Rui Filipe Fernandes Cardoso

Licenciado em Engenharia Electrotécnica e de Computadores

Uma abordagem SDN para o controlo e admissão de tráfego

Dissertação para obtenção do Grau de
Mestre em Engenharia Eletrotécnica e de Computadores

Orientador: Paulo da Costa Luís da Fonseca Pinto, Prof. Dr, FCT-UNL

Júri:

Presidente:	Prof. Doutor Pedro Amaral
Arguente:	Prof. Doutor Henrique Mamede
Vogal:	Prof. Doutor Paulo Pinto

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Julho, 2015

Uma abordagem SDN para o controlo e admissão de tráfego

À minha família, namorada e amigos

Agradecimentos

Em primeiro lugar gostaria de agradecer esta dissertação ao meu orientador, o professor doutor Paulo Pinto, por sempre me ter apoiado durante a sua realização e mesmo nos momentos mais complicados ter acreditado em mim. Demonstrou-se sempre disponível para ajudar, mantendo o contacto mesmo durante as suas viagens ao estrangeiro.

Agradeço também a todos os professores que ao longo deste percurso académico me acompanharam, às vezes ajudando outras vezes complicando, mas sempre contribuindo para o meu desenvolvimento um obrigado. Um agradecimento especial aos professores Pedro Amaral e Luís Bernardo, que apesar de não estarem diretamente ligados à dissertação, demonstraram-se sempre disponíveis para me ajudar.

Gostaria de agradecer a todos os amigos que fui fazendo ao longo destes 5 anos, apesar de durante o percurso ter perdido o contacto com alguns, todos acabaram, de alguma forma, me apoiar. Aqueles com quem diariamente tenho contacto, seja pessoalmente, através da Internet ou telemóvel, um agradecimento especial, passámos muito bons momentos juntos e muitos outros estão para vir. Agradeço também a todas as amizades que fiz durante a minha estadia em Barcelona, em especial a todos os meus colegas da *Casa Awesome*.

À minha namorada, que me tem acompanhado todos os dias durante os últimos quatro anos. Agradeço por todo o apoio dado, especialmente nesta última etapa em que não lhe tenho podido dar tanta atenção quanto merece.

A toda a minha família, especialmente aos meus pais e irmão, por me terem apoiado e ajudado desde sempre, sem eles nunca me teria tornado no que sou hoje e a realização desta dissertação não teria sido possível.

A todos, um obrigado!

RESUMO

Nos últimos anos tem-se verificado uma modificação nos hábitos de consumo da Internet. As suas aplicações necessitam de certas garantias que são cada vez mais exigentes e, para as acompanhar, métodos cada vez mais avançados de classificação de tráfego são utilizados de modo a lhes atribuir os seus requisitos. Este comportamento provoca no entanto um aumento da complexidade da rede, criando problemas na sua gestão e no processamento de toda a informação. Com o aumento de tráfego, a situação tende a ser cada vez pior.

A solução apresentada nesta dissertação passa pela construção de duas lógicas de processamento de controladores Software-Defined Networking (SDN), que atribuem garantias às aplicações utilizando métodos de classificação de tráfego simplistas, utilizando uma metodologia *first-come, first-served* (FCFS), diminuindo o processamento requerido em relação a outras técnicas. Independentemente do tipo de tráfego, caso a rede consiga assegurar garantias, estas serão atribuídas. Existe um comportamento diferenciado na zona central da rede e na zona de ingresso, de modo a criar uma rede mais escalável, em que a zona de ingresso, lidando com um menor número *flows*, prepara-os para serem enviados para a zona central bastando a estes apenas encaminhar o tráfego para o seu destino, retirando carga desta zona onde a quantidade de *flows* é superior.

Obtém-se assim uma rede que garante qualidade de serviço a determinados *flows*, evitando a utilização de técnicas pesadas ao nível de classificação, tornando-a mais escalável.

Palavras-chave: SDN, OpenFlow, Floodlight, qualidade de serviço, classificação de tráfego

Abstract

Over the last few years, there has been a change on the use of the Internet. Its use has become more demanding, with applications needing harder requirements in order to function properly. This has led to the development of exhausting traffic classification techniques, in order to grant them their needs. These techniques have caused an increased complexity on the network, resulting in increased management and processing issues. With the growth of Internet traffic, the situation is getting out of hand.

The solution described in this dissertation consists on constructing two SDN controller processing logic, that grant applications quality of service using simplistic traffic classification methods. It is implemented using a FCFS (First Come First Served) logic, reducing the processing requirements in comparison to other techniques. Any kind of traffic can be protected if the network has enough capacity for it. The two logics are used to elaborate the behaviour of the network at core level and at the ingress area.

The ingress area has fewer flows to track and can exercise some control and prepare them to be received on the network core. The core part has a higher number of flows and the only major operation that it performs is routing the packets through the network. This behaviour leads to a bigger scalability of the network.

With our solution we obtained a network which grants quality of service to some flows, avoiding the use of heavy traffic classification techniques, making it more scalable.

Keywords: SDN, OpenFlow, Floodlight, quality of service, traffic classification

Conteúdo

Lista de Figuras

LISTA DE TABELAS

GLOSSÁRIO

ARP Address Resolution Protocol.

CAC Call Admission Control.

CBR Constraint-based Routing.

CR-LDP Constraint-based Routing Label Distribution Protocol.

CSPF Constrained Shortest Path First.

DPI Deep Packet Inspection.

ERO Explicit Route Objects.

FCFS *first-come, first-served*.

IANA Internet Assigned Numbers Authority.

ICMP Internet Control Message Protocol.

IETF Internet Engineering Task Force.

IP Internet Protocol.

LDP Label Distribution Protocol.

LER Label Edge Router.

LSP label-switched path.

LSR Label Switching Routers.

MAC Media Access Control.

MPLS Multiprotocol Label Switching.

ONF Open Networking Foundation.

OVS Open vSwitch.

P2P Peer-to-peer.

PID Process ID.

QoS Quality of Service.

RSVP Resource Reservation Protocol.

RSVP-TE Resource Reservation Protocol - Traffic Engineering.

RTT Round-trip Time.

SDN Software-Defined Networking.

SIP Session Initiation Protocol.

SSH Secure Shell.

TCP Transmission Control Protocol.

UDP User Datagram Protocol.

VoIP Voice over Internet Protocol.

CAPÍTULO 1

INTRODUÇÃO

1.1 Motivação

A utilização da Internet tem vindo a sofrer uma alteração de comportamento nos últimos anos. As necessidades exigidas pelas aplicações têm-se vindo a alterar e são cada mais exigentes [1]. A solução que se tem usado para assegurar as suas necessidades passa pela identificação do tráfego, atribuindo garantias a uns determinados tipos e retirando-as a outros.

No entanto, este tipo de abordagem tem causado vários tipos de problemas. Certas aplicações, não pretendendo ser classificadas, recorrem a métodos cada vez mais avançados para evitar a sua classificação, tal como a encriptação do conteúdo [33]. Devido a isso, são necessárias novas técnicas de análise de tráfego, utilizando métodos mais avançados de classificação de tráfego que requerem uma maior capacidade de processamento por partes dos *switches* [52]. Aumenta-se desse modo a complexidade da rede, levando ao aumento dos custos dos equipamentos e gestão, mas também a problemas éticos devido a possíveis violações de privacidade, pois alguns métodos requerem a análise dos conteúdos dos pacotes (DPI - *Deep Packet Inspection*) [32].

A situação torna-se ainda mais complicada quando se trata da zona central da rede, onde o volume de dados tratado é maior, tornando estes métodos impraticáveis a longo prazo, tendo em vista o presumível aumento de tráfego na rede, pois para realizar essa classificação é necessário processar um enorme volume de dados num curto espaço de tempo [63].

Uma nova abordagem tem sido tomada através da introdução das SDN, que vêm possibilitar um olhar mais simples sobre a rede e a sua gestão [21]. Através do seu uso, o processamento cai sobre os controladores programáveis SDN, abrindo uma nova perspetiva de gestão de rede e do seu tráfego.

1.2 Objetivos

Esta dissertação tem como objetivo o desenvolvimento de uma lógica de controlo de rede baseada em SDN onde são asseguradas garantias de qualidade a determinados *flows*, fugindo das metodologias tradicionais de classificação de tráfego e utilizando métodos não intrusivos, nem discriminativos podendo ser aplicados a qualquer tipo de tráfego.

Outro objetivo é retirar o processamento da zona central da rede, onde o volume de tráfego é superior, passando a classificação do tráfego para a zona de ingresso e removendo a necessidade de manter estado num interior na rede sem no entanto afetar os pontos anteriormente referidos.

Esta lógica deve suportar a sua implementação em redes de grandes dimensões, sendo escalável e eficiente na utilização dos seus recursos.

1.3 Contribuições do Trabalho

A realização desta dissertação contribuiu para o desenvolvimento de duas lógicas de processamento baseadas em redes SDN utilizando o protocolo OpenFlow.

Houve também uma contribuição para o projeto do controlador utilizado, Floodlight [16], tendo o seu autor ajudado durante o desenvolvimento com correções de *bugs*, e adições de novas funcionalidades. Esta contribuição foi reconhecida no repositório oficial [18], na altura do lançamento da primeira versão final deste controlador.

Esta dissertação é também a base de um artigo a ser entregue para a IEEE GLOBECOM 2015.

1.4 Organização da dissertação

A dissertação encontra-se dividida em 5 capítulos:

- **Capítulo 1**

  No primeiro capítulo encontram-se os motivos que levaram à realização da dissertação, introduzindo qual o problema atual e os objetivos propostos para a sua resolução. São ainda referidas as contribuições provenientes do desenvolvimento da dissertação.

- **Capítulo 2**

  No segundo capítulo é demonstrado o atual estado da arte. São descritas as ferramentas que levaram a que fosse possível realizar o projeto, tais como as ferramentas que atualmente existem com o objetivo de tentar solucionar os problemas apresentados.

- **Capítulo 3**

No terceiro capítulo é apresentada a proposta elaborada para atingir os objetivos propostos. O funcionamento desta, problemas encontrados na sua elaboração e as respetivas soluções são descritas neste capítulo.

- **Capítulo 4**

  No quarto capítulo encontra-se a informação correspondente às simulações. Aqui é descrito qual o ambiente onde foram estas realizadas, os seus resultados e análise dos mesmos, tendo em conta o objetivo proposto para o projeto.

- **Capítulo 5**

  O quinto capítulo é o último da dissertação. Neste é realizado o levantamento final dos resultados alcançados durante a sua elaboração. São referidas também possibilidades de continuação de desenvolvimento do projeto em trabalhos futuros.

CAPÍTULO 2

TRABALHO RELACIONADO

2.1 Software-Defined Networking

O SDN é um recente tipo de arquitetura de redes cuja popularidade tem vindo a aumentar. [49]. Apesar do conceito de SDN já estar presente há algum tempo [67], este tem sofrido uma maior adoção nos últimos anos à medida que foi ganhando o apoio de diversas empresas tecnológicas importantes na área de redes, tais como a HP, a Cisco, a Juniper, entre outras [20].

Tem como objetivo não só resolver algumas das limitações atualmente existentes nas redes mas também tentar dar respostas ao que se prevê ser o futuro destas onde é esperado um aumento da utilização de equipamentos móveis, de serviços *cloud* e virtualização de servidores criando um futuro para o qual estas não foram desenhadas [21] .

Sendo assim, a Open Networking Foundation (ONF) estabeleceu como objetivos para o SDN a criação de uma rede programável, adaptável às novas necessidades e sem as limitações das redes atuais. Alguns dos problemas levantados pela ONF são a enorme complexidade de protocolos que tornam difícil a evolução das redes, uma dependência dos fornecedores de equipamento e uma deficiência de expansão da rede, pois torna-se complicado operadores com milhares de equipamentos se expandirem sendo que a adição de mais equipamento equivale a dezenas de reconfigurações individuais de dispositivos[21].

Com o SDN, como observamos na figura 2.1, o controlo lógico da rede passa a estar centralizado num único ponto, o controlador de SDN. O controlador passará assim a ser um dos componentes mais importante de uma rede SDN. O seu funcionamento encontra-se detalhado na secção 2.1.3.

A centralização do processamento simplifica a rede, não apenas ao nível de desenho e operação, mas também ao nível do equipamento utilizado. Os *switches* e *routers* tornam-se mais simples, necessitando apenas de encaminhar o tráfego utilizando a informação

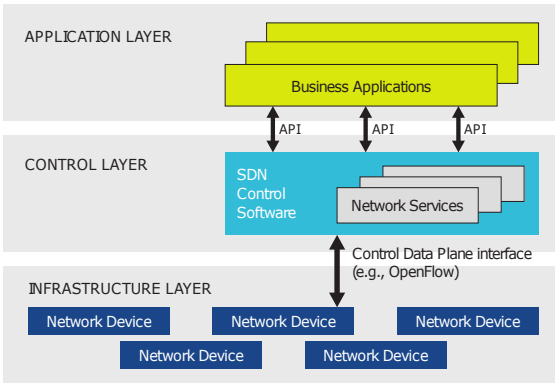

Figura 2.1: Arquitetura do SDN. Adaptado de [21]

recebida do controlador SDN [21]. Os administradores de rede ganham a capacidade de configurar individualmente cada um dos dispositivos da rede a partir de um único ponto, independentemente da sua marca, bastando a este compreender o protocolo usado na interface entre os planos de controlo e dados (figura 2.1).

A rede adquire assim uma maior flexibilidade, passando a ser possível programar o seu comportamento através dos controladores, permitindo o desenvolvimento de aplicações mais especificas ao local onde esta se encontra [43].

O protocolo usado na interface entre os planos de controlo e dados, isto é, entre o controlador e os dispositivos de rede, tem o nome de OpenFlow e é abordado na próxima secção (2.1.1).

2.1.1 OpenFlow

O OpenFlow foi criado pela ONF e consiste no primeiro *standard* de comunicação criado para estabelecer a ligação entre o controlador e o dispositivo de rede [21]. Este possibilita o acesso direto à camada de encaminhamento dos dispositivos de rede, algo que nos *switches* tradicionais não era facilmente possível.

Ao contrário do que acontece nas redes tradicionais, baseadas no modelo TCP/IP de quatro camadas, o OpenFlow é considerado um protocolo *layerless*, onde a definição de camada de aplicação, transporte e Internet é perdida, criando um conceito de uma camada abstrata onde todos os elementos são tidos em conta simultaneamente [25].

Atualmente o protocolo vai na versão 1.4 sendo que o *draft* da versão 1.5 foi submetido em Dezembro de 2014 [44, 45]. Nas especificações do OpenFlow são encontrados quais os componentes e as funções que este protocolo possibilita, sendo que neste documento apenas se descreve as mais interessantes para a elaboração do projeto.

2.1.1.1 Mensagens

O protocolo OpenFlow define 3 categorias de mensagens, trocadas através do *OpenFlow Channel*, que qualquer *switch* que suporte este protocolo possui. As categorias de mensagens são: *controller-to-switch*, correspondendo a mensagens enviadas do controlador para o *switch* sem ser necessário uma resposta por parte deste; *asynchronous*, mensagens enviadas pelo *switch* sem serem requisitadas pelo controlador que geralmente denotam a chegada de um pacote ao *switch* ou uma mudança de estado do mesmo; e as mensagens *symetric* que são enviadas sem nenhuma solicitação, requerendo no entanto uma resposta por parte do *switch* ou do controlador, exceto no caso da mensagem de erro. Na especificação pode ser encontrada a descrição completa e como deverá ser efetuada a sua implementação [44].

Neste capítulo são apenas analisadas as mensagens mais importantes para a compreensão do trabalho elaborado, sendo elas as seguintes:

Hello - Mensagens do tipo *symetric*, trocadas ao ser iniciada uma ligação entre um *switch* e um controlador;

Echo - Mensagens do tipo *symetric*, trocadas entre o switch e o controlador para manter a ligação ativa, mas também utilizada para medir a latência da ligação e a sua largura de banda;

Features - Mensagem do tipo *asynchronous* enviada pelo controlador para o *switch* para saber quais as características deste, como por exemplo qual a versão de OpenFlow que este suporta. É geralmente trocada ao estabelecer a ligação e requer uma resposta do *switch*;

Modify-State - Mensagem do tipo *asynchronous* enviada pelo controlador para gerir o estado do *switch*. A sua principal funcionalidade é adicionar, alterar ou remover *flow entries* (2.1.2). Poderá ainda ser utilizada para modificar as propriedades das portas do *switch*;

Packet-out - Mensagens do tipo *asynchronous* enviadas pelo controlador como resposta ao packet-in. É através deste tipo de mensagens que o controlador encaminha um pacote para uma porta específica no *switch*. A mensagem pode conter uma lista de ações (*actions*) (2.1.2) a serem tomadas pelo controlador; caso não contenha nenhuma *action* o pacote será descartado;

Packet-in - Mensagem do tipo *asynchronous*. É enviada do *switch* para o controlador quando é recebido um pacote em que nenhum *match* (2.1.2) foi correspondido;

Flow-Removed - Mensagem do tipo *asynchronous*. Informa o controlador sempre que uma *flow entry* que possua uma flag `OFPFF_SEND_FLOW_REM` seja removida. As entradas são removidas após um certo período de tempo sem atividade, definido durante a configuração do *flow*, ou através de um pedido do controlador;

Read-State - Mensagem do tipo *asynchronous* originada no controlador com o propósito de recolher algum tipo de informação do *switch*. Este tipo de informação pode ser, por exemplo, o número de bytes transmitido por um certo *flow* ou a quantidade de portas que o *switch* possui.

2.1.2 Switch OpenFlow

Os componentes de um *switch* OpenFlow estão ilustrados na figura 2.2. Estes são uma ou várias tabelas de *flows*, uma tabela de grupos e o canal OpenFlow que possibilita a ligação do *switch*.

Segundo [13], um *flow* é uma sequência de pacotes de uma certa origem para um certo destino. Essa origem e destino são identificados pelo que se chama de *5-tuple*, ou seja, através do seu endereço IP de origem, porta de origem, IP de destino, porta de destino e o protocolo da camada de transporte.

Esta definição é utilizada neste projeto para descrever o funcionamento de uma rede SDN. No entanto, dependendo do contexto em que nos encontramos, este poderá possuir diferentes significados. Ao descrever o funcionamento do projeto, a definição de *flow* passa por uma sequência de pacotes de uma *vlan* e porta de origem e destino.

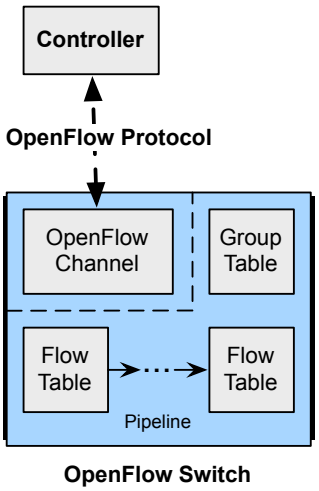

Figura 2.2: Componentes de um switch OpenFlow. Adaptado de [44]

Segundo [44], alguns dos conceitos básicos importantes para conseguir compreender o funcionamento de um *switch* OpenFlow são:

Match - É um campo contra o qual um pacote é comparado, ou seja *matched*. Contém elementos que vão desde a layer 2 até à layer 4. Geralmente ligado a um *match* está uma *action*.

Action - É o elemento que irá encaminhar o pacote para uma certa porta do *switch* ou modificar o conteúdo do pacote. Um pacote pode acumular diferentes *actions*, sendo

que caso duas ações modifiquem o mesmo conteúdo, a última a modifica-lo tem prioridade. Para uma lista de ações possíveis de modificar consulte [44];

Flow Entry - Trata-se de um elemento na tabela de *flows*. Possui um ou mais *match fields* e uma lista de instruções, com uma ou mais *actions*.

Cookie - Uma cookie é um identificador de uma *flow entry*. Torna possível a identificação de diferentes *flow entries* de modo a se filtrar a entrada sobre a qual se deseja atuar, como por exemplo a remover ou obter dados estatísticos.

Flow Tables - É neste componente que as diversas *flow entries* são instaladas. Um *switch* pode ter uma ou mais tabelas de *flows*, isto se utilizar uma versão de OpenFlow superior a 1.1.

Apesar do OpenFlow nos trazer outros conceitos importantes, os acima nomeados são os mais relevantes para se compreender o funcionamento do presente trabalho.

2.1.2.1 Funcionamento

O funcionamento de um encaminhador OpenFlow ao receber um pacote pode ver visualizado na figura 2.3.

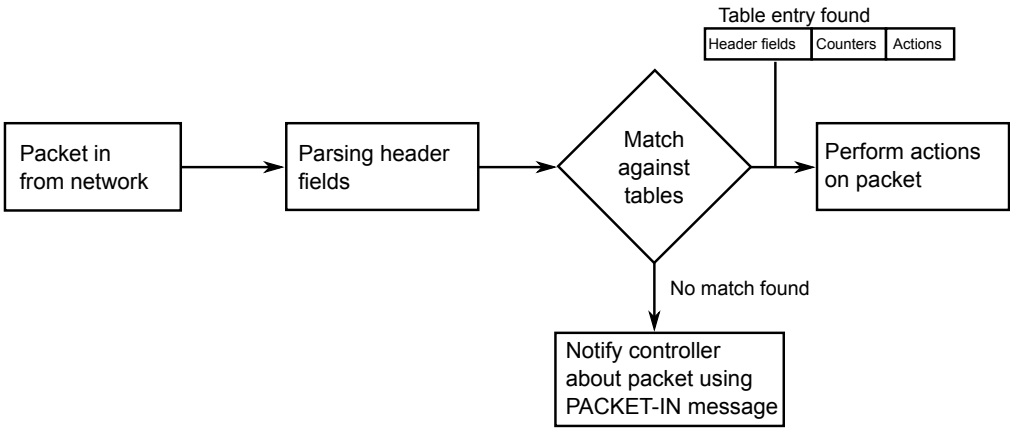

Figura 2.3: Funcionamento de um encaminhador ao receber um pacote. Adaptado de [8]

Ao receber um pacote, após o tratamento dos seus cabeçalhos, o *switch* irá verificar se existe algum *match* na sua tabela de *flows*. Caso exista uma entrada (*flow entry*) com o *match* correspondente, as ações dessa entrada irão então ser tomadas. Caso chegue ao fim de todas as entradas instaladas em todas as listas existentes no *switch*, dependendo da configuração do mesmo, ele poderá encaminhar o pacote para o controlador, utilizando uma mensagem do tipo PACKET-IN ou simplesmente fazer um *drop* ao mesmo.

