Mestrado em Engenharia Informática

Dissertação

Relatório Final

Multiflow WiFi utilizando Software Defined Networking

Filipe Gonçalves Antonio

fantonio@student.dei.uc.pt

Orientador:

Paulo Alexandre Ferreira Simões

Data: 02 de Setembro de 2014

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Agradecimentos

Agradeço o tempo e os conhecimentos que o Doutor Paulo Simões foi disponibilizando que permitiram construir a soluçao apresentada nesta dissertação.

Resumo

As comunicações móveis estão em crescimento acelerado. O aumento de capacidade do *hardware* dos *smartphones* e *tablets* permite o desenvolvimento de aplicações cada vez mais complexas que requerem uma quantidade cada vez mais elevada de dados. Estes dipositivos não se limitam apenas às chamadas de voz, envio e receção de SMS ou MMS, têm de processar vídeo e áudio em virtude dos conteúdos *on-demand* ou *streaming* que os utilizadores têm ao seu dispor via *internet*. Estar conectado em qualquer lugar é um dos requisitos dos utilizadores, pelo que as operadoras de telecomunicações criaram locais, *hotspots* WiFi, em que os dispositivos móveis dos seus clientes podem ser conectados à internet por meio de tecnologia WiFi e assim aceder a estes conteúdos. O aumento do volume de trafego de dados nestes locais, resultante da quantidade de clientes e das exigências de dados das aplicações que estão a executar nos seus dispositivos móveis, representa um desafio que infraestrutura de rede têm de resolver de forma a otimizar o uso dos equipamentos de rede e assim minimizar o montante de investimentos realizados em equipamentos e instalações (*capital expenditure*, CAPEX) e o custo associado à manutenção dos equipamentos e outras despesas operacionais (*operational expenditure*, OPEX).

A solução apresentada nesta Dissertação visa, com recurso a uma arquitetura baseada no paradigma *Software-defined Networking* (SDN), fazer a gestão do tráfego de dados nos *hotspots* WiFi da Portugal Telecom (PT), distribuindo-o por diversos fluxos, que serão distribuídos por vários canais suportados por diferentes *Access Points* (AP) WiFi e/ou diferentes tecnologias de comunicações sem fios (por exemplo WiFi e 4G). A distribuição dos dados em diversos fluxos, e a distribuição destes fluxos pelos diversos APs disponíveis, será efetuada de acordo com as políticas e requisitos definidos pela operadora (por exemplo para oferecer maior largura de banda a clientes *premium* ou para proteger aplicações mais sensíveis do ponto de vista de qualidade de serviço). No âmbito deste trabalho de dissertação foi proposta uma arquitetura baseada em SDN para suportar o modelo de funcionamento descrito. Esta arquitetura foi posteriormente implementada e instalada num *testbed* criado para o efeito (e que corresponde a uma versão simplificada da arquitetura proposta) e que serviu para demonstrar a viabilidade do modelo proposto.

Palavras-Chave

SDN *Controller*, Open vSwitch, OpenFlow, *Software-Defined Networking*.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

AP	Access Point
API	Application Programming Interface
App	Application software
ARP	Address Resolution Protocol
CAPEX	Capital Expanditure
CLI	Command-Line Interface
DHCP	Dynamic Host Configuration Protocol
GRE	Generic Routing Encapsulation
GUI	Graphical User Interface
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
LACP	Link Aggregation Control Protocol
NBI	Northbound Interface
O&M	Operation & Maintenance
OPEX	Operational Expenditure
QoS	Quality of Service
REST	Representational State Transfer
RTT	Round-trip time
SDN	Software-Defined Networking
SNMP	Simple Network Management Protocol
SPAN	Switch Port Analyzer
ToS	Type of Services
VoIP	Voice over IP

Capítulo 1
Introdução

1.1 Motivação

As comunicações móveis estão em franco crescimento, prevendo-se que até 2016 o trafego mundial de dados móveis seja incrementado dezoito vezes, de acordo com as previsões de tráfego global de dados móveis do *Visual Networking Index* (VNI), da Cisco. Cada vez mais os utilizadores querem visualizar vídeos e outros conteúdos multimédia em tempo real, em qualquer lugar e em dispositivos móveis (portáteis, *smartphones*, *tablets*...). Devido aos custos elevados da infraestrutura os operadores de telecomunicações têm de otimizar os equipamentos já instalados de forma a conseguir satisfazer mais clientes. Os utilizadores requerem cada vez mais conteúdos fornecidos *on-demand* ou em *streaming*, em detrimento do conteúdo simplesmente descarregado, pelo que o tráfego de *cloud* em dipositivos móveis irá aumentar 28 vezes entre 2011 e 2016. Com a constante evolução do *hardware*, a capacidade de computação dos dispositivos móveis está cada vez mais avançada, levando à geração e consumo de mais tráfego de dados. Os utilizadores querem também consumir vídeos com maior qualidade, estimando-se que este tipo de conteúdos venhas a representar 71% de todo o tráfego de dados móveis até 2016 [1].
Um *Hotspot* WiFi é um local no qual é possível ter acesso à Internet via tecnologia WiFi (suportada pelas normas IEEE 802.11), sem necessidade de cabos. São normalmente locais públicos, centros comerciais, bares, hotéis, restaurantes, aeroportos. Para utilizar o *Hotspot* WiFi é necessário ter um computador portátil, *smartphone* ou *tablet* que suporte tecnologia *WiFi*. Adicionalmente, caso não se trate de uma rede aberta, é necessário que o utilizador se autentique (na maioria dos casos através de uma palavra-passe). [2]
Em cada *Hotspot* podem existir vários AP, de forma a cobrir uma área mais alargada e/ou a suportar mais utilizadores, causando a sobreposição de redes em determinados pontos, onde um dispositivo móvel tem acesso aos vários AP.

A Portugal Telecom (PT) disponibiliza 300 mil *hotspots* em todo o país através da sua rede WiFi, estando estes *hotspots* distribuídos por duas categorias:

- *Hotspots Premium* WiFi PT, instalados em espaços públicos como hotéis, restaurantes, cafés, esplanadas, estações de correio, centros comerciais, estações de serviço, centros de conferência e estádios de futebol.
- *Hotspots* da Comunidade WiFi PT, disponíveis com o partilhar da ligação doméstica do cliente MEO fibra (resultando assim acesso em mais de 300 mil *hotspots*). Esta comunidade é composta por este perfil de cliente, que transforma a sua ligação num *hotspot* WiFi PT, partilhando os seus recursos de internet livres com outros utilizadores WiFi PT mantendo toda a qualidade, segurança e privacidade da sua rede doméstica. Para garantir os pontos anteriores, são criadas duas redes distintas, a rede privada para os acessos domésticos do cliente e uma rede pública do *hotspot* que tem um endereço IP específico e uma largura de banda limitada de modo a não interferir na qualidade da rede doméstica. É necessário uma autentificação de todos os utilizadores.

Para além da cobertura nacional, a Portugal Telecom disponibiliza este serviço com *roaming*, em parceria com operadores de Telecomunicações de outros países a nível mundial [3-4].

1.2 Proposta

Esta dissertação tem por objetivo desenhar, implementar e testar uma plataforma capaz de fazer a gestão do tráfego de dados para dispositivos móveis com varias interfaces IEEE 802.11, encontrando-se cada uma delas ligada a um AP diferente (mas integrado no mesmo *hotspot)* para que os fluxos de dados sejam distribuídos por vários APs, de acordo as políticas e requisitos do operador (por exemplo privilegiar clientes *premium* ou aplicações *premium*, ou optimizar a carga de cada AP do *hotspot* de modo a maximizar o numero de clientes suportados em simultâneo).

Para esse efeito é proposta uma plataforma cuja arquitetura usa princípios de *Software-Defined Netkorking*, recorrendo para o efeito à ferramenta *OpenFlow* (Figura 8). Com esta solução, o trafego de dados é classificado por fluxos e encaminhado de forma diferenciada para as várias interfaces de rede IEEE 802.11 dos dispositivos dos clientes, seguindo as regras definidas pelo operador e com alterações mínimas na infraestrutura previamente existente.

Para a execução de algumas das tarefas presentes neste projeto, a PT Inovação disponibilizou informação sensível da sua infraestrutura, pelo que este relatório deve ser considerado como confidencial. Esta Dissertação enquadra-se na disciplina de dissertação/estágio do Mestrado em Engenharia Informática.

1.3 Enquadramento

Este estágio enquadra-se no Projeto Multiflow WiFi SDN, realizado em parceria com a PT Inovação e que procura explorar o potencial da combinação do uso de tecnologias SDN com redes WLAN. A PT Inovação é uma empresa tecnológica focada no desenvolvimento e entrega de produtos e serviços avançados para o mercado das telecomunicações e das tecnologias da informação.
Este projeto pretende dar resposta ao caso de uso para terminais móveis multi-rádio IEE 802.11 em ambientes com cobertura sobreposta de vários *hotspots* WLAN (Figura 1), propondo uma arquitetura que, com mecanismos SDN, permita um melhor aproveitamento dos APs.

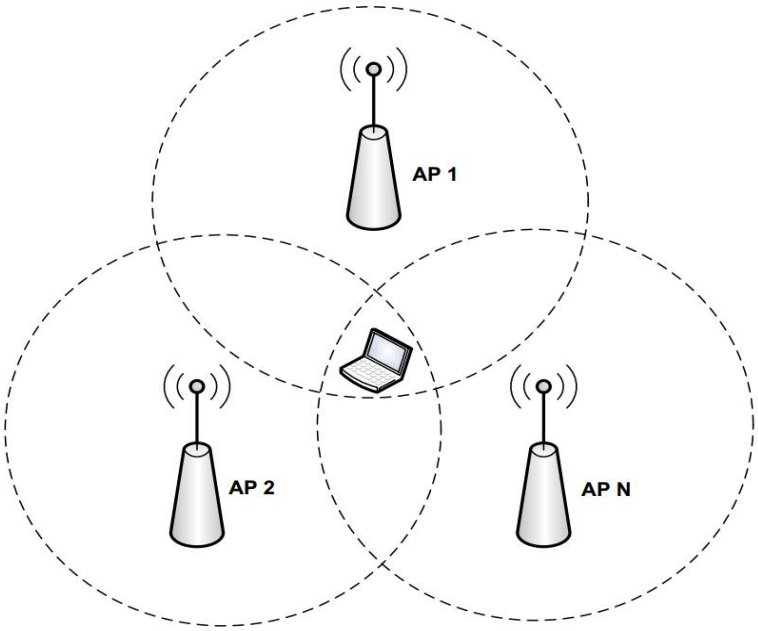

Figura 1 - Sobreposição de áreas de cobertura de *access points*

Num *hotpsot* WiFi, um cliente estabelece uma ligação com um AP selecionado pelo sistema operativo com base em parâmetros fixos (por exemplo maior intensidade de sinal rádio) e que não tomam em conta todos os fatores relevantes. Esse AP pode, por exemplo, ter um sinal rádio bastante forte mas estar congestionado por já ter demasiados utilizadores associados (ou aplicações que requeiram maiores volumes de tráfego). Daqui resulta que uma seleção automática do AP, feita pelo sistema operativo com base num conjunto estático e reduzido de parâmetros poder resultar em piores níveis de Qualidade de Serviço (QoS) para aplicações e utilizadores. Adicionalmente, atualmente assume-se que o dispositivo móvel tem um único rádio WiFi e se associa a um único AP – ainda que seja previsível que no futura o dispositivo possa ter vários interfaces WiFi e/ou associar-se em simultâneo a vários APs (hoje em dia já seria possível, por exemplo, usar em simultâneo os interfaces 4G e WiFi do dispositivo móvel).

O objetivo deste projeto é fazer a gestão do tráfego de dados com uma arquitetura SDN, em que as decisões de encaminhamento sejam centralizadas num ponto lógico, e os dispositivos de rede se limitem a encaminhar os pacotes pelos caminhos definidos pelo operador. Assim um cliente com um terminal móvel multi-rádio 802.11 poderá ligar-se a vários APs de um *hostpot*. O operador, usando as informações do estado dos APs e, eventualmente, do estado das interfaces dos clientes, poderá definir caminhos diferenciados para os vários fluxos de dados, de modo a melhorar o serviço prestado sem que o cliente tenha de ter uma intervenção ativa. Na infraestrutura da rede terão de ser adicionados controladores SDN e dispositivos de rede que suportem o protocolo OpenFlow – no entanto não haverá necessidade de alterar os APs já existentes ou o essencial da infraestrutura atual, um fator relevante para que a solução proposta possa ser gradualmente implantada na infraestrutura de produção dos operadores.

1.4 Estrutura do relatório

Este relatório é constituído por oito capítulos, que seguidamente são descritos de uma forma sumária. No capítulo dois são abordadas as tecnologias que servem de base para a execução deste projeto e alguns trabalhos que também abordam alguns dos aspetos necessários para a realização deste projeto. Os requisitos e casos de uso que estão na origem deste projeto e o algoritmo de decisão são apresentados no capítulo três. No capítulo quatro é apresentada a arquitetura proposta, de acordo com os objetivos previamente enunciados. No capítulo cinco é descrita a estrutura da *testbed* que foi criado, assim como os testes efetuados com vista à validação da plataforma proposta. No capítulo seis são descritos em pormenor os testes a efetuar na *testbed* e os resultados obtidos. No capítulo sete é feita uma descrição breve das tarefas realizadas e as que serão feitas no segundo semestre. No capítulo oito são apresentadas as conclusões desta primeira fase.

Capítulo 2
Estado de Arte

Neste capítulo são introduzidos os conceitos e as tecnologias que servirão de suporte para a definição e implementação da solução proposta. Na secção 3.1 é feita uma breve descrição da arquitetura SDN. Na secção 3.2 é apresentado um estudo comparativo de SDN *controllers*, que serviu para escolher qual o SDN *controller* a implementar. Na secção 3.3 é abordado o protocolo *OpenFlow*, que será utlizado para a troca de mensagens entre o SDN *controller* e o dispositivo de rede compatível com *OpenFlow*. Na secção 3.4 é feita a descrição do *Open vSwitch*, um *software* que permite criar instâncias de *switches* compatíveis com *OpenFlow*.

