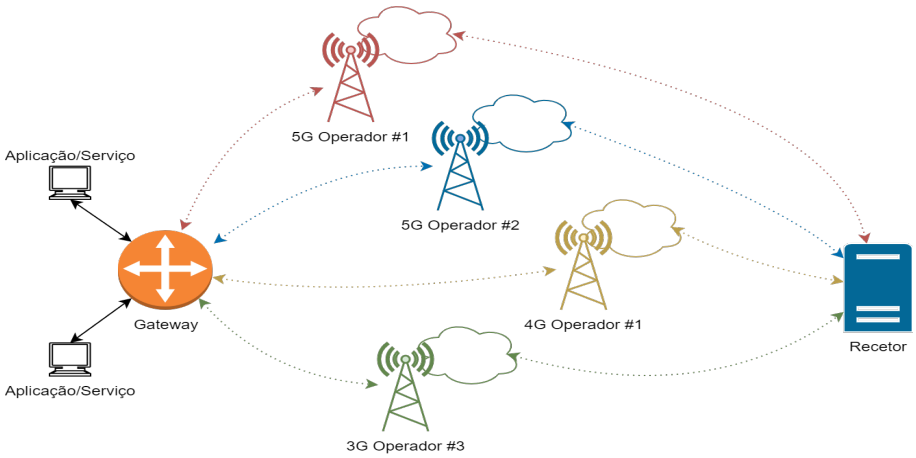
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Departamento de Engenharia de Eletrónica e Telecomunicações e de Computadores



Sistema de encaminhamento resiliente para serviços críticos sobre redes públicas móveis

Carlos Eduardo Governo Rodrigues

(Licenciado)

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

Orientador : Professor Doutor Nuno Miguel Machado Cruz

Júri:

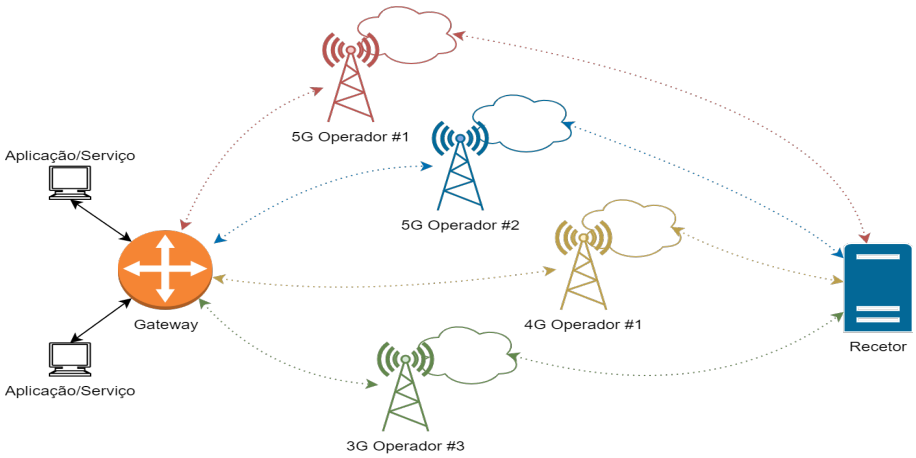
Presidente: Professor Doutor Carlos Jorge de Sousa Gonçalves

Vogais: Professor Doutor Pedro António Marques Ribeiro
Professor Doutor Nuno Miguel Machado Cruz

outubro, 2022

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Departamento de Engenharia de Eletrónica e Telecomunicações e de Computadores



Sistema de encaminhamento resiliente para serviços críticos sobre redes públicas móveis

Carlos Eduardo Governo Rodrigues

(Licenciado)

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

Orientador : Professor Doutor Nuno Miguel Machado Cruz

Júri:

Presidente: Professor Doutor Carlos Jorge de Sousa Gonçalves

Vogais: Professor Doutor Pedro António Marques Ribeiro
Professor Doutor Nuno Miguel Machado Cruz

outubro, 2022

Agradecimentos

No decorrer desta dissertação contei com o apoio de diversas instituições e pessoas, pelo que estou profundamente grato.

Agradeço ao meu orientador, Doutor Nuno Miguel Machado Cruz por todo o apoio, disponibilidade e acompanhamento prestado.

Agradeço à Solvit pelo equipamento, apoio e flexibilidade disponibilizados ao longo da dissertação.

Agradeço a oportunidade de ter participado no projeto Ferrovia 4.0 como bolseiro, um grande contributo para esta dissertação e para as minhas competências técnicas e sentido de responsabilidade.

Agradeço à minha família, aos meus amigos e em especial à minha namorada, pela paciência, ajuda e constante incentivo.

A todos, o meu sincero Obrigado.