2.1.3 Controladores SDN

O controlador é o principal componente num sistema SDN. É neste componente que as decisões são tomadas de maneira a garantir o funcionamento da rede SDN e a atingir os objetivos propostos para a mesma. Apesar de ser geralmente um elemento logicamente centralizado, não significa que seja apenas um equipamento. De maneira a garantir uma maior robustez na rede e poder garantir um melhor desempenho é possível replicá-lo por diferentes dispositivos. Esta situação é especialmente útil ao lidar com informação que necessita de garantias especiais, como acontece no sector bancário e saúde [39].

O controlador pode ser instalado em qualquer dispositivo que possua uma porta compatível com a ligação ao canal OpenFlow. Este dispositivo não tem de possuir grandes características, podendo ir de um computador a um servidor potente, reduzindo assim o custo de instalação de uma rede SDN. Mas sendo um dos elementos centrais, as suas características vão influenciar o comportamento de toda a rede, portanto a sua correta escolha é importante. Contudo, não se trata apenas de uma escolha do hardware do controlador, mas também de programa que o implementa. A diferença entre os vários programas controladores de SDN influencia a eficiência da rede [61], tornando assim a sua escolha importante.

Para além do programa específico existem outras características importantes a ter em conta ao decidir qual o melhor controlador para a rede, dependendo do propósito da rede à qual este se destina. De seguida é feito um breve resumo sobre alguns dos possíveis controladores para a realização deste trabalho. Há que referir que estes podem não ser os melhores controladores para outros projetos, e que existem muitos outros que não são aqui descritos, mas que apresentam outro tipo de vantagens.

2.1.3.1 NOX\POX

O NOX foi o primeiro controlador SDN desenvolvido [27]. Apesar de inicialmente suportar as linguagens C++ e Python, neste momento as versões foram separadas e caso se utilize a versão python tem o nome de POX e a versão C++ possui o nome de NOX.

Olhando para a figura 2.4, apesar de inicialmente bastante adotado e ter sido usado como base para outros controladores [39], o seu uso tem vindo a cair ao longo do tempo. Uma das plausíveis razões é o facto deste controlador não suportar as versões mais recentes do OpenFlow, apenas suportando até ao momento a versão 1.0. Devido às razões apresentadas anteriormente este não foi o controlador escolhido para o projeto.

2.1.3.2 Ryu

O Ryu é outro controlador de SDN [54]. Uma das vantagens deste controlador é o facto de suportar todas as versões de OpenFlow lançadas oficialmente até ao momento, suportando desde a versão 1.0 até à versão 1.4.

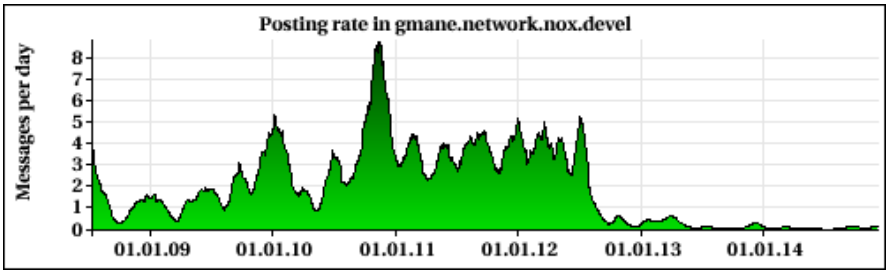

Figura 2.4: Actividade da *mailing list* de desenvolvimento do NOX. Adaptado de [24]

É um controlador em *python* e é baseado em módulos, existindo uma base fixa que tem por objetivo lançar o controlador e encaminhar as mensagens para os diferentes módulos registados, sendo estes a tomar as ações sobre o pacote.

No entanto, este possui pouca documentação acerca do seu funcionamento e uma comunidade pouco ativa. Devido a estes fatores, este também não foi o controlador escolhido, no entanto, deverá ser um controlador a ter em conta devido à sua elevada eficiência [36].

2.1.3.3 Floodlight

O Floodlight é um controlador de SDN baseado em java. A sua primeira versão foi oficialmente lançada no inicio deste ano, que trouxe o suporte para a versão 1.3 do OpenFlow [18].

Tem como base um controlador anteriormente desenvolvido com o nome de Beacon, que foi um controlador anteriormente bastante utilizado [14]. Possui uma documentação bastante completa, mostrando diversos exemplos úteis para o desenvolvimento de novas aplicações, mas também um grupo bastante ativo, onde é possível esclarecer dúvidas diretamente com a comunidade envolvida no desenvolvimento do projeto [17].

Estes fatores fizeram com que este fosse o controlador escolhido para a realização do projeto, sendo analisado em maior detalhe.

Este controlador, tal como o Ryu (2.1.3.2), é um controlador que tem por base o desenvolvimento modular, em que cada um dos módulos tem um objetivo único, simplificando assim a construção de aplicações, pois a sua base é fixa. Através da sua arquitetura, representada na figura 2.5, é possível observar essa metodologia de desenvolvimento, mas também encontrar outras funcionalidades que este controlador possibilita. Mais especificamente, a camada de serviços que este possui, diferencia-o dos restantes controladores apresentados.

Um dos serviços mais interessantes disponibilizados de origem com o controlador é o serviço de armazenamento *NoSQL* [19, 64]. Todos os módulos do controlador podem aceder ao serviço, tornando assim possível aceder a dados que não são apenas locais. Desta maneira conseguimos obter módulos mais eficientes, que através desta comunicação conseguem adquirir informação que não lhes estaria acessível caso estivessem a trabalhar

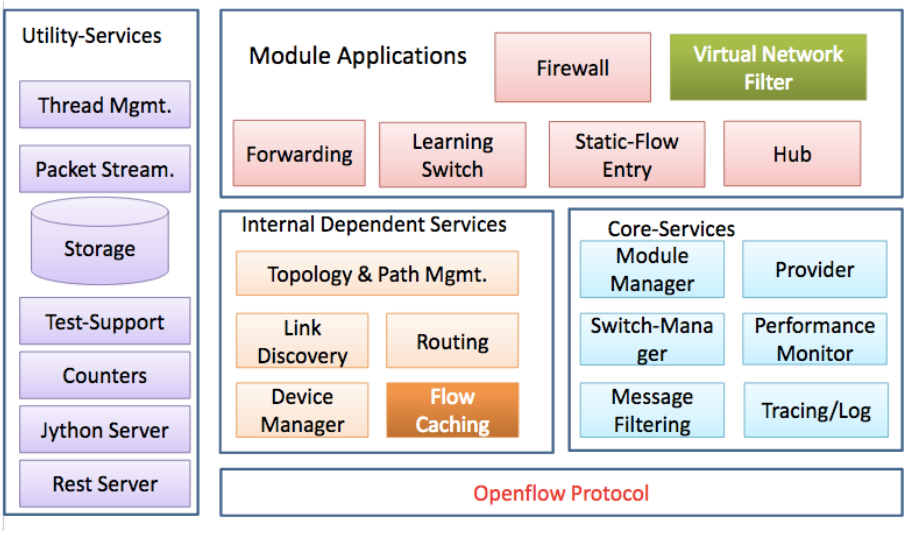

Figura 2.5: Arquitetura do controlador Floodlight. Adaptado de [4]

de modo isolado.

Outras funcionalidades interessantes que o controlador possui são a interface gráfica, bastando ao utilizador inserir o endereço IP do controlador no navegador para obter informação sobre o estado da rede, e uma avançada *REST API* que possui diversas funções completas que possibilitam, desde ver o estado dos *flows* atuais a instalar novos.

O facto deste controlador trazer diversos módulos instalados por defeito, traz consigo duas grandes vantagens: são uma útil fonte base por onde começar a desenvolver um novo módulo e permitem testar de imediato o controlador, possibilitando logo à partida acesso a funcionalidades básicas, como o encaminhamento de pacotes, facilitando a realização de testes inicias.

No entanto, devido a este ser um controlador que se encontra em ativo desenvolvimento, certas funcionalidades ainda não se encontram completas. Uma delas é o suporte a *tags* Multiprotocol Label Switching (MPLS), que na versão utilizada para o desenvolvimento do projeto não são suportadas.

2.1.4 Mininet

O Mininet é um software que possibilita a criação de uma rede simulada que pode correr em qualquer computador [40]. Devido a suportar o SDN, esta ferramenta ganha uma grande importância para o desenvolvimento e o teste de aplicações. Ao correr em Linux, revela-se uma ferramenta muito versátil e de fácil uso, sendo por isso a ferramenta virtual utilizada para a maior parte dos projetos SDN.

Uma característica importante do Mininet é a possibilidade da criação de uma topologia de rede customizada através da Mininet API. Esta API, programada utilizando a linguagem python, possibilita diferentes personalizações, tais como: definir a quantidade de encaminhadores a existir na topologia, quais as ligações que vão ser estabelecidas

entre eles, as caraterísticas dos *switches* configuradas individualmente e a criação de *hosts*, representando dispositivos ligados aos *switches*. Nestes *hosts* podem também ser individualmente configurados atributos tais como o endereço IP e MAC.

Devido aos *hosts* serem uma máquina virtual que corre uma versão simplificada do Linux, quase todas as aplicações deste sistema operativo funcionam nestes *hosts* [48]. Existe assim uma grande quantidade de ferramentas que podem ser utilizadas para testar a rede, sem ser necessário modificações. Algumas das mais famosas, e que serão utilizadas neste projeto, são o iperf [66], o SIPp [23] e o Wireshark [12].

Existe também a possibilidade de instalar diferentes tipos de *switches* no Mininet. No entanto, o software mais utilizado para a sua simulação e o que vem selecionado por origem é o Open vSwitch [48]. O seu funcionamento será detalhado em 2.1.4.1.

Contudo, existem certos problemas que deverão ser tidos em consideração ao criar uma rede no Mininet. Por se tratar de uma rede simulada por *software*, existem problemas na simulação de grandes redes que poderão prejudicar os resultados finais [35], sendo que a eficiência da simulação se encontra limitada pela capacidade de processamento da máquina onde é corrida.

2.1.4.1 Open vSwitch

O Open vSwitch (OVS), tal como o nome indica, é um *switch* virtual cujo código fonte é aberto a toda a comunidade (*open source*)[48]. Este software consegue não só encaminhar o tráfego entre diferentes máquinas virtuais a correr num só equipamento mas também tráfego entre diferentes máquinas virtuais para a rede física [46].

Como referido em 2.1.4, o OVS pode ser escolhido como o *switch* virtual utilizado pela topologia do Mininet, facilitando assim a sua utilização. O OVS permite a configuração de vários elementos dos seus *switches* através da linha de comandos revelando-se assim uma ferramenta versátil perante a criação de topologias de rede personalizadas.

Um desses parâmetros é a criação de *queues*, sendo que uma *queue* é semelhante a uma fila de espera virtual associada a uma porta de saída. Esta fila pode ser configurada com vários parâmetros, descritos em [47]. Um dos parâmetros disponíveis é a prioridade e vai ter um especial relevo neste projeto.

A prioridade permite beneficiar uns pacotes em detrimento de outros e com isso possibilitar a construção de aplicações que diferenciem tráfego. São sobejamente conhecidos os problemas que se podem ter por existirem várias filas de espera com diferentes prioridades. Um deles é uma fila com menor prioridade poder ser levada a *starvation*, ou seja, a uma situação em que nunca chega a ser capaz de enviar os seus pacotes. Em termos de redes de telecomunicações, uma maneira de limitar tal ocorrência é através da utilização de um outro parâmetro que limita o uso máximo de uma *queue* a um certo valor de largura de banda, ou através de uma gestão realizada pelo controlador. Recorde-se que a atribuição de um pacote ou *flow* a uma *queue* é feita através de uma *flow entry* ou diretamente a partir do controlador num *packet-out*.

Apesar de ser uma ferramenta completa, existem certas características que o OVS não suporta. Uma das características não suportadas são os *OpenFlow Meters*. Se existissem permitiriam, por exemplo, a limitação de um certo *flow* a uma certa largura de banda bem como obter métricas mais detalhadas sobre os *flows* diretamente a partir do encaminhador. A sua ausência levou a que certas adaptações tivessem de ser realizadas.

2.2 Gestão de Tráfego

2.2.1 Modelos QoS

O Internet Engineering Task Force (IETF) considera que a qualidade de serviço, normalmente tratada pela sigla QoS (Quality of Service), consiste na habilidade de diferenciar tipos de tráfego com o objetivo de tratar um certo *flow* de forma diferenciada dos restantes [15]. Em [42], o autor contextualiza esta definição no tema de redes, referindo que consiste na habilidade de diferenciar tipos de tráfego uns dos outros, mas tendo em conta o que a aplicação necessita para atingir os seus requisitos de funcionamento. Um olhar mais aprofundado sobre as necessidades e características de diferentes tipos de tráfego é feito em 2.3.

Com o intuito de introduzir Quality of Service (QoS) na rede, o IETF criou dois modelos: o modelo de serviços integrados (2.2.1.1) e o modelo de serviços diferenciados (2.2.1.2).

2.2.1.1 Serviços Integrados

Os serviços integrados, *IntServ* [62], são um modelo que especifica uma reserva de recursos por fluxo. Para que tal aconteça, é necessário que a aplicação em causa interaja em termos de controlo/sinalização. Através dessa interação a rede tem de encontrar uma rota que satisfaça as necessidades dessa aplicação, sendo que após essa rota ser encontrada, os recursos necessários para a estabelecer são registados individualmente em cada um dos *switches* por onde esta passa. Só após a rota estar estabelecida a aplicação pode começar a enviar os pacotes necessários.

Para estabelecer a reserva de recursos na rede, apesar deste modelo não estar diretamente ligado a nenhum protocolo, o Resource Reservation Protocol (RSVP) (Resource ReSerVation Protocol) é geralmente indicado como o utilizado para a sinalização do *IntServ* e encontra-se detalhado em 2.2.3.

A arquitetura dos serviços integrados define três classes de serviços que, ordenados desde o que dá mais garantias para o que garante menos, são:

- **Garantia de Serviço** - garante largura de banda, atraso limitado e sem perdas de pacotes;
- **Carga controlada** - esta classe não define nenhum valor especifico de largura de banda ou de atraso. É direcionada para tráfego que se consegue adaptar às características da rede em tempo real.

- ***Best Effort*** - sem nenhum tratamento específico, sendo que a rede vai dar o melhor que puder para a sua transmissão mas sem garantias que esta irá ser bem sucedida.

No entanto uma das desvantagens da utilização desta arquitetura é o facto de necessitar que todos os *switches* na rede tenham a informação sobre todos os fluxos que passam por ele. Apesar de em redes pequenas não representar um problema, em redes de grandes dimensões tal torna-se impraticável fazendo com que esta arquitetura não seja escalável.

2.2.1.2 Serviços Diferenciados

Os serviços diferenciados, DiffServ [26], têm como objetivo diferenciar o tráfego num pequeno número de classes e disponibilizar recursos para cada uma dessas classes. Deste modo, é eliminado um dos problemas existentes nos serviços integrados (2.2.1.1) que era a impossibilidade de monitorizar individualmente cada um dos *flows* em redes de grandes dimensões, sendo apenas necessário a monitorização das classes existentes. Torna-se assim num serviço escalável e implementável em redes de grandes dimensões.

Ao entrar no *router* de ingresso da rede, o tráfego é classificado com o intuito de obter o mesmo tratamento ao longo da rede. Para isso, será aplicado um identificador ao pacote, que irá transportar a sua prioridade no byte *Type-of-Service* do seu cabeçalho. Usando este método, deixa de ser requerido aos *switches* centrais realizarem a identificação, aliviando o processamento na zona central da rede.

A simplicidade do *DiffServ* contudo torna-se um dos seus problemas, pois aumenta a configuração necessária na rede à priori. Outro problema é a dificuldade em prever o relacionamento entre as diferentes classes, pois cada uma das classes provoca um efeito noutra, situação que se torna especialmente evidente em grande redes [65]. Devido a este problema, por norma, apenas são utilizadas duas classes, uma classe do tipo *best-effort*, não dando nenhumas garantias de serviço, e uma classe para tráfego de voz, garantindo os requisitos necessários para este tipo de classe, como um *delay* e *jitter* baixo.

2.2.2 CAC

O controle de admissão de chamadas, mais vulgarmente chamado de Call Admission Control (CAC), tem como objetivo regular a rede com o intuito de proteger uma conexão de flutuação da largura de banda disponível e de perdas de ligação, com o fim de salvaguardar a qualidade da chamada. O conceito de CAC destina-se unicamente ao tráfego de voz, não sendo aplicável em tráfego de dados [11].

Para o fazer, o CAC protege as chamadas já em curso, sendo que ao se realizar uma nova, o estado da rede irá ser analisado de modo a saber se esta é suportada sem estragar a qualidade das chamadas já existentes. Assim sendo, uma chamada só será iniciada após a sua rota ser estabelecida, com a garantia que esta não irá deteriorar a qualidade das chamadas já existentes.

Existem diversos algoritmos CAC, cabendo ao administrador da rede decidir qual o mais indicado para a sua [22, 69]. Estes tentam maximizar a utilização das ligações

disponíveis, não diminuindo a qualidade das chamadas, tendo em conta as características dos protocolos utilizados, como por exemplo a sua inatividade e as suas variações de utilização de largura de banda ao longo do tempo.

2.2.3 RSVP

O RSVP é um protocolo de reserva de recursos que foi definido inicialmente em [41]. Ao longo do tempo tem sido alvo de alterações e extensões [29].

Tem como principio de funcionamento um modelo *soft-state*, em que para ser considerado que um certo fluxo se mantém ativo, os *switches* necessitam de enviar periodicamente mensagens que o indiquem. Caso as mensagens deixem de chegar, a reserva para um determinado fluxo será automáticamente eliminada, existindo no entanto a possibilidade de remover uma reserva por solicitação.

Para uma reserva ser realizada utilizando este protocolo, um *switch* de ingresso ao receber uma ligação irá enviar mensagens de sinalização ao longo do percurso dos pacotes, sendo que as reservas vão sendo estabelecidas sequencialmente *switch* a *switch* até chegar ao final da rota.

A rota não é calculada diretamente pelo RSVP, sendo definida pelos *switches*, utilizando as suas tabelas de encaminhamento. Quando um pedido de reserva chega a um *switch*, para que seja estabelecida será necessário estabelecer a comunicação entre os módulos de admissão, que irá verificar se atualmente existe capacidade no *switch* para acomodar a nova reserva, e um de controlo de acesso, que irá autentificar o pedido de QoS. Só após ambos os passos serem finalizados e bem sucedidos a reserva ficará efetuada. O processo é realizado através de uma troca de mensagens PATH e RESV.

A mensagem PATH é criada a partir do emissor, percorrendo o caminho até chegar ao seu destino. A informação que esta contém é guardada em cada um dos *switches* pelos quais a mensagem vai passando até chegar ao seu destino. Nesta primeira fase ainda não existe uma reserva ativa, mas apenas a intenção. Após a mensagem PATH chegar ao destino, este envia uma mensagem RESV, tendo como objetivo confirmar a reserva em cada um dos *switches* intermédios, tornando a reserva ativa, e confirmando o estabelecimento da reserva ao *switch* emissor. A partir deste momento, a ligação está estabelecida e os pacotes podem começar a ser trocados.

A falta de escalabilidade deste protocolo, devido a ser necessário a troca das mensagens para estabelecer a reserva por cada fluxo a criar, levou a que fosse adaptado para suportar reservas de diferentes fluxos em simultâneo [5]. Deste modo, a quantidade de mensagens trocadas pelo protocolo para a reserva de *flows* diminui.

2.2.4 MPLS

O MPLS [53] veio como resposta da IETF para tentar melhorar a engenharia de tráfego utilizada nas redes. Apesar de o protocolo também suportar outro tipo de tráfego como o ATM, a sua aplicação em outras redes que não IP não será aqui analisada.

Um dos problemas e virtudes das redes IP consiste no facto de os seus pacotes serem encaminhados tendo em conta o seu endereço IP de destino. Como resultado, certas ligações congestionam-se introduzindo atrasos e perdas de pacotes.

Para o MPLS ser implementado é necessária a introdução de um novo campo no cabeçalho de um pacote IP. Este é apenas válido dentro de uma rede que suporte MPLS e é inserido entre a *Layer 3* e a *Layer 2*. É com base neste novo cabeçalho, após a entrada no núcleo da rede, que o encaminhamento passará a ser feito, sendo que ao dispensar a leitura dos restantes campos, aumenta-se a velocidade de processamento deste pacotes.

O nome dado aos *switches* que se encontram na fronteira entre a rede MPLS e a rede exterior é Label Edge Router (LER). São o ponto de entrada e de saída da rede e deverão preparar os pacotes, inserindo ou removendo o cabeçalho MPLS caso estes estejam, respetivamente, a entrar ou sair da rede.

Aos *switches* que se encontram no centro das redes MPLS dá-se o nome de Label Switching Routers (LSR). Têm a característica de apenas lerem o *label* de entrada para encaminhar os pacotes e de inserir uma *label* ao pacote na saída, de forma a garantir que este é encaminhado corretamente. Contrariamente ao que acontece numa rede IP, no interior de uma rede MPLS os pacotes não mantêm o mesmo *label* e este não tem de ser único na globalidade da rede.

A distribuição de etiquetas poderá ser realizada através do protocolo Label Distribution Protocol (LDP) [2]. Foi criado pela IETF com o objetivo de distribuir as etiquetas dentro de uma rede MPLS com o intuito de construir e manter as ligações. Às ligações estabelecidas através de *labels* dá-se o nome de label-switched path (LSP) e estas começam e acabam num LER.

Umas das características do MPLS é suportar o encaminhamento com restrições, Constraint-based Routing (CBR). Este tem como intenção calcular rotas tendo em conta métricas tais como a largura de banda e regras administrativas. De maneira a suportar o CBR é necessário assegurar os seguintes mecanismos: um que construa um caminho na origem; um para ter a informação para estabelecer o caminho em qualquer *switch*; um mecanismo de suporte de encaminhamento explícito e outro para a reserva de recursos.

O estabelecimento do caminho é realizado através do algoritmo Constrained Shortest Path First (CSPF). Para traçar o encaminhamento explícito, apesar de inicialmente o IETF apoiar dois protocolos diferentes, sendo eles o Resource Reservation Protocol - Traffic Engineering (RSVP-TE) e o Constraint-based Routing Label Distribution Protocol (CR-LDP), o último já não é suportado [3].