2.1. SDN

O crescimento de dispositivos móveis e respetivos conteúdos, assim como o crescimento dos *cloud services*, estão entre os fatores que levaram a indústria a repensar as arquiteturas tradicionais de redes. As redes convencionais são hierárquicas, construídas com *switchs ethernet* distribuídos em camadas definindo uma estrutura em forma de árvore. Este design está de acordo com o modelo de computação cliente-servidor, no entanto para computação dinâmica e armazenamento nos *data centers*, *campus* e operadoras, esta arquitetura estática não se adequa às suas necessidades. Alguns dos conceitos de computação que requerem um novo paradigma de rede:

- A mudança dos padrões de trafego - Nas aplicações cliente-servidor, quase toda a comunicação centra-se entre um cliente e um servidor contrastando com as aplicações atuais que requerem acesso a diferentes bases de dados e servidores. Os utilizadores estão a mudar o tipo de padrão de trafego ao aceder a conteúdos e aplicações através de diversos tipos de dispositivos, ligando-se em qualquer ponto e a qualquer hora.
- O aumento da oferta de *cloud services* - O mercado de *cloud services* públicos e privados está em franca expansão. As empresas que fornecem este tipo de servições necessitam de agilizar o acesso às aplicações, infraestruturas e outros recursos de IT. Para além de que no planeamento de *cloud services* devem ser tomados em conta aspetos de segurança.
- "*Big Data*" - Tratar "*Big Data*" ou mega *datasets* requer processamento paralelo massivo recorrendo a milhares de servidores, que precisam de estar conectados entre eles. Com o crescimento dos *datasets*, a rede destes *data centers* tem de ser escalável de forma a poder crescer e conseguir manter a conetividade entre os servidores.

As tecnologias das redes atuais não vão de encontro aos requisitos do mercado. Com orçamento cada mais reduzidos e a pressão de fazer cada mais com menos, os departamentos de IT tentam a todo o custo acomodar a necessidade de uma maior resposta da rede nos equipamentos existentes. Até à data, as tecnologias de redes são maioritariamente um conjunto de protocolos que visam ligar *hosts* de forma confiável sobre distâncias arbitrárias, velocidades de ligação e topologias. Os protolocos evoluíram de forma a incrementar a performance, confiabilidade, conetividade e segurança da rede. Um protocolo tem por objetivo(s) solucionar problemas específicos, limitando assim a possibilidade de abstração. Esta característica levou à complexidade das redes. Ao adicionar ou mover um dispositivo de uma rede implica alterar a configuração de *switches*, *routers*, *firewalls*, portais web de autenticação, para além de ser necessário atualizar ACLs, VLANs e QoS. As redes tendem a ser estáticas com vista a minimizar o risco de indisponibilidade, contrastando com o dinamismo criado com a virtualização de servidores. Com a

virtualização, o número de *hosts* que necessitam de ligação à rede aumentou, alterando a perceção de localização física de *hosts*. Muitas empresas dispõem de um número limitado de endereços públicos, pelo que é comum o trafego de voz, dados e vídeo partilharem o mesmo IP. É possível definir níveis diferenciados de QoS para cada tipo de aplicação, no entanto, a reserva desses recursos é pouco automatizada. Requer a configuração separada de cada equipamento de acordo com as suas especificações e o ajuste de parâmetros tais como largura de banda, QoS por sessão. Devido à sua natureza estática, as redes não têm capacidade de se adaptar dinamicamente às mudanças de trafego, aplicações e pedidos dos utilizadores. À medida que a rede cresce para ir de encontro com as necessidades das empresas, aumenta a complexidade da mesma, fruto da adição de novos equipamentos que precisam de ser configurados e geridos. Em empresas de grandes dimensões, com centenas de servidores, não é possível fazer a configuração manual da rede por cada novo servidor, pois a rede atual é pouco escalável.

As empresas têm de inovar em termos de ofertas e serviços aos seus clientes, mas nem sempre os equipamentos de rede suportam novas funcionalidades. Alguns aspetos que limitam a adaptação tais como a falta de padrões, interfaces livres levam a que os operadores de redes não possam moldar a rede de acordo com as necessidades de cada um.

As capacidades das redes não satisfazem totalmente os requisitos do mercado, pelo que surgiu a arquitetura *Software-Defined Networking* (SDN).

SDN é uma arquitetura de rede em que o controlo é desassociado do encaminhamento e é diretamente programável. A migração do controlo, anteriormente incorporado nos dispositivos de redes, para dispositivos de computação acessíveis permite que a infraestrutura subjacente seja abstraída com aplicações e serviços de rede, que a podem tratar como entidade lógica ou virtual [5].

A arquitetura SDN é constituída por três camadas (Figura 2): camada aplicacional, camada de controlo e camada de infraestrutura.

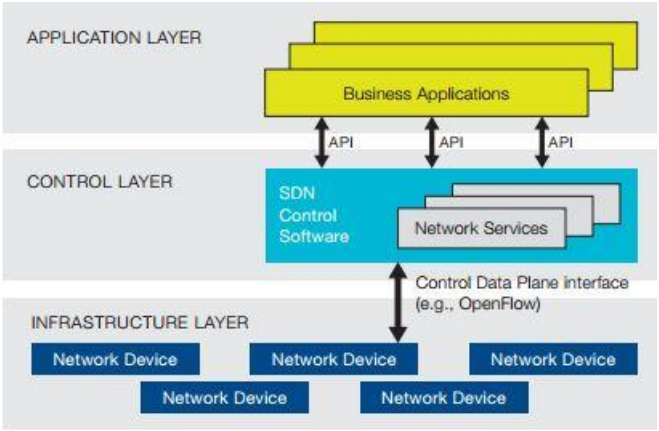

Figura 2 - Arquitetura SDN [5]

A camada aplicacional, ou SDN App, é constituída por aplicações SDN que são programas que permitem comunicar com o SDN *controller* via *Northbound Interfaces* (NBI's), e nos quais são definidos os requisitos e comportamentos pretendidos para a rede.

A camada de controlo é constituída por um SDN *controller*. É uma entidade logica centralizada que é responsável converter os pedidos da camada aplicacional e enviar essa informação para a camada de infraestrutura.

A camada de infraestrutura é constituída pelos dispositivos de rede que encaminham o tráfego de acordo com as indicações recebidas do SDN *controller* [6].

A comunicação entre as camadas da arquitetura SDN é suportada por Application Programming Interface (API) (Figura 3). Elas disponibilizam um canal através do qual as instruções de programação podem ser enviadas para o dispositivo que se pretende programar. As APIs são chamadas de *northbound* ou *southbound*, dependendo da sua função na arquitetura SDN. As APIs que são disponibilizadas pelo SDN *controller* e que são usadas por aplicações para enviar instruções para o SDN *controller* são *northbound*, porque a comunicação ocorre a norte do SDN *controller*. As APIs *southbound* são disponibilizadas pelos dispositivos de rede, como *switches*, que são usados pelo SDN *controller* para fazer a gestão da rede. O nome está relacionado com o fato da comunicação ocorrer a "sul" do SDN *controller*.

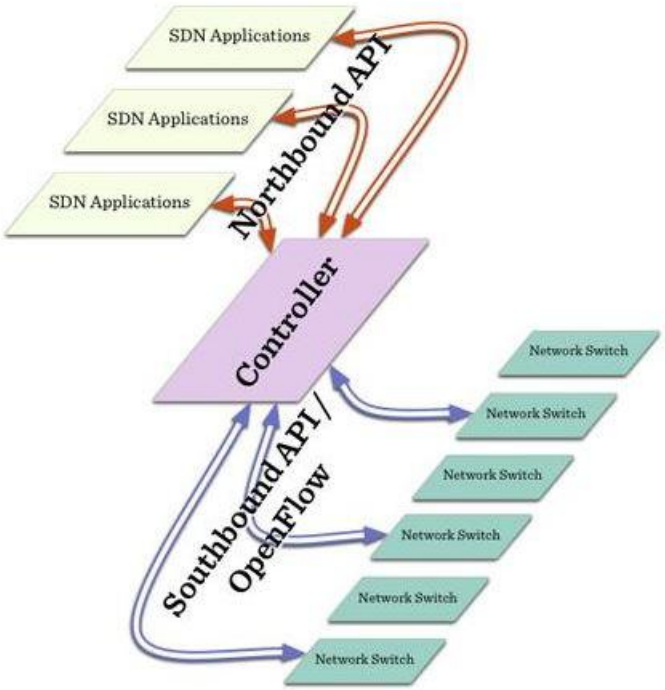

Figura 3 - APIs na arquitetura SDN [7]

A inteligência da rede está centralizada nos SDN *controllers* da camada de controlo, que têm a visão global da rede. Este fato leva a que a rede, tanto para as aplicações como para os motores de policiamento, aparente ser um *switch* lógico. Com SDN as empresas e operadores adquirem o controlo, que é independente de qualquer fabricante, das suas redes a partir de um ponto lógico, simplificando o desenho e funcionamento das mesmas. Com SDN, a gestão dos dispositivos de rede também é simplificada, pois não é necessário compreender nem configurar os protocolos padrão e proprietários desses dispositivos: basta para tal que aceitem instruções dos SDN *controllers*. Mas mais importante é a capacidade dada aos administradores e operadores de redes de configurar esta abstração da rede de forma programática, em vez de os obrigar a percorrer todos os dispositivos e configura-los um a um. A centralização do controlo da rede num ponto permite alterar o comportamento da rede em tempo real e implantar novas aplicações e serviços de rede num relativo curto espaço de tempo comparativamente ao tempo atualmente gasto. O SDN permite aos administradores de rede configurar, gerir, proteger e otimizar os recursos da rede de forma

dinâmica, com programas SDN automatizados. Para além disso, há a liberdade para escrever programas e criar assim as funcionalidades pretendidas, em vez de esperar que os fabricantes as disponibilizem nos seus equipamentos (por vezes ainda de forma fechada) de modo a satisfazer os requisitos pretendidos. As arquiteturas SDN disponibilizam um conjunto de APIs que permitem implementar serviços de redes tais como *routing, multicast*, segurança, controlo de acesso, gestão da largura de banda, QoS, otimização de armazenamento e processamento, utilização de energia e políticas de gestão.

Este tipo de arquitetura adequa-se a diversos tipos de organizações:

- *Campus* – Controlo centralizado e automatizado com convergência de dados, voz e vídeo, a qualquer hora e em qualquer lugar, permitindo aplicar políticas de forma consistentes na rede de cablagem estruturada e sem fios. Reserva de recursos de acordo com o tipo de aplicação, perfil de utilizador.
- *Data center* - As arquiteturas SDN facilitam a virtualização da rede, tornando-a muito escalável, a migração automatizada das *Virtual Machines*, menor consumo de energia e otimização da largura de banda
- *Cloud* - Num ambiente de *cloud* privada ou híbrida, o SDN permite que os recursos de rede sejam alocados de uma forma altamente elástica, permitindo uma rápida reserva de *cloud services* e uma transição mais flexível para a *cloud* externa de um fornecedor.
- Operadoras e fornecedores de serviços – O SDN oferece a escalabilidade e automação necessárias para a implementação a funcionalidade de um modelo de computação *IT as a Service*. Com o controlo centralizado e automatizado da rede e um modelo de reserva de recursos, torna-se possível otimizar o uso dos recursos de rede, reduzir o CAPEX e OPEX e aumentar o valor e velocidade dos serviços. [5]

Em 2011 foi criada a Open Networking Foundation (ONF), uma organização sem fins lucrativos dedicada à promoção e adoção do *Software-Defined Networking* suportado por padrões abertos. Nos seus associados encontram-se fabricantes de *hardware*, *software*, prestadoras de serviços de rede, operadoras de telecomunicações entre outros, a nível mundial.

2.2 Controladores SDN

Nas subsecções seguintes temos uma breve descrição de SDN *controllers* disponíveis e as características que foram tidas em conta e que explicam a escolha do controlador que melhor se adaptou à realização deste projeto.

2.2.1 Ryu

O Ryu é uma *Framework* para SDN baseada em componentes. As componentes de *software* disponibilizam APIs bem definidas, permitindo aos desenvolvedores criarem novas aplicações de gestão e controlo de rede. O Ryu suporta vários protocolos de gestão de dispositivos de rede como *OpenFlow*, Netconf, OF-config. Relativamente ao *OpenFlow*, o Ryu suporta as especificações 1.0, 1.2, 1.3 e Extensões Nicira. A linguagem de programação é o *Python*, sendo a versão 3.5 a mais recente.

O código está disponível de forma gratuita com uma licença Apache 2.0 [8-9].

3.2.2 NOX

O NOX é uma plataforma para construir aplicações para controlo da rede. Inicialmente foi desenvolvido na Nicira Networks, sendo o primeiro controlador *OpenFlow*. Em 2008, A Nicira doou o NOX para a comunidade de investigadores e tem sido a base para muitos e variados projetos de pesquisa. O NOX seguiu duas linhas distintas de desenvolvimento, o NOX-Classic e o NOX, a.k.a., o novo NOX. O NOX- Classic é a linha de desenvolvimento que está disponível com licença GPL desde 2009. Tem suporte para *Python* e C++. De acordo com as indicações do site, os responsáveis não pretendem desenvolver mais para esta linha.

O novo NOX só suporta C++, tem menos aplicações do que NOX-Classic, no entanto é muito mais rápido e tem um código fonte muito mais limpo. Disponibiliza uma API *OpenFlow* 1.0 em C++ e está disponível para as distribuições Linux mais recentes. Os responsáveis desta plataforma também disponibilizam o POX, sobre a qual estão a focar o seu trabalho. [10].

2.2.3 POX

O POX é uma plataforma desenvolvida pelos responsáveis do NOX, mas mais recente do que esta. O requisito que levou à sua criação foi o desenvolvimento rápido e prototipagem de *software* para controlo de rede utilizando Python, para concorrer com o número crescente de *frameworks* que permitem criar um controlador *OpenFlow*. Disponibiliza uma interface *OpenFlow* "Pythonic". Tem exemplos de componentes reutilizáveis para a seleção do caminho, a descoberta da topologia. Compatível com os sistemas operativos Linux, Mac OS e Windows. Suporta as ferramentas de visualização e GUI do NOX [11].

2.2.4 Beacon

O Beacon é escrito em Java e corre em muitas plataformas, desde servidores Linux *multi-core* de alto desempenho até smartphones Android. É open source, estando licenciado sob uma combinação da licença GPL v2 e da *License Exception* v1.0 da Universidade de Stanford FOSS. É um controlador dinâmico, em que pacotes de código podem ser inicializados, parados, atualizados e instalados em tempo de execução, sem interromper outros pacotes que não dependam do Beacon. Este controlador é *multithread*[12].

