioning strategy can also provide a real-time network view. These two features together can permit the intelligent and automatic network resource allocation to satisfy different SLA requisites on a per-user or per-flow basis.

2.4.2 PAY FOR QOS

The network traffic offered to network providers is rising. The change in data traffic patterns driven by events, such as the intensification of video content consumption and the proliferation of mobile devices creates a challenge of higher traffic unpredictability. [13]

In this situation of massive traffic growth, operators have two options: they either make extra investments on over provisioning the network infrastructure to bare the worst-case scenario or resign to a best-effort connectivity service and, thus, user-experience degradation. If the investments are not sustainable or profitable, then network providers will chose towards service degradation. [13]

Now, considering a service degradation scenario, it is reasonable to think that some costumers or content providers who expect higher Quality of Experience (QoE) standards will be willing to pay more for extra value from the network.

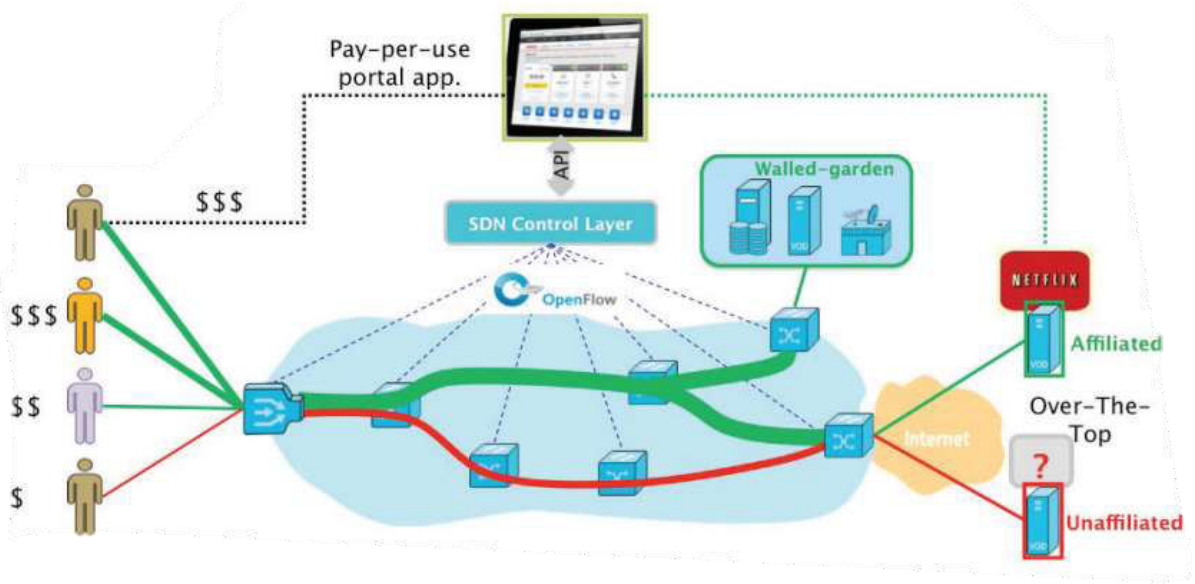


Figure 2. 9: Differentiated and Monetizable services [13]

SDN permits the transform of the network from a bit transportation commodity to a valuable resource based on network intelligence. This differentiation, basically, consists of flow prioritization, and can easily be implemented with the deployment of OpenFlow switches on the network borders. In this way, “premium” or special flows could be sent through optimized paths, and standard or unknown flows could be forwarded through a best-effort path. [13]

In the long run, this strategy permits operators to extract value and generate the necessary revenue to support incremental infrastructure improvement, and thus traffic growth, in a more profitable and sustainable manner.

3 ARCHITECTURAL MODEL

This chapter describes the architectural model used to study the experimentation of a software-defined network solution to offer service provisioning automation in service providers. We briefly detail the main components and auxiliary components as well.

3.1 OVERVIEW

In this work, we present a study case to evidence the applicability of software-defined networking solutions to solve real challenges of service providers, specifically, service provisioning automation. We propose a useful network model based on SDN that can fasten service activation and ease the delivery of customized services, therefore, allowing the creation and utilization of new SLA standards. To do that we apply an OpenFlow-based SDN architecture, as described in Fig. 3.1, to control the network and reinforce network policies of bandwidth utilization and data volume constraints.

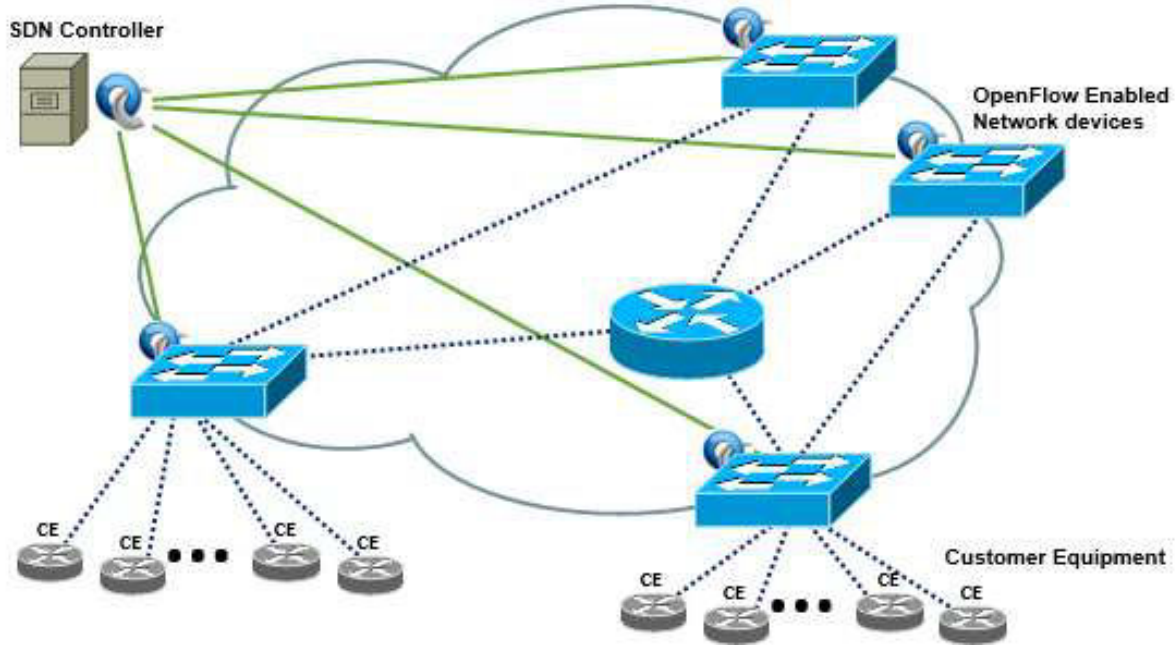


Figure 3. 1: Illustration of an OpenFlow-based SDN architecture for Service Providers

The network behavior is controlled by the implemented network applications in the controller: service activation, speed control and data volume monitoring. External network policies are reinforced according to an external database by means of the network applications. Figure 3.2 illustrates the architecture.

Additionally, the proposed architecture is only applied to border switches, decreasing deployment costs and permitting an incremental execution of its strategies. This approach can still reach the purposes of automatic service provisioning, and network policies reinforcement.

The system works basically as follow: every time a packet arrives in a switch, the switch checks for a corresponding entry in its flow-table. If that packet has no match, thus “unknown”, then the switch forwards that packet towards the controller. The controller identifies the user based on its IP address and its physical location then confirms service authorization for that user. If confirmed, then network traffic for that user will be selectively permitted and limited according to the network policies, using the meter OpenFlow 1.3 functionality to limit the rate for that user. At the same time, the controller repeatedly requests all the network devices for metering statistics and then updates its usage statistics. Next, according to these updates, the controller will decide about service deactivation, reactivation or service

upgrading. If that user is not allowed in the network according to the network policies database, then its flows will be instructed to drop packets as the correspondent action.

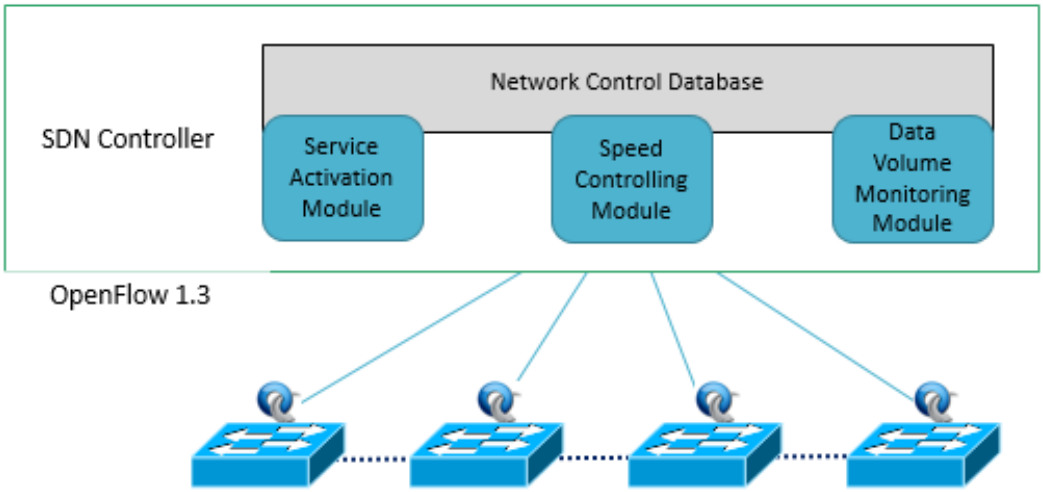


Figure 3. 2: Illustration of system architecture

3.2 OPENFLOW SDN ARCHITECTURE

OpenFlow is a communication protocol that allows a controller to coordinate and customize the forwarding plane of network devices. It does that by allowing the modification of the flow tables in an OpenFlow enabled switch. OpenFlow is the standard Southbound Interface according to the Open Networking Foundation, as illustrated in Fig. 2.6.