O RSVP-TE é uma adaptação do RSVP, com o objetivo de permitir a sua utilização para esta função do MPLS. Ao passo que no RSVP, tal como referido em 2.2.3, ve o seu encaminhamento ser feito através das tabelas dos *switches*, nesta adaptação é pretendido que um caminho pré-definido através de outro protocolo, por exemplo o CSPF, seja utilizado. Para que tal aconteça, um objeto chamado Explicit Route Objects (ERO), contendo o caminho explicito, é anexado à mensagem PATH que este protocolo possui.

2.2.5 Classificação de tráfego

Ao falarmos de QoS torna-se importante a classificação de tráfego. É a diferenciação que possibilita assegurar melhores garantias a uma ligação em comparação com outra.

2.2.5.1 Classificação por porta

Inicialmente uma das técnicas mais comuns para a classificação de tráfego passava por saber que porta estava a ser utilizada e o seu respetivo registo para utilização no Internet Assigned Numbers Authority (IANA).

Nos dias de hoje, as aplicações tentam cada vez mais esconder o que elas são com o intuito de fugir às restrições impostas ao seu tipo de tráfego, sendo que uma das aplicações mais populares a querer omitir a sua identidade são as aplicações P2P [34].

Como consequência, este tipo de classificação tem uma precisão menor que 70% mas continua, porém, a ser utilizado devido a ser o método mais rápido e simples de classificação de tráfego [37].

2.2.5.2 Classificação por conteúdo

Outra técnica utilizada para classificação de tráfego são os classificadores que avaliam o conteúdo do pacote, sendo que à sua abordagem se dá o nome de *Payload-based approach*.

Este tipo de abordagem vai mais longe que o anterior, sendo que analisa o conteúdo do pacote em estudo para conseguir calcular que tipo de pacote se trata. Para que tal aconteça é necessário que as características das diversas aplicações que pretendem ser classificadas tenham sido anteriormente analisadas e extraídas criando o que se chama de assinatura da aplicação. Esta extração pode ser feita manualmente ou automaticamente [28].

A classificação por conteúdo é um dos métodos mais precisos no que toca à classificação de tráfego, chegando a atingir valores próximos dos 100%, dependendo também da qualidade da assinatura da aplicação criada. No entanto, necessita de um grande processamento por parte dos *switches* o que se reflete numa implementação dispendiosa e difícil em grandes redes [52]. Outro problema que este tipo de classificação traz são os problemas de privacidade, pois são analisados os conteúdos dos pacotes de utilizadores.

2.2.5.3 Classificação por Flow

A classificação por *flows* tem como base as características dos *flows* a classificar. As mais utilizadas são a duração do *flow*, o número de pacotes transmitidos e o intervalo médio entre a chegada destes [6].

O tempo que leva a classificar o tráfego é um dos grandes problemas deste tipo de classificadores. Quanto maior for o tempo a analisar o *flow* maior será a precisão da avaliação. No entanto, existe um interesse limitado em calcular a classificação de um *flow* quando estiver terminado [6]. É necessário então calibrar devidamente o controlador para não analisar durante demasiado tempo o *flow*, correndo o risco de este terminar,

mas também de o analisar durante tempo suficiente para obter uma maior eficiência na classificação.

2.3 Características do Tráfego

Apesar do tráfego de uma rede IP ser realizado através de pacotes, já não basta garantir que cada um deles chegue ao seu destino. Na Internet de hoje cada aplicação possui requisitos diferentes com consequências sobre o modo como os pacotes circulam na rede.

Se em 2.2 se estudou alguns dos mecanismos utilizados para conseguir dar às aplicações as garantias que necessitam, esta secção aborda a composição do tráfego da rede e qual a razão dessas proteções serem necessárias.

A Sandvine [55] realizou em 2014 um estudo sobre a utilização da Internet [58]. Através desse estudo, é possível compreender os hábitos de consumo da Internet.

As principais categorias identificadas foram as seguintes:

- **P2P** - Tráfego P2P. Ex.: *BitTorrent*
- **Web** - Páginas de internet, inclusive alojamento de ficheiros mas excluindo streaming de vídeos e áudio. Ex.: www.google.pt, Dropbox.
- **Streaming** - *Streaming* de vídeos e de áudio. Ex.: Youtube, NetFlix.
- **VoIP** - Voz sobre IP. Ex.: Skype.

No entanto nem todas as classes possuem o mesmo volume de tráfego. Olhando para a tabela 2.1, verificamos que dependendo da região, os hábitos de consumos são diferentes.

Classe	Europa	América do Norte	América do Sul	Ásia
P2P	16,28%	5,63%	12,63%	35,63%
Web	32,88%	19,67%	34,40%	14,41%
Streaming	38,15%	63,25%	41,35%	40,23%
VoIP	4,35%	< 1%	1,74%	2,25%
Outro	8,34%	11,45%	9,88%	7,48%

Tabela 2.1: Proporção de classes por região. Dados retirados de [58]

Contudo, é preciso considerar que o que é realidade num ano pode não ser realidade noutro. Comparando os resultados obtidos no ano de 2009, em [60], com os obtidos mais recentemente em 2014 [58], verifica-se a alteração das caraterísticas do tráfego com o tempo.

Outra característica importante é o facto dos dados referidos serem uma média. Durante, eventos desportivos ou épocas de férias por exemplo, os hábitos de consumo são diferentes dos habituais, como se verifica na figura 2.6, em que se pode visualizar a comparação da utilização da rede entre um período de aulas e um período de férias. Durante

a sua ocorrência, o tráfego exibe um maior consumo desde mais cedo e até mais tarde do que o habitual. Outro aspeto interessante é o facto de ser possível verificar que na noite de Natal, exibe um valor menor do que o habitual.

Torna-se assim importante criar uma rede que consiga acomodar este comportamento flutuante que a rede demonstra.

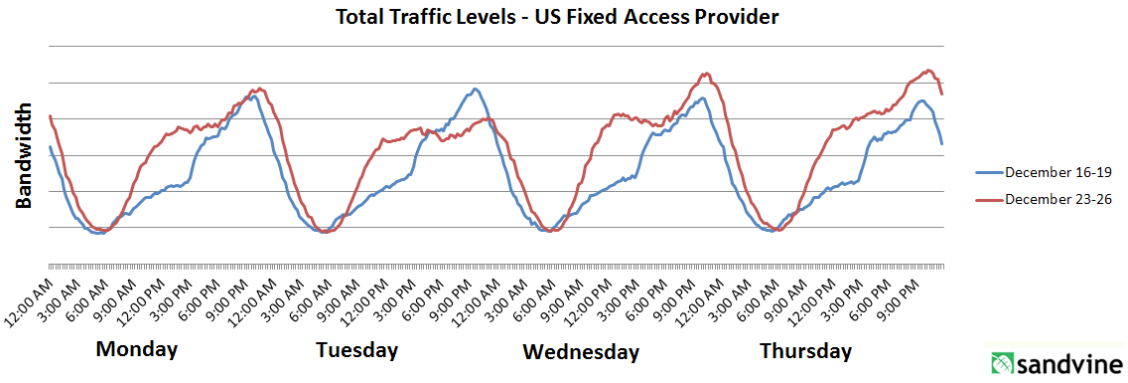

Figura 2.6: Variação do tráfego durante um período de férias. Adaptado de [59]

Vão-se aprofundar de seguida as principais categorias: o P2P, o Web, o *streaming* de vídeos e áudio e o VoIP.

Apesar de o VoIP possuir um peso menor em comparação com as restantes classes, olhando para a tabela 2.1, também será analisado com um maior detalhe devido a ser uma classe com necessidades mais específicas. Durante esta análise, é tida em consideração o tráfego europeu, devido a ser o local geográfico onde este trabalho se encontra a ser realizado.

2.3.1 Tráfego P2P

O tráfego P2P já foi o tráfego com maior peso na Internet, com uma percentagem superior a 50%. No entanto, a sua percentagem tem vindo a diminuir ao longo do tempo devido ao aumento dos serviços de *streaming* e de serviços de alojamento de ficheiros [60].

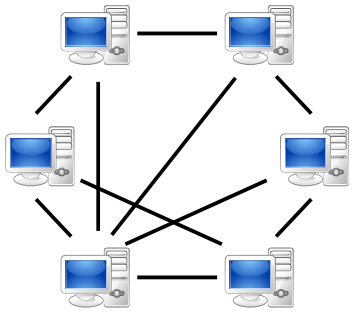

Figura 2.7: Arquitetura de uma rede P2P. Adaptado de [7]

A estrutura descentralizada, representada na figura 2.7, deste tipo de tráfego leva a

que as conexões, ao efetuar a transferência de um ficheiro, sejam realizadas diretamente a outros utilizadores criando uma rede de ligações semelhante à representada. Teoricamente, quanto maior for o número de ligações, maior será a velocidade de transferência, e maior será o número de *flows*, levando a que este tipo de tráfego seja muito intensivo, quer em termos de largura de banda, quer em termos de processamento, pois múltiplas conexões são estabelecidas por cada um dos clientes. No entanto, não necessita de nenhum tratamento em especial, sendo que *delay* na rede ou a perda de pacotes não lhe trará grande impacto, para além da diminuição da taxa de transferência.

2.3.2 Tráfego Web

O tráfego *web* tem mantido uma posição de relevância na rede ao longo do tempo [38, 58, 60]. Corresponde ao tráfego que é gerado ao visualizar páginas na Internet, incluindo redes sociais como o Facebook e páginas de armazenamento de ficheiros tais como a Dropbox, mas excluindo sites de visualização de vídeos tais como o YouTube (esse tipo de tráfego será analisado em 2.3.3).

Tem como principais características o facto de grande parte das suas ligações terem durações curtas, dando-se o nome de *dragonflies*, sendo que a percentagem de flows cuja duração é menor que dois segundos é aproximadamente de 30% [9, 38].

Apesar de em termos de largura de banda este não ser o tráfego mais exigente, em relação aos outros tratados neste documento, este deverá possuir um *delay* inicial curto. Isto acontece pois o utilizador comum irá sentir uma degradação da sua experiência caso as páginas que quer visitar demorarem demasiado tempo a abrir, levando a que o utilizador passe a consumir menos o serviço [50].

2.3.3 Tráfego Streaming

O tráfego de visualização ou audição de conteúdos em tempo real, mais vulgarmente conhecido por *streaming*, é o tipo de tráfego que tem registado um maior crescimento nos últimos anos [58]. Um dos principais serviços de *streaming*, e uma das razões da expansão deste tipo de tráfego, é o YouTube. A figura 2.8 representa o crescimento que a plataforma tem obtido com o passar do tempo. Observando-a verifica-se que grande percentagem de todo o tráfego na Internet é gerado apenas devido a este serviço e tambem que o seu volume tem sofrido uma evolução ao longo dos anos, principalmente nas plataformas móveis.

Existe uma tendência para que o tráfego *streaming* venha ocupar uma parte da rede cada vez maior. Uma das causas é o aumento médio da qualidade dos vídeos transmitidos, sendo que a visualização de videos em FullHD ou 2k corresponde a um aumento de até 50% da utilização da largura de banda em relação aos vídeos normais, e estes são cada vez mais comuns [57].

Um dos problemas deste tipo de tráfego é que necessita de várias características para que o utilizador consiga obter uma boa experiência de visualização, tornando-se assim

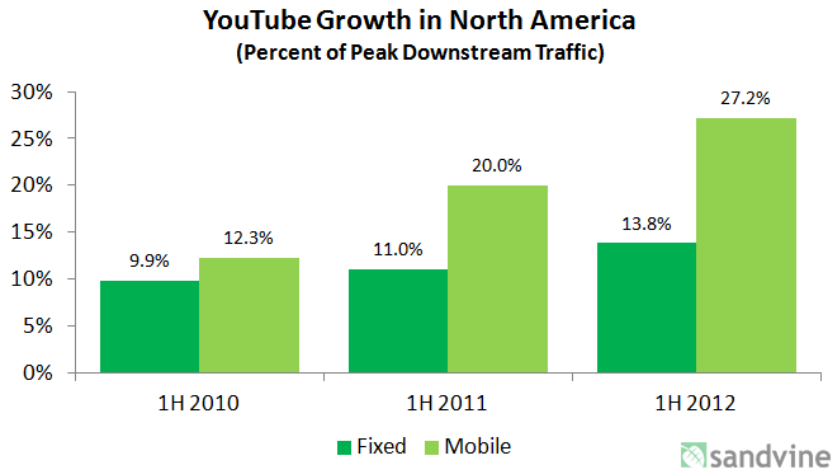

Figura 2.8: Percentagem de tráfego de origem do YouTube durante a hora de ponta na América do Norte. Adaptado de [56]

muito exigente para a rede. Necessita de uma grande largura de banda e um baixo número de pacotes perdidos, caindo sobre o risco de perda de qualidade caso o seu valor seja elevado.

Outro problema deste tipo de tráfego é o seu funcionamento. Por vezes denota um comportamento do estilo *On-Off*, representado na figura 2.9.

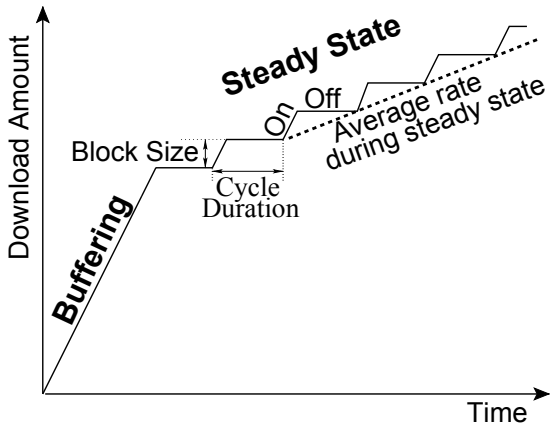

Figura 2.9: Tráfego *On-Off*. Adaptado de [51]

Este comportamento provoca que, ao invés de utilizar uma largura de banda fixa ao longo do tempo, tenha um comportamento de *burstiness*. Apenas pequenas sessões do vídeo ou áudio são descarregadas momentaneamente durante o *buferring* e períodos *on*, não existindo transferência de dados nos restantes períodos (*off*).

Deste modo pode ser criada uma ineficiência na rede, podendo se estar a reservar largura de banda que na verdade não se encontra em uso. Um estudo aprofundado sobre as características deste tipo de tráfego pode ser encontrado em [51].

2.3.4 Tráfego VoIP

O tráfego Voice over Internet Protocol (VoIP), apesar de representar uma pequena fatia de todo o tráfego da Internet, possui certas características interessantes que tornam relevante o seu estudo. O VoIP necessita de um fluxo contínuo de dados que deverá ter um baixo *jitter*, ou seja, não deverá existir uma variação no tempo esperado de receção entre cada pacote.

O atraso na receção de pacotes é outro dos elementos importantes neste tipo de tráfego. Segundo a Cisco, o valor de atraso ideal para uma chamada VoIP possuir qualidade, falando em termos de atraso, é um valor inferior a 150 milissegundos [10]. No entanto é referido que um valor entre 150 e 400 milissegundos é tolerável caso o utilizador tenha consciência desse facto, apesar da experiência do utilizador já ser considerada afetada. Qualquer valor acima de 400 milissegundos é considerado inaceitável.

O número de pacotes perdidos também deverá ser idealmente zero, de modo a manter a qualidade da chamada. Ao se perderem pacotes a qualidade da chamada começará a piorar, dificultando a conversa.

No entanto, este tipo de tráfego não necessita de um grande valor de largura de banda. O valor necessário varia conforme a codificação e o *bit rate* utilizado na chamada, mas por norma situa-se abaixo dos 100Kbs [68].

CAPÍTULO 3

DESCRIÇÃO DO PROJETO

3.1 Introdução

Neste capítulo é descrita a arquitetura desenvolvida. Os pontos mais importantes tomados em consideração foram os seguintes:

- Garantir qualidade de serviço;
- Minimizar o processamento dos controladores;
- Remover o estado da zona central da rede;
- Escalabilidade.

Tendo em vista esses pontos, foram desenvolvidas duas lógicas de controladores diferentes, sendo uma destinada a controlar o núcleo da rede e outra a zona de ingresso.

A razão pela qual existe um processamento separado deve-se ao facto de no núcleo da rede existir um grande número de *flows*. O controlo individual dos *flows*, como se verificou no caso do *IntServ*, torna-se impraticável pois provocaria uma sobrecarga nos controladores situados na zona central, que necessitariam de lidar com um elevado número de *flows*, tornando a rede não escalável.

Os controladores de ingresso, por sua vez, ao lidarem com uma menor quantidade de *flows*, podem manter esse nível de controlo. Possuem a tarefa de identificar os pacotes que por eles passam, retirando o processamento na zona central da rede.

No entanto, existe uma necessidade de passar a identificação realizada pelos controladores de ingresso para os controladores na zona central. Tal deveria ser feito colocando alguma informação adicional no pacote.

O protocolo OpenFlow, na sua versão 1.3, não possibilita a adição de cabeçalhos customizados e os controladores que utilizam o Floodight não possuem suporte para

a utilização de *tags* MPLS. A solução encontrada para passar a identificação de um controlador para outro foi através da utilização de *tags vlan*. Deste modo, estas *tags* não estão a ser utilizadas para o seu fim habitual mas sim como identificadores especiais dentro da rede. Estas *tags* nunca saem para fora da rede.

Foram definidas diferentes tipos de *vlan*, cada uma com um significado único. As diferentes existentes, encontram-se identificadas na tabela 3.1

Vlan	Funcionalidade
0x	Pacote pertencente à classe 1 com destino x
1x	Pacote pertencente à classe 2 com destino x
2x	Pacote pertencente à classe 3 com destino x
3x	Pacote pertencente à classe 4 com destino x
4x	Pacote que possui a resposta a um pedido de reserva de largura de banda originado por x
99	Pacote que possui um pedido de reserva de largura de banda com destino a x
100	Pacote transportador de uma mensagem com a largura de banda total reservada num controlador

Tabela 3.1: Funcionalidade das diversas vlans.

Tal como referido na tabela 3.1, cada uma das *vlans* possui um significado diferente. Para simplificar o encaminhamento no interior da rede, todos os *hosts* existentes foram identificados de 1 a 6. Deste modo um *flow* será encaminhado para o *host* cujo valor se encontra identificado na casa das unidades, identificados na tabela por X, da *tag vlan* que este possuí. Por exemplo uma *tag vlan* 34, corresponde a um *flow* cujo destino é o *host* que se encontra na posição 4.

Já a casa das dezenas, representa qual a classe a qual este *flow* pertence ou um pacote de comunicação entre controladores. Se utilizarmos o exemplo referido anteriormente, de uma *tag vlan 34*, sabemos agora que estamos na presença de um pacote que pertence à quarta classe com destino ao *host* 4.

No caso dos pacotes de comunicação entre controladores, o funcionamento é diferente (à exceção dos pacotes com 4 na casa das dezenas) sendo que o próprio pacote contem informação à cerca do seu destino.

É através da existência das diferentes classes que se torna possível no interior da rede a identificação do tipo de *flow* com o qual os controlador está a lidar e qual a garantia que este lhe deverá atribuir. Sendo esta classificação realizada pelos controladores de ingresso, deixa de ser necessário os controladores centrais a realizarem ou manterem um estado à cerca dos mesmos, bastando olhar para a *tag vlan* que os *flows* transportam.

3.2 Qualidade de Serviço

Atribuir qualidade de serviço a determinados *flows* é um dos objetivos deste trabalho. Como foi referido em 2.2.5, nas redes atuais existe uma tendência de aumento do processamento efetuado pelos *switches* [52] com o intuito de descobrir que tipo de aplicação está a gerar um certo *flow* para assegurar que este tem as suas necessidades asseguradas, ou negadas.

Existe ainda o problema de alguns tipos de classificadores criarem problemas de invasão de privacidade dos utilizadores [32]. Esta situação acontece pois alguns classificadores acedem à informação contida no pacote (Deep Packet Inspection (DPI)), estando assim a analisar informação privada.

Neste trabalho, para diferenciar tipos de tráfego de modo a garantir qualidade de serviço, é efetuada uma abordagem do tipo FCFS, com a exceção do tráfego VoIP por possuir características especiais. Deste modo, deixa de ser necessária uma classificação exaustiva da informação, aliviando o processamento nos *switches*, criando uma arquitetura mais escalável e não recorrendo a processos que podem ser considerados menos éticos, mantendo ainda assim garantia ao tráfego já existente. Com esse propósito, foram definidas quatro classes de serviço, em que cada uma possui um objetivo diferente:

- **Classe 1** - É automaticamente garantida qualidade de serviço;
- **Classe 2** - É garantida a qualidade de serviço ao *flow* após análise;
- **Classe 3** - Classe *Best Effort*;
- **Classe 4** - Classe de sondagem de características do *flow*;

3.2.1 Descrição de Classes

Cada uma das classes é constituída por um ou mais *flows*, sendo que dentro da mesma classe todos eles possuem a mesma prioridade.

3.2.1.1 Classe 1

Um *flow* pertencente à primeira classe tem automaticamente garantida a sua qualidade de serviço, não sendo necessário verificar as suas características. Esta classe tem como objetivo proteger as mensagens básicas para o funcionamento da rede e o tráfego VoIP.

A proteção das mensagens básicas da rede assegura que a rede tem as suas funções básicas a funcionar corretamente, mesmo estando saturada. A garantia do tráfego VoIP, tem como objetivo assegurar que este tipo de tráfego tem as suas garantias asseguradas.

Existe uma diferenciação do tráfego VoIP para os restantes pois é um tráfego bem comportado, tendo um fluxo contínuo de dados transmitidos sem grandes oscilações e de baixa largura de banda, necessitando no entanto de baixos atrasos (*delay*) e baixos atrasos relativos entre os pacotes (*jitter*). A existência de mais uma ou menos uma ligação deste

género não trará grande peso para a rede devido à baixa largura de banda, mas possui grandes vantagens para os seus utilizadores, não sendo deste modo prejudicados mesmo em alturas de elevado tráfego na rede. Estas características levaram a que este tráfego possuísse esta regalia.

A um *flow* que tenha sido atribuída a classe 1, terá como identificador uma *tag vlan* 0x, sendo-lhe assim atribuída a garantia ao longo do percurso.

3.2.1.2 Classe 2

A um *flow* que tenha sido atribuída a classe 2, é garantida qualidade de serviço durante o seu percurso ao longo da rede, sendo a única classe para além da classe 1 que tem esta regalia. A principal diferença entre esta classe e a classe 1 é que, ao invés de ser garantida automaticamente, ela é atribuída através do *switch* de ingresso, utilizando o método descrito de seguida, em 3.2.2. A sua atribuição é independente do tipo de tráfego.

Os *flows* desta classe são identificados através da *tag vlan* 1x.