2.2.5 Floodlight

O Floodlight é um *Open* SDN *controller* que funciona com *switches OpenFlow* e *switches OpenFlow* virtualizados. É baseado em java e está licenciado com uma licença Apache. É suportado por uma comunidade de desenvolvedores, incluindo engenheiros da Big Switch Networks. Disponibiliza um sistema de carregamento de módulos, permitindo ampliar e melhorar o controlador. É fácil de configurar e requer dependências mínimas. Permite a gestão de redes *OpenFlow* e não *OpenFlow* misturadas. Projetado para alto desempenho, é o núcleo de um produto comercial de Redes da Big Switch Networks. [13]

2.2.6 MuL

O MuL comporta-se como um núcleo de uma infraestrutura *multi-thread* baseado em C. Disponibiliza NBIs que permitem a interação com aplicações. Essas interfaces são do tipo ML-API and RESTful NB-API. Suporta o protocolo *OpenFlow* 0x4 (do tipo 1.3.x), bem como 0x1 (1.0). MuL está licenciado com GPLv2. [14-15]

2.2.7 OpenDaylight

O OpenDaylight (Figura 4) é uma plataforma aberta para programação de redes para a implementação dos conceitos de SDN e Network Functions Virtualization em redes de qualquer tamanho e escala. Este *software* é uma combinação de componentes, na qual se inclui um controlador, interfaces de protocolo de plug-ins e aplicativos. As interfaces Northbound e Southbound estão bem definidas e as APIs documentadas.

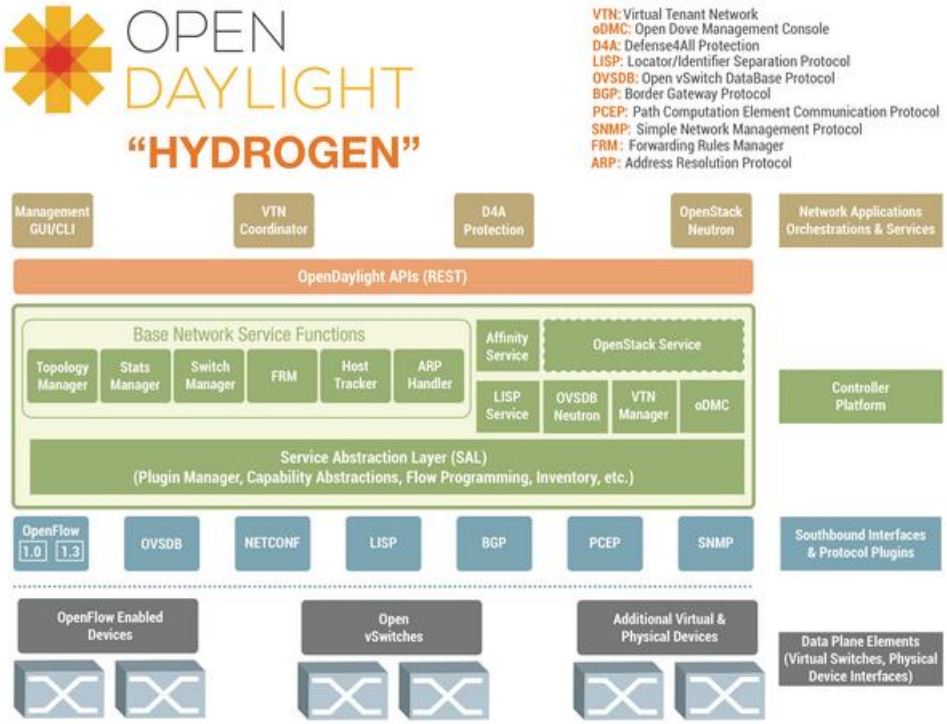

Figura 4 - Estrutura do OpenDaylight [16]

Em 4 de Fevereiro de 2104 foi lançada a primeira versão do Opendaylight com a denominação de hydrogen.

2.2.8 Escolha do SDN Controller

Na escolha do controlador SDN Controller a utilizar no testbed foram tidas em conta os requisitos seguintes: ferramentas de acesso remoto ao Controlador SDN para definir os fluxos (*northbound* API), sistema operativo de suporte ao controlador, versão de *OpenFlow* implementada para transferência dos fluxos para o OF switch (*southbound* API), empresa de suporte ao controlador, envolvimento da comunidade, linguagem de desenvolvimento, licença de uso (Tabela 1). Da lista de controladores analisados Ryu, NOX, POX, Beacon, Floodlight e Mul e Opendaylight, a escolha recaiu sobre o Opendaylight. Na arquitetura do

Tabela 1 - Comparativo das soluções de SDN *controller*

Característica	SDN Controller						
	ryu	NOX	POX	Beacon	OpenDaylight	Floodlight	Mul
Northbound Interface	JSON-RPC	não	não	REST API	REST API	REST API	ML-API e RESTful NB-API
Suporte de sistemas operativos específicos	Integrado em OpenStack	Linux	Windows, Mac OS, Linux	Mac, Windows, multi-core Linux servers a Android phones	Todas as plataformas que suportem Java	Linux, Mac	linux
Condições de licenciamento	GNU General Public License version 2.0 (GPLv2)	GPL	GPL	GPL v2 license e a Stanford University FOSS License Exception v1.0	Eclipse Public License – v 1.0 (EPL)	Apache License Version 2.0	GPLv2
Empresa de Suporte	NTT laboratories OSRG group	ICSI	ICSI	Stanford University	Linux Foundation	BigSwitch	Kulcloud Networks
Linguagens suportadas pelos controladores	Python	C++/python	Python	Java	C, C++, Java, Python	Java, Python	C
Tem suporte por parte da comunidade?	SIM	não	SIM	SIM	SIM	SIM	SIM
Versões *OpenFlow* Suportadas	1.0, 1.2, 1.3	1.0	1.0	1.0	1.0, 1.3	1.0	1.0, 1.3

sistema, a gestão do controlador SDN é feita remotamente por um servidor dedicado, de forma a fazer a gestão e definição dos flows de acordo com políticas definidas pelo O&M. O Linux é o sistema operativo da maioria dos *open source controllers*, o que permitiria aceder via ssh, mas implicaria criar uma conta com alguns privilégios de administração do *host* do *controller*. Está em desenvolvimento a gestão dos *controllers*, recorrendo a apps remotas, mas para tal, os *controllers* tem de disponibilizar mecanismos que permitam essa interação. Esses mecanismos são denominados de *northbound* APIs. Essas APIs permitem enviar instruções ao *controller* para definir os fluxos que serão enviados aos *switchs OpenFlow* (ou implementações destes), em que estão definidos as ações a efetuar de acordo com cada tipo de fluxo. Dos *controllers* estudados, o ryu disponibiliza JSON-RPC, o beacon, o Floodlight e o Opendaylight disponibilizam REST API (ver figura 4) e o mul que disponibiliza ML-API and RESTful NB-API, o que excluiu alguns dos controladores. Na arquitetura, os *switches OpenFlow* e no cliente serão implementados a partir do software open vSwtich. A versão do *OpenFlow* que o Open vSwtich suporta é a 1.3 pelo que na escolha do controller foi necessário ter em conta quais as versões de *OpenFlow* usado nas comunicações *southbound*. Embora esteja previsto que no futuro o open vSwitch suporte versões mais recentes de *OpenFlow*, não está disponível a data para quando tal irá acontecer. O Opendaylight suporta as especificações 1.0 e 1.3 de Open Flow, versões essas que são um requistiot de desenvovimeto para este projeto. Relativamente à linguagem de programação, o

Opendaylight é um dos poucos desenvolvido em java e é programável em java, phyton, C e C++, que, dado a característica das aplicações em java, "*run everywhere*", devido ao jvm, torna versátil a sua instalação e execução. O Opendaylight é um *open source controller* que pertence à Linux Foundation, cujo projeto tem contribuição de granseds empresas ligadas à informatica. Por este conjunto de características o Floodlight foi a opção escolhida para implementar o *controller*.

2.3. *OpenFlow*

O *OpenFlow* é a primeira interface de comunicação padrão definida para estabelecer a ligação entre as camadas de controlo e encaminhamento da arquitetura SDN. O *OpenFlow* permite o acesso direto e manipulação dos planos de encaminhamento dos dispositivos de rede, tais como *switches* e *routers*, tanto físicos como virtualizados. Nenhum outro protocolo padrão permite mover o controlo da rede por parte dos *switches* de rede para um software de controlo lógico centralizado. O protocolo *OpenFlow* é implementado nas interfaces dos dispositivos de rede da infraestrutura e no *software* da camada de controlo do SDN. O *OpenFlow* usa o conceito de fluxos para identificar o trafego da rede com base em regras pré-definidas que podem ser programadas dinamicamente ou por estatísticas, via *software* da camada de controlo do SDN. Este conceito permite ao administrador da rede definir o modo como o tráfego irá circular pela rede, tendo por base os padrões de uso, aplicações e recursos da *cloud*. Como o *OpenFlow* permite programar a rede com base em fluxos, uma arquitetura SDN baseada neste protocolo disponibiliza um controlo extremamente granular, permitindo à rede adaptar-se em tempo real às exigências das aplicações, utilizadores e sessões. O encaminhamento *IP Based* atualmente utilizado não permite este tipo de controlo. [5].

2.3.1 *OpenFlow Switch*

Um *switch* OpenFlow é um *switch* que cumpre as especificações do OpenFlow Standard e pode ser dividido em três componentes(Figura 5):

- Uma tabela de fluxos com a informação de como processar cada fluxo. A cada fluxo estão associados ações
- Um canal seguro de comunicação entre o *switch* e o dispositivo remoto de controlo *(controller)*. A comunicação é feita de acordo com o protocolo *OpenFlow*.
- O Protocolo *OpenFlow* disponibiliza um padrão de comunicação entre o *switch* e o *controller*. Especifica uma interface padrão através da qual as entradas da tabela de fluxos podem ser definidas externamente.

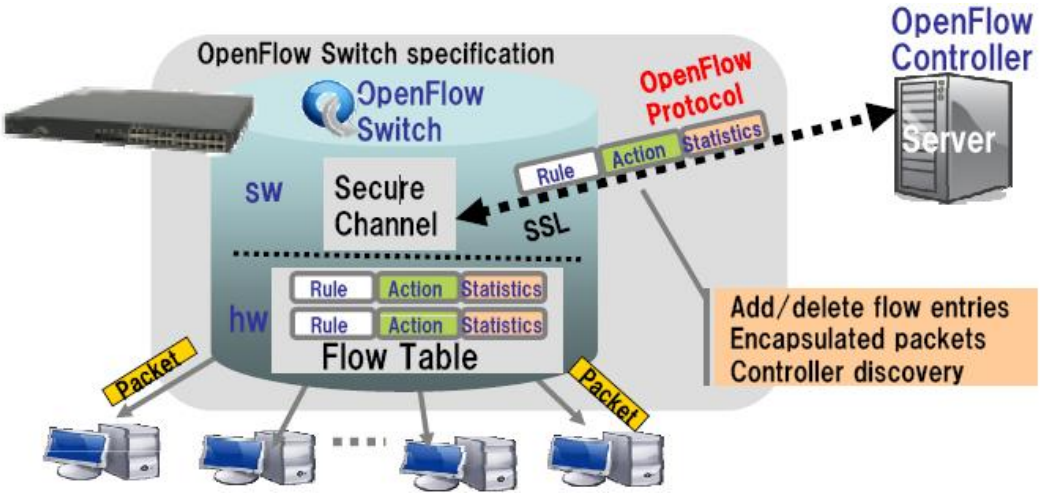

Figura 5 - Estrutura de um switch OpenFlow [17]

Um fluxo de uma tabela é constituído por quatro campos (Figura 6). O campo *header* define quais os campos e os valores que o pacote tem de ter para pertencer ao fluxo. O campo *counters* serve para contabilizar o número de pacotes já passaram e pertenciam ao fluxo. O campo *Actions* define as ações a aplicar ao pacote daquele fluxo. O campo *Priority* define a ordem dos fluxos com os quais é feita a comparação dos campos do pacote.

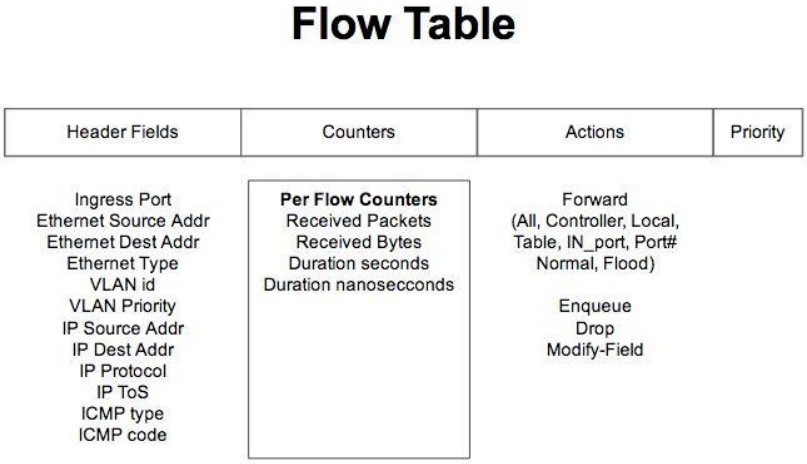

Figura 6 - Estrutura de um fluxo de uma tabela [18]

Quando um *switch* OpenFlow recebe um pacote, procura na tabela qual é o fluxo ao qual aquele pacote pertence. Se houver uma correspondência, executa as operações definidas na ação. Caso contrário envia o pacote para o *controller* (Figura 7).

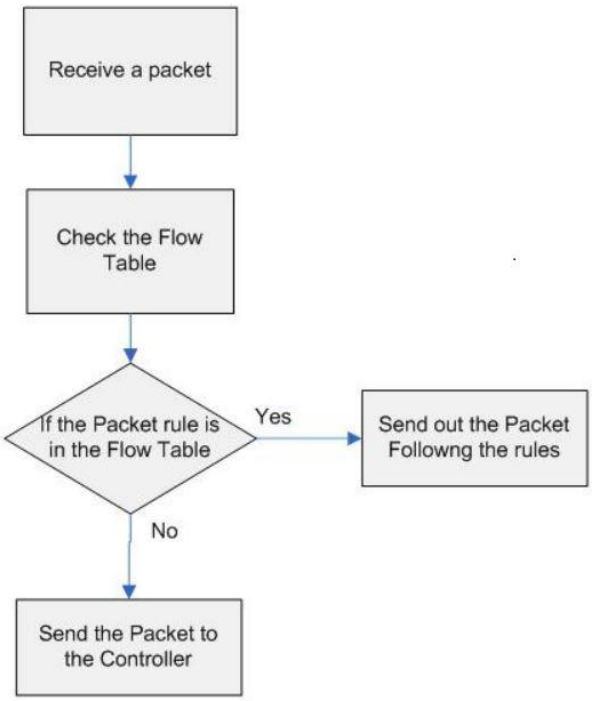

Figura 7 – Diagrama de Fluxos de um pacote num switch OpenFlow [12]

A primeira especificação de um Openflow Switch foi disponibilizada em Dezembro de 2009.