The OpenFlow Specification version 1.3.3 was released on December, 18th, 2013 by the Open Networking Foundation, while its 1.3.4 version is on ratification at the time of this work [8]. Several vendors are already selling OpenFlow 1.3 enabled switches, including HP, Centec, Edge-core and Pica8 [9] [10] [11] [12]. This incredibly fast acceptance by the market of the protocol shows its popularity as a tool for innovation in the network area. The companies supporting it, are doing it mainly because the proposed "revolution" in the network industry might be strongly beneficial to them.

3.2.1 OPENFLOW ENABLED SWITCH

An OpenFlow switch behaves as described in Fig 3.3. For every flow, if a matching entry is already defined in any of its flow-tables, then the switch applies the corresponding actions. If a flow has no matching entry in any of the switch's tables, then, the switch will search for an action on the table-miss entry. If no is action defined for that entry, then the packet is dropped. [8]

Commonly, the table-miss entry is used to redirect new packets to the controller. The controller, then, resolve the forwarding action of the switch, from a holistic view of the network.

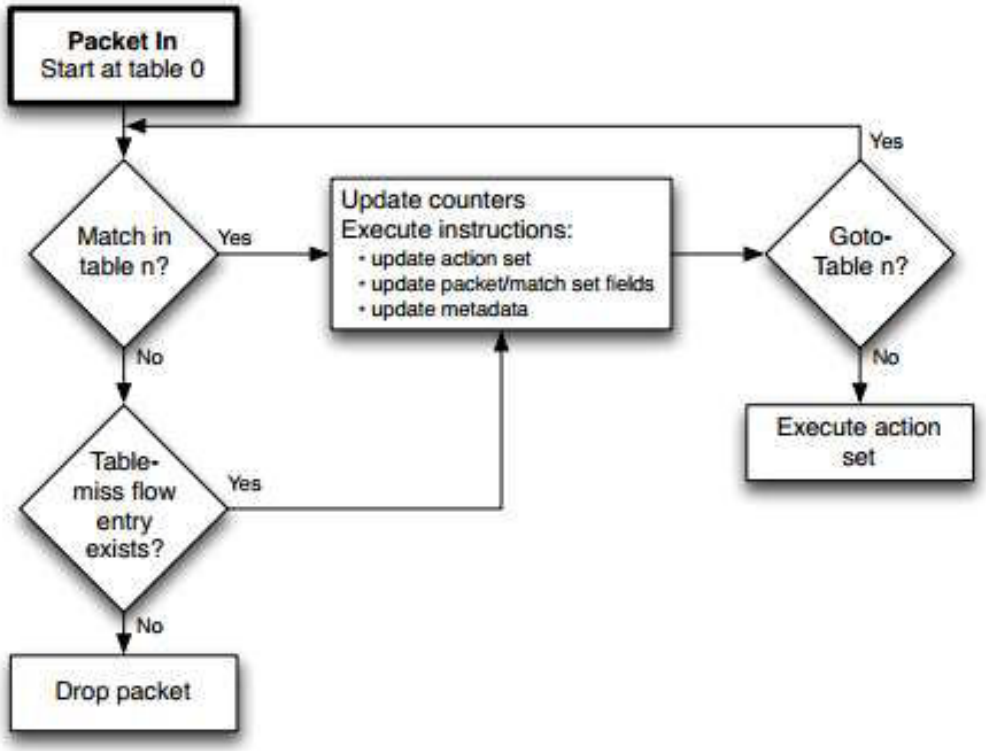
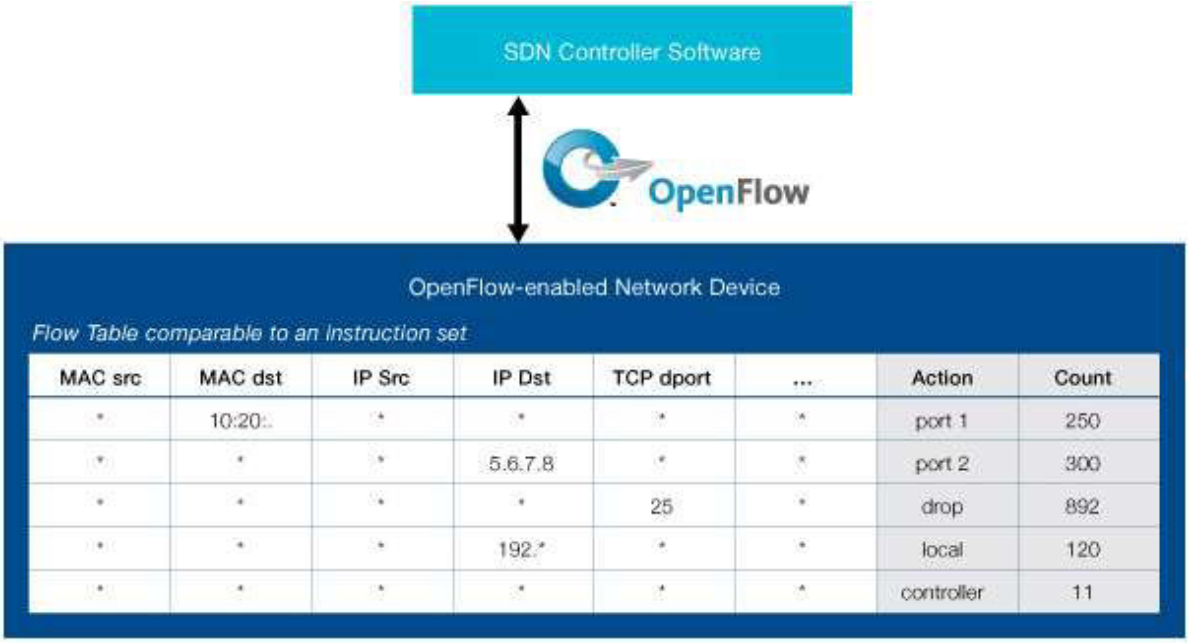


Figure 3. 3: Flowchart detailing packet flow through an OpenFlow switch [8].

A flow is described by its match fields, which includes, at least, 12 matching fields according to the OpenFlow 1.3 specification. The match fields are exemplified in Fig. 3.4 and include, but are not limited to, IP addresses, TCP and UDP ports, Ethernet addresses and type. Further information can be found in the OpenFlow specification itself. [8]



MAC src	MAC dst	IP Src	IP Dst	TCP dport	...	Action	Count
*	10:20:..	*	*	*	*	port 1	250
*	*	*	5.6.7.8	*	*	port 2	300
*	*	*	*	25	*	drop	892
*	*	*	192.*	*	*	local	120
*	*	*	*	*	*	controller	11

Figure 3. 4: Example of Flow table and instruction set [6].

The last entry of the table in Fig. 3.4 typifies the table-miss entry.

3.2.3 OPENFLOW CONTROLLER

An OpenFlow Controller can interact with the OpenFlow switches on several forms. The controller can add, modify or delete flows according to the received packets. The controller can also add modify or delete meters for a switch. It is capable of sending custom packets through its OpenFlow channel to the network. An OpenFlow switch will not stop working when the OpenFlow connection between controller and switch is disabled. This feature is important because it increases the resilience of the network. [6]

3.2.4 METER

According to the OpenFlow 1.3 specification published by the Open Networking Foundation, a meter is a switch element that can reassure and control the rate of packets. The meter triggers a meter band if the byte rate (or packet rate) passing through the meter exceeds a predefined threshold. The meter bad can eventually drop a packet, in that case, it is denominated a Rate Limiter. [8]

The meter table existent in OpenFlow 1.3 devices permits a per-flow granularity and, thus, allow the implementation of several QoS operation from rate-limiting to DiffServ schemes. Any flow entry can specify a meter in its instruction set. The meter measures and controls the rate of the aggregation of all flow entries to which it is attached. [8]

3.3 NETWORK APPLICATIONS

The three network applications implemented in the controller provide an interface to external conditions through the Network Control Database, as illustrated in Fig. 3.2. This database basically defines the permitted users, their respective bandwidth limits, and their data volume constraints. Additionally, the controller, maintains a real-time representation of the network state in its “memory”. The behavior of the three modules are based on the data structures described in Fig. 3.5.