3.2.1.3 Classe 3

A classe 3 é a classe *best effort* deste trabalho. É a classe menos prioritária entre as quatro classes existentes.

A um *flow* atribuído a esta classe, não é assegurada nenhuma garantia. Tal como na classe 2, um *flow* é atribuído a esta classe através do *switch* de ingresso. No entanto, ao contrário do que sucede na classe 2, esta é atribuída quando algum dos *switches* na rota deste *flow* não lhe consegue assegurar as garantias.

O seu identificador é a uma *tag vlan* 2x.

3.2.1.4 Classe 4

Ao chegar a um *switch* de ingresso, todos os *flows* que não sejam considerados classe 1, são colocados nesta classe.

É uma classe de sondagem, sendo que nenhuma garantia é atribuída a nenhum dos seus *flows*. No entanto, possui maiores garantias do que a classe 3. Significa isto que, mesmo estando a rede sobrelotada, desde que o tráfego com prioridade não seja prejudicado, um *flow* nesta classe consegue demonstrar os seus requisitos de largura de banda, tornando assim a atribuição de reserva para o *flow* por parte do *switch* de ingresso mais correta.

É uma classe transitória, onde se encontram todos os *flows* que ainda não viram a sua classe final ser atribuída. Esta é atribuída pelos *switches*/controladores de ingresso.

Um *flow* nesta classe contém uma *tag vlan* do estilo 3x.

3.2.2 Atribuição de classe

A atribuição de classes é feita unicamente pelo *switch*/controlador de ingresso, consultando no entanto os restante *switches* existentes no percurso do *flow*. Após ser atribuída uma

determinada classe a um determinado *flow*, esta não será reavaliada, permanecendo o *flow* associado à classe inicialmente definida até que deixe de existir.

Um *switch* de ingresso ao receber um pacote, verifica se possui uma entrada correspondente a esse pacote na sua tabela de *flows*. Caso não exista um *match* ao pacote, significa que este ainda não foi classificado iniciado-se o processo de atribuição de classe. O funcionamento de todo o processo é observável através do fluxograma na figura 3.1.

Ao ser recebido num controlador de ingresso, um pacote é examinado para verificar se este se trata de um pacote de tráfego VoIP.

A classificação do tráfego VoIP neste trabalho é realizada através do protocolo utilizado, caso o pacote possua o protocolo SIP é considerado tráfego VoIP. Apesar do método utilizado não representar a mais correta classificação de tráfego, este não representa o elemento mais importante deste projeto, sendo que no caso da sua implementação real um classificador mais preciso deverá ser utilizado, como os identificados anteriormente em 2.2.5. No entanto, este procedimento baseado na identificação do protocolo SIP demonstrou-se suficiente para a realização das simulações de teste do projeto.

A classe 1 é atribuída ao *flow* caso ele seja classificado como um tráfego VoIP, sendo atribuída a classe 4 nos restantes casos. No primeiro caso o pacote é enviado para o *switch* e o controlador deixa, a partir deste momento, de manter qualquer controlo individual sobre este *flow*, sabendo apenas o agregado de todos os *flows* nesta classe. Caso seja atribuída a classe 4, o pacote é enviado para o *switch*, como no caso anterior, mas o controlador continuará a analisar o seu comportamento.

A próxima ação do controlador em relação a este *flow* será apenas tomada passados 5 segundos. A razão desta espera deve-se ao facto de uma considerável quantidade de *flows* na Internet, cerca de 30%, serem de curta duração [9, 38], dando assim tempo para que estes *flows* esporádicos terminem, evitando todo o tráfego e processamento gerado durante a alocação de reservas de um *flow*.

Após esse tempo de espera o controlador verifica se o *flow* ainda se mantém ativo. Este controlo é possível devido à *cookie* (2.1.1) correspondente a este *flow* ter sido guardada e ao OpenFlow permitir recolher informação sobre o estado dos diferentes *flows* ativos através de uma mensagem OFFlowStatsRequest. Em resposta a esta, é recebida uma mensagem OFFlowStatsReply, contendo informação da duração do *flow*, o número de bytes transmitidos por ele, entre outros.

Para confirmar se um *flow* se encontra ativo verifica-se se o número de bytes transmitidos, antes e após a pausa, se mantém igual. Caso os valores sejam iguais, é considerado que o *flow* se encontra inativo e este é colocado na classe 3, não lhe sendo atribuídas garantias, caso contrário, o comportamento do *flow* continua a ser estudado.

Com o intuito de obter a taxa média de transferência que o *flow* em análise necessita, são guardados os números de bytes que esse *flow* já transmitiu em dois instantes diferentes, de modo a se tornar possível efetuar uma conta da média da taxa de transferência que o *flow* requer. O tempo de espera entre cada uma das leituras teve de ser ponderado, pois um tempo demasiado curto leva a leituras incorretas da taxa e um tempo demasiado longo

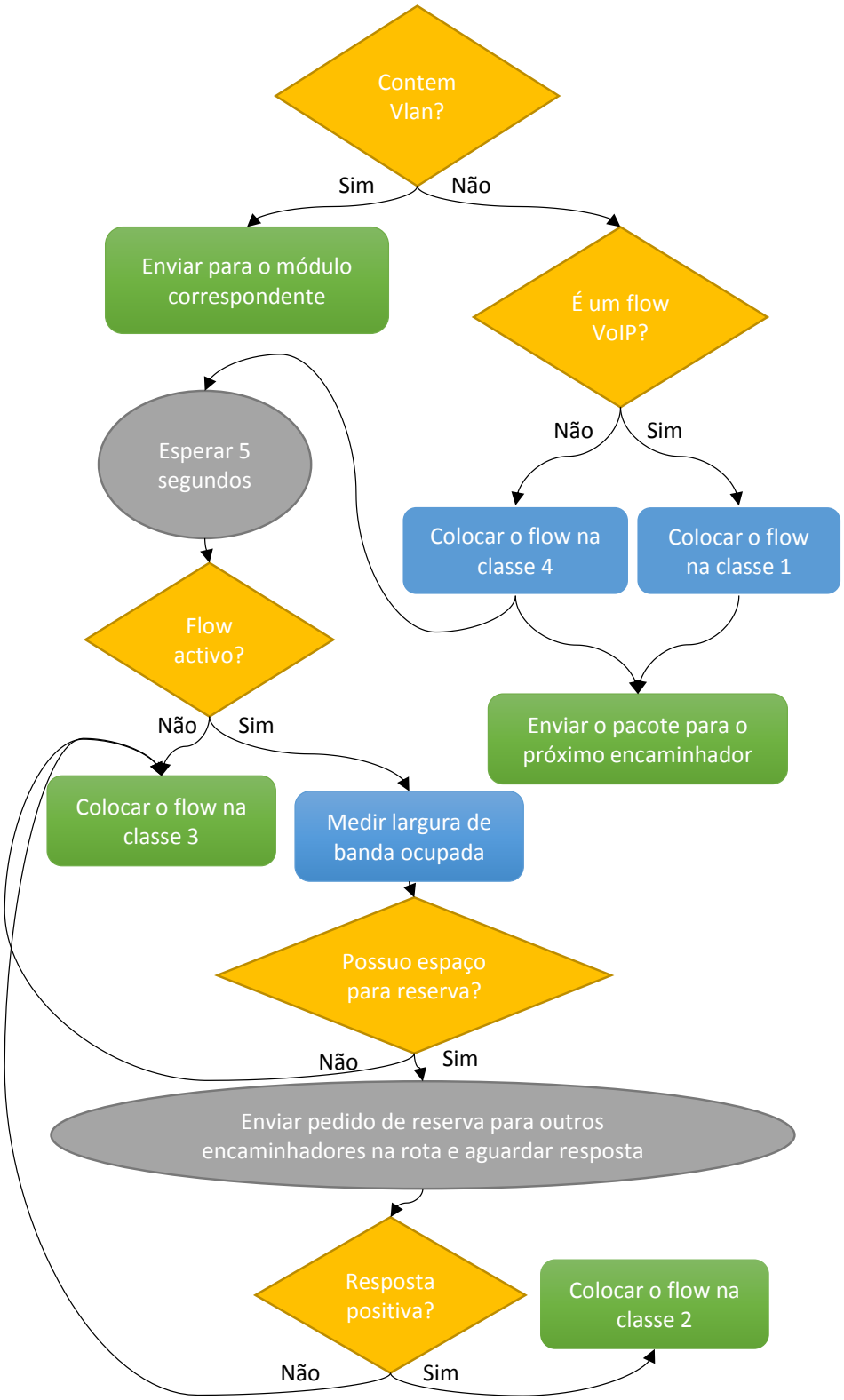

Figura 3.1: Fluxograma do funcionamento da classificação de um pacote.

leva a que o processo de classificação demore demasiado tempo, correndo o risco do *flow* terminar enquanto este ainda se encontra a decorrer. É utilizado um tempo de espera de 1 segundo entre cada leitura, correspondendo ao intervalo por defeito de um *ping* entre duas máquinas.

Possuindo a taxa média de transferência do *flow*, é verificado se o *switch* possui largura de banda suficiente na classe 2, de modo a albergar este novo *flow* com as suas garantias asseguradas. Esta verificação é efetuada utilizando dois métodos diferentes, representados na figura 3.2.

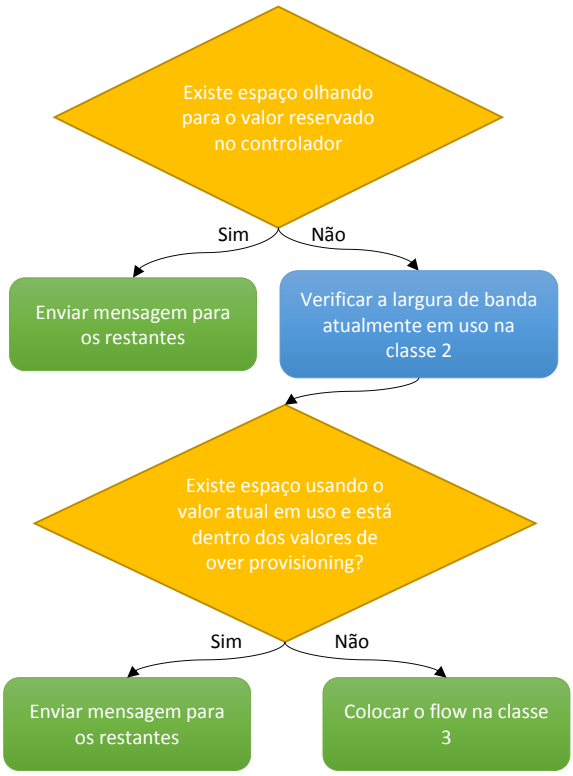

Figura 3.2: Fluxograma da verificação de largura de banda num controlador.

O primeiro método utiliza o valor da largura de banda que o controlador vai mantendo guardado. Esta é proveniente da informação das reservas que ele próprio já efetuou mas também da informação periódica recebida de outros controladores, contendo o valor agregado de todas as reservas destes.

A necessidade de se saber especificamente a largura de banda guardada provém do facto do volume de tráfego numa rede não ser constante no tempo, existindo momentos em que este demonstra uma utilização menor que outros. Deste modo, esta característica do tráfego é tida em consideração com o intuito de evitar que sejam reservados mais recursos do que os suportados.

Se o controlador verificar que existe largura de banda suficiente para assegurar este novo *flow*, são guardadas as suas características no controlador, o pedido de reserva é reenviado para o próximo *switch* através de um pacote de pedido de reserva (3.3.1.1) e

aguarda a receção da resposta à reserva. Caso não possua espaço para o *flow*, é utilizado o segundo método.

O segundo método verifica qual a largura de banda com qualidade assegurada (a de classe 2) que atualmente se encontra em utilização. Este valor é obtido diretamente do *switch* a pedido do controlador. Caso o valor obtido seja superior numa certa margem, chamada de margem de excesso, relativamente ao valor de largura de banda que o controlador possui guardado, é considerado que o *switch* apresenta capacidade para o novo *flow*.

Ao efetuar este procedimento, o *switch* é colocado num estado de *over provisioning*, em que é garantida uma maior largura de banda do que a que realmente possui. Deste modo aumenta-se a taxa de utilização de um *link*, sendo reutilizada uma parte da ligação com recursos que não estavam em utilização mas que estavam reservados. Em contrapartida, quanto maior for a margem de excesso, maior será o risco do *switch* não conseguir assegurar a largura de banda a que se comprometeu.

Caso utilizando este método o controlador verifique que suporta o novo *flow*, as suas caraterísticas são salvas na base de dados do controlador, é enviado um pedido de reserva para os restantes *switches* e uma resposta ao pedido é aguardada. Caso esta não seja aceite, o *flow* é colocado na classe 3, não lhe assegurando nenhuma garantia.

O pedido de reserva vai sendo recebido pelos controladores dos *switches* que se encontram na rota até chegar ao *switch* destino (*egress*).

Todos os *switches* na rota passam pelo processo descrito anteriormente, representado na figura 3.2. A única diferença é, ao invés do *flow* ser colocado na classe 3, é enviada uma mensagem com a sinalização de que a reserva não é suportada de volta ao *switch* que inicialmente criou o pedido. No caso de ser suportada, a mensagem de pedido de reserva é encaminhada sucessivamente até esta chegar ao destino.

O controlador de ingresso que criou o pedido, ao receber a resposta da mensagem de reserva, identificada pela *tag vlan* 4x, verifica se a rota pode ser estabelecida através do campo "*available*"que a mensagem contém.

No caso da resposta ser negativa, identificado pelo campo "*available*"com o valor *false*, o *flow* é colocado na classe 3 e a suas características anteriormente salvas são apagadas da base de dados.

Ao ser bem sucedida, sendo o valor do campo "*true*", o *flow* é colocado na classe 2 e o controlador mantém a informação sobre ele guardada.

Esta informação apenas será apagada quando este *flow* deixar de existir, informação esta que é passada ao controlador ao receber uma mensagem OpenFlow do tipo *Flow-Deleted*, que contém o *cookie* identificador da *flow* apagada.

3.2.3 Informação guardada

Como verificado em 3.2.2, existe uma necessidade de guardar informação nos controladores para assegurar qualidade de serviço na rede. São guardadas informações acerca dos

flows que possuem garantias de serviço atribuídas e do agregado dos valores de reservas estabelecido por controladores.

3.2.3.1 *Flows* com garantias

A informação sobre os *flows* que possuem garantias, ou seja, que tiveram atribuída a classe 2, é salva unicamente no controlador de ingresso desse *flow*. Para salvar esta informação é criada uma entrada na base de dados global do Floodlight, significando que todos os módulos de um controlador conseguem visualizar esta informação. No entanto, é importante referir que apenas o controlador que guardou essa informação a consegue obter, sendo que os restantes controladores não possuem acesso a ela.

Uma nova entrada na base de dados é criada cada vez que um pedido de reserva de um *flow* é criado. Caso o pedido de reserva falhe, esta informação é eliminada.

A estrutura de uma entrada encontra-se representada na figura 3.3.

Nome	Cookie	Origem	Destino	Valor Reservado
Tipo	long	int	int	float

Figura 3.3: Estrutura de uma entrada na base de dados de um *flow* garantido.

Existe a necessidade de guardar informação sobre os *flows* ativos num controlador, pois é necessário saber qual a taxa média de transferência de cada um para poder calcular a largura de banda ocupada pelo seu agregado. Um agregado corresponde ao somatório de todas as taxas médias de transferência para um determinado destino. Esta informação é enviada de 5 em 5 segundos para cada um dos controladores que se encontram na rota desse destino.

Quando um *flow* é removido do *switch*, o controlador recebe uma mensagem *Flow-Removed*, contendo a sua respetiva *cookie* e a entrada correspondente é eliminada da base de dados. Deste modo é apenas mantido o controlo dos *flows* que se encontrem atualmente ativos num *switch*.

3.2.3.2 Estado de reservas

A informação sobre o estado das reservas é obtida pelos controladores ao receberem uma mensagem de informação sobre as reservas (3.3.1.2). Tal como a anterior, é guardada na base de dados global do Floodlight para ser acedida por todos os módulos.

A informação é atualizada ao receber uma mensagem com uma *tag vlan* 100, contendo informação sobre o agregado da largura de banda reservada de um determinado *switch* de ingresso para outro de destino.

Esta informação tem como objetivo os controladores centrais conseguirem manter o controlo da quantidade de largura de banda que foi reservada. Tal é importante pois os *flows* por vezes não mantêm um comportamento constante ao longo do tempo (consultar 2.3) tornado uma simples leitura da largura de banda atualmente em uso não confiável.

Orig\Dst	1	2
1	x	x
2	x	x

Figura 3.4: Organização das reservas guardadas.

Estes dados são guardados numa tabela com a estrutura representada em 3.4, sendo o número de origens e destinos variável conforme a topologia criada. A mensagem recebida possui nos seus campos informação sobre quem a criou e o valor total reservado por essa origem para um destino. Sempre que uma nova mensagem é recebida, os valores correspondentes da tabela são atualizados.

Esta informação é utilizada pelos controladores para calcularem qual a largura de banda total reservada para uma determinada porta. O valor total reservado é a soma dos valores de todas as colunas cuja rota para o destino passa por essa porta.

3.3 Arquitetura dos controladores

3.3.1 Comunicação entre controladores

A comunicação existente entre os diferentes controladores na rede é efetuada com mensagens especialmente criadas para esse fim. Existem dois tipos de mensagens que podem ser trocadas: para o estabelecimento de uma reserva, ou para informar um controlador sobre o estado das reservas atuais de outro.

3.3.1.1 Estabelecimento de uma reserva

Sempre que existe a necessidade de efetuar uma reserva para um *flow*, ocorre uma troca de mensagens entre os controladores na sua rota para saber se é possível atribuir garantias a esse *flow*.

Esta mensagem é enviada através de um pacote identificado pela *vlan* 99, caso seja o pedido inicial, ou pela *vlan* 4x (em que x é o criador da mensagem) caso contenha a resposta final. A estrutura da mensagem encontra-se detalhada na tabela 3.2.

O pacote possui diversos campos identificadores do *flow*. Neste caso, em vez de ser utilizada uma abordagem *5-tupple*, em que é guardada a informação do IP, a porta de origem e destino e o protocolo de transporte, é utilizado um método mais simples, reduzindo assim o tamanho do pacote e o consequente *overhead* criado.

Nome	srcVlan	srcPort	dstVlan	dstPort	Requirement	Available	Cookie	startTime
Tipo	int	int	int	int	float	boolean	long	long

Tabela 3.2: Estrutura de um pacote para efetuar o estabelecimento de uma reserva.

É apenas guardada a origem e o destino, através das vlans identificativas destes, e a porta do protocolo de transporte de origem e destino. No pacote é ainda incluída qual a *cookie* identificadora do *flow*.

A *cookie* faz parte do protocolo OpenFlow e representa um identificador criado pelo controlador para um ou mais flows. Neste projeto a sua função será a identificação de um *flow*, tornando-se assim num identificador único para cada um.

Apesar da parte identificadora ser apenas necessária no *switch* de ingresso, visto o núcleo da rede não guardar o estado, a sua transmissão deve-se ao facto de, ao receber a resposta, ser necessário conhecer qual o *flow* que a originou.

O campo *requirement* corresponde ao valor de largura de banda que o *flow* que motivou a reserva necessita para ser assegurada a sua qualidade de serviço. Este possui o seu valor em *megabits*.

Caso o *switch* verifique que possui espaço suficiente para um *flow* com a largura de banda requisitada, o pacote continuará a ser encaminhado até chegar ao seu destino. Caso contrário, o campo "*available*"passa a ter o seu valor a *false*, a vlan do pacote será modificada para 4x, e o pacote é encaminhado de regresso à origem.

O tempo decorrido para o processamento da reserva pode ser calculado através do campo *startTime*, cujo valor corresponde às horas em que o pacote de reserva foi criado.

3.3.1.2 Informação sobre as reservas

Os controladores necessitam de saber qual é o valor total de largura de banda reservada para assegurar que não é reservado um valor de tráfego com qualidade garantida maior que a capacidade da ligação. Esta informação é passada através de uma mensagem identificada pela *vlan* 100 e é criada pelos controladores de ingresso.

Idealmente esta mensagem iria conter o agregado da largura de banda total reservada no controlador criador da mensagem para todos os seus destinos. No entanto, neste trabalho foi implementada uma versão simplificada, criado-se uma mensagem individual por destino.

Nome	srcVlan	srcPort	reservedValue
Tipo	int	int	float

Tabela 3.3: Estrutura de uma mensagem para informar um *switch* sobre o estado das reservas.

Na tabela 3.3 é possível encontrar a estrutura de uma mensagem deste tipo, que contém a sua origem, através da *srcVlan*, e qual o valor total das reservas que essa origem tem, valor representado em *megabits* no campo *reservedValue*, para um certo destino.

3.3.2 Controlador de Ingresso

É no controlador de ingresso que se concentra a maior parte do processamento, com o fim de retirá-lo do núcleo da rede. Na figura 3.5, encontramos um fluxograma do esquema de funcionamento de um controlador de ingresso. É possível verificar que a complexidade é superior ao fluxograma de um controlador central (figura 3.6), criando o efeito esperado, e pretendido, desta topologia de duas zonas.

Figura 3.5: Fluxograma do funcionamento de um controlador de ingresso.

Ao receber um pacote do *switch*, é verificado que tipo de mensagem OpenFlow se

trata. Ao contrário do que verifica num controlador central, em que só são recebidas mensagem do tipo *Packet-In*, aqui também são recebidas mensagens do tipo *Flow-Removed*. Esta mensagem é recebida sempre que um *flow* é removido do *switch*, tornando assim possível o controlo dos *flows* que se encontram ativos por parte do controlador. Ao recebê-la, o controlador verifica, através da sua *cookie*, se esta corresponde a um dos *flows* com espaço reservado. Caso corresponda, este *flow* é eliminado da base de dados. Caso seja um *flow* não existente, esta mensagem é ignorada.

Se a mensagem for do tipo Packet-In, existem 4 ações que podem ser tomadas, dependendo do valor da *Vlan*. Essas ações são analisadas de seguida.

3.3.2.1 Vlan 100

Se o pacote pertence à *vlan* 100, ele transporta informação acerca do estado das reservas de um *switch*. Ao ser recebido o controlador atualiza o valor de largura de banda que o criador desse pacote possui reservado e descarta o pacote, pois ele já chegou ao seu destino.

3.3.2.2 Vlan 99

Caso o pacote recebido contenha uma *tag vlan* 99, significa que transporta um pedido de estabelecimento de reserva. Ao receber este pacote, o controlador verifica se o seu *switch* possui capacidade para um novo *flow* com as características indicadas.