2.4. Open vSwitch

O Open vSwitch é um *software* que permite criar um *switch* virtual multicamada, estando licenciado com *open source* Apache 2.0. Possibilita a automatização de uma rede de grande dimensão e é compatível com interfaces padrão de gestão e protocolos (NetFlow, sFlow, Switch Port Analyzer (SPAN), Remote SPAN, CLI, LACP, IEEE 802.1ag). A primeira referência a Open vSwitch surge em 20 de Maio de 2009, estando disponível num repositório público em Julho de 2009. O Open vSwitch disponibiliza uma base de dados (OVSDB) com a informação do estado da rede. Por ser compatível com as especificações OpenFlow, torna possível o acesso remoto com o objetivo de controlar o trafego, permitindo a descoberta da rede global através de protocolos do tipo Link Layer Discovery Protocol, Cisco Discovery Protocol e Open Shortest Path First. O Open vSwitch inclui vários métodos para especificar e manter regras de marcação, acessíveis remotamente. Estas regras são armazenadas de forma otimizada, evitando estar associado a dispositivos de rede complexos. Suporta uma implementação de Generic Routing Encapsulation (GRE) que permite gerir milhares de túneis GRE simultaneamente bem como a criação e gestão remota dos mesmos.
Uma das prioridades do Open vSwitch é manter o código no *kernel* tão pequeno quanto possível e reutilizar os subsistemas já existentes, por exemplo no caso da pilha de QoS. O *kernel* do Linux tem um módulo do Open vSwitch, estando disponíveis pacotes para Ubuntu, Debian e Fedora. A próxima versão de Open vSwitch também será compatível com FreeBSD.

A maior parte do código é escrito em C, independente de plataforma e pode ser facilmente portado para outros ambientes.

Os componentes principais do Open vSwitch são:

- ovs-vswitchd - *Daemon* que implementa o *switch* juntamente com o módulo do *kernel* Linux responsável pelo *switching* com base em fluxos.
- ovsdb-server - Servidor de base de dados a quem o ovs-vswitchd faz consultas de forma a obter a configuração.
- ovs-dpctl - Ferramenta para configurar o modulo *switch* do kernel do Linux.
- *Script* e especificações para o RPM do Citrix XenServer e Red Hat Enterprise Linux.
- ovs-vsctl - Funcionalidade para a consulta e atualização da configuração do ovs-vswitchd.
- ovs-appctl - Funcionalidade para enviar comandos aos *daemons* Open vSwitch que estão a executar.

Para além destes componentes, o Open vSwitch disponibiliza outras ferramentas:

- ovs-ofctl - Funcionalidade para a consulta e controlo dos *switches* OpenFlow e *controllers*.
- ovs-pki – Funcionalidade para a criação e gestão da infraestrutura de *public-key* para *switches* OpenFlow.
- patch para o analisador de pacotes tcpdump, permitindo a análise de mensagens OpenFlow.

A partir do momento em que é configurada uma ligação entre o Open vSwitch e um SDN *controller*, a criação de todos os fluxos no vSwitch passa a ser da responsabilidade do *controller*. Periodicamente são trocados mensagens entre eles, pelo que se a conexão entre o switch e o *controller* falhar, no instante em que o switch não receber a terceira mensagem seguida vinda do *controller*, assumirá dois comportamentos distintos dependendo do modo que estiver configurado. No modo standalone, o ovs-vswichd irá assumir a responsabilidade pela criação de fluxo, configurando o *switch* para que faça a comutação dos pacotes com base no MAC-learning. O *ovs-vswitchd*, em background, continuará a tentar conectar-se ao controlador e assim que a conexão for bem restabelecida, interromperá seu comportamento *standalone*. No modo *secure*, que é o outro dos modos, fará com que o ovs-vswitchd não crie novos fluxos em caso da falha da conexão.

À medida que foram sendo disponibilizadas novas especificações de OpenFlow Switch, as versões do Open vSwitch foram incorporando essas especificações, estando disponíveis para *download* (Figura 8).

Open vSwitch	OF1.0	OF1.1	OF1.2	OF1.3	OF1.4	OF1.5
1.9 and earlier	yes	---	---	---	---	---
1.10	yes	---	[*]	[*]	---	---
1.11	yes	---	[*]	[*]	---	---
2.0	yes	[*]	[*]	[*]	---	---
2.1	yes	[*]	[*]	[*]	---	---
2.2	yes	[*]	[*]	[*]	[%]	[*]
2.3	yes	yes	yes	yes	[*]	[*]

[*] Supported, with one or more missing features.
[%] Experimental, unsafe implementation.

Figura 8 - Versão do Open vSwitch vs Especificações de OpenFlow Switch [19]

A última versão estável, Open vSwitch 2.3.0, foi disponibilizada em Agosto de 2014. [20]

2.5. Trabalhos relacionados

As tecnologias utilizadas para elaborar a solução com base na arquitetura proposta têm sido abordadas em trabalhos académicos e na indústria. De forma a aproveitar todas as interfaces *wireless* de dispositivos móveis para melhorar a conetividade à rede, Kok-Kiong Yap instalou uma instância do Open vSwitch num smartphone com interfaces 3G, IEE 802.11 e WiMax. Este artigo tem por objetivo explorar o modo como se pode utilizar todas as tecnologias de redes disponibilizadas e no qual foram apresentados os benefícios e limitações no uso simultâneo das tecnologias *wireless* num telefone com sistema Android[21].

Peter Dely desenvolveu um sistema, denominado de BEST-AP, para estimar a largura de banda em APs em que ocorre sobreposição de cobertura. Foi instalada uma instância do Open vSwitch numa *station*, permitindo que estivesse associada a vários APs e com recurso ao BEST-AP, ia comutando para o AP com menor *overhead* [22].

A Google implementou uma *wide area network* privada, denominada de B4, para ligar os seus *data centers* espalhados pelo mundo. Devido às exigências de trafego, esta empresa recorreu à arquitetura SDN, utlizado o Openflow para controlar os *switches* relativamente simples. [23]

Capítulo 3
Análise de Requisitos

Neste capítulo são discutidos os principais requisitos identificados, para a plataforma a conceber e desenvolver e o algoritmo de decisão para satisfazer esses requisitos

3.1 Requisitos

Foram identificados três requisitos aplicáveis ao tráfego de dados em *Hotspots* WiFi com o objetivo incrementar a qualidade de serviço prestada aos clientes:

- Requisito 1 - Encaminhar o trafego dos utilizadores pelos APs de acordo com o perfil de Cliente acordado com a operadora.
- Requisito 2 - Encaminhar o trafego de dados pelos APs de acordo com os requisitos de QoS das aplicações dos clientes.
- Requisito 3 -Fazer balanceamento de carga pelos APs de forma a maximizar a utilização dos recursos de rede disponíveis.

Para satisfazer o requisito 1, um cliente liga-se a vários APs, ficando as informações das ligações do cliente na máquina O&M. Periodicamente são consultadas as métricas de QoS dos APs por Simple Network Management Protocol (SNMP) pela máquina O&M, que avalia por qual dos APs o trafego de cada cliente, de acordo com o seu perfil, irá fluir. Com esses dados, o SDN *controller* atualiza as entradas dos fluxos dos *switches* OpenFlow, o que permite encaminhar o trafego de determinado cliente pelo AP com melhores métricas de QoS.

Para satisfazer o requisito 2, um cliente liga-se a vários APs, ficando as informações das ligações do cliente na máquina O&M. Periodicamente são consultadas as métricas de QoS dos APs por SNMP pela máquina O&M. Dependendo das características de rede exigidas pela aplicação do cliente, é definido qual o AP mais adequado para essa aplicação. Com esses dados, o SDN *controller* atualiza as entradas dos fluxos dos *switches* OpenFlow, o que permite encaminhar os pacotes dessa aplicação por esse AP.

Para satisfazer o requisito 3, os clientes ligam-se simultaneamente a vários APs. As informações das ligações dos clientes ficam registadas na máquina O&M. Periodicamente são consultadas as métricas de QoS dos APs por SNMP pela máquina O&M. Se é detetado congestão num AP, a máquina O&M distribui trafego pelos outros APs. Com esses dados, o SDN *controller* atualiza as entradas dos fluxos dos *switches* OpenFlow, o que permite distribuir o trafego dos clientes pelos APs.

3.2 Algoritmo

A definição dos fluxos e encaminhamento dos pacotes para os clientes com OVS é da responsabilidade de um algoritmo, que está descrito no algoritmo 1. Entre as linhas 1-41, a função é responsável por distribuir os fluxos relativos a cada cliente com OVS pelos APs a que estão conectados com o objetivo de maximizar os APs de um hostspot. Na linha dois é feito um ciclo para percorrer todos os clientes com OVS. Nas linhas 3-9 é percorrido um array bidemensianal em que cada linha corresponde a um interface do cliente e na qual estão guardado os últimos RTT lidos. Com o IP da interface é obtido o RTT atual daquela interface, linha 5, valor que é colocado na última posição da linha apos todos os RTT se

terem deslocado para a esquerda, eliminando assim o RTT mais antigo. Deste modo a linha contem as leituras dos RTT mais recentes. Na linha 7 é calculada a média dos RTT e colocado num array. Esse array contém as médias de todas as interfaces do cliente com OVS. Nas linhas 11-15 estão flags para sinalizar os fluxos que já foram verificados no ciclo definido nas linhas 17-39. Na linha 16 são ordenados os índices numa lógica do maior para o menor RTT e são colocados num array auxiliar, permitindo com o valor desse índice obter o

Algoritmo 1 Distribuir os fluxos das stations com OVS pelos APs a que estão ligados e/ou transferir determinado fluxo para um AP

```
1:  if (update_station==1)
2:      for each station in stationsList do
3:          for each row in arrayAVGrttStation do
4:              IPAddress <- getIPAddressStation(index)
5:              LastIPrtt <- getAtualRTT(IPAddress)
6:              setLastPositionRow(LastIPrtt)
7:              AVGrttInterface<- calcAVGrtt(row)
8:              add_Average_rtt_Interface_To_List(arrayAVGrttInterface)
9:              index=index+1
10:         end for
11:         Voice_flows_flag
12:         IntVideo_flows_flag
13:         StreamVideo_flows_flag
14:         Transdata_flows_flag
15:         bulkdata_flows_flag
16:         order_AVGrtt_decrease_by_index(auxArray,arrayAVGrttInterface)
17:             for i in range (0,len(arrayAVGrttInterface)-1)
18:                 if (Voice_flows_flag==0)
19:                   if((arrayAVGrttInterface.get_value_of_avg_RTT(i)/arrayAVGrttInterface.get_value_of_avg_RTT(i+1))> beta))
20:                       Voice_flows_add(stationIP[auxArray.get_value(i)],stationMACaddr[auxArray.get_value(i)])
21:                     Voice_flows_flag=Voice_flows_flag+1
22:                 elif  (IntVideo_flows_flag==0)
23:                     if(arrayAVGrttInterface.get_value_of_avg_RTT(i)/arrayAVGrttInterface.get_value_of_avg_RTT(i+1))> beta)
24:                         IntVideo_flows_add(stationIP[auxArray.get_value(i)],stationMACaddr[auxArray.get_value(i)])
25:                         IntVideo_flows_flag=IntVideo_flows_flag+1
26:                     elif (StreamVideo_flows_flows_flag==0)
27:                         if(arrayAVGrttInterface.get_value_of_avg_RTT(i)/arrayAVGrttInterface.get_value_of_avg_RTT(i+1))> beta)
28:                             StreamVideo_flows_add(stationIP[auxArray.get_value(i)],stationMACaddr[auxArray.get_value(i)])
29:                             StreamVideo_flows_flag=StreamVideo_flows_flag+1
30:                         elif (Transdata_flows_flag==0)
31:                             if(arrayAVGrttInterface.get_value_of_avg_RTT(i)/arrayAVGrttInterface.get_value_of_avg_RTT(i+1))> beta)
32:                                 Transdata_flows_add(stationIP[auxArray.get_value(i)],stationMACaddr[auxArray.get_value(i)])
33:                                 Transdata_flows_flag=Transdata_flows_flag+1
34:                             elif (bulkdata_flows_flag==0)
35:                                 if(arrayAVGrttInterface.get_value_of_avg_RTT(i)/arrayAVGrttInterface.get_value_of_avg_RTT(i+1))> beta)
36:                                     bulkdata_flows_add(stationIP[auxArray.get_value(i)],stationMACaddr[auxArray.get_value(i)])
37:                                     bulkdata_flows_flag=bulkdata_flows_flag_flag+1
38:                 end if
39:             end for
40:     end for
41: end if
42: if (reserve_AP==1)
43:         arrayIPstationAPlist
44:         arrayMACstationAPlist
45:         for i in range (0,len(arrayIPstationAPlist))
46:             definenewflow(tos,arrayIPstationAPlist[i],arrayMACstationAPlist[1])
47: end if
```

RTT medio e a interface a que corresponde aquele índice. Na função definida entre as linhas 17-39, são comparados todos os RTT das interfaces. Se a diferença entre os RTT das duas

interfaces for superior a beta, então o tipo de fluxo imediatamente a seguir ao último que foi verificado é direcionado para a interface correspondente do cliente OVS. Esta operação de verificação repete-se até atingir os últimos dois elementos do array auxiliar. Estes processos estão em loop.
A função definidada nas linhas 42-47 tem por objetivo encaminhar um tipo específico de aplicações por um determinado AP.