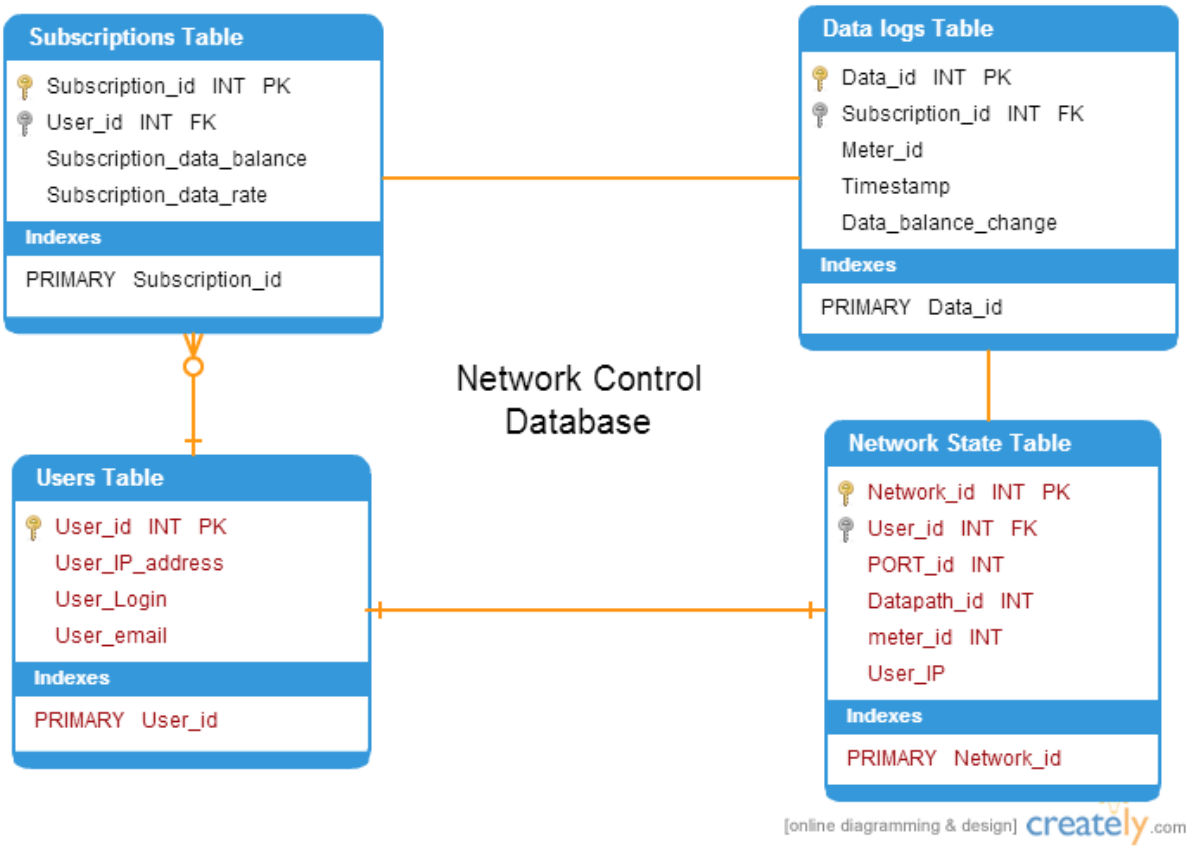


Figure 3. 5: Network Control Database

3.3.1 SERVICE ACTIVATION MODULE

The service activation module is connected to the subscriptions database. The module works in two ways: in a responsive manner and in a proactive mode. Responsively, when an unfamiliar packet arrives in the network it is sent to the controller to request for instructions. The service activation module identifies the user according to its IP address and network location and then validates either: if that user should be provided with network access or not. If positive, then it computes the forwarding action corresponding to that specific network device and sends an "add flow" message through its OpenFlow Interface.

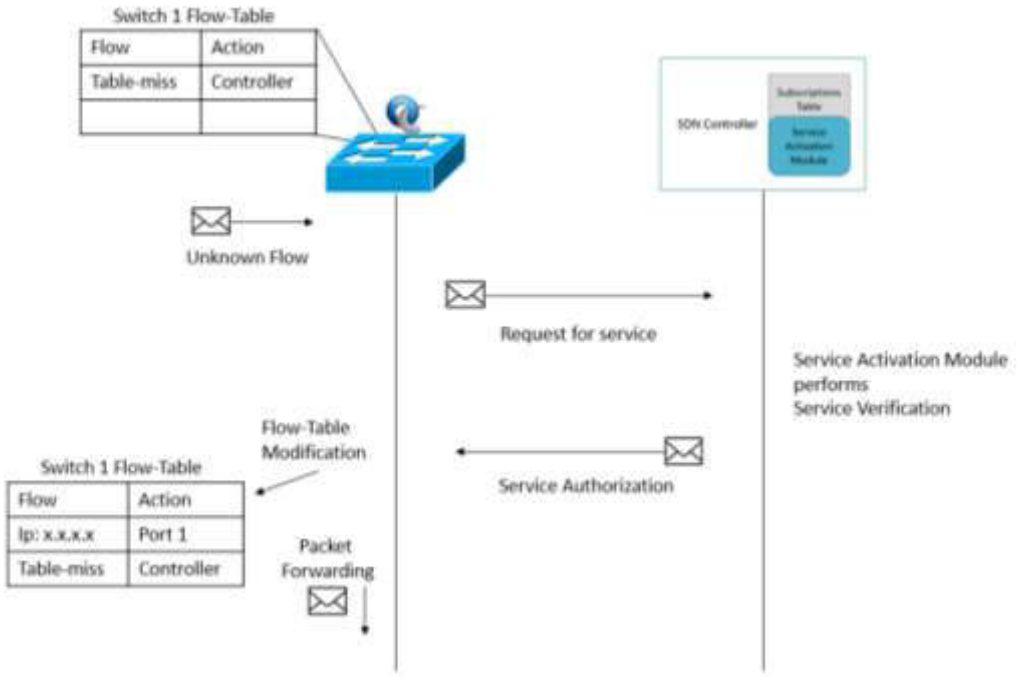


Figure 3. 6: Service Activation illustration

In negative case, instead of adding a forwarding action, the controller sends an instruction to drop the packets for that flow.

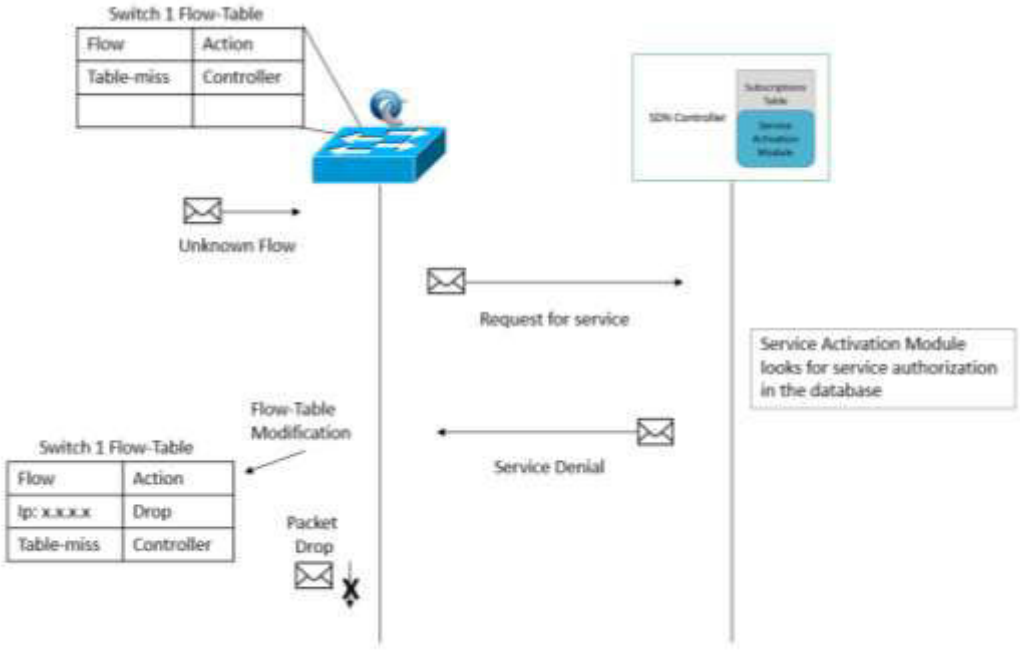


Figure 3. 7: Service Denial illustration

Proactively, the application persistently checks if the state of each subscription in the subscription database has changed in order to identify the necessity to reactivate services for users with deactivated service.

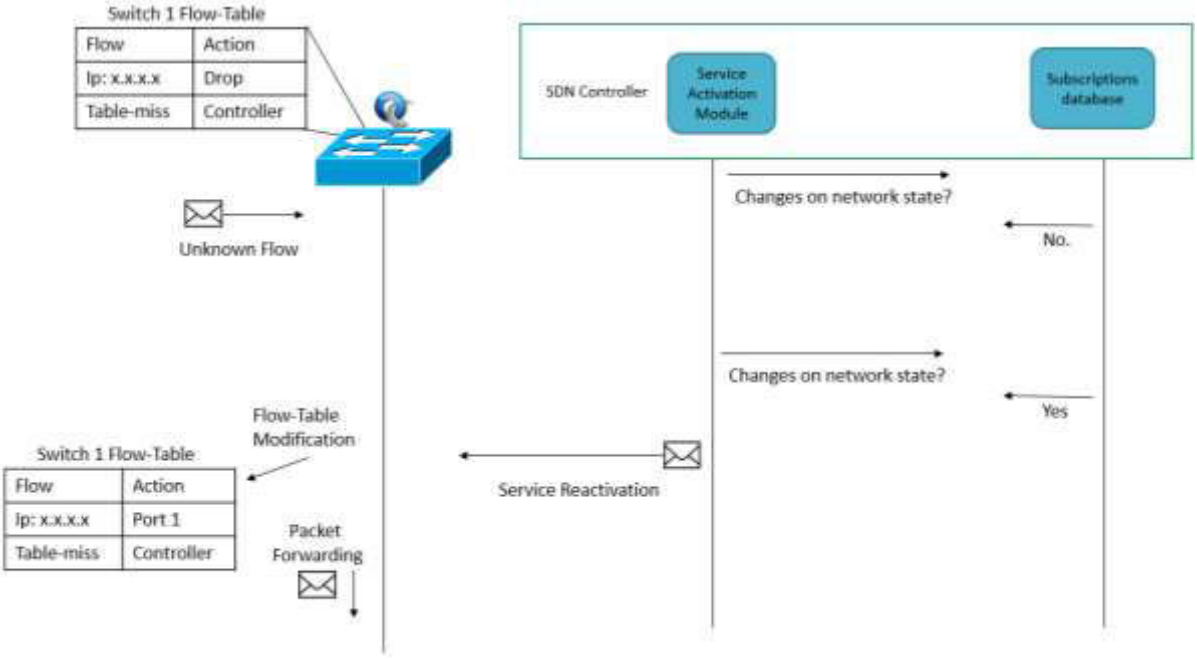


Figure 3. 8: Service Reactivation illustration

3.2.2 SPEED CONTROL MODULE

The speed control module is also dependent on the subscription table and relies on the rate limiting capability provided by the OpenFlow 1.3 Protocol. OpenFlow 1.3 allows a per-flow based metering strategy. It is capable of dropping packets based on the rate that the packets are arriving to that meter.