Existem dois métodos utilizados para verificar se o *switch* possui capacidade para o novo *flow*, definidos em 3.2.2 e representados em 3.2. Estes consistem em verificar a informação guardada no controlador sobre a largura de banda ocupada por flows com garantias, e em verificar qual a utilização atual da porta de saída necessária para estabelecer o novo *flow*. Existindo espaço suficiente para o *flow* utilizando um destes dois métodos, é considerado que o *switch* possui capacidade para o novo *flow*.

Sendo capaz de assegurar as garantias necessárias, o pacote é encaminhado para o próximo controlador na rota. Caso contrário, o pacote é retornado à origem, com a informação que não é possível estabelecer a nova rota.

3.3.2.3 Vlan 4x

Um pacote numa *vlan* 4x é indicador de uma resposta a uma mensagem de estabelecimento de reserva.

A razão pela qual a resposta tem uma *tag* diferente do pedido é devido a esta não necessitar de nenhuma resposta por parte dos *controladores* no seu caminho de regresso, sendo que as ações a serem realizadas por estes foram decididas anteriormente.

Não é necessário informar os controladores acerca do fracasso ou sucesso da reserva pois estes possuem um estado *soft* e adquirem a informação sobre as reservas estabelecidas através das mensagens provenientes dos controladores de ingresso (3.3.1.2). Esta informação ser-lhes-á informada periodicamente, de cinco em cinco segundos.

Ao receber uma mensagem com esta *tag*, o controlador verifica se o pedido obteve uma resposta positiva ou negativa, verificando o campo "*available*"que esta contém. Sendo mal sucedida, este campo possui valor *false*, e o *flow* em causa será colocado na classe 3, sem garantias de serviço, as informações do *flow* salvas anteriormente no controlador são eliminadas e nenhuma informação acerca deste é mantida. Sendo bem sucedida, o *flow* é colocado na classe 2, garantindo-lhe qualidade de serviço e mantendo os seus dados guardados na base de dados.

3.3.2.4 Sem tag Vlan

Receber um pacote sem nenhuma *tag vlan* num *switch* de ingresso significa que este teve origem numa zona exterior da rede. O pacote é encaminhado para o seu destino, colocando-lhe uma *tag vlan* provisória correspondente à classe 4, sendo o seu comportamento analisado durante 5 segundos.

Após o período de análise, é tratado de modo a lhe ser atribuída uma classe de garantias, utilizando os procedimentos descritos anteriormente em 3.2.2.

3.3.3 Controlador Central

O controlador central, tal como o nome indica, encontra-se ligado aos *switches* centrais da rede. O seu funcionamento é simplificado em relação aos controladores de ingresso, sendo esse um dos objetivos iniciais.

Na comparação entre ambos os fluxogramas, representados na figura 3.6 e 3.5, correspondendo respetivamente ao funcionamento de um controlador central e de ingresso, essa simplificação torna-se clara.

Ao receber uma mensagem OpenFlow do tipo *Packet-in*, o controlador verifica qual a sua *vlan*. Nos controladores centrais, apenas existe a possibilidade de receber um pacote pertencente à *vlan* 99 ou 100. As restantes *vlans* não são tratadas pelo controlador, pois o *switch* possui, na sua tabela de *flows*, a entrada para estas.

Caso o pacote possua a *vlan* 99, significa que se trata de uma mensagem para estabelecimento de reserva, e esta é enviada para o módulo "*fowarding*"para ser tratada.

Neste módulo, o *switch* verifica se consegue suportar os requisitos da reserva, indicados na mensagem recebida. O modo como tal é verificado foi referido anteriormente em 3.2.2. Se não existir espaço suficiente, esta mensagem será retornada ao *switch* anterior, colocando o campo "*available*"da mensagem de reserva a "false", indicando assim que as condições para o seu estabelecimento não estão garantidas. Caso exista espaço, a mensagem será reencaminhada para o próximo *switch*, sem modificações.

No caso do pacote possuir a *vlan* 100, trata-se de uma mensagem de informação sobre as reservas, sendo que a sua informação será guardada na tabela correspondente (consulte 3.2.3.2).

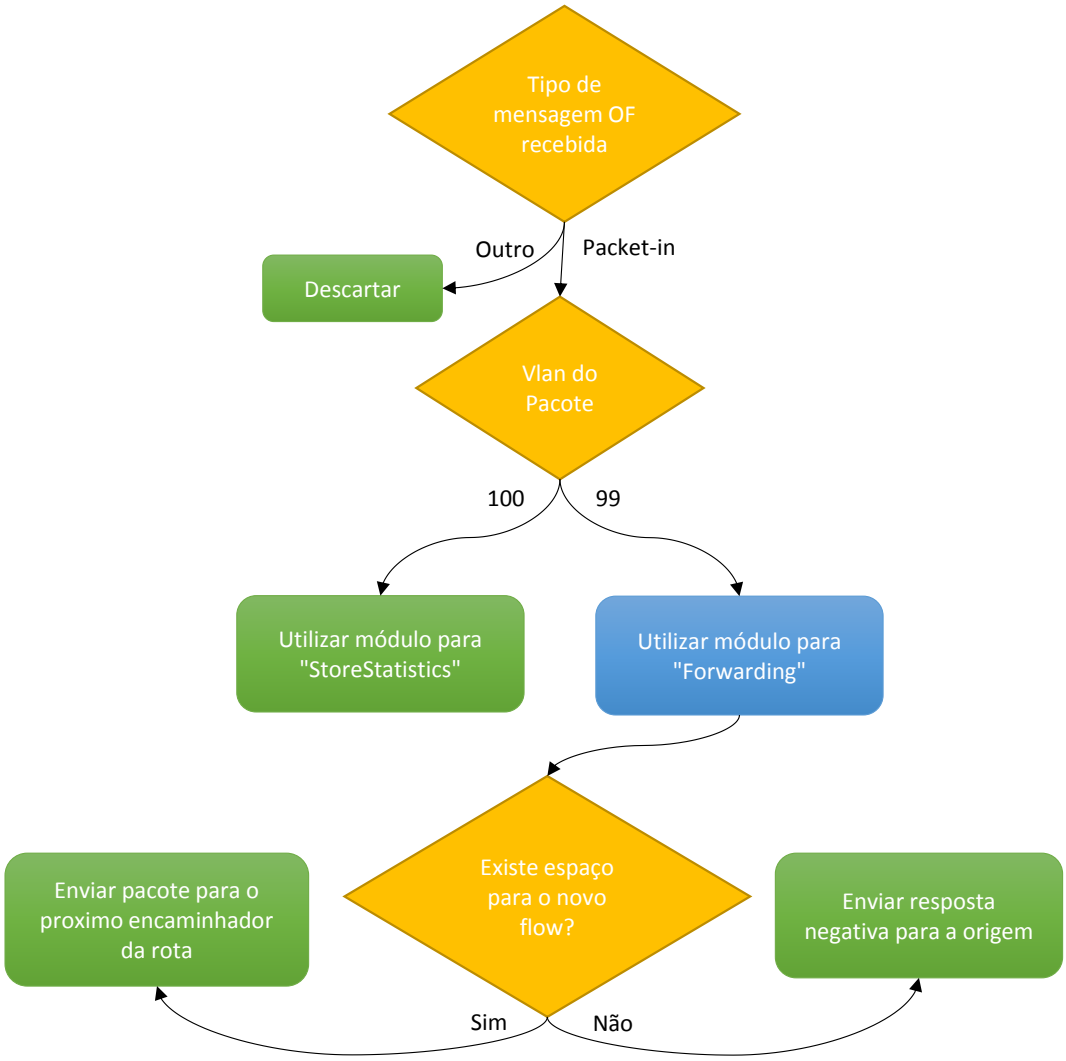

Figura 3.6: Fluxograma do funcionamento de um controlador central.

3.3.4 Encaminhamento

O encaminhamento do tráfego do projeto é fixo não se modificando conforme o estado da rede. Este comportamento, apesar de não ser ideal, foi propositado na arquitetura do controlador.

Um dos objetivos do projeto é demonstrar que é possível garantir qualidade de serviço numa rede, sem existir a necessidade de manter estado no interior desta e sem técnicas pesadas de classificação de tráfego, com o intuito de criar uma rede escalável. Caso fosse utilizado um algoritmo no *switch* ou controlador que procurasse a rota mais curta ou com menor utilização de tráfego, o foco do projeto estaria a ser perdido. Existem protocolos com esse objetivo e esses deverão ser utilizados caso venha a ser feita uma implementação numa rede real, mas neste caso, em que se pretende analisar o comportamento da rede em situações não favoráveis em que existe demasiada carga na rede, tal comportamento não é desejável.

Deste modo, o encaminhamento é realizado de modo fixo tendo em vista sobrecarregar certos troços da rede. Através das possibilidades do OVS, foram configuradas as rotas estáticas para todo o espaço interior da rede, mas não nos *switches* de ingresso. Na zona de ingresso, é desejado manter o controlo individual sobre os *flows* e não apenas no agregado como na zona central. Caso as rotas já estivessem estabelecidas tal não seria possível, pois o *switch* não iria comunicar com o controlador por já conhecer a rota.

CAPÍTULO 4

SIMULAÇÕES

4.1 Introdução

Para analisar o comportamento do sistema foram realizadas diferentes simulações.

Estas consistiram em verificar as características de um determinado troço em diferentes condições da rede, especificamente quando esta se encontra em estado de sobrecarga e num estado de carga normal. Para cada um dos momentos, os comportamentos desse troço são avaliados através das seguintes características:

- Distribuição do tráfego pelas diferentes classes
- Número de pacotes perdidos pelas diferentes classes
- O *jitter* e *delay* dos pacotes
- Utilização das ligações com garantias

Alguns elementos, tais como a eficiência na distribuição do tráfego pela rede e a capacidade de processamento dos *switches* não serão tidos em conta. Como referido anteriormente, a distribuição do tráfego foi realizada de modo manual tendo em vista saturar um certo ponto na rede e é considerado que os *switches* são capazes de lidar com a largura de banda total máxima sem provocar atrasos ao nível de processamento de pacotes.

4.2 Ambiente de Simulação

4.2.1 Controlador

O controlador escolhido foi o Floodlight pelas razões indicadas em 2.1.3.3. Foi utilizada uma versão beta da versão 1.0, ao invés de ser utilizada uma versão estável, pois esta

já possuía suporte para algumas das características das versões OpenFlow superiores à 1.0, como uma melhor gestão das filas de espera. No entanto, esta decisão trouxe alguns problemas, pois certas funcionalidades não existiam ou continham erros.

Foi necessário estabelecer contacto com o principal contribuidor do projeto para que estes erros fossem corrigidos, o que acabou por fazer com que o projeto desta tese contribuísse para o desenvolvimento do controlador.

Uma das partes em que foi necessário o seu desenvolvimento foi na obtenção das características dos *switches*. Esta parte não se encontrava completa pois não suportava a extração direta das características através dos módulos, sendo necessário utilizar a *REST API*. Encontraram-se também dificuldades no módulo da base de dados, que possuía problemas de concorrência, manifestáveis ao simular uma rede com múltiplas ligações a serem estabelecidas em simultâneo.

Para a realização da simulação, certos módulos foram alterados tornando possível verificar o seu comportamento de uma forma mais completa. O método encontrado para tal foi criar diversos pontos onde a informação é salva em ficheiros, para serem lidos e analisados posteriormente. A extração da informação para o ficheiro é realizada de 5 em 5 segundos para não sobrecarregar o controlador.

Para simplificar a simulação foram apenas utilizados dois controladores, sendo um destinado à zona central da rede e outro à zona de ingresso. No entanto, apesar de um controlador deste modo ser ligado a múltiplos *switches*, o conhecimento que o controlador possui de um *switch*, como o número de *flows* reservados, não é transmitido para outro. É assim simulada a utilização de um controlador por *switch*.

Os controladores correm em Eclipse 3.8, numa máquina com um processador Intel i5-4200u e com 8GB de RAM.

4.2.2 Topologia de Rede

Para a criação de uma rede simulada foi utilizado o Mininet na sua versão 2.1.0p2. Esta versão do Mininet possui pré-instalados *switches* do OVS que suportam a maior parte das funcionalidades do OpenFlow 1.3. Algumas das características, como as *OpenFlow Meter Bands* não são suportadas, tendo sido realizados outros métodos para substituir a sua funcionalidade (identificados em 3.2.2).

O Mininet corre numa máquina virtual de 2 núcleos e com 1GB de RAM, utilizando o VMWare.

A topologia criada encontra-se representada na figura 4.1. Possui um total de 6 hosts, ligados a 6 *switches* de ingresso. Estes encontram-se ligados entre si unicamente através da zona central da rede, onde se encontram localizados os *switches* centrais.

As ligações estabelecidas entre os *switches* possuem uma largura de banda de 80 *megabits por segundo*. Dessa largura de banda, 50 encontram-se idealmente destinados a serem utilizados pela segunda classe. Este valor pode ser aumentado caso o *switch* entre em estado de *over-provisioning*, ou reduzido caso nenhum *flow* se encontre ativo nesta e

existam outros a necessitar da largura de banda. As restantes classes não possuem valores máximos, podendo até ocupar a ligação por completo, mas nunca prejudicando *flows* na classe 1 e 2.

Um aspeto importante dos *switches* de ingresso são que estes estão ligados ao seu *host* com o mesmo número, ou seja o *switch* S1 está ligado diretamente ao H1, S2 a H2 e assim sucessivamente. O número do *host* está também ligado ao seu IP e ao seu *MAC Address*, sendo que estes se encontram especificados na tabela 4.1. Estas relações foram estabelecidas de maneira a facilitar o cálculo das rotas.

Host	Vlan	IP	MAC Address
H1	1	10.0.0.1	aa:aa:aa:aa:aa:aa
H2	2	10.0.0.2	aa:aa:aa:aa:aa:bb
H3	3	10.0.0.3	aa:aa:aa:aa:aa:cc
H4	4	10.0.0.4	aa:aa:aa:aa:aa:dd
H5	5	10.0.0.5	aa:aa:aa:aa:aa:ee
H6	6	10.0.0.6	aa:aa:aa:aa:aa:ff

Tabela 4.1: Detalhes da configuração dos *hosts*.

Em relação ao encaminhamento na topologia, como referido anteriormente, é introduzido manualmente no núcleo da rede. Este é comunicado aos *switches* centrais após estes serem criados, utilizando a Mininet API.

Ao contrário do que se pretende numa rede, nesta simulação é pretendido sobrecarregar certos pontos. Como tal, as rotas introduzidas não são por norma as mais curtas, mas sim rotas que passam por pontos específicos. Nesta simulação, o ponto analisado foi a conexão entre o *switch* 7 e o *switch* 12, sendo os resultados provenientes dessa ligação.

Todas as rotas são reversíveis, significando que o caminho de um pacote de um *host* origem para um *host* destino é o mesmo do caminho dos pacotes vindos do *host* destino para o *host* origem.

4.2.3 Geração de tráfego

A geração automática de tráfego na rede não é suportada pelo Mininet. No entanto, este suporta a utilização de outros programas que podem ser utilizados para esse fim.

Os programas utilizados foram os seguintes:

- iperf3 [30] - Utilizado para a geração da maior parte dos fluxos sendo bastante adaptável, suportando a configuração de diferentes características. Foi utilizada uma versão menos comum, iperf3, pois esta permite a limitação de largura de banda em tráfegos TCP e UDP, algo que a versão 1 não permite;

- iperf - A versão do iperf instalada por defeito na máquina mininet, tendo sida utilizada inicialmente para testes. Esta versão acabou por ser utilizada para simulações

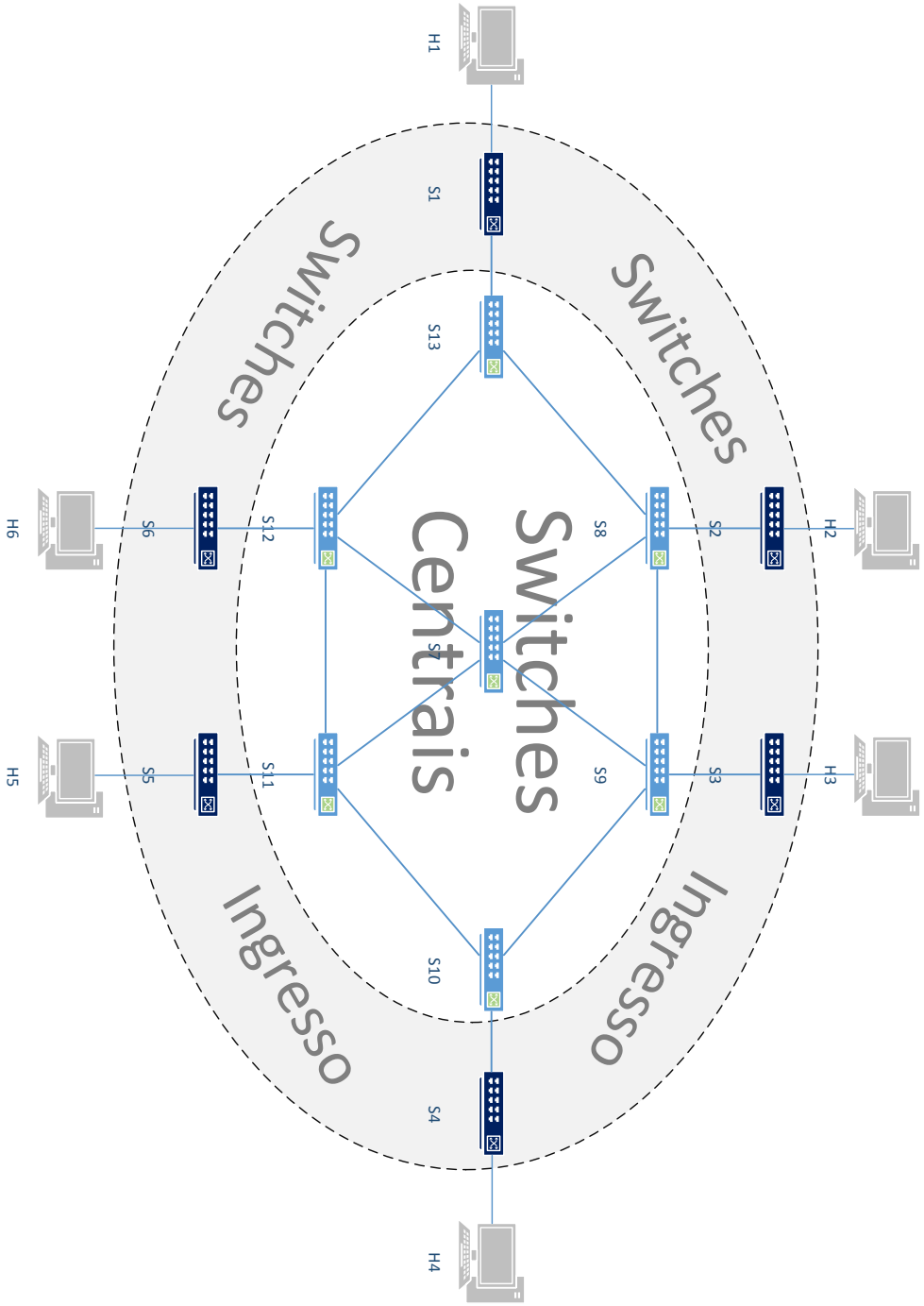

Figura 4.1: Topologia da rede criada.

de chamadas, devido a problemas na utilização do programa SIPp, e devido ao tráfego UDP gerado pelo iperf3, necessitar de uma componente TCP.

- SIPp [23] - Simulador de chamadas SIP, utilizado para recriar chamadas VoIP. Inicialmente pretendido para utilização na simulação, problemas no lançamento de múltiplas chamadas a partir de um *host* em momentos diferentes fizeram que este não pudesse ser utilizado na simulação final, mas apenas para testes;

- ping - O ping é um software que vem instalado por defeito no Mininet, tendo como objetivo testar o RTT de um *host* para outro, utilizando o protocolo ICMP. Este programa, devido à sua simplicidade e rapidez, foi muito utilizado no desenvolvimento com o intuito de testar as conexões. Foi através do ping que foram obtidos os valores de *delay* dos *flows*, tendo este sido alterado manualmente para ser atribuído às diferentes classes, dependendo da classe em estudo;

- PackETH [31] - Gerador de pacotes com uma interface gráfica. Utilizado para testar a rede durante o seu desenvolvimento devido à conveniência da interface gráfica, não sendo utilizado no entanto na simulação final.

4.2.4 Tráfego Simulado

Para a simulação de tráfego de rede foram considerados os dados anteriormente descritos em 2.3. Existem 4 tipos de tráfego que possuem maior relevo, sendo estes o tráfego P2P, web, *streaming* e VoIP.

Destes, foi efetuada uma diferenciação no caso do tráfego *streaming* e *web*. No caso do *streaming* existem dois tipos, um com uma utilização de largura banda fixa ao longo do tempo e outro com um comportamento On-Off. No tráfego *web*, como existe uma considerável quantidade de tráfego de curta duração, foram criados *flows* com uma curta duração média e outros com uma duração média longa. Estas diferenciações foram criadas para atingir condições de simulação semelhantes às características do tráfego reais.

Os valores específicos para cada um dos *flows* encontram-se representados na tabela 4.2.

Os valores apresentados correspondem ao valor médio da duração das chamadas. Para atingir um comportamento mais exemplificativo de uma rede real, os momentos para a criação de tráfego e a sua duração foram obtidos utilizando uma distribuição Exponencial e de *Poisson*, respetivamente, garantido assim valores aleatoriamente gerados mas tendendo para o valor médio pretendido.

A simulação divide-se em dois momentos, um em que a rede se encontra num estado normal, tendo capacidade para garantir qualidade de serviço a todos os *flows* existentes, e outro momento em que esta se encontra saturada, sendo que o tráfego total gerado ocupa uma largura de banda superior ao suportado pela rede. Estes dois momentos possibilitam verificar o comportamento de tráfego numa altura em que não deverão ocorrer problemas,

Tipo	Duração (segundos)	Largura de Banda (megabits)	Peso (%)
P2P	30	2	18
Web longo	61	2	26
Web curto	2	2	10
Streaming normal	143	3	21
Streaming on-off	143	1,5	21
VoIP	143	0,1	4

Tabela 4.2: Características dos diferentes tipos de tráfego a serem simulados.

e outro em que apenas o tráfego a que foram dadas garantias deverá, em princípio, ser mantido sem problemas.