Capítulo 4
Arquitetura Proposta

Para fazer a gestão do tráfego de dados nos hotspots WiFi, a solução apresentada recorre a uma arquitetura SDN baseada em OpenFlow. A solução tem de incluir dispositivos de rede que suportem o protocolo Openflow e SDN controllers. Para além disso, a solução tem de ser compatível com a estrutura já existente ou minimizar o custo de integração nessa mesma estrutura, sob pena ser inviável. A solução de arquitetura que contem apenas os elementos-chave consiste (Figura 9):

- Cliente - Com múltiplas interfaces físicas de rede e uma interface virtual, no qual está a ser executado uma instância do Open vSwitch. Todas as interfaces são consideradas portas para o Open vSwitch. É necessário uma interface virtual com IP para que as aplicações possam comunicar de forma transparente, evitando o descarte de pacotes devido à IPs distintos. Este cliente está ligado ao SDN *controller*, para que a tabela de fluxos do Open vSwitch possa ser atualizada.
- Access Points – Acessos à rede WiFi, aos quais as interfaces físicas de rede do cliente estão ligadas, na relação de 1:1. Os APs estão ligados a um WiFi *controller* e enviam informação acerca do seu estado de funcionamento a uma unidade de O&G.
- Unidade de O&G - É responsável pela recolha da informação dos APs e o respetivo tratamento. Nesta unidade estão definidas as regras e comportamento da rede de acordo com as políticas da empresa. Nesta unidade são tomadas as decisões de alterar a rede que são transmitidas ao Gestor Multiflow.
- Gestor Multiflow - É responsável por converter as decisões recebidas da unidade de O&G, e comunicar as definições de tráfego ao SDN *controller* via APIs.
- SDN Controller - Unidade responsável pela gestão do trafego da rede. Está ligado aos *switches* openflow e ao controlador Multiflow.
- WiFi Controller - Unidade responsável por encaminhar o trafego de dados, de e para os APs, com base na camada 3. Está ligado aos APs e ao *switch* OpenFlow.
- Switch OpenFlow - Unidade responsável pela ligação do *hotspot* ao core da Internet. Os caminhos dos dados até ao cliente são definidos neste *switch*. Está ligado ao SDN *Controller*, WiFi *Controller* e *core*.

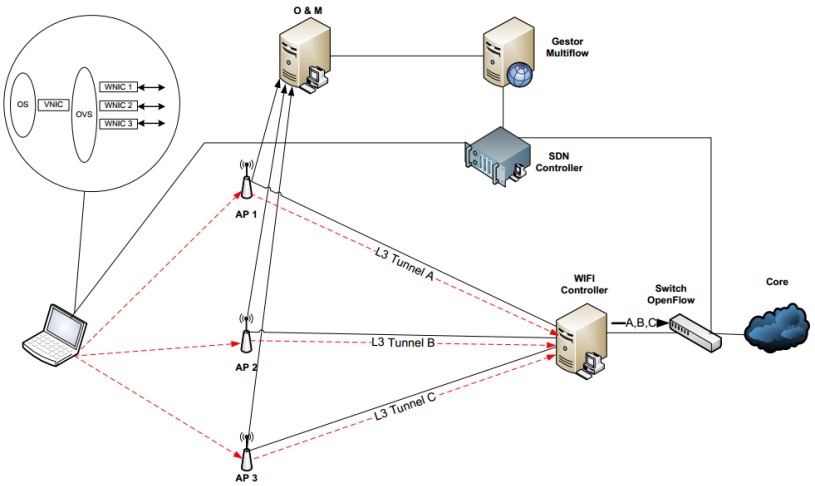

Figura 9 - Arquitetura com interfaces IEEE 802.11

Numa breve descrição, um cliente tem varias interfaces físicas, em que cada uma se liga a um único AP. Cada interface física e virtual fica com um IP único. A unidade de O&M recebe informação dos APs e com base nessa informação e políticas da empresa, define as regras para o trafego de dados para esse cliente e transmite essas regras ao controlador Multiflow. O gestor Multiflow converte essas regras, com o recurso a APIs, num formato que o SDN *controller* possa processar. O SDN *controller* envia esses fluxos para as tabelas que constam nos *switches* OpenFlow.
Esta arquitetura permite escolher por qual dos APs o tráfego de dados, do cliente em direção ao core, irá fluir bem como no sentido inverso, mesmo que a interface física de entrada e saída no cliente não seja a mesma. Todo o trabalho de desenvolvimento e implementação deste projeto teve por base esta arquitetura.

Após a apresentação desta arquitetura, foi levantada a questão de como teria de ser alterada esta arquitetura de forma a tirar partido das interfaces físicas IEEE 802.11 e interfaces 3/4 G do cliente. A solução passa por uma nova arquitetura (Figura 10) em que a gestão do tráfego na rede 3/4 G tem de ser controlado pelo *switch* OpenFlow, que assim gere todo o trafego de dados quer da rede WiFi que da rede 3/4 G. Neste projeto não foi desenvolvido trabalho com base nesta arquitetura.

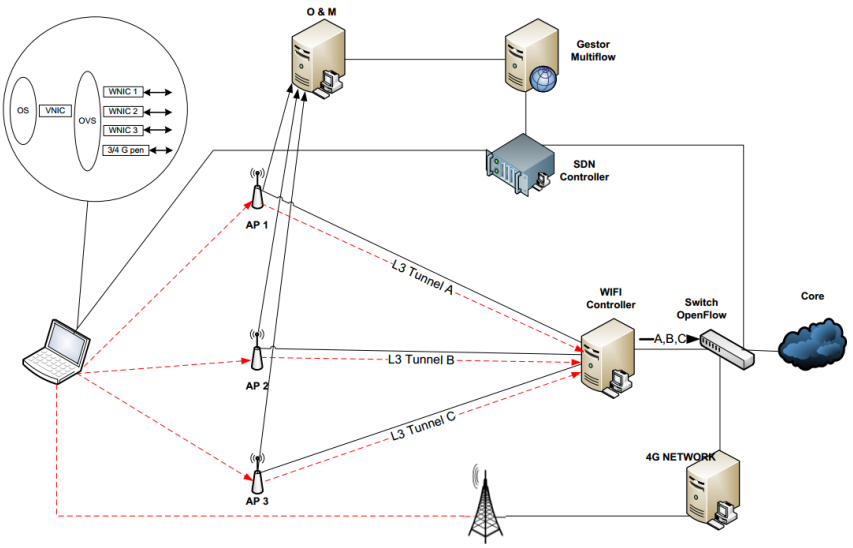

Figura 9 - Arquitetura com interfaces IEEE 802.11 e 3/4 G

Capítulo 5
Trabalho de implementação e validação

Neste capítulo é apresentado o trabalho de implementação e validação que foi efetuado. Em primeiro lugar é descrito o testbed de referência que foi instalado no Laboratório – que reproduz um *hotspot* no qual se integra a plataforma proposta. De seguida é descrito o trabalho de integração e validação prévia que foi conduzido ainda antes do desenvolvimento da plataforma propriamente dito, que é abordado na Secção 5.3.

5.1. Testbeds

A construção do testbed que refletisse a arquitetura proposta neste projeto (ver Figura 8) foi feita de forma incremental, de forma a identificar e resolver eventuais problemas que pudessem surgir. O *testbed* 1 (Figura 10) teve como intuito verificar se o Open vSwitch no cliente permitiria fazer a gestão das suas interfaces IEEE 802.3, focando-se no *end-to-end* e estruturado para a execução de testes funcionais. É constituído por um cliente que tem duas interfaces físicas IEEE 802.3 e no qual foi instalado uma instância do Open vSwitch para fazer a gestão do trafego de entrada e saída do cliente. Foi adicionada uma interface virtual ao Open vSwitch, à qual foi atribuído um IP para que as aplicações do cliente poderem comunicar com o exterior de forma transparente. O cliente está ligado a dois *switches* domésticos nos quais não foi efetuada nenhuma configuração. Por sua vez, esses *switches* estão ligados a uma máquina com OS linux que se comporta como um *router*. Esta máquina está ligada a uma outra tem uma instância do Open vSwitch instalada de forma a comportar-se como um *switch*. Por fim este *switch* está ligado a uma máquina que tem como função simular a *internet*.

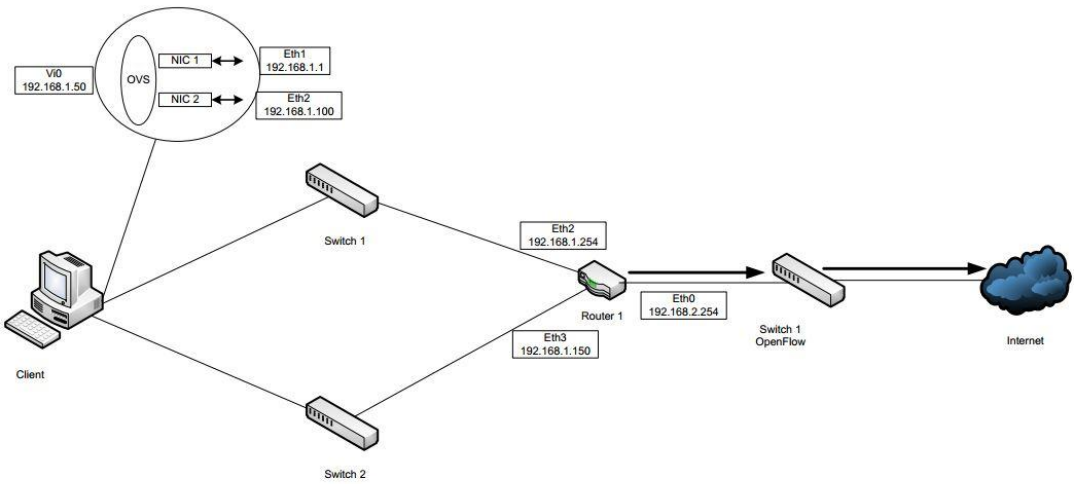

Figura 10 - *Testbed* 1: cliente com interfaces IEEE 802.3

No *testbed* 2 (Figura 11), a ligação do *client* à rede passou a ser da responsabilidade de duas interfaces IEEE 802.11. Os Switch 1, Switch 2 foram substituídos por APs da Cisco. A atribuição de endereços IP às interfaces do cliente passou a ser dinâmica, com recurso ao protocolo *Dynamic Host Configuration Protocol* (DHCP). O DHCP é um protocolo que oferece a configuração dinâmica de terminais, com a atribuição de endereços IP de host, máscara de sub-rede e *Default Gateway*, pelo que foi instalado o serviço DHCP no router 1 para a atribuição de endereço IP às interfaces do *client*.

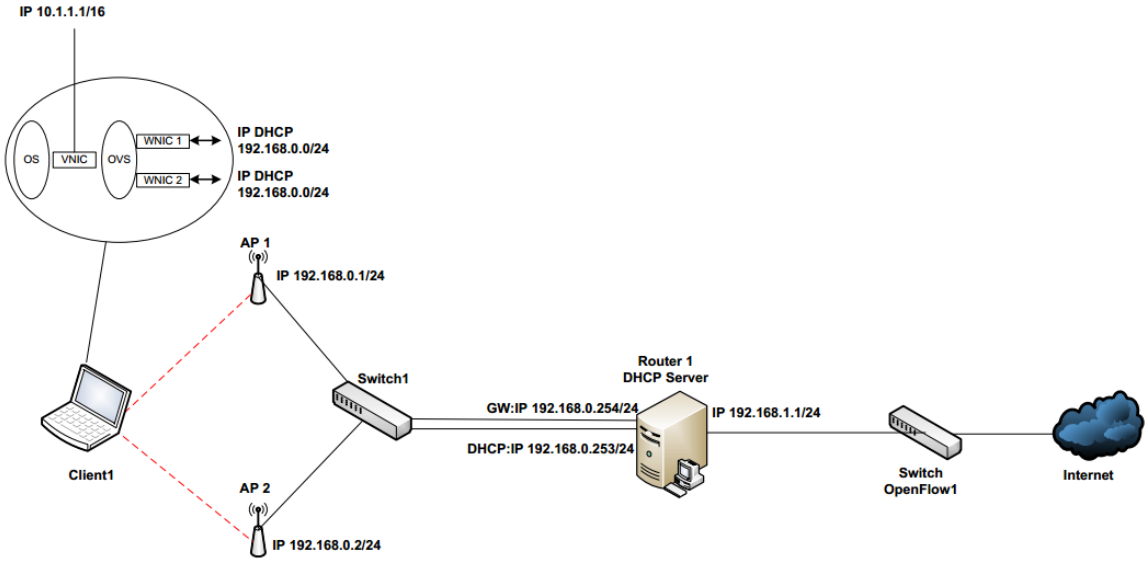

Figura 11 - *Testbed* 2: cliente com interfaces IEEE 802.11

No *testbed* 3 (Figura 12) foi adicionado um cliente com uma interface IEEE 802.11, que se ligou a um dos AP com a atribuição de endereço IP à sua interface via DHCP. Foram adicionados um SDN *controller* e um Gestor Multiflow cujas funções estão descritas no capítulo anterior.

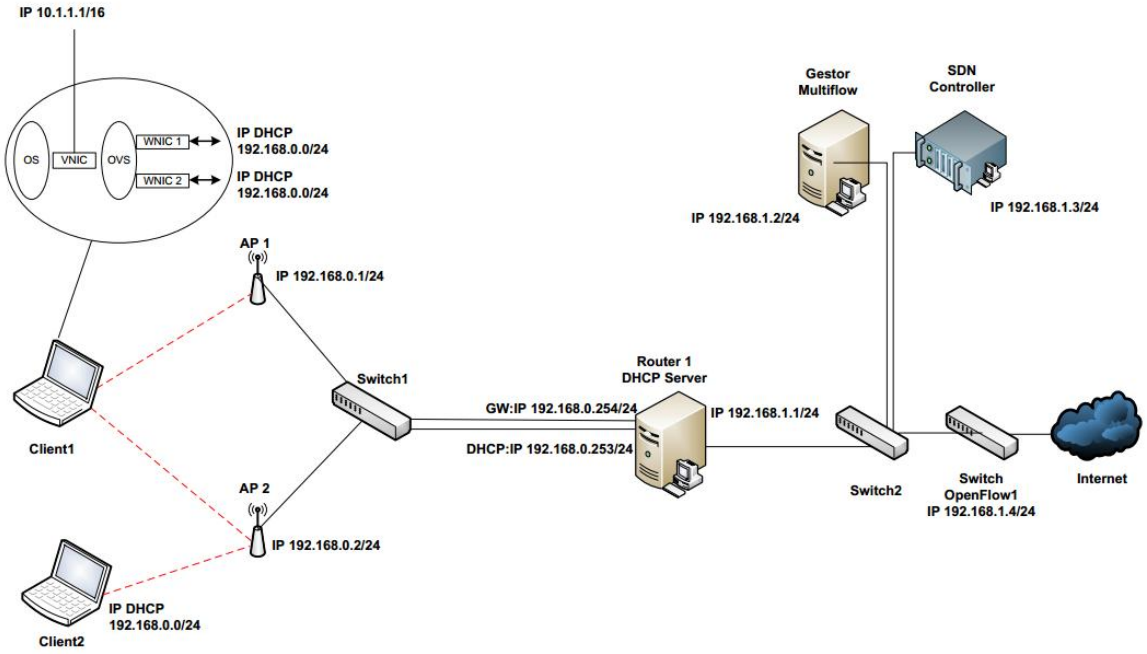

Figura 12 - *Testbed* 3: Topologia da rede para testar a eficacia da solução

5.2. Trabalho de integração e validação prévia

Para construir o primeiro testbed, ligaram-se os vários componentes por cabos. As máquinas do cliente, router 1, switch 1 OpenFlow e Internet foram configuradas da seguinte forma:

- Máquina Cliente - Tem como sistema operativo o CentOS 6.3, no qual foi instalado a versão 1.9.0 do Open vSwitch. Foi criada uma interface virtual e

atribuído um IP estático, tendo sido definida uma *default route* para esta interface. Nas interfaces físicas foram atribuídos IPs estáticos. Os IPs pertencem à mesma rede. Todas as interfaces foram adicionadas ao Open vSwitch, que passaram a ser tratadas como portas de um *switch*.