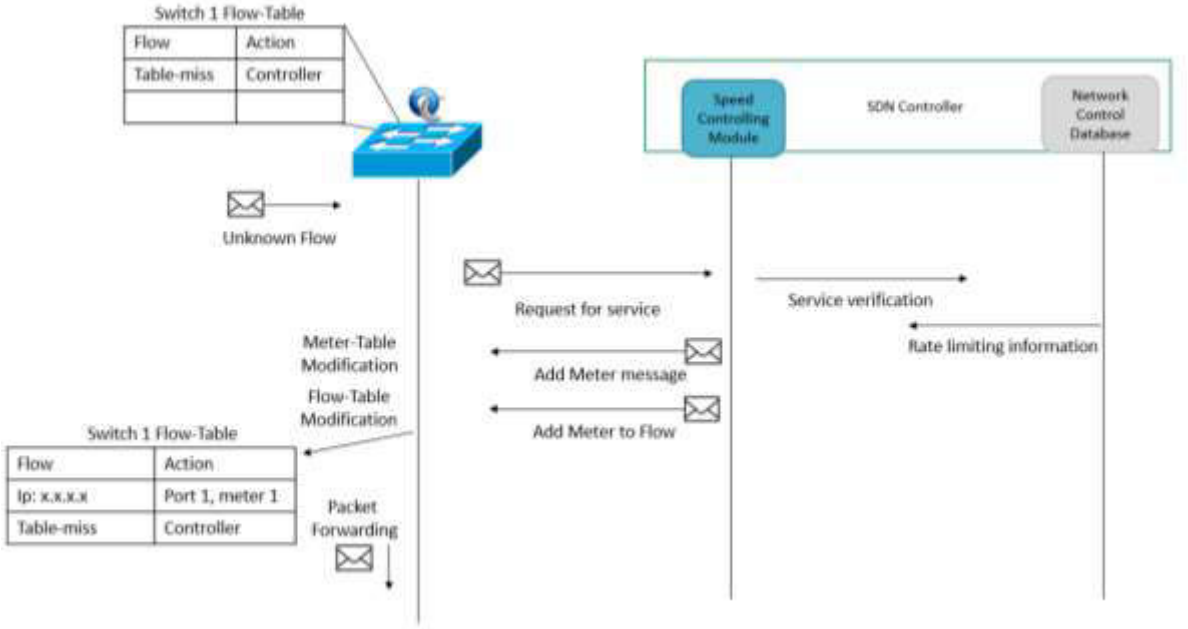


Figure 3. 9: Speed Controlling Module operation

The main challenge of this task is to coordinate the creation of multiple meters according to the network control database. The presence of multiple subscription entries for a specific user also have to be managed by the module.

3.2.3 DATA VOLUME MONITORING MODULE

The data volume monitoring module keeps track of data usage on a per-user basis. According to the network state table, the controller query all switches for the meter statistics related to each of its active users. After receiving that information, the controller updates its statistics table. Later, if a user has a negative balance the service provisioning module will deactivate the service for that user.

3.3 OTHER COMPONENTS RELATED

In this section, the supplementary components necessary to the deployment of the test bed for the study case will be described. The SDN controller used in the work was a RYU controllers. The OpenFlow switches is the Openflow 1.3 software switch provided by CPqD. [15] And the network was emulated on Mininet, version 2.1. We also had to use a few performance testing tools to verify the right behavior of the network.

Host	Ip address
h1	10.0.0.1
h2	10.0.0.2
h3	10.0.0.3
h4	10.0.0.4

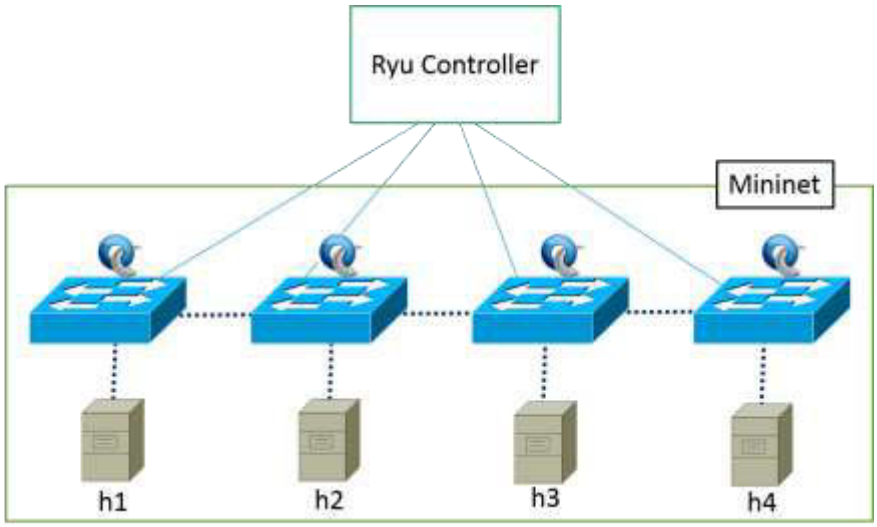


Figure 3. 10: Linear topology with 4 hosts on Mininet

3.3.1 RYU CONTROLLER

The RYU controller was chosen because it is a very lightweight controller written in python. It is relatively well documented and its learning curve is very comfortable, since python is a very straightforward and easy programming language. The RYU Controller fully supports OpenFlow 1.3. Further information on the RYU Controller can be found at [14].

3.3.2 OPENFLOW 1.3 SOFTWARE SWITCH

The CPqD reference switch has been used by the ONF as the standard software implementation of the OpenFlow 1.3 switch and it is a virtual switch. In our case the software was chosen because it was the only virtual switch implementing OF 1.3 metering capabilities at the time of the development of this work. It can be easily integrated to Mininet. Further information on the OpenFlow Switch can be found at [15].

3.3.3 MININET

Mininet is a relatively recent and increasingly popular tool for network emulation. It is very lightweight and mostly written in python, what provides a very fast learning curve. It is very easy to use and customize according to your needs.

Mininet emulates hosts using Linux containers with isolated namespaces and network interfaces, it emulates switches using isolated Linux containers running the virtual switches software, and it emulates

links by using network interfaces. Mininet automatically configures the Openflow Communication path to each of the virtual switches.

By means of isolated namespaces, Mininet can easily permit the isolated execution of software on different emulated hosts as can be seen in Fig. 3.11 below. This feature was used to ease performance evaluation of the network.

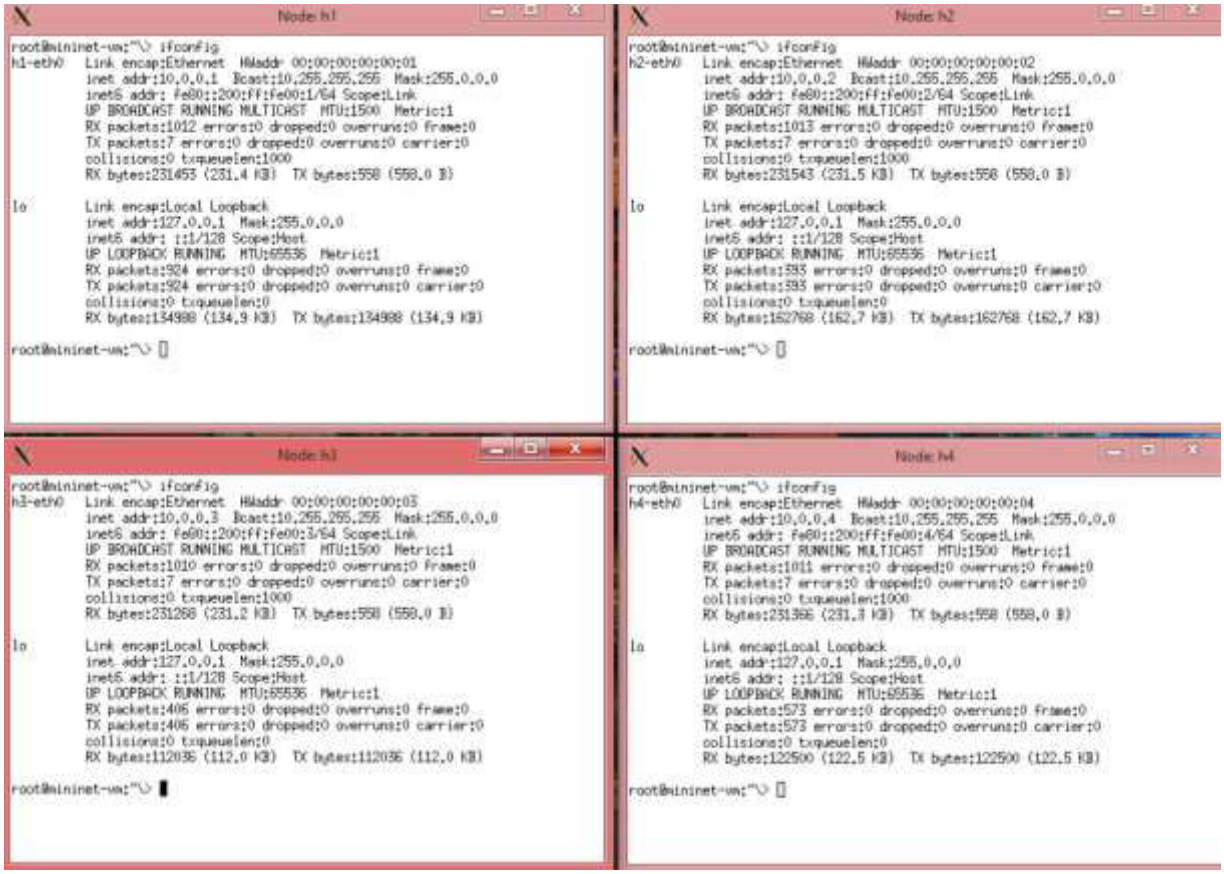


Figure 3. 11: Mininet hosts' terminals and its network interfaces.

3.4 NETWORK PERFOMANCE MEASUREMENT TOOLS

3.4.1 SMOKEPING

Smokeping is an Open Source Linux software to measure, monitor and analyze connectivity performance of a server. It has great visualization tools.

3.4.2 IPERF

IPerf is another Open Source Linux tool to measure, network performance. It is a very complete tool, it can measure TCP throughput, TCP RTT, UDP loss rate, UDP throughput, UDP delay, UDP Jitter. It is possible to create an arbitrary number of simultaneous flows at a configurable rate. It does not have great visualization tools.