Tipo de tráfego	Normal	Saturação	Total
P2P	12	82	94
Web de curta duração	95	676	771
Web de longa duração	4	62	66
Streaming On-Off	5	22	27
Streaming Normal	8	27	35
Voip	13	66	79

Tabela 4.3: Tráfego simulado (número de *flows*)

No momento inicial, correspondendo ao tráfego normal, a utilização esperada do link situa-se nos 30 *megabits por segundo*. Este valor é atingido através da criação de diferentes *flows*, descritos na tabela em 4.3. O momento de início e a duração desses *flows* são obtidos através das distribuições descritas anteriormente. Na mesma tabela encontra-se o número de *flows* gerados para a simulação do segundo momento, onde é pretendido saturar a rede gerando uma largura de banda média de 140 *megabits por segundo*.

4.2.5 Limitações da simulação

Ao tentar criar a simulação foram encontradas algumas limitações, que são abaixo descritas.

4.2.5.1 Lançamento remoto de *flows*

Existiu a necessidade de criar um ponto central de onde a simulação pudesse ser coordenada, lançando os vários *flows* com as suas características nos momentos exatos. Para o tornar possível, é necessário a comunicação dos *hosts* com a máquina que os vai lançar.

Para que a comunicação se realizasse, foi necessário a criação de um túnel por cada *host* existente ligado ao *switch* de ingresso correspondente. Foi configurado o controlador para suportar características anteriormente não requeridas pela rede, como acontece no caso dos pedidos ARP, cuja implementação não havia sido necessária.

Estando os túneis de rede criados e os controladores reconfigurados, as *routing tables* da máquina e dos respetivos *hosts* foram atualizadas. Torna-se assim possível efetuar a ligação com os *hosts* a partir de uma máquina remota utilizando o protocolo Secure Shell (SSH).

De seguida foi criado um pequeno programa em java, que se conecta aos *hosts* do Mininet, carregando os ficheiros com os dados obtidos das distribuições, transformado-os em *flows* com as características especificadas, executando-os quando necessário.

4.2.5.2 Limitar a largura de banda por *flow*

Outro problema que surgiu ao preparar a simulação deveu-se ao facto dos *switches* criados pelo OVS não suportarem as *meter-bands*. Deste modo, não é possível limitar a largura de banda ocupada por um *flow*. Ao não ser limitado, um *flow* poderia ocupar uma maior largura de banda do que originalmente estava destinado, podendo causar problemas de qualidade de serviço.

Para o problema ser resolvido, foi necessária a modificação no controlador de modo a ele poder controlar qual a largura de banda utilizada pela origem. O controlador, ao colocar um *flow* numa das classes após a conclusão da análise das suas características, estabelece uma ligação ao *host* diretamente através de uma ligação SSH.

São verificados todos os processos atualmente em execução no *host* através do comando `ps -ef`, com um *grep* para filtrar, passando apenas a incluir os processos do *iperf3* nos resultados obtidos. Deste modo, são devolvidos todos os *flows* simulados através deste programa.

Para conseguir saber qual dos processos corresponde ao *flow* que se pretende limitar, foi necessário que para cada *iperf3* criado, uma porta diferente fosse utilizada. Deste modo, visto o controlador saber qual a porta em utilização pelo *flow*, este irá procura-la, obtendo o *Process ID* (Process ID (PID)) do processo.

O processo é cancelado e é relançado um novo *iperf3*, mas desta vez com a largura de banda que o controlador garantiu a esse *flow* na altura da atribuição da sua classe, e com o tempo que falta para este ser terminado. Esse tempo é obtido pois o controlador possui o momento em que o *flow* foi inicialmente criado na mensagem de resposta à reserva, que teve de ter este campo acrescentado para efetuar este cálculo.

Contudo, devido ao elevado número de processos criados, o curto tempo existente para obter a resposta do *host* e a quantidade de ligações necessárias a estabelecer em determinadas alturas, nem sempre é encontrado o PID correspondente. Deste modo todos os *flows* são criados com um limite de largura de banda.

É importante referir que tal situação só se verifica no ambiente de simulação, sendo que ao utilizar *switches* reais que suportem OpenFlow 1.3, a situação referida não aconteceria pois através das *meter-bands* é possível definir qual a largura de banda máxima utilizada por um *flow*.

4.3 Resultados das simulações

Através das distribuições exponenciais e de Poisson foram obtidos os momentos de arranques dos *flows* e as suas durações necessárias para a atingir a largura de banda requisitada, 30 ou 140 *megabits por segundo* dependendo do momento simulado. Os *flows* são estabelecidos entre os diferentes *hosts* existentes na topologia cuja rota passe no troço entre o *switch* 7 e 12.

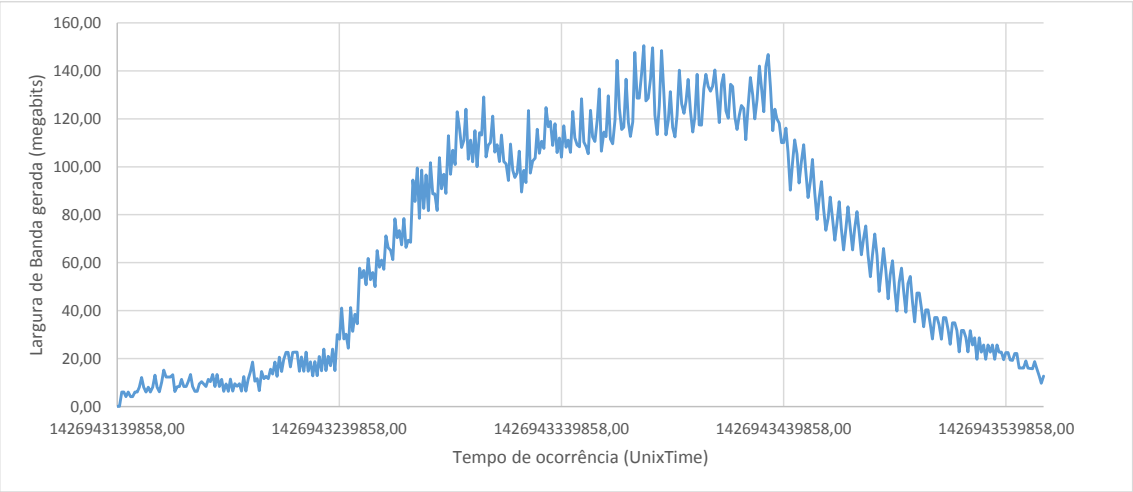

Figura 4.2: Largura de banda da carga oferecida ao troço em estudo.

Os resultados obtidos estão representados na figura 4.2. A diferença existente entre os valores simulados e as duas referências para a largura de banda deve-se evidentemente ao facto de se estar a usar métodos aleatórios.

Na figura 4.3 encontra-se a largura de banda verificada no troço em análise, em resposta à carga oferecida. Apesar das ligações apenas estarem definidas para suportarem 80 *megabits por segundo*, verifica-se que em certos momentos esse valor é ultrapassado. Isto deve-se ao facto do iperf produzir *flows* que por vezes demonstram um comportamento superior ao esperado, apesar da sua média final tender para a largura de banda pretendida; ao facto da leitura ser realizada através de amostras de cinco em cinco segundos e devido à limitação virtual imposta na criação das ligações possivelmente permitir alguma flexibilidade quando existe um pico de uso.

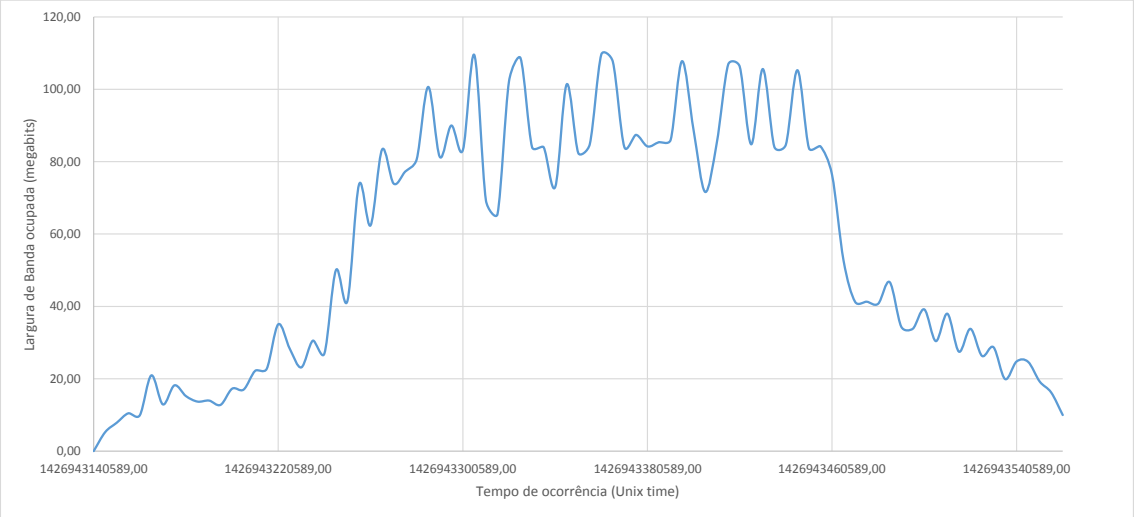

Figura 4.3: Largura de banda em uso no troço em estudo.

No entanto, a simulação demonstra o comportamento esperado, onde é possível verificar os dois momentos distintos da simulação, um inicial em que o tráfego gerado não ultrapassa a largura de banda das ligações e outro em que este valor é ultrapassado.

Para a realização das simulações, que estão descritas de seguida, foi utilizado sempre o mesmo modelo de modo a ser possível efetuar comparações.

4.3.1 Qualidade de Serviço

Um *flow* com garantias de qualidade de serviço possui um baixo número de pacotes perdidos, idealmente zero, um tempo de receção de pacotes curto (*delay*) e um baixo atraso na diferença de tempo entre a receção dos pacotes (*jitter*). Estas são as características utilizadas como base para avaliar o comportamento do tráfego na rede.

No caso do tráfego VoIP, que se trata de um tráfego mais exigente, uma chamada é considerada de boa qualidade se possuir um valor de *delay* inferior a 150 milissegundos [10], um valor de *jitter* inferior a 2 milissegundos e a ausência de pacotes perdidos.

Na figura 4.4 encontra-se o resultado do valor de *jitter* de uma chamada VoIP, ocorrida na altura de menor acesso à rede. Os valores observados são baixos, tendo o pico mais alto o valor de 0,12 milissegundos. As oscilações existentes, provenientes de diferenças de ritmo de transmissão, quer da própria chamada, quer por influência de outras, não influenciam a qualidade da chamada, encontrado-se os seus valores dentro do estipulado para uma chamada de voz decorrer com qualidade.

O mesmo se sucede em relação ao *delay*, representado na figura 4.5, em que se verifica que os seus valores se encontram dentro dos parâmetros para assegurar uma chamada de voz com qualidade.

Sendo estes casos correspondentes a alturas em que a rede se encontrar a funcionar abaixo das suas capacidades, este resultado também seria esperado num *flow* ao qual não

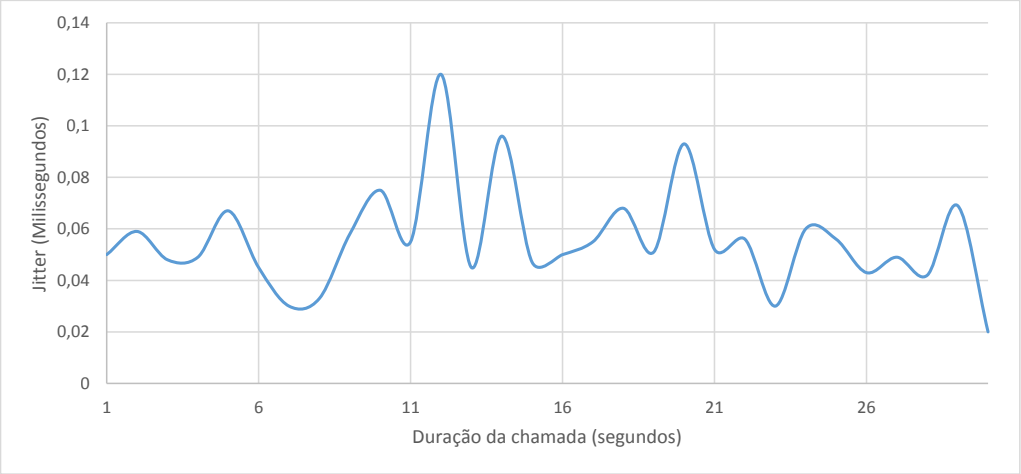

Figura 4.4: *Jitter* de uma chamada VoIP em condições de tráfego normal.

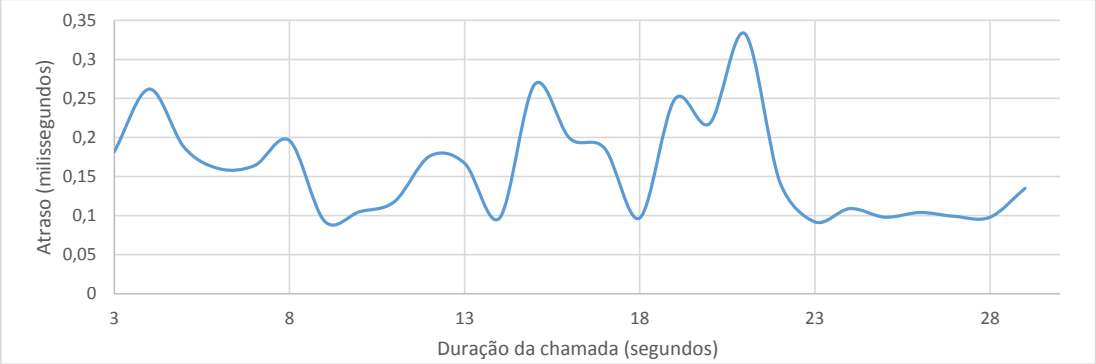

Figura 4.5: *Delay* de uma chamada VoIP em condições de tráfego normal.

fossem atribuídas garantias.

Nas figuras 4.6 e 4.8 é possível encontrar o comportamento de uma chamada VoIP durante o momento de saturação da rede.

Na figura 4.6 verifica-se que os valores de *jitter* entre os pacotes continua a estar abaixo do limite estabelecido pela Cisco, sendo que no pior caso apenas se verificou um *jitter* de 0,5 milissegundos, valor abaixo do limite estipulado de 2 milissegundos. O número de pacotes perdidos na chamada também se mantém em zero, o mesmo verificado nas condições anteriores.

É possível deste modo afirmar que estamos perante uma chamada que demonstra qualidade de serviço, mesmo em alturas de saturação da rede.

A figura 4.7 mostra a comparação entre os atrasos de uma chamada VoIP perante as duas condições da rede. Uma foi criada durante uma altura de uma utilização de rede normal, dentro das suas capacidades, e outra foi criada numa altura de saturação da rede, em que a largura de banda total em uso se encontra acima das suas capacidades. Ambas as chamadas foram criadas com as mesmas características, possuindo uma duração de 30 segundos.

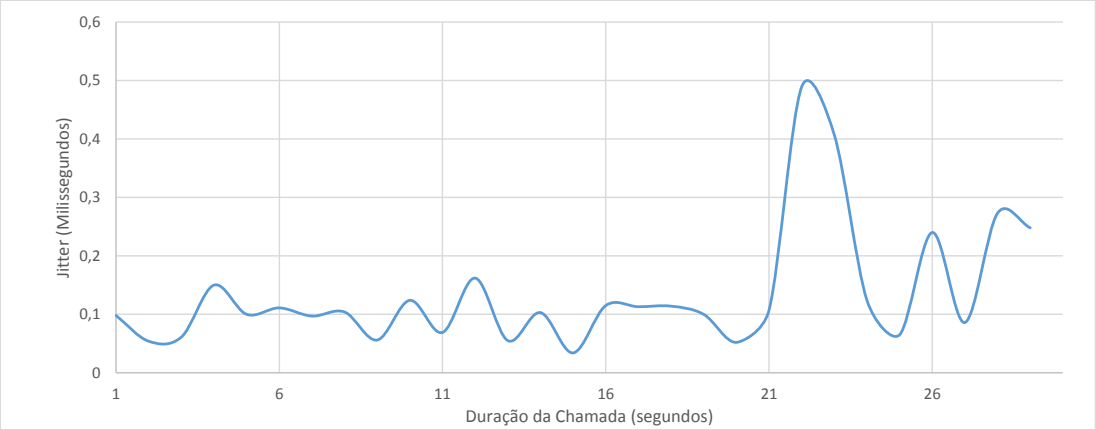

Figura 4.6: *Jitter* de uma chamada VoIP em condições de saturação da rede.

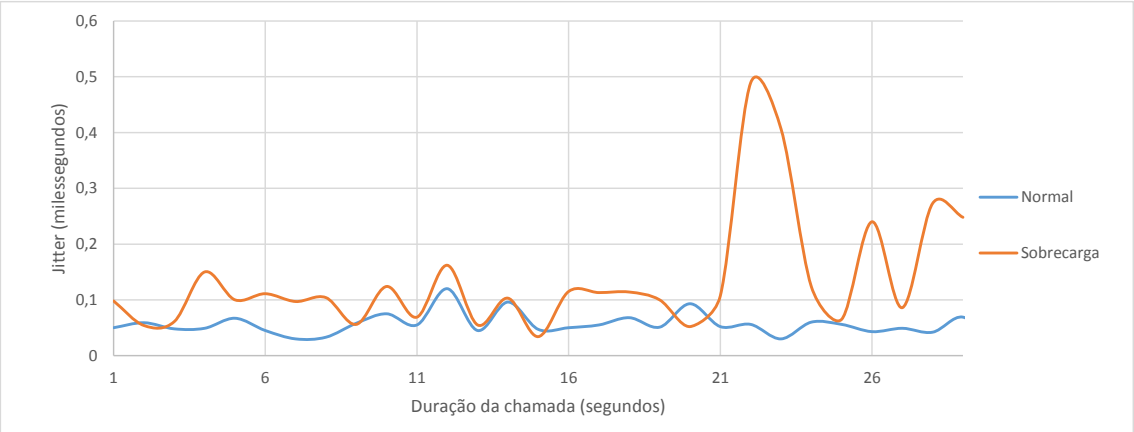

Figura 4.7: Comparação de *jitter* de uma chamada VoIP realizada em condições normais e de saturação da rede.

É possível verificar que, mesmo sendo os valores de *jitter* ligeiramente maiores numa rede saturada, estes não são grandes o suficiente para afetar a qualidade da chamada. A razão pela qual o *jitter* da chamada aumentou com a rede saturada, ainda que por valores pequenos, deve-se ao facto de estarem a decorrer múltiplas chamadas de voz na classe, cerca de 66 no melhor dos casos, ao contrário do primeiro caso em que o número era menor, no máximo 13.

O facto dos resultados serem tão semelhantes entre si, comprovam que a rede garante qualidade de serviço a tráfego VoIP em todas as ocasiões, decorrendo as chamadas sem alterações até perante condições de saturação da rede.

À semelhança do que se sucede com o *jitter*, também o *delay*, representado na figura 4.8, demonstra um comportamento aceitável, dentro dos parâmetros definidos para a chamada decorrer com qualidade. É observável que o seu valor se encontra dentro dos limites de até 150 milissegundos de atraso.

É possível verificar a existência de um *delay* inicial superior ao existente durante o resto da chamada. Este corresponde à necessidade de comunicação do *switch* com o controlador

num momento inicial, para a instalação da *flow entry* correspondente ao *flow* no *switch*. No entanto, o atraso inicial verificado no estabelecimento da ligação não chega a perturbar a qualidade da chamada.

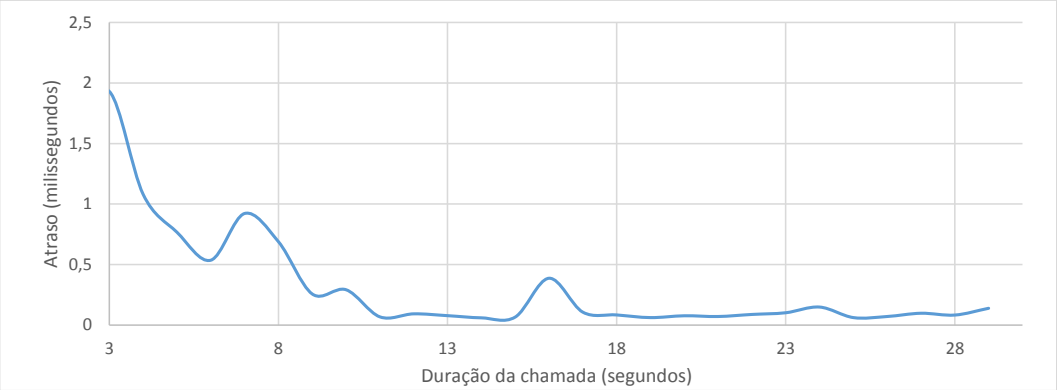

Figura 4.8: *Delay* de uma chamada VoIP em condições de saturação da rede.

Quando se fala de um *flow* cujas garantias não foram asseguradas, sendo-lhe atribuído a classe *best effort*, os resultados são diferentes.

Na figura 4.9 encontra-se uma comparação do *jitter* entre dois *flows*, que não sendo VoIP tiveram atribuídas as suas classes de garantia dependendo do estado da rede. Enquanto que a um dos *flows* foi atribuída a classe 2, assegurando-lhe garantias, ao outro foi atribuída a classe 3, sendo assim um *flow best effort*.

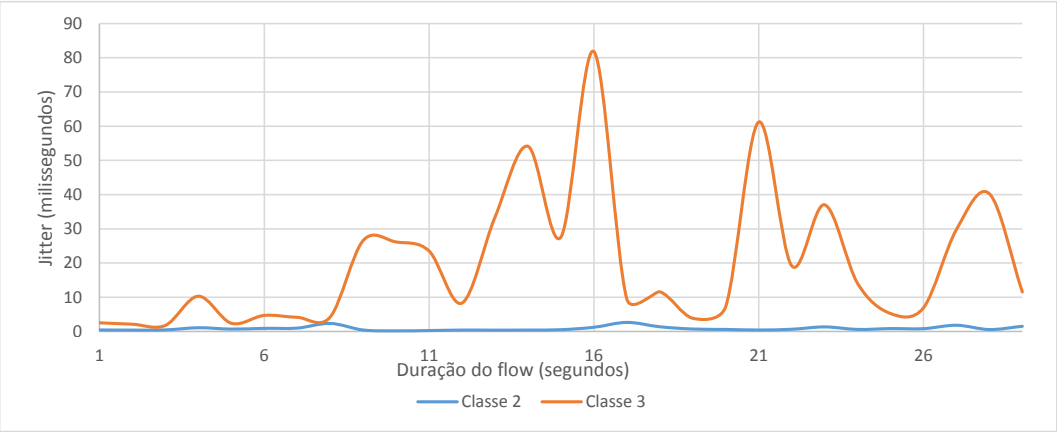

Figura 4.9: Comparação de *jitter* de dois *flows* iniciados durante saturação da rede mas com classes diferentes.