- Router 1 - Tem como sistema operativo o Fedora 16. Tem três interfaces físicas com IPs estáticos. Os endereços IPs das duas interfaces físicas estão na gama de endereços da LAN em que se encontra o cliente. O endereço IP da terceira interface física está na gama definida para a rede entre o router e a Internet.
- Switch 1 OpenFlow - Tem como sistema operativo o CentOS 6.3, no qual foi instalado a versão 1.9.0 do Open vSwitch. As duas interfaces físicas foram adicionadas ao Open vSwitch, que passaram a ser tratadas como portas de um *switch*.
- Internet – Foi configurado um IP estático na sua interface física na gama definida para a rede entre o *router* e a Internet.

No primeiro testbed, a atribuição de IPs estáticos às interfaces físicas, virtual do *Client* e às interfaces do *router*, todos pertencentes à mesma rede devido ao protoloco Address Resolution Protocol (ARP). Ao definir o IP estáticos, são enviados pacotes ARP pelo *Client* para proceder ao preenchimento da sua tabela de ARP. Se os IPs estáticos e o IP virtual não estiverem na mesma gama, são necessários fluxos específicos para este tipo de protocolo de forma a alterar algumas dos campos do pacote de ARP. Para esta fase inicial de testes, optou-se por ainda não definir estes fluxos.

No terminal *Client* fez-se um teste de conetividade com a máquina *Internet* através de um *echo request*, ao qual foi respondido, com sucesso, com um *echo reply*. Com o analisador de pacotes tcpdump a executar nas interfaces do *client* e na interface da máquina *Internet*, foi possível analisar os IPs de origem e destino dos pacotes. Os pacotes que chegam à máquina *Internet* vindos da máquina *Client* têm como IP de origem o IP da interface virtual da mesma. Os pacotes que chegam à máquina *Client* vindos da máquina *Internet* têm como IP de origem o IP da interface física da máquina *Internet*. As implementações do Open vSwitch comportam-se como *switches*, garantindo a conectividade *end-to-end*.

Após construir o segundo *testbed*, os endereços IP das interfaces IEEE 802.11 dos terminais que se ligassem à rede a qual os APs estão associados passaram a ser atribuídos de forma dinâmica. A ligação dos terminais à rede passou de cablada para sem fios, transição essa que obrigou a que no *Client* fosse necessário definir fluxos para manipular os pacotes ARP que continham dados da interface virtual, cuja existência é desconhecida para lá do *Client*. Para criar esses fluxos foi necessário substituir no *Client* a versão 1.9.0 do Open vSwitch por outra que suportasse a especificação 1.3 do OpenFlow, de forma que foi instalada a versão 2.1.90. Efetuaram-se com sucesso os testes de conetividade entre o *Client* e a máquina *Internet*. No *Router1* utilizaram-se adaptadores usb/ethernet que criaram conflitos com o sistema operativo existente, optando-se por instalar o sistema operativo Linux Debian.

Na construção do *testbed* 3 foram adicionadas as restantes máquinas. Na tabela 2 estão resumidos as características do *harware*, o *software* e as respetivas versão que são relevantes para o funcionamento do *testbed*.

Tabela 2 – *Hardware* e *software* utilizado na montagem do *testbed* 3

Função	Hardware	Sistema Operativo	Software
Client1	Portatil asus EEE 1001-PX-H	CentOS 6.5	Open vSwitch Versão 2.1.90
Client2	Portatil asus EEE 1001-PX-H	CentOS 6.5	
Ap1	Aironet Cisco 1100 series	------	
Ap2	Aironet Cisco 1130 series	------	
Router / Servidor DHCP	Portatil Acer Aspire 1642 LMi	Debian 7.0	
Switch 1 / Switch 2	LevelOne com 5 portas Fast Ethernet	------	
Gestor Multiflow	Server Pentium Dual Intel	CentOS 6.5	
SDN Controller	Server Pentium 4 Intel	CentOS 6.5	OpenDaylight Hydrogen Base Edition V1
Switch 1 Openflow	Server Quad Core Intel Q9400	CentOS 6.5	Open vSwitch Versão 2.1.90
Internet	Desktop Acer Aspire M3640 Quad Core Intel Q6600	CentOS 6.5	

5.3. Desenvolvimento

O *tesbed 1*, sem configuração de fluxos, permite a conetividade entre os extremos, pelo que se avançou para a fase de testes. Com o primeiro teste pretendeu-se que o tráfego que chega aos *switches* OpenFlow seja classificado em fluxos aos quais são aplicadas as respetivas ações. As máquinas do cliente, router 1, switch 1 OpenFlow e Internet foram configuradas da seguinte forma:

- Máquina Cliente – Foram definidos dois fluxos de trafego e as respetivas ações:
    - Aos pacotes que entram pela interface virtual, o campo IP origem e o campo MAC address de origem são alterados para os valores correspondentes da interface Eth1. Os pacotes modificados saem pela porta da interface Eth1.
    - Aos pacotes que entram pela interface Eth2, o campo IP de destino e o campo MAC address de destino são alterados para os valores da interface virtual correspondentes. Os pacotes modificados saem pela porta da interface virtual.
- Router 1 – Foi definida um rota estática para cada um dos IPs das interfaces físicas da máquina cliente.

- Switch 1 OpenFlow - Foi definido um fluxo de trafego e as respetivas ações:
    - Aos pacotes que entram pela interface física ligada à máquina Internet e cujo IP destino seja a interface eth1 da máquina cliente, o campo IP destino é alterado para o valor da interface Eth2 da máquina cliente. Os pacotes modificados saem pela porta da interface física ligada ao *router* 1.

No terminal do cliente repetiu-se o teste de conetividade com a máquina Internet através de mensagens *echo request* do protocolo Internet Control Message Protocol (ICMP), ao qual foi respondido, com sucesso, com mensagens *echo reply do* ICMP (Figura 13). Com o analisador de pacotes tcpdump a executar nas interfaces do cliente e na interface da máquina Internet, foi possível analisar os IPs de origem e destino dos pacotes. Os pacotes que chegam à máquina Internet vindos da máquina Cliente têm como IP origem o IP da interface física eth1 da máquina Cliente. A máquina Internet faz o echo reply preenchendo o campo IP destino com o IP da interface física eth1 do cliente. No switch 1 OpenFlow, o campo IP destino do pacote é alterado para o valor da interface física eth2. O router encaminha o pacote pela interface física que tem ligação com a interface física eth2 da máquina cliente. Os pacotes que chegam à máquina Cliente na interface física eth2 são modificados nos valores de IP e MAC address destino para os valores da interface virtual e enviados para a porta da interface virtual. No terminal do cliente, os pacotes têm como IP origem o IP da interface física da máquina Internet. Com este teste constatou-se que encaminhado um pacote de resposta para uma interface física da máquina cliente que não a mesma por onde saiu o pedido, não houve descarte do pacote quando chegou à aplicação do cliente. O *router* 1 e a máquina Internet não têm conhecimento do IP virtual da máquina Client.

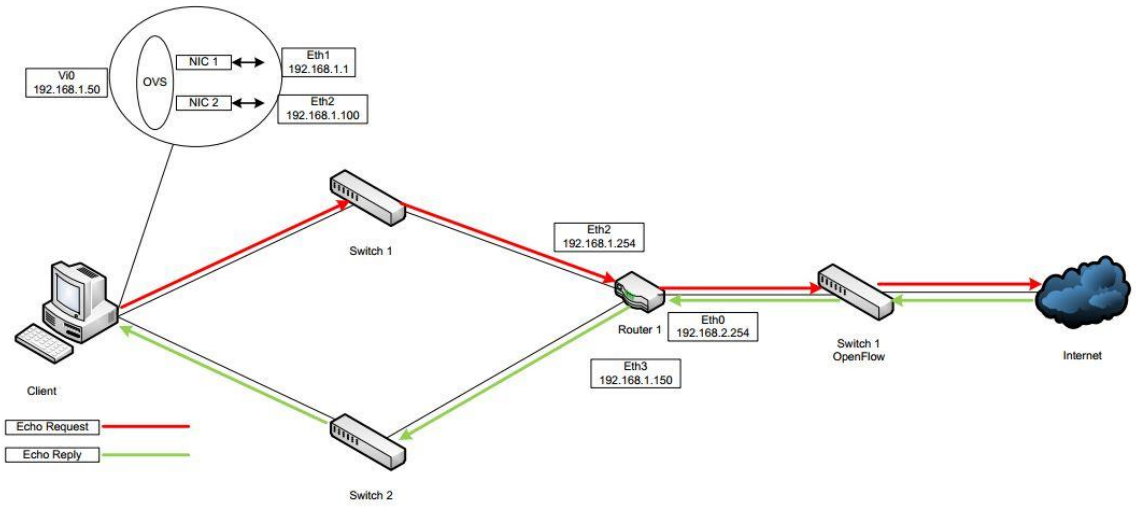

Figura 13 - Esquema do teste com manipulação dos IPs dos pacotes.

Para a definição do algoritmo que será responsável pelos critérios utilizados na elaboração dos fluxos, pesquisaram-se soluções para obter informação relevante tendo como requisito principal a não intervenção do cliente. As soluções encontradas são métricas de QoS dos APs obtido por SNMP, o round-trip time desde as portas do openvswitch do cliente ao Gestor Multiflow de forma a calcular a latência, as estatísticas das portas do openvswitch do cliente, recolhidas pelo SDN Controller e acedidas com northbound APIs pelo Gestor Multiflow.

O round-trip time (RTT), também chamado de round-trip delay time, é o tempo necessário para um pacote ou um impulso de sinal percorrer a rede de uma fonte específica para um

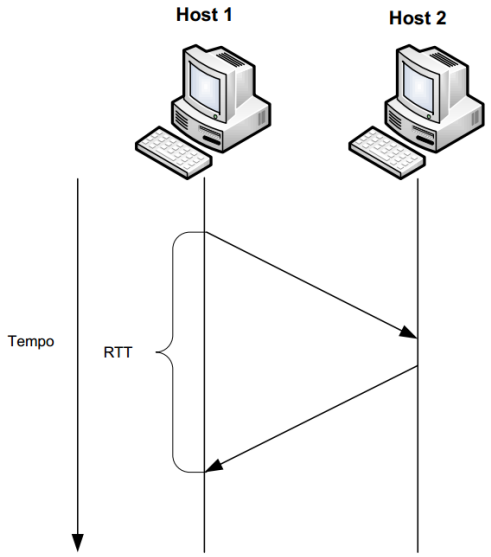

Figura 14 - round-trip time

destino especifico e regressar à fonte de origem (Figura 14). Um dos protocolos utilizados é o ICMP (RFC 792). Este protocolo permite gerir as informações relativas aos erros nas máquinas conectadas. Essas informações são obtidas através de mensagens ICMP encapsuladas em datagramas IP.
O SNMP é um protocolo da camada de aplicação desenvolvido para transportar informações de gestão de rede entre os dispositivos que implementem o SNMP e os sistemas de gestão de redes, permitindo que administradores de rede controlem o desempenho da uma rede monitorizando interfaces, processadores, memórias de equipamentos como *routers*, *switches*, dispositivos wireless e servidores. A primeira versão do SNMP foi adotada como padrão em 1989 e quatro anos depois teve uma atualização para a versão 2. As versões 1 e 2 do SNMP não são seguras, pelo que foi criado o SNMPv3 (versão 3), para solucionar as questões de segurança, fornecendo acesso seguro às informações de gestão por meio de autenticação e criptografia de pacotes.

A escolha da solução que permitisse avaliar o estado da rede centrou-se em dois aspetos, *overhead* na rede e confiabilidade dos valores para caracterizar o estado da rede. Na tabela 3 estão descritas as características de cada uma das soluções. As politicas de QoS nos AP da Cisco não vêm definidas de fabrica, pelo que dependem do operador, pelo que esta solução foi descartada. O RTT é rápido de implementar, é utilizado para avaliar a qualidade da rede e não acrescenta mais *overhead* do que a informação das portas do Open vSwitch no cliente, o que levou a ter sido escolhido.

Tabela 3 - Comparativo das soluções de recolha dados para definição dos fluxos

Tipo de Solução	Forma de Recolha	Alvo	Tipo de informação	*Overhead* na rede (Bytes)
Metricas de QoS	Simple Network Management Protocol (SNMP)	Access Point (AP)	Nº de falhas em frames por classe (Voice, Video, Background, BestEfford) Nº de frames descartadas por classe (Voice, Video, Background, BestEfford)	(484+20+26) x 2 x 2 x 4 Classes
Round-Trip Time	Internet Control Message Protocol (ICMP)	Interfaces IEEE 802.11 do cliente	Tempo que decorre entre o envio de um pacote a um host e a receção da resposta do mesmo.	(40+20+26) x 2 = 172 x ∑ (n Clientes x m Interfaces)
Dados das portas do openvswitch no cliente	Nortbound APIs do SDN Controller	Portas do openvswitch do cliente	receiveDrops transmitDrops receiveErrors transmitErrors receiveFrameError receiveOverRunError	(64 bytes pedido + 96 bytes dos contadores) x ∑ (n Clientes x m Interfaces)

Depois de escolhido o método de recolha de informação da rede que servirá de base para definir os caminhos que os pacotes irão percorrer na rede, construiu-se uma aplicação no Gestor MultiFlow, que com base nos RTT ao *client* com OVS, define fluxos que transmite ao SDN controller via NBI.

Capítulo 6
Validação e Avaliação

Neste capítulo discute-se o trabalho que foi efetuado para validar e avaliar a plataforma desenvolvida.

Para avaliar a eficácia do algoritmo implementado no gestor Multiflow, pesquisaram-se ferramentas de rede que permitissem recolher métricas resultantes da análise do trafego.