4 STUDY CASE

In this chapter we describe the executed procedures and testing experiments to evidence the proposed concept of automatic service provisioning using a software-defined network.

4.1 OVERVIEW

To demonstrate the applicability of the proposed solution, we emulate a simple network on Mininet and connected it to a RYU controller with the developed network applications of service activation, speed control, and data volume monitoring. To demonstrate the effectiveness of the proposed solution, we summed the study case to 5 scenarios:

- START: Automatic service activation, constrained to rate limiting policies.
- STOP: Automatic service deactivation, constrained to data volume limits.
- RESTART: Automatic service reactivation.
- STOP AGAIN: Service deactivation
- UPGRADE: Automatic service reactivation, according to new policies.

4.1.1 INITIAL SETUP

For our study case we run RYU and Mininet on two virtualized servers in a private data-center to eliminate possible bottlenecks of using a limited personal computer.

The initial state of the subscriptions database is described in Table 4.1.

Table 4.1: Initial state of subscriptions database

Subscription id	User IP address	Data volume limit (KB)	Rate limit (Kbps)
1	10.0.0.1	10000	2000
2	10.0.0.2	1000	3000
3	10.0.0.3	1000	4000
4	10.0.0.4	10000	4000

The simulated topology was a linear topology with 4 switches in chain and 1 host attached to each switch as shown in Fig. 14. Detailed procedures for this operations are detailed in this reference.

We setup and start Smokeping in host 1. Smokeping send 5 ping every 5 seconds to all hosts from host 1. It records its latency time and displays then in a few graphs with basic statistics as it will be seen in Fig. 4.1.

4.1.2 START PHASE

Before the start phase all network devices have empty flow tables and, hence, cannot forward traffic successfully. After initiating the controller, it will insert a table-miss entry on every switch's table. During, the start phase we ping all hosts, and record its first five ping responses including latency time. We expect that the controller will permit traffic of the permitted users according to the network subscriptions database. In this phase, it is important to register the response time of the network, which is the latency of the first ping. Later, we use this information to verify the operation of the service

4.1.3 STOP PHASE

After basic connectivity tests, it is necessary to verify the operation of the speed control module and the data volume monitoring volume. For that we generate network traffic from one host to another and monitor its network performance using the iperf tool: UDP traffic is generated at the rate of 5Mbps; its throughput, loss rate, and jitter are monitored on the server side using the iperf tool. This procedure is repeated a total of 10 times to guarantee some reliability to the measurement.

It is very important to measure the throughput to validate the correct operation of the speed control module and also measure its accuracy. It is also necessary to measure the volume of data transmitted in the network to examine the accuracy of the data volume monitoring module, and also verify its correct operation.

The expected behavior of the system is to interrupt service when the data volume transmitted goes beyond the permitted value stored in the database.

4.1.4 RESTART PHASE

It is necessary to evaluate the reactivation of the system after verifying the service deactivation feature of the network. For that, a valid subscription is reinserted in the subscription database. Then, to account the response time of the reactivation module, the timestamps of the deactivation system and the timestamp of the reactivation button are compared to each other. This experiment was also repeated 10 times, 5 times for host 1 and 5 times for host 2.

4.1.5 STOP AGAIN PHASE

Next, TCP traffic is generated in the system to measure the TCP throughput of the network. It is also important to measure the volume of data transmitted in the network to examine the accuracy of the data volume monitoring module. Again, the expected behavior of the system is to interrupt service when data volume transmitted goes beyond the permitted value. This experiment was repeated 5 times for host 1.

4.1.6 UPGRADE PHASE

Now, in order to evidence that the software defined network can be easily adapted to dynamic requisites, a new entry is inserted in the database with different rate limits. Again, TCP traffic is generated to analyze the throughput of the network. It is important to notice that the system behaves as dictated by the network control database and, therefore, we permit the network to be dynamically controlled by allowing the database to be controlled by external systems.

4.2 OUTCOMES

Here we present the data obtained from the experiments described in the section 4.1.

4.2.1 SUMMARY

In this subsection, we summarize the behavior of the system with the graph generated by Smokeping throughout the whole sequence of experiments. Figure 4.1 describes the labels for graphs described next in this section.

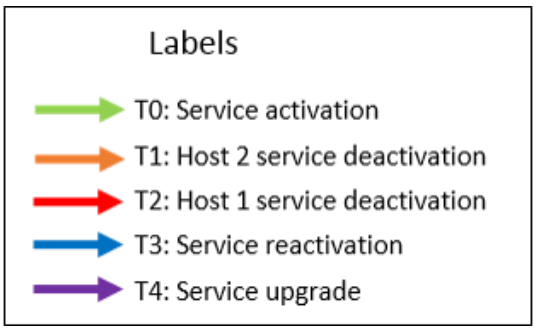


Figure 4. 1: Labels for the graphs described in section 4

Figure 4.1 displays the sequence of experiments and the behavior of the whole network. The blue lines represent connectivity times from host 1 to each of the hosts. In the Figures 4.2.a); 4.2.b); 4.2.c); 4.2.d), the connectivity time for hosts 1, 2, 3 and 4 are, respectively, represented. The blue rectangles shadowing the curve represent the variance of the round-trip time. The T0 arrow in green indicates service activation for all hosts. Notice that, after that time, the graph displays a blue line for all hosts, thus, all hosts are reachable.

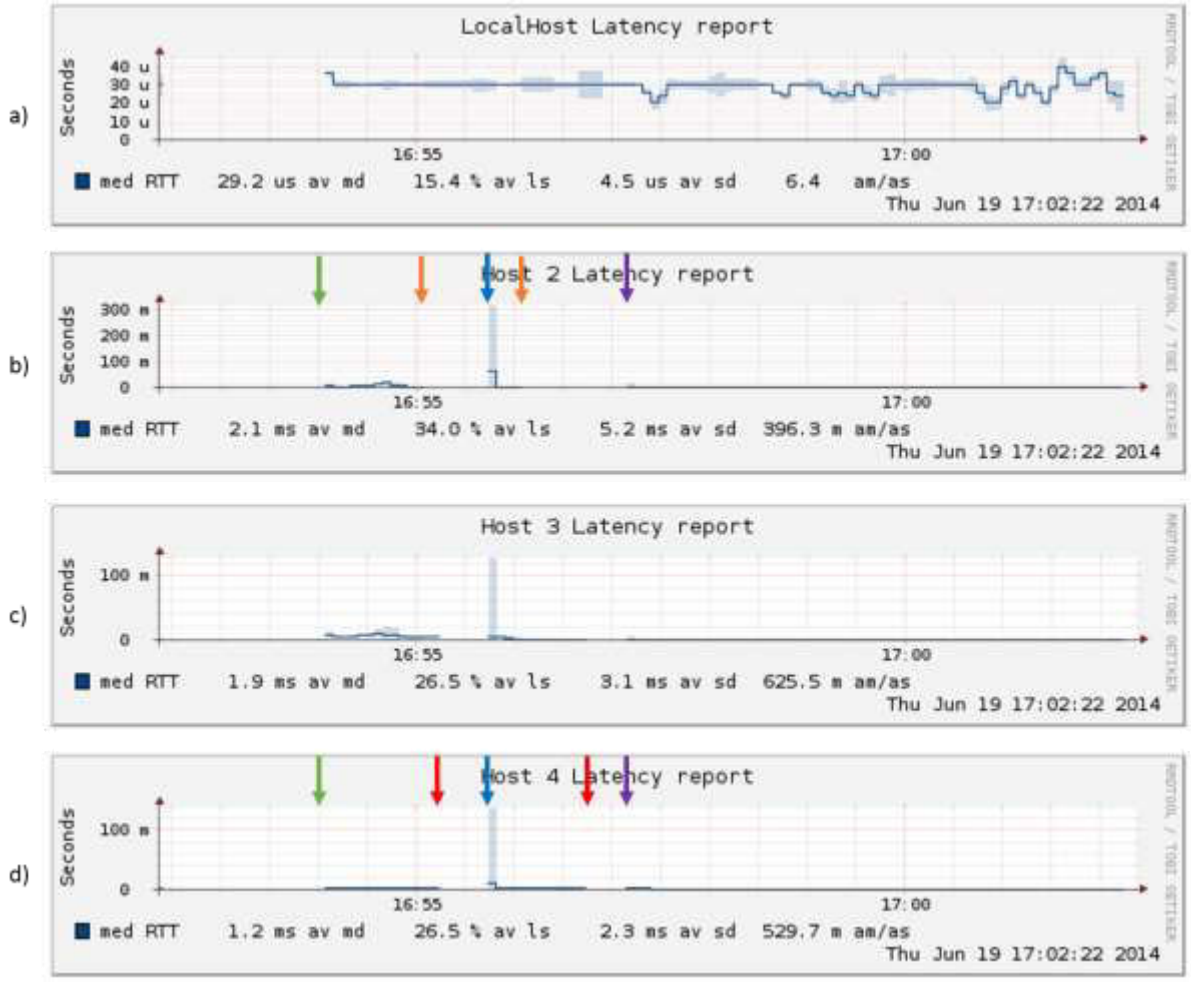


Figure 4. 2: Network connectivity throughout the sequence of experiments

The T1 arrows, in orange, indicate service deactivation times for host 2. The T2 arrows, in red, indicate service deactivation times for host 1. Observe that service is deactivated at different times for hosts 1 and 2; after that, the connectivity line (blue line) disappears.

The T3 arrow, in blue, indicates service reactivation. Notice that network connectivity is restored simultaneously for all hosts at time T3. The T4 arrows, in purple, indicate the upgrading time. Observe the overall behavior of the system: Service activation at T0, deactivation at different times T1

and T2, then reactivation at T3, another deactivation and finally reactivation due to service upgrading. Figures 4.3 and 4.3 illustrates the network connectivity for host 4 and 2, respectively, in further detail.