A figura 4.9 representa a comparação dos valores de *jitter* de dois *flows* com as mesmas características ocorridos durante a saturação da rede. É observável que os valores de ambos são consideravelmente diferentes.

A discrepância verificada entre os dois *flows* ocorre pois ao *flow* atribuído à classe 3 é negado o acesso a garantias de qualidade. Numa altura em que as conexões em curso ultrapassam a largura de banda máxima das ligações, os *flows* nesta classe vêem

a sua qualidade decair pois o *switch* prioriza as outras ligações existentes nas classes com garantia. Deste modo, estando à mercê dos restantes *flows* com garantias atribuídas durante a sua ocorrência, o *flow* possui grandes variações de *jitter*.

Em comparação, o *flow* atribuído à classe 2, possuindo a garantia de que existe largura de banda para a sua existência, demonstra um comportamento com poucas oscilações.

No entanto, a qualidade dos *flows* não se verifica apenas através da análise dos atrasos, pois o número de pacotes perdidos também tem um aspeto importante na sua avaliação.

Se durante a simulação de *flows* VoIP com qualidades asseguradas não existiu a perda de nenhum pacote, o mesmo não se verifica quando se fala de *flows* sem qualidades asseguradas.

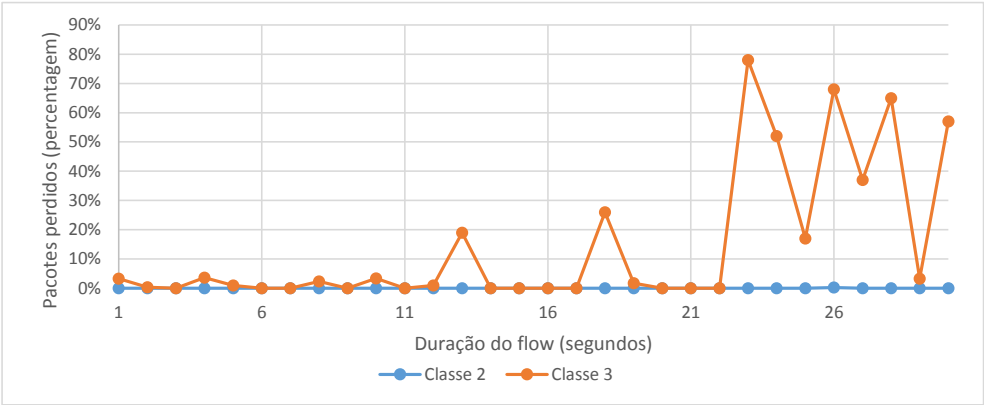

Figura 4.10: Comparação da percentagem de pacotes perdidos em condições normais e de saturação da rede.

Na figura 4.10, encontramos novamente a comparação entre dois *flows* ocorridos durante a saturação da rede, mas em classe de garantias diferentes, deste vez tendo em consideração a percentagem de pacotes perdidos.

Mais uma vez, e tal como sucedeu com o atraso entre pacotes, um *flow* que veja as suas garantias negadas numa altura de saturação da rede tem a sua qualidade afetada. Os *switchs* não conseguem processar todos os *flows* ativos e os que possuem uma menor garantia vêem os seus pacotes serem perdidos. Observando a figura 4.10 é presumível que durante o final da existência deste *flow* tenha existido um aumento do número de *flows* com garantias, pois a percentagem de perdas de pacotes aumentou drasticamente.

Por sua vez, os *flows* com garantias atribuídas, possuindo espaço reservado para a sua existência, não se demonstram afetados pela utilização da rede, obtendo uma percentagem de pacotes perdidos durante a sua existência próxima de 0%.

A necessidade de atribuir garantias a certos *flows* torna-se deste modo evidente, especialmente em alturas de grande afluência à rede, pois tal como se verificou, um *flow* sem garantias de serviço vê a sua qualidade deteriorar rapidamente, através do aumento do atraso entre pacotes e a perda dos mesmos.

Mesmo sendo a classe 2 uma classe de garantias, a existência de um elevado número de *flows* nesta durante a altura de saturação, provoca atrasos maiores, observáveis na figura

Figura 4.11: *Jitter* de um *flow* na classe 2 durante a saturação.

4.11. Estes valores são por vezes superiores aos desejáveis num *flow* VoIP, justificando a existência de uma classe de garantias superiores destinadas apenas a este tipo de tráfego, mantendo num entanto um valor suportável para os restantes tipos de tráfego menos exigentes que o VoIP.

É de referir ainda a existência de um pico de *jitter* aos 8 segundos, correspondendo ao momento em que este *flow* lhe viu atribuída uma classe. Já os restantes aumentos de *jitter* ocorrem devido ao comportamento dos restantes *flows* não demonstrarem uma utilização de largura de banda contínua ao longo do tempo, acabando por influenciar o *jitter* dos restantes.

4.3.2 Eficiência na gestão de recursos

A eficiência na gestão dos recursos de uma rede é outro aspeto relevante. É importante atribuir garantias à maior quantidade de tráfego possível para melhorar a eficiência da rede, maximizando a taxa de ocupação das classes que garantem qualidade de serviço.

Existindo apenas uma classe controlável na qual os *flows* possuem garantias de serviço, é nesta classe que os recursos existentes devem ser aproveitados ao máximo.

Na figura 4.12 encontramos representada a taxa de utilização da classe 2 usando uma abordagem normal, sem um estado de *over-provisioning*.

Na primeira parte da figura, correspondendo a uma altura de pouco tráfego na rede, a taxa de ocupação da classe 2 tem valores baixos devido à quantidade de *flows* existentes não ocuparem uma largura de banda suficiente para encher esta classe. Na segunda parte, os valores médios são superiores pois já existe uma maior quantidade de *flows*, suficiente para o limite de largura de banda da classe ser atingido.

Tal como seria de esperar, a taxa de utilização não chega aos 100% pois os *flows* não estão a gastar a totalidade dos recursos que para estes foram reservados, fazendo com que outros *flows*, que possivelmente poderiam aceder a esta classe sem afetar o tráfego já existente, sejam colocados numa classe *best-effort*. A média de utilização da classe 2

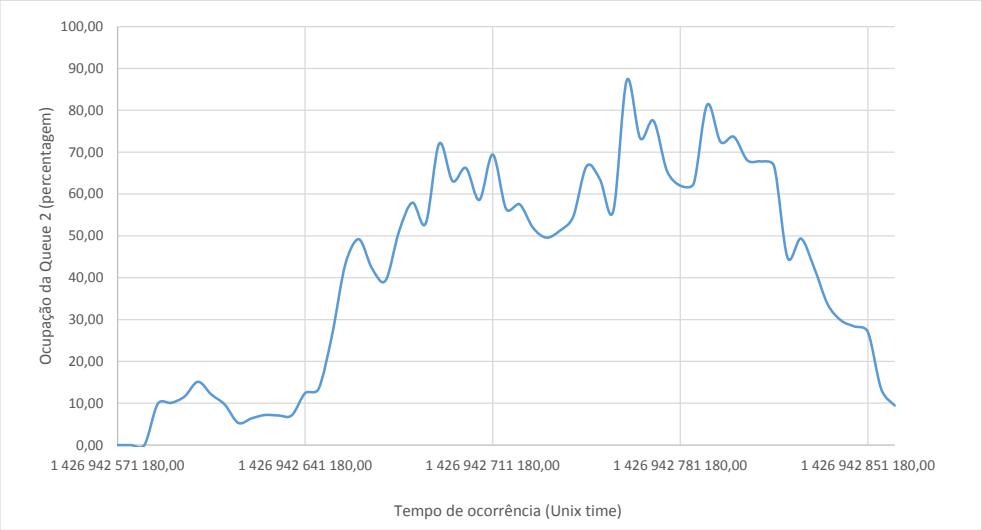

Figura 4.12: Taxa de ocupação da classe 2 sem over-provisioning.

neste caso situa-se nos 42,5%, sendo que durante o pico de uso da rede esta se situa aproximadamente nos 72%, longe da ocupação ideal de 100%.

A introdução do estado de *over-provisioning* nos controladores traz uma maior flexibilidade à rede, permitindo aos controladores alocarem um maior valor de largura de banda com garantia do que o permitido, tendo em conta a utilização dos recursos.

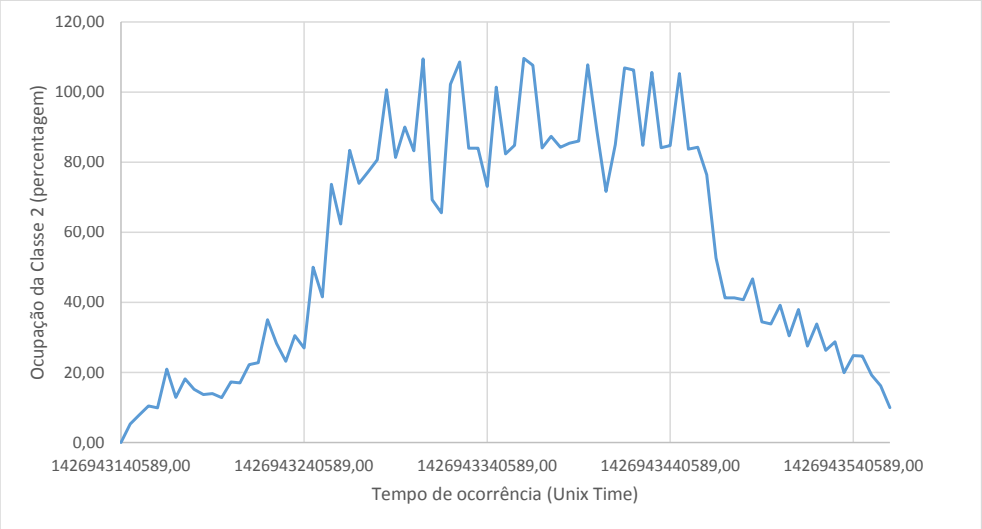

Figura 4.13: Taxa de ocupação da classe 2.

Na figura 4.13 encontra-se novamente a taxa de ocupação da classe 2, desta vez efetuando *over-provisioning*.

Tal como na figura anterior, nos momentos iniciais existindo um baixo valor de largura de banda em uso, esta classe encontra-se com uma taxa de ocupação baixa. Após um certo tempo, com a existência de uma maior quantidade de *flows* na rede, a sua ocupação

torna-se mais alta, entrando o mecanismo de *over-provisioning* em uso.

É, logo à partida, possível verificar que a taxa de utilização é superior à verificada anteriormente, atingindo aproximadamente, durante o pico da rede, uma ocupação de 94%. Este valor é bastante superior ao que se verificou no caso anterior. No entanto, a utilização deste método possui alguns impactos negativos, também estes visíveis na figura.

Em alguns momentos, a taxa de ocupação chega a ser superior aos 100%, significando que esta classe durante essas ocorrências estará a utilizar uma largura de banda superior à que lhe foi inicialmente estipulada. No entanto, esta ocorrência não irá afetar o desempenho de *flows* nesta classe, mas sim diminuir o espaço disponível para as restantes onde não há garantias de qualidade de serviço, nomeadamente as classes 3 e 4.

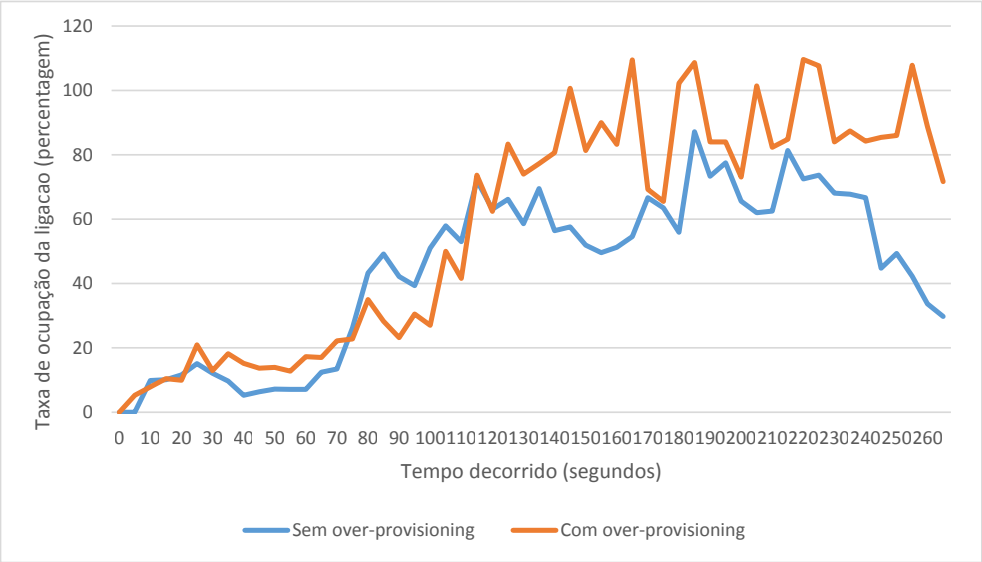

Figura 4.14: Comparação da taxa de ocupação da classe 2 entre controladores com over-provisioning e sem.

As diferenças descritas anteriormente encontram-se visíveis na comparação entre controladores com e sem *over-provisioning* na figura 4.14. Como se verificou, ao utilizar este método, a taxa de ocupação é superior com a sua utilização.

Através do *over-provisioning* é assim obtida uma rede mais eficiente, que utiliza e gere melhor os seus recursos conseguindo assim atribuir garantias a um maior número de *flows*, elevando a qualidade geral do tráfego na rede.

4.3.3 Atribuição de classe

Neste projeto existem quatro diferentes classes correspondentes a diferentes níveis de qualidade de serviço. A evolução da utilização das diferentes classes durante a simulação encontra-se representada na figura 4.15.

Verifica-se, tal como esperado, que todo o tráfego a chegar a um *switch* inicia-se na classe 4, correspondendo à classe de *probing*. Um *flow* só fica atribuído a esta classe enquanto uma resposta sobre a sua classe final não regressar dos restantes controladores.

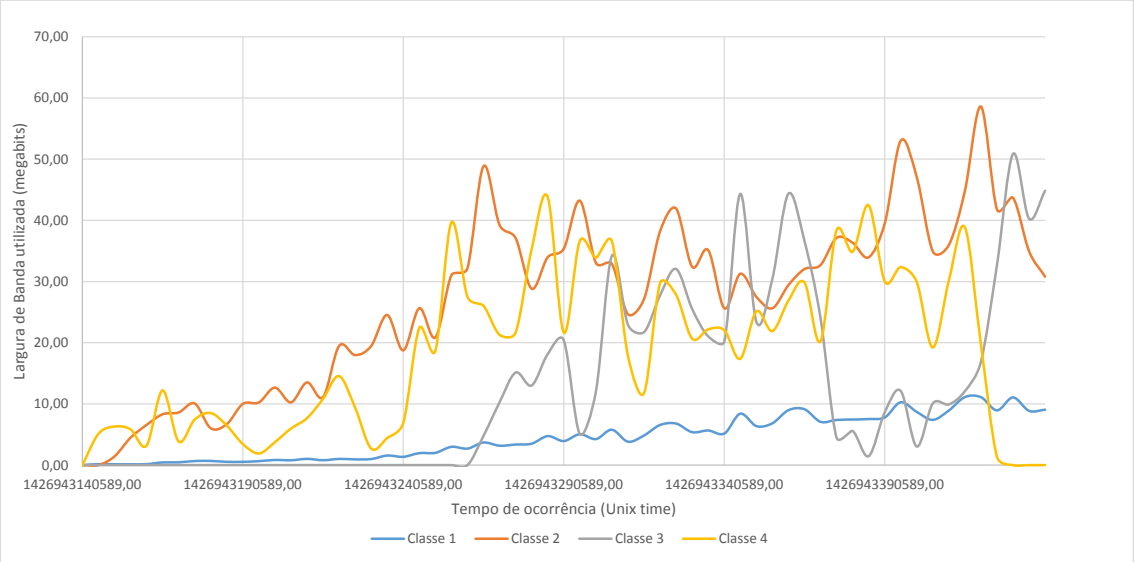

Figura 4.15: Largura de banda ocupada por classe.

Este processo demora algum tempo a ser efetuado devido aos diversos passos necessários para a atribuição de classe, como a medição das suas caraterísticas.

Na figura 4.16 encontram-se representados os tempos necessários para a atribuição de classes a *flows*.

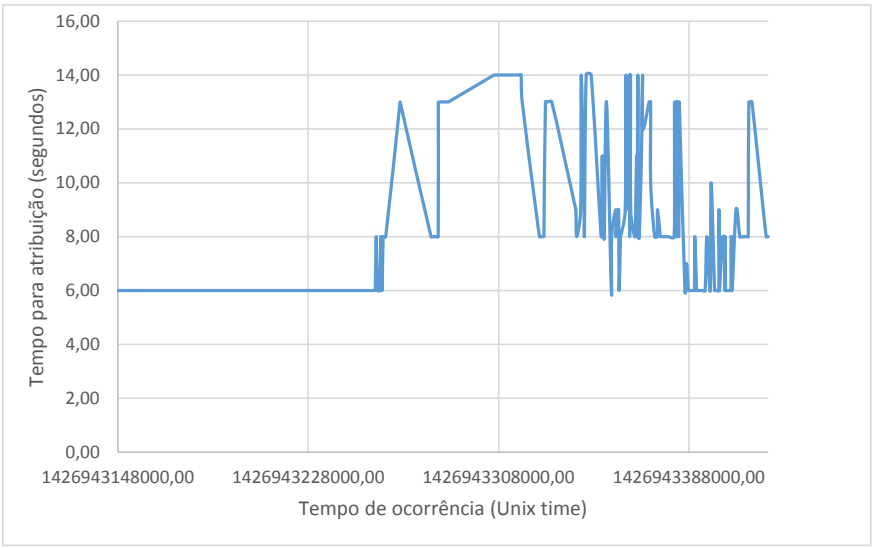

Figura 4.16: Tempo para atribuição de uma classe a um *flow* com *over-provisioning*.

É possível verificar que os tempos necessários para a atribuição de classes são inferiores no início e superiores no final da experiência, passando de 6 segundos para a atribuição da classe, para até um máximo de 14 segundos, correspondendo a mais de o dobro. O período em que esta situação acontece é durante o período em que a rede se encontra mais congestionada.

No entanto, a razão pela qual tal sucede, não é devido ao aumento do número de *flows* com que os *switches* têm de lidar ou com atrasos causados nos pacotes pelas ligações estarem saturadas. Esta situação acontece devido à utilização do *over-provisioning*.

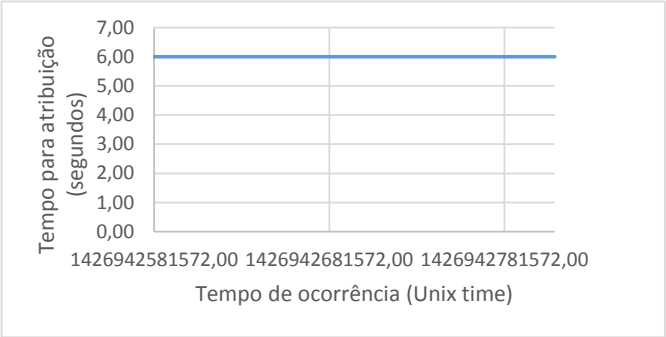

Figura 4.17: Tempo para atribuição de uma classe a um *flow* sem *over-provisioning*.

Olhando para a figura 4.17, correspondendo ao tempo de atribuição de uma classe numa topologia com a mesma carga na rede mas sem utilizar o *over-provisioning*, verificamos que o tempo para atribuição se mantém constante, independentemente da utilização da rede.

Os pacotes de reserva não são afetados pelo estado pois pertencem à classe 1, possuindo assim uma prioridade superior aos restantes, não sofrendo atrasos. No entanto, ao se utilizar o *over-provisioning*, cada *switch* necessita de efetuar mais medições acerca do seu estado atual, provocando o atraso verificado na figura 4.16.

Voltando a olhar para a figura 4.15, é visível que a proximidade da classe 2 do seu limite de 50 *megabits por segundo* leva a um aumento de tráfego na classe 3. Sendo a classe 4 provisória, qualquer *flow* nela presente irá ser atribuído a outra classe quando a resposta à sua reserva chegar. Não existindo espaço na classe 2, este será atribuído à terceira classe, originando o aumento verificado.

Já as oscilações de largura de banda observáveis nas diferentes classes são originadas por *flows* que não possuem uma utilização constante ao longo do tempo. Os principais contribuidores para as oscilações são os *flows* web de curta duração e o *streaming* do tipo on-off. Estas oscilações não se manifestam na classe 1 pois esta classe apenas contém tráfego VoIP, tráfego este que possui uma utilização constante ao longo do tempo e uma longa duração.

4.3.4 Estimações da largura de banda protegida

Sendo este um projeto que utiliza um estado *soft* na zona central da rede, torna-se interessante verificar se o valor esperado recebido através de informações de outros controladores é semelhante ao valor realmente em utilização no *switch*.

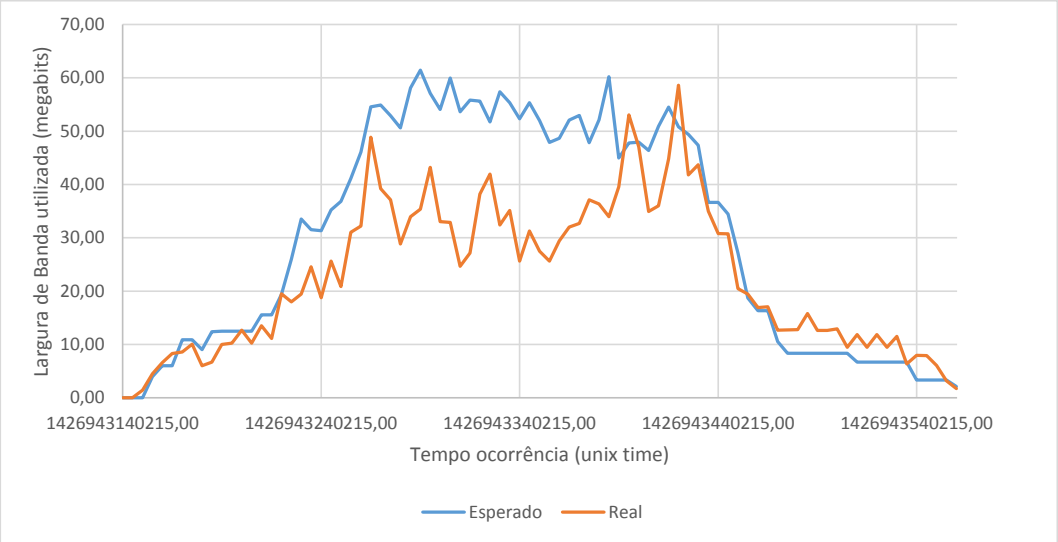

Figura 4.18: Comparação entre o valor de tráfego esperado e o real.