O D-ITG é uma plataforma que permite gerar tráfego IPv4 e IPv6, ao nível das camadas de rede, transporte e aplicacional. É capaz de replicar as propriedades estatísticas do tráfego de diferentes aplicações bem conhecidas: Telnet, Voice over IP - G.711, G.723, G.729, deteção de Atividade de Voz, *Compressed* RTP – DNS e jogos em rede (*Counter Strike, Quake 3*). Relativamente à camada de transporte, o D-ITG atualmente suporta TCP (Transmission Control Protocol), UDP (User Datagram Protocol), SCTP (Stream Transmission Control Protocol) e DCCP (Datagram Congestion Control Protocol). Ao nível da camada de rede é possível definir o TOS (DS) do cabeçalho IP. Em simultâneo funciona como ferramenta de medição das métricas mais comuns para avaliar o desempenho de redes: taxa de transferência, atraso, *jitter*, perda de pacotes.

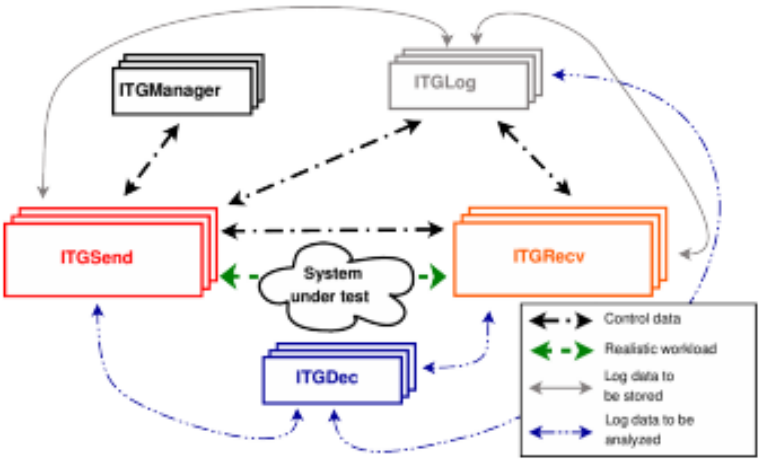

Figura 15 - Componentes da plataforma D-ITG.[24]

Os componentes principais do D-ITG são o ITGSend e ITGRecv. O ITGSend é o componente responsável pela geração de tráfego com destino ao ITGRecv (Figura 15). Por suportar multithreadeds, o ITGSend pode enviar vários fluxos de tráfego paralelos para várias instâncias do ITGRecv. O ITGRecv é o componente responsável por receber os fluxos de tráfego paralelos de uma ou mais instâncias de ITGSend. O ITGSend e ITGRecv podem criar ficheiros de log contendo informações detalhadas sobre todos os pacotes enviados recebidos [24].

O Iperf é uma ferramenta simples e muito eficiente para medir a largura de banda e a qualidade de um determinado link de rede (Figura 16). Permite obter dados relativos à largura de banda, *jitter* e perda de datagramas [25]. Precisa de ser executado tanto no cliente como no servidor. É compatível com sistemas operativos Unix/Linux e Windows.

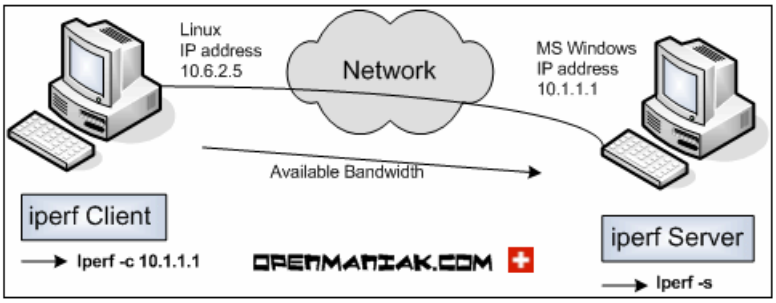

Figura 16 - Exemplo de um teste à rede com Iperf [26]

Com o recurso a funcionalidades do D-ITG, definiu-se um conjunto de três fluxos para serem enviados da máquina Internet para cada um dos *clients*, de acordo com o *testbed3*. O fluxo nº1 simula o tráfego de uma chamada VoIP. O fluxo 2 simula o tráfego de um jogo online e o fluxo 3 tem como função provocar a perda de pacotes e assim levar a que o Gestor Multiflow tome decisões acerca do modo como os pacotes do cliente OVS são encaminhados. Os fluxos estão descritos na tabela 4. Para as decisões tenham o efeito

Tabela 4 - Características do tráfego enviado para os *clients*

Nº de Fluxo	Tipo de trafego	Tempo de Duração (s)	Bandwidth (kb/s)	Packet size (Bytes)	ToS (dec)
1	VoIP (Codec G.711)	300	96.1	120	184
2	Jogo Interativo (Quake 3)	300	107	93	144
3	Trafego UDP	300	400	540	56

pretendido, o Gestor Multiflow tem de conseguir diferenciar os fluxos e assim definir rotas para cada um deles. Um campo que pode ser utilizado para identificar cada tipo de fluxo é o Type of Service (ToS).

O ToS é um campo de 8 *bits* do cabeçalho do datagrama IP (Figura 17).

Version (4 bits)	IHL (4 bits)	Type of Service (8 bits)	Total Length (16 bits)	
Identification (16 bits)			Flags (3 bits)	Fragment Offset (13 bits)
Time to Live (8 bits)		Protocol (8 bits)	Header Checksum (16 bits)	
Source Address (32 bits)				
Destination Address (32 bits)				
Options and Padding (multiples of 32 bits)				

Figura 17 - Cabeçalho do datagrama IP [27]

Este campo faz parte do cabeçalho IP desde o início, mas raramente foi utilizado até à introdução de serviços diferenciados (Diff-Serv). O ToS foi descrito na RFC 791 (Internet Protocol, em setembro de 1981). De seguida, foi especificado na RFC 1349 (Type of Service in the Internet Protocol, Julho de 1992). Este campo foi definido com duas partes, um valor de precedência e os ToS bits (Figura 18). O valor de precedência ocupa os 3 bits mais à

esquerda e foi concebido para fornecer uma forma de prioridade filas. Os ToS bits definem como a rede deve fazer trade-offs entre o throughput, atraso, confiabilidade e custo. [28]

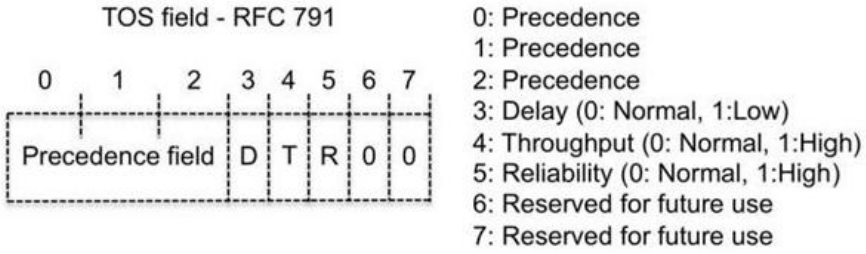

Figura 18 - Bits do campo Type of Service.[29]

Ambos os APs da Cisco suportam velocidades de 54 Mb/s nas interfaces IEEE 802.11, pelo que um deles teve a velocidade limitada a 2 Mb/s para simular um AP congestionado. Nos primeiros ensaios utilizou-se o D-ITG na máquina internet para gerar tráfego com destino aos *clients*. Nas configurações do D-ITG nos *clients* ativou-se a opção de gerar os logs com as métricas dos fluxos. No *client* sem OVS, esse ficheiro de log continha as métricas da rede com valores, no *client* com OVS, essas mesmas métricas estavam vazias. Utilizou-se o analisador de rede **Wireshark** a capturar pacotes nas interfaces dos *clients* de modo tentar identificar os pacotes que estariam a ser descartados pelo *client* com OVS. Não foi detetada diferença nos pacotes que chegavam às interfaces, pelo que se optou por utilizar o Iperf e os fluxos com as características descritas na tabela 4. Os relógios dos *clients* e da máquina Internet foram sincronizados entre eles.

A elaboração do plano de ensaios, descrito na tabela 5, teve por objetivos verificar qual o efeito do *overhead* dos RTT entre o Gestor Multiflow e o *client* OVS na rede, mais concretamente nos *clients* e que vantagens se obteriam na utilização de SDN.

No ensaio nº 1, o *client* sem OVS está ligado ao AP com 2 Mb/s e o *client* com OVS está ligado aos dois APs. O *client* com OVS recebe tráfego apenas do AP com 2 Mb/s, o que vai provocar congestionamento e perda de pacotes. Este ensaio é a base de referência para os ensaios seguintes e permite determinar se há implicações na performance ao instalar o OpenvSwitch num cliente.

No ensaio nº 2, para além das condições descritas no ensaio nº1, o Gestor Multiflow começa a enviar mensagens ICMP a todas as interfaces do *client* com OVS com intervalos de tempo de 10 segundos. Com este ensaio pretende-se avaliar o impacto que o RTT terá na rede, mais especificamente nos *clients*.

No ensaio nº 3, para além das condições descritas no ensaio nº2, o Gestor Multiflow começa a fazer a gestão dos fluxos com base no RTT das interfaces do client com OVS. A partir deste instante o *client* com OVS pode receber pacotes pelas duas interfaces, condição que depende das decisões do Gestor Multiflow. Com este ensaio pretende-se avaliar a eficácia da solução proposta.

Tabela 5 – Planos dos ensaios a realizar no *testbed 3*

Nº de ensaio	Condições do ensaio	Métricas a recolher
1	- 1 Client ligado ao AP com 2 Mb/s - 1 Client com OVS ligado aos dois AP com trafego a circular apenas pelo AP no qual está ligado o outro cliente	- *Packet loss* Fluxo/Cliente
2	- 1 Client ligado ao AP com 2 Mb/s - 1 Client com OVS ligado a dois AP - Gestor Multiflow a calcular o RTT de 10 em 10 segundos	
3	- 1 Client ligado ao AP com 2 Mb/s - 1 Client com OVS ligado aos dois AP com trafego a circular apenas pelo AP no qual está ligado o outro cliente - Gestor Multiflow a executar o algoritmo e a calcular o RTT de 10 em 10 segundos	

Cada ensaio foi repetido dez vezes, tendo sido calculado a média e o desvio padrão.

Um dos objetivos do cenário 1 é detetar se existe alguma interferência na performance das interfaces de um equipamento com uma instância do Open vSwitch a executar. Tanto o cliente como o cliente com OVS estabeleceram ligações com o mesmo AP. Do gráfico da figura 19, a primeira evidência é que os desvios-padrão são muito elevados, o sucedeu em todas as medições pelo que essa constatação é abordada no fim deste capítulo. Focando-se nos valores, não é possível relacionar a percentagem dos pacotes perdidos com o facto de um cliente estar a executar uma instância do OVS. O cliente com OVS perdeu maior percentagem nos fluxos Trafego UDP e VoIP e o cliente perdeu uma maior percentagem no fluxo Jogo.

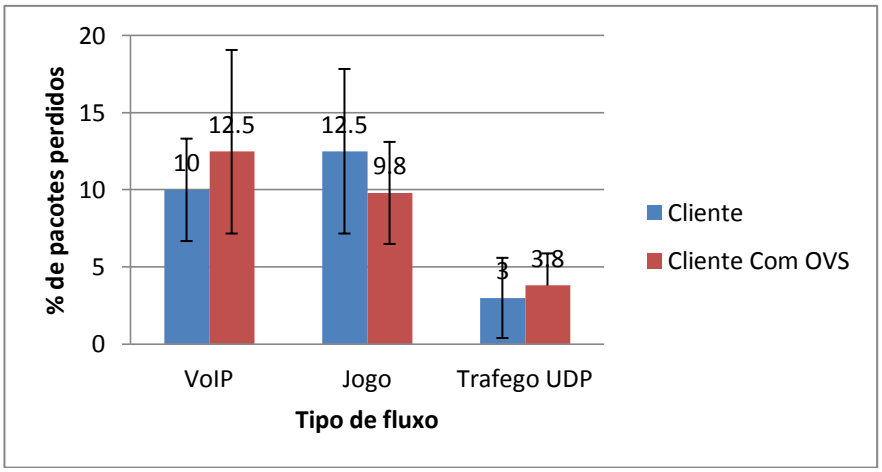

Figura 19 - Percentagem de pacotes perdidos pelo *client* e client com OVS no primeiro ensaio

O número de pacotes perdido no *client* e no *client* com OVS por cada fluxo variou de acordo com o ensaio, como pode ser observado nos gráficos da Figura 20 e Figura 21. No ensaio 2, a quantidade de pacotes a circular no *testbed 3* aumentou devido à execução no Gestor Multiflow de uma aplicação que calculava os RTT, refletindo-se no aumento de pacotes

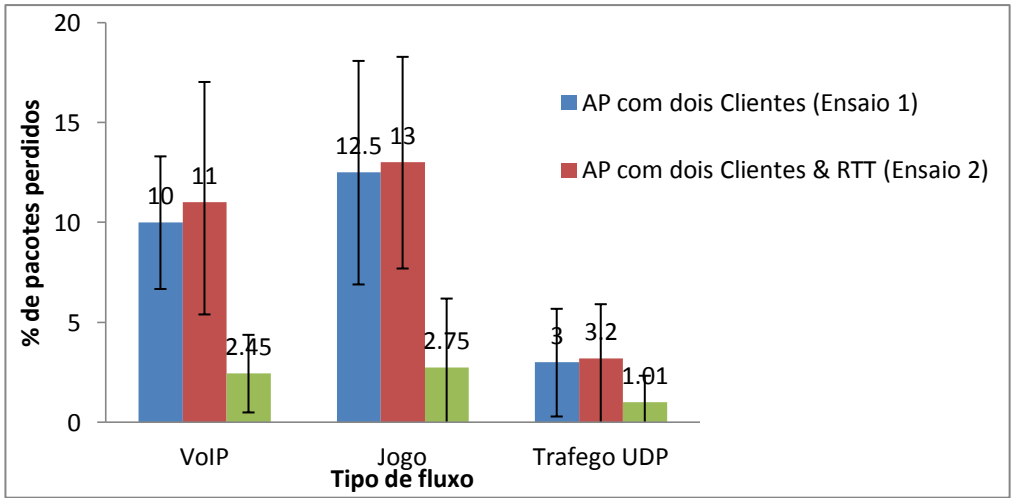

Figura 20 - Percentagem de pacotes perdidos pelo *client* nos três ensaios

perdidos tanto no *client* como no *client* com OVS. No ensaio 3, a perda de pacotes no *client* e no *client* com OVS foi muito menor comparativamente ao ensaio 1 devido à execução no Gestor Multiflow do algoritmo de decisão.