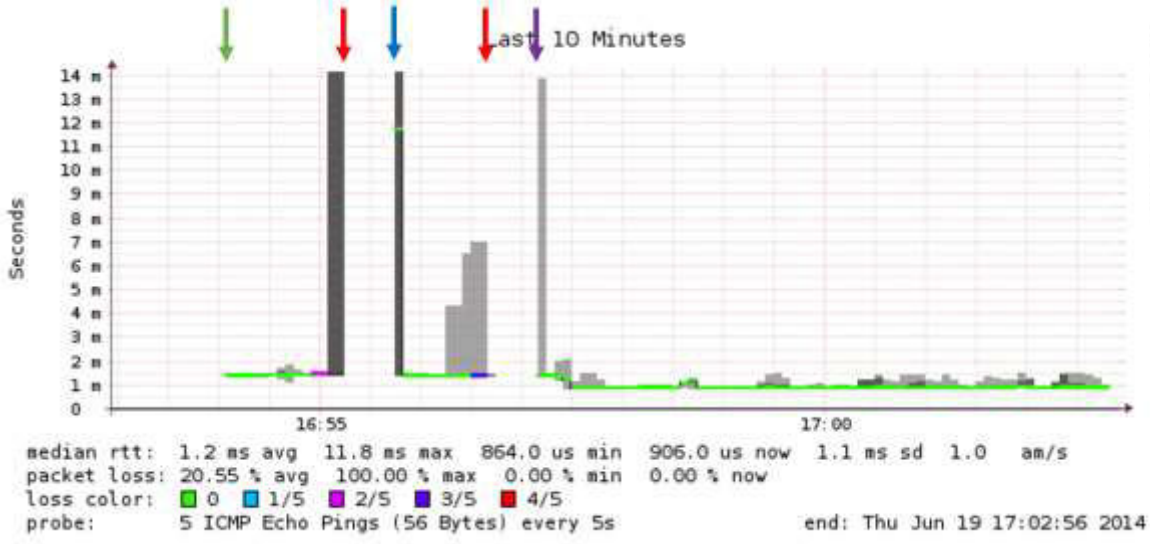


Figure 4. 3: Host 4 Latency report detailed

Observe that the variance of the delay for T4 is very higher that the standard ones in the graph. Variance is high during activation and deactivation times, as expected. Again, T3 indicates service activation, red arrows indicate deactivation; and the purple ones indicate the final reactivation and service upgrading.

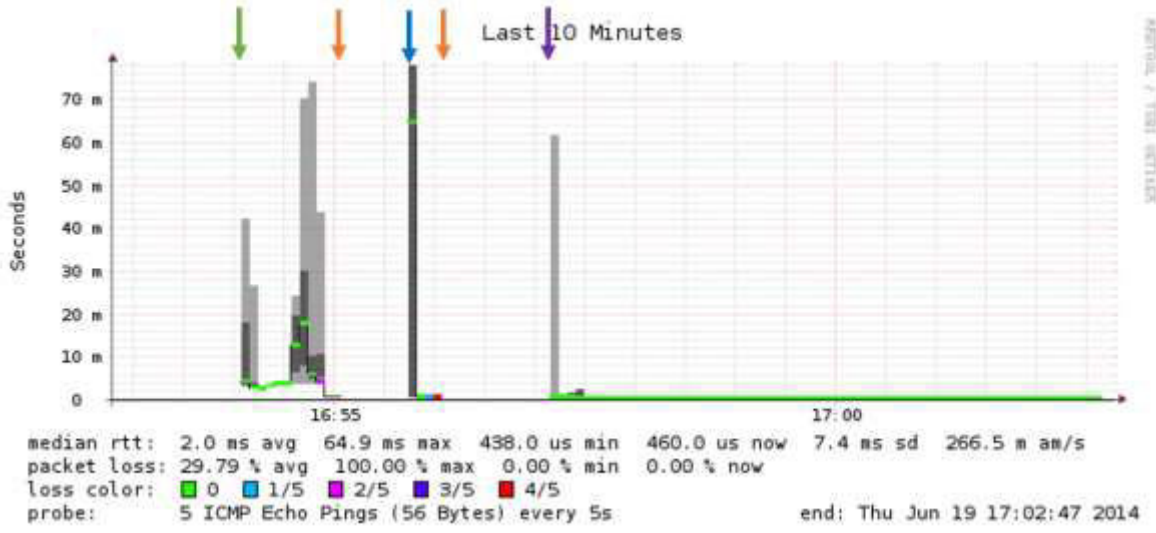


Figure 4. 4: Host 2 Latency report detailed

Notice that the graphs are alike. In this case, T3 represent reactivation, and T4 represent service upgrade, but now, T1 represents deactivation. Looking closely, it is possible to notice that the time difference between the last deactivation and reactivation are the different. Host 2 can be accessible for less time than host 4. This behavior is right as expected and it is a consequence of the data limits set in the initial database described in table 4.1

4.2.2 START PHASE OUTCOMES

In the start phase every host ping each other host to verify the response time for that link, then the system is restarted to repeat the procedure. Figure 4.6 shows one iteration of the procedure.

Note in Fig. 4.6 that the time for the first ping request is always much higher than the time for the subsequent ones. The initial delay happens because the switch does not have any entry on its table (for that flow) to decide which action to take. The so-far unknown user have to be verified by the controller. The network response time is the time of the first ping. As stated before, the experiment is repeated several times. The acquired data is displayed in Fig. 4.5.

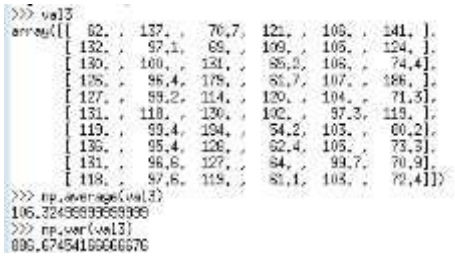


Figure 4. 5: Response Time distribution – 106 ms mean, variance 30 ms

Note that the maximum response time in 100 experiments was smaller than 0.2 seconds. This result is very satisfactory and exemplifies that a software defined networking solution can deliver, in real-time, service delivery dynamically.

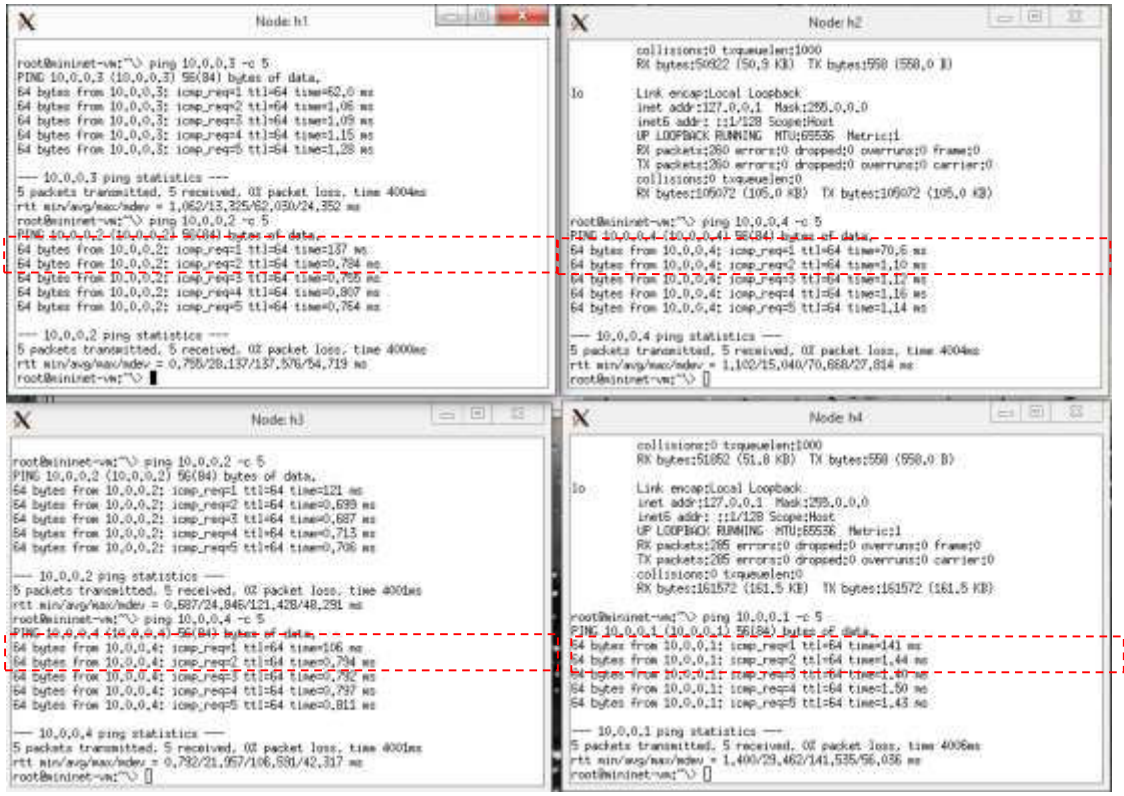


Figure 4. 6: Response time measurement screenshot

4.2.3 STOP PHASE OUTCOMES

During the stop phase, the data volume monitoring module and the speed control module are tested. For that, 5 flows of 1Mbps of traffic are generated in hosts 1 and 2 towards host 4 and 3 respectively. A total incoming traffic of 5 Mbps per host. This experiment was repeated a few times for each host. Figures 4.6 and 4.7 show the result of one experiment repetition for host 2 and 4, respectively.