Na figura 4.18 encontramos essa comparação. É possível verificar que existe uma certa discrepância entre o valor de largura de banda reservado no *switch* e o valor que este realmente se encontra a utilizar. A razão pela qual esta diferença acontece deve-se à oscilação verificada em 4.3.3, consequência de *flows* curtos no tempo ou irregulares.

Ao acontecer, a rede fica sub-aproveitada pois, apesar de esta considerar que já não possui capacidades para atribuir garantias a mais nenhum *flow*, tal não sucede. A utilização de um *over-provisioning* maior pode resolver esta situação, arriscando no entanto que toda a ligação fique reservada para a classe 2 e aumentando o tempo para atribuição de uma classe.

4.3.5 Tráfego de controlo gerado

Durante a geração de reservas e para a informação do estado dos *switches* de ingresso é criado tráfego de controlo adicional.

Na figura 4.19 podemos encontrar a quantidade total de tráfego de controlo introduzido na rede. É observável um aumento da largura de banda ocupada por pacotes de controlo na parte final da figura, correspondendo ao momento de utilização da rede, em que existe um aumento do número de *flows* novos levando à criação de um maior número de pacotes de pedido de reserva. Este trata-se de um comportamento indesejável numa rede, pois é nas alturas de maior utilização que a rede precisa de aproveitar ao máximo a sua capacidade, estando deste modo a diminuir a sua capacidade agravando a situação.

Figura 4.19: Tráfego de controlo introduzido na rede.

Observando a figura 4.20, representando a relação existente entre o tráfego da rede e o introduzido na rede devido ao controlo, verifica-se que a quantidade de tráfego adicional gerado diminui em relação ao tráfego existente na rede. Este, no entanto, é superior nos momentos de menor uso. A razão pela qual tal acontece é devido a uma componente do tráfego de controlo, contendo a informação das reservas ativas nos controladores, utilizar uma largura de banda fixa, não sendo dependente do número de *flows* na rede.

Para diminuir o valor da largura de banda fixa, que na implementação atual escala conforme a quantidade de destinos existentes, o controlador de ingresso deveria agrupar todos os destinos que possui num só pacote e enviá-lo, cabendo aos controladores centrais separar o seu conteúdo conforme os destinos e encaminhá-lo.

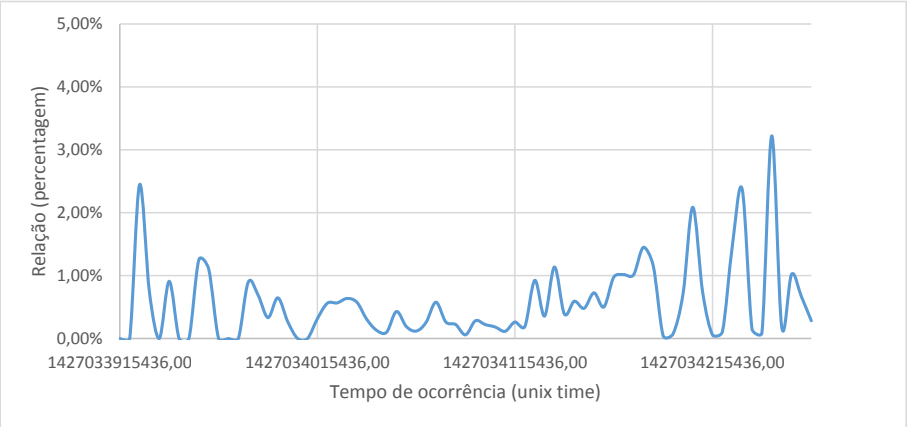

Figura 4.20: Relação entre o tráfego da rede e de controlo.

No final do gráfico verificam-se ainda três picos, sendo consequência do período de amostragem da simulação. O facto deste ser demasiado curto, levou a que num período não fossem lidos dados de tráfego de controlo, sendo lidos no seguinte dois períodos de controlo criando a ilusão de que numa certa altura houve um súbito aumento do tráfego de controlo, sendo que na verdade este se manteve.

CAPÍTULO 5

CONCLUSÕES

5.1 Conclusões

Nesta dissertação foi proposta a criação de uma lógica de rede que possibilitasse assegurar qualidades de serviço a certos *flows*, utilizando métodos não intrusivos nem discriminativos, que necessitassem de baixas capacidades de processamento, de modo a aumentar a escalabilidade, e garantisse a maximização da utilização dos recursos da rede.

Para esse fim, foram desenvolvidas duas lógicas de processamento para controladores SDN utilizando o Floodlight, sendo uma destinada à zona de ingresso da rede e outra à zona central.

No capítulo 4, é possível encontrar os resultados das simulações obtidos. Estes confirmam que os objetivos para este projeto foram cumpridos. São atribuídas garantias a determinados *flows*, sem a necessidade de os classificar e independentemente do tipo de tráfego que transportam. Observou-se que um a *flow* a que tenha sido garantido qualidade de serviço, essa é mantida até este deixar de existir, independentemente das condições pelas quais a rede passe.

Utilizando este método, surgem comportamentos interessantes na rede. No caso do tráfego P2P, que tem como características ser um tráfego que cria uma grande quantidade de *flows*, estes entraram em competição consigo mesmo, pois um *flow* criado por um *host*, tenta obter garantias tais como outros *flows* oriundos do mesmo *host*.

Uma especial atenção é atribuída ao tráfego de voz, que vê as suas garantias serem cumpridas qualquer que seja o estado da rede sem a necessidade de verificar o seu estado. Deste modo, a um *flow* deste tipo é assegurado que nunca será afetado por aumentos dos valores de atraso dos pacotes ou aumentos da taxa de perda de pacotes, garantindo uma chamada sem quebras. Mesmo durante o pico de utilização da rede, em que existiam até 66 chamadas em simultâneo a decorrer, todas decorreram com valores de atraso dentro

dos parâmetros de tempo de atraso propostos.

Através das mensagens de controlo trocadas entre os diferentes controladores, foi possível atribuir as garantias aos *flows* na zona central da rede sem a utilização de estado rígido.

Deste modo, e tal como pretendido, o processamento nesta região ficou reduzido aumentando a escalabilidade da rede. No entanto, a utilização desta abordagem provocou a necessidade da troca de mensagens de uma forma constante, criando uma largura de banda fixa utilizada para a troca de informações entre controladores.

No entanto, observou-se que a taxa de ocupação das ligações protegidas foi elevada, acima dos 93%, obtidas através do uso de técnicas de *over-provisioning*. Estes permitem que um controlador assegure um maior volume de tráfego com qualidade de serviço do que aquele que supostamente é suportado pelo *switch*. Em comparação com uma rede onde esta técnica não seria utilizada, existiu um aumento de 20% de largura de banda com garantidas, obtendo deste modo uma rede mais eficiente.

5.2 Trabalho Futuro

Nem todos os aspetos cobertos por esta dissertação foram detalhados, existindo alguns que poderão ser alvo de melhorias ou mais aprofundados num trabalho futuro.

- **Classificação VoIP**

  A classificação de tráfego VoIP utilizada no projeto passou por verificar se o tráfego em questão utiliza o protocolo Session Initiation Protocol (SIP). Em situações reais esta situação não se verifica, pois nem todo o tráfego SIP corresponde a tráfego VoIP e nem todo o tráfego VoIP utiliza este protocolo. Outro método deverá ser implementado para a sua classificação, mantendo no entanto a utilização de métodos não intrusivos.

- **Identificadores dos pacotes**

  A identificação dos pacotes para a zona central da rede foi efetuada utilizando cabeçalhos *vlan*. Seria interessante num trabalho futuro que outro método fosse utilizado, utilizando por exemplo *tags* MPLS ou cabeçalhos personalizados, caso exista uma nova versão do OpenFlow que os suporte.

- **Encaminhamento dinâmico**

  O encaminhamento utilizado no trabalho é introduzido manualmente, sendo interessante num trabalho futuro a utilização de um algoritmo para implementar automaticamente as rotas. A utilização da informação recebida dos controladores acerca da utilização das ligações para a sua estimação seria um tópico interessante.

- **Agregação de pedidos de reserva**

Sempre que um novo *flow* chega ao controlador e existe capacidade no *switch* de ingresso para este, é criado um novo pedido de reserva. Seria interessante, e reduziria o tráfego de controlo introduzido na rede, se estes pedidos fossem agrupados de modo a ser enviado um conjunto de pedidos, em vez de pedidos individuais, caso vários cheguem ao controlador num curto espaço de tempo.

- **Agregação de informações do estado de reservas**

  Atualmente as informações do estado de reservas são enviadas individualmente por destino e origem. Com o intuito de diminuir o tráfego de controlo na rede e aumentar a sua escalabilidade, este deveria ser agrupado por origem, fazendo com que um *controlador* de ingresso apenas envie um pacote com toda a informação para todos os destinos que este possui. Seria assim reduzida a largura de banda fixa utilizada pelos controladores, que atualmente escala conforme a quantidade de destinos existentes.

  Utilizando esta agregação, tal deixaria de suceder pois um só pacote iria conter informação sobre todos os destinos do controlador de ingresso.

- **Reavaliação de *Flows***

  Após ser atribuída uma determinada classe a um *flow*, esta será a sua classe até ser terminado. Apesar de este ser o comportamento desejado, seria interessante, no caso do *flow* estar numa classe sem garantias, que este tivesse a possibilidade de ser reavaliado, podendo após algum tempo ser-lhe atribuído a classe com garantias caso existisse disponibilidade para tal.

- **Utilização de *Flow Meters***

  O ambiente de rede criado para o teste e simulação do trabalho não possibilitava a utilização de *Flow Meters*. No entanto, estes representam um método de obtenção das características dos *flows* mais rápido e simples, sem a necessidade de possuir intervalos para amostragem, que levam a que por vezes valores incorretos sejam obtidos e provocam um aumento do tempo necessário para a sua leitura.

  Num trabalho futuro, a utilização de uma versão do controlador que já os suportasse seria interessante, melhorando o tempo de atribuição de uma classe e diminuindo o número de pedidos efetuados ao *switch*.

BIBLIOGRAFIA

[1] I. F. Akyildiz, A. Lee, P. Wang, M. Luo e W. Chou. "A Roadmap for Traffic Engineering in SDN-OpenFlow Networks". Em: *Comput. Netw.* 71 (out. de 2014), pp. 1–30. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2014.06.002. URL: http://dx.doi.org/10.1016/j.comnet.2014.06.002.

[2] L Andersson. *Minei, I., Ed., and B. Thomas, Ed.,"*. Rel. téc. LDP Specification", RFC 5036, 2007.

[3] L. Andersson e G. Swallow. *The multiprotocol label switching (MPLS) working group decision on MPLS signaling protocols*. Rel. téc. RFC 3468, February, 2003.

[4] *Arquitectura do Floodlight*. 2015. URL: http://thenewstack.io/sdn-series-part-v-floodlight/.

[5] F. Baker, C. Iturralde, F. Le Faucheur e B. Davie. "Aggregation of RSVP for IPv4 and IPv6 reservations". Em: *Work in Progress* (2001).

[6] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule e K. Salamatian. "Traffic Classification on the Fly". Em: *SIGCOMM Comput. Commun. Rev.* 36.2 (abr. de 2006), pp. 23–26. ISSN: 0146-4833. DOI: 10.1145/1129582.1129589. URL: http://doi.acm.org/10.1145/1129582.1129589.

[7] M. Bieg. *P2P-network*. Online; accessed 03-March-2015. 2010. URL: http://commons.wikimedia.org/wiki/File:P2P-network.svg.

[8] W. Braun e M. Menth. "Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices". Em: *Future Internet* 6.2 (2014), pp. 302–336.

[9] N. Brownlee e K. Claffy. "Understanding Internet traffic streams: dragonflies and tortoises". Em: *Communications Magazine, IEEE* 40.10 (2002), pp. 110–117.

[10] Cisco. *Understanding Delay in Packet Voice Networks*. 2015. URL: http://www.cisco.com/c/en/us/support/docs/voice/voice-quality/5125-delay-details.html.

[11] Cisco. *VoIP Call Admission Control*. 2015. URL: http://www.cisco.com/c/en/us/td/docs/ios/solutions_docs/voip_solutions/CAC.pdf.

[12] G. Combs et al. "Wireshark". Em: *Web page: http://www. wireshark. org/last modified* (2007), pp. 12–02.

[13] A. Dainotti, A. Pescape e K. C. Claffy. "Issues and future directions in traffic classification". Em: *Network, IEEE* 26.1 (2012), pp. 35–40.

[14] D. Erickson. "The beacon openflow controller". Em: *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM. 2013, pp. 13–18.

[15] D. ETSI. *TIPHON-05001, Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON); General Aspects of Quality of Service (QoS)*. Rel. téc. TR 101 329 Ver. 1.2. 5, October, 1998.

[16] *Floodlight SDN Controller*. 2015. URL: `https://groups.google.com/a/openflowhub.org/forum/#!msg/floodlight-dev/l5r71SZcD8g/jRXzvgzHoygJ`.

[17] *Floodlight SDN Controller Community*. 2015. URL: `https://groups.google.com/a/openflowhub.org/forum/#!forum/floodlight-dev`.

[18] *Floodlight SDN Repository*. 2015. URL: `https://github.com/floodlight/floodlight`.

[19] *Floodlight Storage Service*. 2015. URL: `http://www.openflowhub.org/display/floodlightcontroller/MemoryStorageSource+(Dev)`.

[20] O. N. Foundation. 2015. URL: `https://www.opennetworking.org/our-members`.

[21] O. N. Fundation. "Software-defined networking: The new norm for networks". Em: *ONF White Paper* (2012).

[22] S. Garg e M. Kappes. "Can I add a VoIP call?" Em: *Communications, 2003. ICC'03. IEEE International Conference on*. Vol. 2. IEEE. 2003, pp. 779–783.

[23] R. Gayraud e O. Jacques. "SIPp". Em: *SIPp Tool (http://sipp. sourceforge. net)* (2004).

[24] GMANE. *NOX network control platform developers' mailing list activity*. 2015. URL: `http://dir.gmane.org/gmane.network.nox.devel`.

[25] G. Goth. "Software-defined networking could shake up more than packets". Em: *Internet Computing, IEEE* 15.4 (2011), pp. 6–9.

[26] I. N. W. Group et al. "RFC 2475 An Architecture for Differentiated Services". Em: *IETF* (1998).

[27] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown e S. Shenker. "NOX: towards an operating system for networks". Em: *ACM SIGCOMM Computer Communication Review* 38.3 (2008), pp. 105–110.

[28] P. Haffner, S. Sen, O. Spatscheck e D. Wang. "ACAS: automated construction of application signatures". Em: *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*. ACM. 2005, pp. 197–202.

[29] IETF. *RFC 2205 - Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification*. 2015. URL: http://tools.ietf.org/html/rfc2205.

[30] *iperf3: A TCP, UDP, and SCTP network bandwidth measurement tool*. 2015. URL: https://github.com/esnet/iperf.

[31] M. Jemec. *packETH–Ethernet packet generator*. 2012.

[32] S. Jordan. "Some traffic management practices are unreasonable". Em: *Computer Communications and Networks, 2009. ICCCN 2009. Proceedings of 18th Internatonal Conference on*. IEEE. 2009, pp. 1–6.

[33] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy e M. Faloutsos. "Is p2p dying or just hiding?[p2p traffic measurement]". Em: *Global Telecommunications Conference, 2004. GLOBECOM'04. IEEE*. Vol. 3. IEEE. 2004, pp. 1532–1538.

[34] T. Karagiannis, A. Broido, M. Faloutsos et al. "Transport layer identification of P2P traffic". Em: *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. ACM. 2004, pp. 121–134.

[35] K. Kaur, J. Singh e N. S. Ghumman. "Mininet as Software Defined Networking Testing Platform". Em: ().

[36] R. Khondoker, A. Zaalouk, R. Marx e K. Bayarou. "Feature-based comparison and selection of Software Defined Networking (SDN) controllers". Em: *Computer Applications and Information Systems (WCCAIS), 2014 World Congress on*. IEEE. 2014, pp. 1–7.

[37] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos e K. Lee. "Internet Traffic Classification Demystified: Myths, Caveats, and the Best Practices". Em: *Proceedings of the 2008 ACM CoNEXT Conference*. CoNEXT '08. Madrid, Spain: ACM, 2008, 11:1–11:12. ISBN: 978-1-60558-210-8. DOI: 10.1145/1544012.1544023. URL: http://doi.acm.org/10.1145/1544012.1544023.

[38] M.-S. Kim, Y. J. Won, H.-J. Lee, J. W. Hong e R. Boutaba. "Flow-based characteristic analysis of Internet application traffic". Em: *Workshop Chair*. 2004.

[39] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama et al. "Onix: A Distributed Control Platform for Large-scale Production Networks." Em: *OSDI*. Vol. 10. 2010, pp. 1–6.

[40] B. Lantz, B. Heller e N. McKeown. "A network in a laptop: rapid prototyping for software-defined networks". Em: *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM. 2010, p. 19.

[41] A. Mankin, F. Baker, B. Braden, S. Bradner, M. O'Dell, A. Romanow, A. Weinrib e L. Zhang. "Resource ReSerVation Protocol (RSVP) Version 1 Applicability Statement Some Guidelines on Deployment". Em: *RFC2208, Sep* (1997).

[42] M. Marchese. *QoS over heterogeneous networks*. John Wiley & Sons, 2007.

[43] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker e J. Turner. "OpenFlow: enabling innovation in campus networks". Em: *ACM SIGCOMM Computer Communication Review* 38.2 (2008), pp. 69–74.

[44] *OF 1.4 Spec*. 2013. URL: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf.

[45] *OF 1.5 Draft*. 2014. URL: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf.

[46] *Open vSwitch FAQ*. 2015. URL: https://github.com/openvswitch/ovs/blob/master/FAQ.md.

[47] *Open vSwitch vsctl*. 2015. URL: http://manpages.ubuntu.com/manpages/natty/man8/ovs-vsctl.8.html.

[48] Openvswitch.org. *Open vSwitch*. 2015. URL: http://openvswitch.org/.

[49] S Ortiz. "Software-Defined Networking: On the Verge of a Breakthrough?" Em: *Computer* 46.7 (2013), pp. 10–12.

[50] J. Ramsay, A. Barbesi e J. Preece. "A psychological investigation of long retrieval times on the World Wide Web". Em: *Interacting with computers* 10.1 (1998), pp. 77–86.

[51] A. Rao, A. Legout, Y.-s. Lim, D. Towsley, C. Barakat e W. Dabbous. "Network characteristics of video streaming traffic". Em: *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*. ACM. 2011, p. 25.

[52] F. G. O. Risso, M. Baldi, O. Morandi, A. Baldini e P. Monclus. "Lightweight, payload-based traffic classification: An experimental evaluation". Em: (2008).

[53] E. Rosen, A. Viswanathan, R. Callon et al. "Multiprotocol label switching architecture". Em: (2001).

[54] *Ryu SDN Controller*. 2015. URL: http://osrg.github.io/ryu/.

[55] Sandvine. *Sandvine - Intelligent Broadband Networks*. 2015. URL: https://www.sandvine.com/.

[56] I. U. SANDVINE. *Global Internet Phenomena Report*. 2012. URL: https://www.sandvine.com/downloads/general/global-internet-phenomena/2012/1h-2012-global-internet-phenomena-report.pdf.

[57] I. U. SANDVINE. *Global Internet Phenomena Report*. 2013. URL: https://www.sandvine.com/downloads/general/global-internet-phenomena/2013/2h-2013-global-internet-phenomena-report.pdf.

[58] I. U. SANDVINE. *Global Internet Phenomena Report*. 2014. URL: https://www.sandvine.com/downloads/general/global-internet-phenomena/2014/2h-2014-global-internet-phenomena-report.pdf.

[59] I. U. SANDVINE. *Global Internet Phenomena Report*. 2014. URL: https://www.sandvine.com/downloads/general/global-internet-phenomena/2014/1h-2014-global-internet-phenomena-report.pdf.

[60] H. Schulze e K. Mochalski. "Ipoque internet study 2008/2009". Em: *ipoque. com* (2009).

[61] S. A. Shah, J. Faiz, M. Farooq, A. Shafi e S. A. Mehdi. "An architectural evaluation of SDN controllers". Em: *Communications (ICC), 2013 IEEE International Conference on*. IEEE. 2013, pp. 3504–3508.

[62] S Shenker. "RFC 1633 Integrated Services Architecture June 1994". Em: (1994).

[63] I. Stoica. *Stateless Core: A Scalable Approach for Quality of Service in the Internet: Winning Thesis of the 2001 ACM Doctoral Dissertation Competition*. Vol. 2979. Springer Science & Business Media, 2004.

[64] C. Strauch, U.-L. S. Sites e W. Kriha. "NoSQL databases". Em: *Lecture Notes, Stuttgart Media University* (2011).

[65] TACS. *DiffServ Architecture*. 2015. URL: http://tacs.eu/Analyses/Internet/ip%20qos%20architectures/diff-serv_architecture.htm.

[66] A. Tirumala, F. Qin, J. Dugan, J. Ferguson e K. Gibbs. "Iperf: The TCP/UDP bandwidth measurement tool". Em: *htt p://dast. nlanr. net/Projects* (2005).

[67] S. University. *Ethane*. 2006. URL: http://yuba.stanford.edu/ethane/index.html.

[68] *VOICE OVER IP - PER CALL BANDWIDTH CONSUMPTION*. 2015. URL: http://www.cisco.com/c/dam/en/us/support/docs/voice/voice-quality/7934-bwidth-consume.pdf.

[69] H. Zhai, J. Wang e Y. Fang. "Providing statistical QoS guarantee for voice over IP in the IEEE 802.11 wireless LANs". Em: *Wireless Communications, IEEE* 13.1 (2006), pp. 36–43.