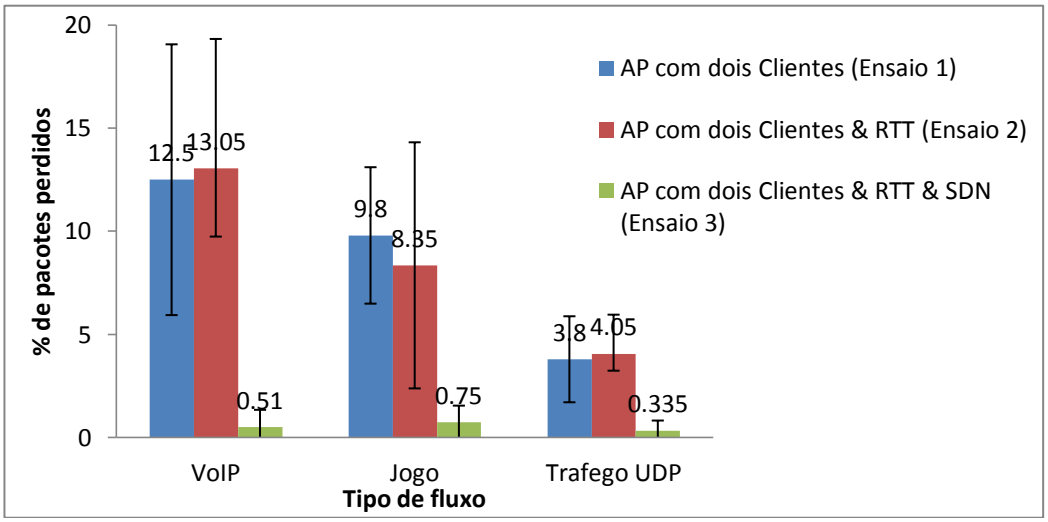

Figura 21- Percentagem de pacotes perdidos pelo *client* com OVS nos três ensaios

No terceiro ensaio, o Gestor Multiflow esteve a executar uma implementação do algoritmo. Como se pode observar na Figura 22, a perda de pacotes no Cliente com OVS é muito inferior quando comparada com a perda de pacotes no Cliente. A perda de pacotes nos fluxos do Cliente com OVS é explicada pelo modo como o Gestor Multiflow toma a decisão de encaminhar os fluxos. Entre execuções do RTT no Gestor Multiflow, decorrem 10 segundos nos quais os fluxos entre a máquina Internet e o Cliente com OVS seguem a rota definida quando foi estabelecido a ligação. O Gestor Multiflow só altera essa rota após avaliar se não haverá outra que ofereça mais garantias. É de notar que também o Cliente beneficiou com a mudança de rota, as perdas diminuíram drasticamente.

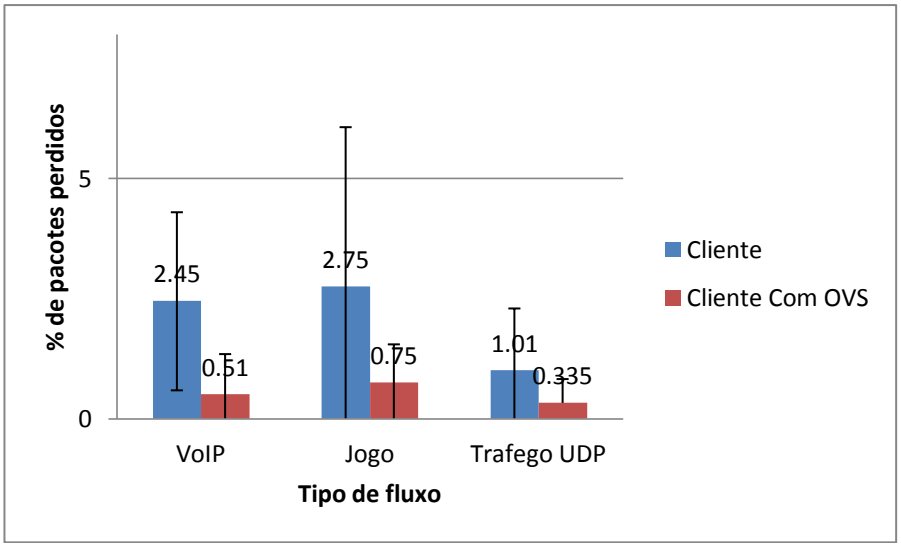

Figura 22 - Percentagem de pacotes perdidos pelo *client* e client com OVS no terceiro ensaio

Uma das causas para o elevado desvio-padrão das medições poderá estar relacionada com a entropia introduzida no sistema. Ao utilizar APs e switches comerciais, um servidor DHCP para atribuir de forma automatizada os endereços IP às interfaces radio dos *clients*, a ligação entre o equipamento com Open vSwitch e o SDN *controller,* leva a que circulem na rede uma grande quantidade de pacotes que condicionam a repetibilidade dos testes.

Capítulo 7
Execução das Atividades Previstas

Neste capítulo é discutida a execução das atividades previstas, em termos de planeamento inicial e desvios ao planeamento definido.

Foram elaborados dois planos de atividade, um para cada semestre.

7.1 Plano de atividades do 1º Semestre

Comparando o trabalho realizado até à data e as etapas definidas para o 1º semestre (Figura 23), pode-se observar que as etapas estão a ser executadas de acordo com o previsto. As tecnologias para a realização do projeto já estão definidas, a arquitetura já foi apresentada, discutida e ajustada com a PT Inovação para ir de encontro aos objetivos pretendidos. Foi criado um *testbed* com o objetivo de confirmar as capacidades das tecnologias escolhidas em satisfazer os requisitos definidos pela PT Inovação.

	2013				
Etapas	Setembro	Outubro	Novembro	Dezembro	Janeiro
- Familiarização com o tema do trabalho, com o projeto e com o estado da arte (7 semanas).					
- Escrita da primeira versão da arquitetura a propor (3 semanas).					
- Discussão e refinamento da arquitetura (3 semanas).					
- Documentação e Escrita do Relatório Intermédio de Estágio (2 semanas).					

Figura 23 - Plano de atividades para o 1º semestre

7.2 Plano de atividades do 2º Semestre

No segundo semestre, O *testbed* evoluiu de forma iterativa, para se aproximar da arquitetura proposta (Figura 24). No *testbed* 1 foram adicionados novos componentes que estão incluídos na proposta de arquitetura, entre os quais APs, que levou à transição da tecnologia IEEE 802.3 para a tecnologia IEEE 802.11 no *client*.

Etapas	2014			
	Março	Abril	Maio	Junho
- Desenvolvimento e integração da arquitetura proposta (7 semanas)				
- Validação dos componentes desenvolvidos e respectiva integração (5 semanas).				
- Documentação final e Escrita do Relatório/Dissertação (3 semanas).				

Figura 24 - Plano de atividades para o 2º semestre

7.3 Constrangimentos

As maiores dificuldades surgiram a partir do momento em que no *client1* se fez a transição da ligação à rede por cabo para *wireless*. Foi necessário adicionar fluxos específicos no Open vSwitch do *client* para encaminhar os pacotes do protocolo ARP. Só as versões do software Open vSwitch que suportem a especificação OF 1.3 é que permitem criar esses fluxos. À data da elaboração do *testbed 1*, a versão Open vSwitch disponibilizada não suportava essa especificação, pelo que após várias tentativas e finalmente o recurso à mailing list levou a instalação de uma versão nova que suportasse a OF 1.3. Após definir as ligações no Open Switch do *client1* e do *switch Openflow1* ao SDN *controller*, constatou-se que ambos os equipamentos constam da página web que é disponibilizada pelo OpenDaylight, no entanto a ligação entre o OpenDaylight e o *client1* é instável, o que impede a gestão de fluxos de forma remota e leva a que no *client1* todos os fluxos sejam eliminados, perdendo a ligação à rede. Tentou-se identificar a causa desta intermitência, sem sucesso, pelo que se optou por, nesta fase, não incluir a gestão Open vSwitch do *client1* via SDN Controller. Devido a estas dificuldades, algumas das etapas prolongaram-se mais do que o previsto, como se pode observar no diagrama de gantts com a execução real do projeto (Figura 25).

	2013					2014					
Etapas	**Setembro**	**Outubro**	**Novembro**	**Dezembro**	**Janeiro**	**Março**	**Abril**	**Maio**	**Junho**	**Julho**	**Agosto**
- Familiarização com o tema do trabalho, com o projeto e com o estado da arte (7 semanas).											
- Escrita da primeira versão da arquitetura a propor (3 semanas).											
- Discussão e refinamento da arquitetura (3 semanas).											
- Documentação e Escrita do Relatório Intermédio de Estágio (2 semanas).											
- Desenvolvimento e integração da arquitetura proposta (9 semanas)											
- Validação dos componentes desenvolvidos e respectiva integração (9 semanas).											
- Documentação final e Escrita do Relatório/Dissertação (3 semanas).											

Figura 25 – Execuçao do projeto

Capítulo 8
Conclusões

Para fazer a gestão dinâmica de trafego de dados nos *hotspots* WiFi da PT de acordo com as politicas e regras da empresa, a arquitetura SDN baseada em OpenFlow apresenta-se como solução porque implica poucas alterações na infraestrutura existente, para alem de introduzir dinamismo na rede. Elaborou-se um *testbed* com base na arquitetura proposta, recorrendo a alguns equipamentos comerciais, o que aproximou este projeto a uma solução aplicável à indústria. Dos resultados obtidos, pode-se concluir que não são apenas os terminais com Open vSwitch instalado que tiram proveito deste tipo de arquitetura, mas todos os outros terminais que estão associados aos APs acabam por beneficiar por haver um balanceamento de carga pelos APs.

8.2 Trabalhos Futuros

Da solução apresentada com base na arquitetura proposta, o SDN *controller* está ligado ao *switch* 1 OpenFlow, o que permite definir por qual das suas interfaces radio um cliente com OVS instalado e ligado a vários APs irá receber pacotes. Como o cliente não está ligado ao SDN *controller,* a definição dos fluxos no cliente é estática, impedindo que o operador possa fazer a gestão da rede do cliente para o exterior do *hotspot.* Para uma gestão mais eficiente dos recursos, implementar a ligação entre os clientes com o SDN controller representa uma mais-valia que deve ser explorada.

Referências

[1] - White Paper, *"Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2012-2017",* 6 February 2013

[2] - http://www.significados.com.br/hotspot-wifi/, acedido em 09-01-2014

[3] - https://www.wifipt.pt/pt/oquee/Pages/OQueE.aspx, acedido em 09-01-2014

[4] - http://meo.pt/suporte/artigo/wifi-pt-o-meu-hotspot/?context=servicos-adicionais, acedido em 09-01-2014

[5] - OPEN NETWORKING FOUNDATION, *"Software-Defined Networking: The New Norm for Networks", ONF* White Paper April 13, 2012

[6] - OPEN NETWORKING FOUNDATION, *"SDN Architecture Overview",* Version 1.0, December 12, 2013

[7] - Susan Fogarty, "*7 Essentials Of Software-Defined Networking*", http://www.informationweek.com/infrastructure/networking/7-essentials-of-software-defined-networking/d/d-id/898899?image_number=5, acedido em 09-01-2013

[8] - ryu, http://osrg.github.io/ryu/, acedido em 09-01-2013

[9] - ryu, https://pypi.python.org/pypi/ryu, acedido em 09-01-2013

[10] - NOX, http://www.noxrepo.org/nox/about-nox/, acedido em 09-01-2013

[11] – POX, http://www.noxrepo.org/pox/about-pox/, acedido em 09-01-2013

[12] - Guillermo Romero de Tejada Muntaner, *"Evaluation of OpenFlow Controllers",* October 15, 2012

[13] - Floodlight, http://www.projectfloodlight.org/floodlight/, acedido em 09-01-2013

[14] - Mul, http://sourceforge.net/projects/mul/, acedido em 09-01-2013

[15] - kulcadmin, "*MuL architecture in a nutshell*", http://kulcloud.wordpress.com /2012/10/02 /mul-architecture-in-a-nutshell/

[16] - OpenDaylight, http://www.opendaylight.org/software, acedido em 28-07-2014

[17] - http://ltgjamaica.wordpress.com/2012/04/29/software-defined-networking-first-look-at-openflow-30/, acedido em 28-07-2014

[18] - Steven Wallace, "OpenFlow Workshop", https://wikispaces.psu.edu/display/EmergingTechnologies/OpenFlow+Workshop

[19] - https://raw.githubusercontent.com/openvswitch/ovs/master/FAQ, acedido em 28-07-2014

[20] - Open vSwitch, http://openvswitch.org/, acedido em 28-07-2014

[21] - Kok-Kiong Yap, Te-Yuan Huang, Yiannis Yiakoumis, Nick McKeown, Sachin Katti, Guru Parulkar, *"Making use of all the networks around us: a case study in android",* 2012 ACM SIGCOMM workshop on Cellular networks: operations, challenges, and future design, Pages 19-24

[22] - Peter D., Andreas K., Lawrence C., Nico B., Hans E., Christoph P., Nicholas B., *"BEST-AP: Non-intrusive estimation of available bandwidth and its application for dynamic access point selection",* Journal Computer Communications Volume 39, February, 2014 Pages 78-91

[23] - Sushant J., Alok K., Subhasree M., Joon O., Leon P., Arjun S., Subbaiah V., Jim W., Junlan Z., Min Z., Jon Z., Urs H., Stephen S., *Amin V.,"B4: experience with a globally-deployed software defined wan",* ACM SIGCOMM 2013 conference on SIGCOMM Pages 3-14, Newsletter ACM SIGCOMM Computer Communication Review Volume 43 Issue 4, October 2013 Pages 3-14

[24] - D-ITG 2.8.1 Manual, http: // traffic. comics. unina. it/ software/ ITG, October 28, 2013, acedido em 28-07-2014

[25] - Iperf, https://iperf.fr/, acedido em 28-07-2014

[26] - Iperf, http://openmaniak.com/iperf.php, acedido em 28-07-2014

[27] - http://www.telecomworld101.com/IPheader.html, acedido em 28-07-2014

[28] - TOS, Adrian Farrel, "A INTERNET E SEUS PROTOCOLOS, uma análise comparativa", EDITORA CAMPUS

[29] - http://what-when-how.com/qos-enabled-networks/classifiers-qos-enabled-networks-part-1/ , acedido em 12-08-2014