a)

```
------------------------------------------------------------
Client connecting to 10.0.0.3, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  4] local 10.0.0.2 port 59477 connected with 10.0.0.3 port 5001
[  8] local 10.0.0.2 port 40126 connected with 10.0.0.3 port 5001
[  5] local 10.0.0.2 port 41763 connected with 10.0.0.3 port 5001
[  6] local 10.0.0.2 port 56037 connected with 10.0.0.3 port 5001
[  7] local 10.0.0.2 port 52998 connected with 10.0.0.3 port 5001
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0- 2.0 sec   247 KBytes  1000 Kbits/sec
[  4] Sent 172 datagrams
[  8]  0.0- 2.0 sec   247 KBytes  1000 Kbits/sec
[  8] Sent 172 datagrams
[  5]  0.0- 2.0 sec   247 KBytes  1000 Kbits/sec
[  5] Sent 172 datagrams
[  6]  0.0- 2.0 sec   247 KBytes  1000 Kbits/sec
[  6] Sent 172 datagrams
[  7]  0.0- 2.0 sec   247 KBytes  1000 Kbits/sec
[  7] Sent 172 datagrams
[SUM]  0.0- 2.0 sec  1.21 MBytes  5.00 Mbits/sec
[  4] Server Report:
[  4]  0.0- 2.0 sec   168 KBytes   680 Kbits/sec   0,303 ms   58/  172 (34%)
[  7] Server Report:
[  7]  0.0- 2.0 sec   162 KBytes   656 Kbits/sec   0,372 ms   59/  172 (34%)
[  5] Server Report:
[  5]  0.0- 2.0 sec   164 KBytes   663 Kbits/sec   0,204 ms   58/  172 (34%)
[  8] Server Report:
[  8]  0.0- 2.0 sec   162 KBytes   657 Kbits/sec   0,454 ms   59/  172 (34%)
[  6] Server Report:
[  6]  0.0- 2.0 sec   161 KBytes   648 Kbits/sec   0,727 ms   60/  172 (35%)
root@mininet-vm:~\> 
```

b)

```
root@mininet-vm:~\> iperf -s -u -i 0.5
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  4] local 10.0.0.3 port 5001 connected with 10.0.0.2 port 59477
[  5] local 10.0.0.3 port 5001 connected with 10.0.0.2 port 52998
[  6] local 10.0.0.3 port 5001 connected with 10.0.0.2 port 41763
[  7] local 10.0.0.3 port 5001 connected with 10.0.0.2 port 40126
[  8] local 10.0.0.3 port 5001 connected with 10.0.0.2 port 56037
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagr
[  4]  0.0- 0.5 sec  64.6 KBytes  1.06 Mbits/sec   0.287 ms    0/   42 (0%)
[  5]  0.0- 0.5 sec  60.3 KBytes   988 Kbits/sec   0.058 ms    0/   42 (0%)
[  6]  0.0- 0.5 sec  58.9 KBytes   964 Kbits/sec   0.229 ms    1/   42 (2.4%)
[  7]  0.0- 0.5 sec  58.9 KBytes   964 Kbits/sec   0.285 ms    1/   42 (2.4%)
[  8]  0.0- 0.5 sec  58.9 KBytes   964 Kbits/sec   0.086 ms    1/   42 (2.4%)
[SUM]  0.0- 0.5 sec   301 KBytes  4.94 Mbits/sec
[  4]  0.5- 1.0 sec  21.5 KBytes   353 Kbits/sec   0.263 ms   28/   43 (65%)
[  6]  0.5- 1.0 sec  23.0 KBytes   376 Kbits/sec   0.179 ms   27/   43 (63%)
[  7]  0.5- 1.0 sec  23.0 KBytes   376 Kbits/sec   0.226 ms   27/   43 (63%)
[  5]  0.5- 1.0 sec  21.5 KBytes   353 Kbits/sec   0.049 ms   28/   43 (65%)
[  8]  0.5- 1.0 sec  21.5 KBytes   353 Kbits/sec   0.090 ms   28/   43 (65%)
[SUM]  0.5- 1.0 sec   111 KBytes  1.81 Mbits/sec
[  4]  1.0- 1.5 sec  60.3 KBytes   988 Kbits/sec   0.269 ms    0/   42 (0%)
[  5]  1.0- 1.5 sec  60.3 KBytes   988 Kbits/sec   0.093 ms    0/   42 (0%)
[  6]  1.0- 1.5 sec  60.3 KBytes   988 Kbits/sec   0.190 ms    0/   42 (0%)
[  7]  1.0- 1.5 sec  60.3 KBytes   988 Kbits/sec   0.282 ms    0/   42 (0%)
[  8]  1.0- 1.5 sec  60.3 KBytes   988 Kbits/sec   0.109 ms    0/   42 (0%)
[SUM]  1.0- 1.5 sec   301 KBytes  4.94 Mbits/sec
[  4]  1.5- 2.0 sec  18.7 KBytes   306 Kbits/sec   0.287 ms   30/   43 (70%)
[  7]  1.5- 2.0 sec  17.2 KBytes   282 Kbits/sec   0.355 ms   31/   43 (72%)
[  8]  1.5- 2.0 sec  17.2 KBytes   282 Kbits/sec   0.227 ms   31/   43 (72%)
[  5]  1.5- 2.0 sec  17.2 KBytes   282 Kbits/sec   0.098 ms   31/   43 (72%)
[  6]  1.5- 2.0 sec  18.7 KBytes   306 Kbits/sec   0.159 ms   30/   43 (70%)
[SUM]  1.5- 2.0 sec  89.0 KBytes  1.46 Mbits/sec
```

Figure 4. 7: Network measurements for UDP traffic from host 2. Figure a) represents the iperf client, and Fig. b) represents the iperf server

a)

```
Node: h1
[  4] Sent 1362 datagrams
[  5]  0.0-16.0 sec  1.91 MBytes  1000 Kbits/sec
[  5] Sent 1362 datagrams
[  8]  0.0-16.0 sec  1.91 MBytes  1000 Kbits/sec
[  8] Sent 1362 datagrams
[  6]  0.0-16.0 sec  1.91 MBytes  1000 Kbits/sec
[  6] Sent 1362 datagrams
[SUM]  0.0-16.0 sec  9.55 MBytes  5.00 Mbits/sec
[  7] Server Report:
[  7]  0.0-16.0 sec   873 KBytes   447 Kbits/sec   0.350 ms  755/ 1362 (55%)
[  7]  0.0-16.0 sec  1 datagrams received out-of-order
[  5] Server Report:
[  5]  0.0-16.0 sec   874 KBytes   448 Kbits/sec   0.242 ms  754/ 1362 (55%)
[  5]  0.0-16.0 sec  1 datagrams received out-of-order
[  8] Server Report:
[  8]  0.0-16.0 sec   871 KBytes   446 Kbits/sec   0.218 ms  756/ 1362 (56%)
[  8]  0.0-16.0 sec  1 datagrams received out-of-order
[  6] Server Report:
[  6]  0.0-16.0 sec   871 KBytes   446 Kbits/sec   0.340 ms  756/ 1362 (56%)
[  6]  0.0-16.0 sec  1 datagrams received out-of-order
[  4] Server Report:
[  4]  0.0-16.0 sec   889 KBytes   455 Kbits/sec   0.394 ms  744/ 1362 (55%)
[  4]  0.0-16.0 sec  1 datagrams received out-of-or
root@mininet-vm:~\> 
```

b)

```
Node: h4
[  4] 13.0-14.0 sec  51.7 KBytes   423 Kbits/sec   0.229 ms   49/   85 (58%)
[  5] 13.0-14.0 sec  53.1 KBytes   435 Kbits/sec   0.178 ms   48/   85 (56%)
[  6] 13.0-14.0 sec  51.7 KBytes   423 Kbits/sec   0.195 ms   49/   85 (58%)
[  7] 13.0-14.0 sec  51.7 KBytes   423 Kbits/sec   0.181 ms   49/   85 (58%)
[  8] 13.0-14.0 sec  51.7 KBytes   423 Kbits/sec   0.236 ms   49/   85 (58%)
[SUM] 13.0-14.0 sec   260 KBytes  2.13 Mbits/sec
[  4] 14.0-15.0 sec  51.7 KBytes   423 Kbits/sec   0.183 ms   49/   85 (58%)
[  5] 14.0-15.0 sec  53.1 KBytes   435 Kbits/sec   0.395 ms   48/   85 (56%)
[  6] 14.0-15.0 sec  51.7 KBytes   423 Kbits/sec   0.508 ms   49/   85 (58%)
[  7] 14.0-15.0 sec  51.7 KBytes   423 Kbits/sec   0.277 ms   49/   85 (58%)
[  8] 14.0-15.0 sec  51.7 KBytes   423 Kbits/sec   0.174 ms   49/   85 (58%)
[SUM] 14.0-15.0 sec   260 KBytes  2.13 Mbits/sec
[  4]  0.0-16.0 sec   873 KBytes   447 Kbits/sec   0.351 ms  755/ 1362 (55%)
[  4]  0.0-16.0 sec  1 datagrams received out-of-order
[  5]  0.0-16.0 sec   889 KBytes   455 Kbits/sec   0.394 ms  744/ 1362 (55%)
[  5]  0.0-16.0 sec  1 datagrams received out-of-order
[  6]  0.0-16.0 sec   874 KBytes   448 Kbits/sec   0.243 ms  754/ 1362 (55%)
[  6]  0.0-16.0 sec  1 datagrams received out-of-order
[  7]  0.0-16.0 sec   871 KBytes   446 Kbits/sec   0.219 ms  756/ 1362 (56%)
[  7]  0.0-16.0 sec  1 datagrams received out-of-order
[  8]  0.0-16.0 sec   871 KBytes   446 Kbits/sec   0.340 ms  756/ 1362 (56%)
[  8]  0.0-16.0 sec  1 datagrams received out-of-order
[SUM]  0.0-16.0 sec  4.28 MBytes  2.24 Mbits/sec
```

Figure 4. 8: Network measurements for UDP traffic from host 1. Figure a) represents the iperf client and Fig. b) represents the iperf server

Notice that the total volume of data transmitted by host 1 (4.28 Mbytes), presented in Fig 4.8, is slightly bigger than the predefined value (4 Mbytes). Also note that different loss rates are presented: a 35% loss rate to host 2 and a 55% loss rate to host 1 according to Fig 4.7 and Fig. 4.8 respectively. Different throughputs are presented: a 2.24 Mbps throughput for host 1 and a 3.44 Mbps throughput for host 2, according to Fig. 4.7 and Fig 4.8 respectively. Therefore, the loss rate and throughput depend on the limiting rate as expected.

The measured throughput is also slightly higher than the predefined limiting rate. To measure this imprecision we repeated the experiment a few times for each host, the measured values are presented in Tab. 4.2.

Table 4.2- Network measurements for 5Mbps of UDP Traffic generated at host 1 for 16 seconds

Throughput (Mbps)	Jitter (ms)	Loss Rate (%)	Total transmitted data (Mbyte)
2.19	31	55	4.57
2.18	30	54	4.3
2.2	1.5	55	4.68
2.18	30	55	4.3
2.2	0.5	55.8	4.2
2.24	0.31	55,4	4.28
2.15	31	55.8	4.24

Table 4.3 - Network measurements for 5Mbps of UDP Traffic generated at host 2 for 3.5 seconds

Throughput (Mbps)	Jitter (ms)	Loss Rate (%)	Total transmitted data (Mbyte)
3.81	0.24	21	1.16
3.46	0.24	29	1.08
3.44	0.24	34	1.06
3.75	0.4	25	1.12
3.56	0.5	28.6	1.07
3.58	0.29	28.6	1.07

According to the measurements presented in Table 4.2, the host 1 presented: a mean throughput of 2.19 Mbps, an average Total of transmitted data of 4.37 Mbytes. Therefore, the proposed solution presented an average imprecision of 9.5% on the speed controlling module and average imprecision of 9.25% on the data volume monitoring module for host 1. On the other hand, according to the measurements presented in Table 4.3, the host 2 presented a mean throughput of 3.6 Mbps; and an average total of transmitted data of 1.09 Mbytes. Thus, host 2 presented an average imprecision of 20% on the speed controlling module, and average imprecision of 9% on the data monitoring module.

Finally, the modules work as expected and control the speed and total volume of data transmitted in the network with some imprecision. This result exemplifies how a software defined networking solution can be used to provide customized SLA levels on a per-user basis in a service provider like scenario.

4.2.4 RESTART PHASE OUTCOMES

After verifying that basic features of the system are working fine: service activation, speed controlling, data volume monitoring. We want to test if the system can be usable by testing the reactivation response time.

After service deactivation, the system automatically deletes the subscription entry for the user who went through his data limit.

We test the reactivation time, by reinserting an entry in the subscriptions database and then measuring the time from the insertion to service activation.

Table 4.4 – Response time for reactivation

Repetition	Measured reactivation time
1	392 ms
2	604 ms
3	136 ms
4	860 ms
5	204 ms

According to our measurements, our solution presents a maximum response time smaller than 1 s. This result is very satisfactory and shows that a software defined networking solution can deliver real-time service activation and reactivation to customers.

4.2.5 STOP AGAIN OUTCOMES

Now, TCP traffic is generated to investigate the accuracy of the proposed solutions under different constraints.

Table 4.5: Network measurements for TCP Traffic generated at host 1for 32 seconds

Throughput (Mbps)	Total transmitted data (Mbyte)
2.14	8.25
2.13	8.00
2.16	8.25
2.15	8.25
2.14	8.25

Again, the system worked as expected, with some imprecisions. According to Table 4.5, the mean throughput for TCP is 2.14Mbps and the total transmitted data was 8.20 Mbytes. Thus, the speed controlling module presented an average imprecision of about 7%, and the data volume monitoring module was, now, inversely biased, and showed an average imprecision of 18%. Again, this study case exemplifies how a software defined networking solution can be used to provide customized SLA levels.

4.2.6 UPGRADE PHASE OUTCOMES

After inserting different entries in the subscriptions database we re-evaluate the network performance generating TCP traffic. The subscription database state is described in the Table 4.6.

Table 4.6: Final state of subscriptions database

Subscription id	**User IP address**	**Data volume limit (MB)**	**Rate limit (Kbps)**
5	10.0.0.1	100	4000
6	10.0.0.2	10	4000
7	10.0.0.3	10	4000
8	10.0.0.4	100	4000

Table 4.7: Network measurements for TCP Traffic generated at host 1 for 17 seconds

Throughput (Mbps)	Total transmitted data (Mbyte)
4.18	9.25
4.15	8.75
4.41	9.25
4.34	9.00
4.74	9.62
4.66	9.62
4.69	9.75

According to the measurements in the Table 4.7, the network presented an average throughput of 4.45Mbps. Therefore, an average imprecision of 11.25%.

Nevertheless, the system behaved as expected and it increased the average throughput with a simple update in the subscription database. This exemplifies how a software defined networking solution can provide customized services on a per-user basis in an automatic manner.

5 CONCLUSION

In this section we conclude this work and briefly go through the main achieved results. Possibilities for future works are also discussed briefly.

We successfully developed a software defined networking solution and demonstrated by simulation how the architecture can be used to fasten and ease service provisioning. Additionally, we designed and provided a solution that allows operators to apply new network monetization strategies by delivering customized service levels automatically to different users.

The automatic behavior of the system was exemplified in the tests of section 4 and it is based on the usage of a network database to interface with end systems. This strategy permits innumerous possibilities for automation such as the creation of activation systems providing a direct interface to ISP customers. In that scenario, the system can be design in such way that the billing systems can also be automatically connected to the network provisioning model. Moreover, it permits the implementation of Bandwidth on Demand strategies to providers

In this work, we presented a service provisioning solution with activation response times smaller than 200 milliseconds and reactivation times smaller than 1 second. We also implemented speed controlling and data volume monitoring in a software defined network with small imprecisions under than 10% and 20%, respectively. We could verify that the metering feature of the OpenFlow 1.3 can be applied to provide custom service levels to customer and can be used to control speed for both UDP and TCP traffic.

Additionally we simulated a software defined networking architecture on the widely used network emulator Mininet, and applied some features of OpenFlow 1.3 protocol using the CPqD reference switch and the RYU network controller.

The solution and technologies are very new and insipient, and there is still plenty of room for innovation. The results of this work are very constrained by the data structures and specificities of the written code. The precision of the controlling modules can be improved with better coding strategies, additionally, the response time can also be decreased with better data structures and system architectures. For example, the response time for activation (200 milliseconds) can be surely reduced with better database look-up strategies. The speed controlling module can also be made more precisely with further analysis of the specificities of the OpenFlow 1.3 metering capability.

Future works can design different new strategies for service provisioning automation to deliver new types of features to service providers that can be better suited for real world implementation. For instance, the Pay for QoS feature could be implemented by increasing the granularity of the flow definitions.

REFERENCES

[1] L.L. Peterson and B. S. Davie, *Computer Networks: A System Approach*, Fifth edition, 2012. Morgan Kaufman Publishers,

[2] William B. Norton, "Internet Service Providers and Peering". 2000. Paper available at <http://web.univ-pau.fr/~cpham/ENSEIGNEMENT/COMMUN/norton-peering.pdf > accessed on June, 10th, 2014.

[3] Verizon Enterprise. "Service Level Agreement Internet Dedicated Services". 2014. Website available at <http://www.verizonenterprise.com/terms/us/products/internet/sla/> accessed on June, 10th, 2014.

[4] Verizon Enterprise, "Verizon Voice over IP ("VoIP") SLA". 2014. Website available at <http://www.verizonenterprise.com/terms/us/products/advantage/> accessed on June, 10th, 2014.

[5] Caroline Chappell, "Unlocking Network Value: Service Innovation in the Era of SDN", in Heavy Reading, 2013. White paper available at <http://www.cisco.com/web/solutions/trends/open_network_environment/docs/hr_service_innovation.pdf> accessed on June, 10th, 2014.

[6] Open Networking Foundation, "Software-Defined Networking: The New Norm for Networks" 2012. White paper available at <https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf> accessed on June, 10th, 2014.

[7] Cisco Systems, "Cisco Open Network Environment: Adaptable Framework for the Internet of Everything". 2013. White paper available at <http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/white_paper_c11-727538.pdf> accessed on June, 10th, 2014.

[8] Open Networking Foundation, "OpenFlow 1.3.3 Switch Specification", 2013. White paper available at <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.3.pdf> accessed on June, 15th, 2014.

[9] Hewlett-Packard Development Company, L.P. "Quick Specs HP 2920 Switch Series". Datasheet available at <http://h18000.www1.hp.com/products/quickspecs/14499_div/14499_div.pdf> accessed on June, 15th, 2014

[10] Centec Networks Co. "V330 OpenFlow Switch Reference System". 2014. Datasheet available at <http://www.centecnetworks.com/en/DownList.asp?ID=9&s0=114&s1=115&s2=121&sr=> accessed on June, 15th, 2014.

[11] Edge-Core Networks. "AS4600-54T White-Box Switch Datasheet". Datasheet available at <http://www.edge-core.com/temp/ec_download/1105/AS4600-54T%20DCSS_DS.pdf> accessed on June, 15th, 2014

[12] Pica8 Inc. "P3290 Switch Datasheet". 2014. Datasheet available at <http://www.pica8.com/documents/pica8-datasheet-48x1gbe-p3290-p3295.pdf> accessed on June, 15th, 2014

[13] Open Networking Foundation, "Operator Network Monetization Through Openflow-Enabled SDN", 2013. White paper available at <https://www.opennetworking.org/solution-brief-operator-network-monetization-through-openflow-enabled-sdn> accessed on June, 19th, 2014.

[14] Nippon Telegraph and Telephone Corporation, "Ryu 3.10 documentation", 2014. Website available at < http://ryu.readthedocs.org/en/latest/ > accessed on June, 19th, 2014.

[15] Fernandes, E. L., Rothenberg, C. E. R., "Openflow 1.3 Software Switch", Simpósio Brasileiro de Redes de Computadores e Sistemas distribuídos, SBRC 2014. Paper available at < http://sbrc2014.ufsc.br/anais/files/salao/SF-ST3-1.pdf> accessed on June 19th, 2014

[16] Balakrishnan, H., Feamster, N. "Interdomain Internet Routing", 2005. Paper available at < http://pages.cs.wisc.edu/~akella/CS740/F08/740-Papers/hari-bgp-notes.pdf > accessed on July 7th , 2014

[17] Cisco Systems, “Cisco Open Network Environment: Network Programmability and Virtual Network Overlays”, 2012. White paper available at <http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/white_paper_c11-707978.pdf> accessed on July 6th, 2014