Resumo

Existem cada vez mais serviços críticos assentes sobre ligações móveis fornecidas por operadores públicos. No entanto, estas ligações não oferecem, muitas vezes, *Service Level Agreement* (SLA) alinhados com a criticidade do serviço que as utiliza. O 5G veio oferecer nativamente mecanismos que permitem fornecer este tipo de garantias a um serviço e aplicação. No entanto, isto carece de negociação com o operador em causa, com custos não negligenciáveis. Adicionalmente, em situações de falha do operador, a resiliência continua a não estar assegurada. Assim, neste trabalho pretende-se desenvolver, sobre um sistema operativo de código aberto, um sistema de comunicação (*gateway*) que permita o encaminhamento de *flows* de pacotes, com elevada resiliência e segurança, através de múltiplas ligações WAN baseadas em redes públicas móveis, com a *gateway* estática ou em movimento. Identificam-se como grandes resultados para este trabalho: 1) A integração, através de um *stack* de *software*, num sistema operativo baseado em Linux, da capacidade de agregação automatizada de diferentes ligações WAN de forma transparente para as aplicações através de um canal seguro; e, 2) A avaliação da Qualidade de Serviço oferecida, para determinar quais as melhores ligações a usar utilizando para tal as *Application Programming Interface* (API) existentes no Kernel de um sistema operativo Linux.

Palavras-chave: Resiliência; Serviços Críticos; *Virtual Private Network*; Redes móveis; *Multipath*; Encaminhamento dinâmico; Avaliação Ativa; Avaliação Passiva

Abstract

The number of critical services based on mobile connections provided by public operators has been rising significantly. However, these connections often do not offer Service Level Agreements (SLAs) in line with the criticality of the service that uses them. 5G has natively offered mechanisms that allow this type of guarantee to be provided to a service and application. Nonetheless, this requires negotiation with the operator, with non-negligible costs, and in situations of operator failure, resilience is not guaranteed. Thus, the present work aims to develop a gateway based on an open-source operating system that allows packet flow forwarding, with high resilience and security, through multiple WAN connections based on public mobile networks, with a static or a moving terminal. The main goals for this work are: 1) The integration, through a software stack, in a Linux-based OS, of the automated aggregation capacity of different WAN connections transparently to the applications through a secure channel; and, 2) The evaluation of the Quality of Service offered, in order to establish the best connections using the Application Programming Interfaces (APIs) existing in the Kernel of a Linux operating system.

Keywords: Resilience; Critical Services; Virtual Private Network; Mobile Networks; Multipath; Dynamic Routing; Active Measurements; Passive Measurements

Índice

Índice de Figuras

Índice de Tabelas

Índice de Listagens

Índice de Abreviaturas e Siglas

ACK Acknowledgment. 10, 11, 21
AEAD Authenticated Encryption with Associated Data. 60
AES Advanced Encryption Standard. 24, 25, 27, 28, 29, 30, 54, 55, 56, 57, 58, 60
AES-NI Advanced Encryption Standard New Instruction. 27, 28
AH Authentication Header. 25, 26
API Application Programming Interface. vii, ix

BTC Bulk Transfer Capacity. 7, 15, 84, 85, 93

CBC Cipher Block Chaining. 25, 27, 29, 30, 54, 56, 57, 60
CLTCP Congestion Level TCP. 23
CMT Concurrent Multipath Transfer. 33
CNN Convolutional Neural Networks. 20
cwnd Congestion Window. 61

DSCP Differentiated Services Code Point. 18, 71, 72, 79, 80, 84, 90, 94, 95

ECIO Energy per chip to Interference power ratio. 69
ECMP Equal Cost Multipath. 32
ECT Equal Cost Tree. 32
ESP Encapsulating Security Payload. 26, 27

GCM Galois/Counter Mode. 24, 25, 27, 28, 29, 30, 54, 55, 56, 57, 58
GNSS Global Navigation Satellite System. 61, 88, 89, 95, 96

ICMP Internet Control Message Protocol. 12, 13, 16, 18, 54
IP Internet Protocol. 12, 25, 26, 33
IPsec Internet Protocol Security. 3, 25, 27, 28, 29

KPI Key Performance Indicator. 61
KRR Kernel Ridge Regression. 20

LACP Link Aggregation Control Protocol. 32
LTE Long Term Evolution. 20, 22, 23, 31, 69

MPTCP Multipath TCP. xv, xvi, 3, 33, 34, 35, 37, 38, 39, 40, 41, 42, 43, 45, 46, 49, 50, 51, 52, 63, 64, 65, 66, 68, 71, 94
MQTT Message Queuing Telemetry Transport. 62, 70, 81, 86, 89, 90, 96
MSS Maximum Segment Size. 61
MTU Maximum Transmission Unit. 10

NAT Network Address Translation. 34
NEXT New Enhanced Available Bandwidth Estimation Technique. 18, 19
NR New Radio. 22, 31, 69

OSI Open Systems Interconnection. 6, 13, 23, 25, 26, 31
OWAMP One-Way Active Measurement Protocol. 16, 17

PGM Probe Gap Model. 14, 15
PRM Probe Rate Model. 15, 19
PSTN Public Switched Telephone Network. 33
pTCP parallel TCP. 33

QoS Quality of Service. 3, 9

RF Random Forests. 20

RSCP Received Signal Code Power. 69
RSRP Reference Signal Received Power. 69, 70
RSRQ Reference Signal Received Quality. 69, 70
RSSI Received Signal Strength Indicator. 69, 70
RTO Retransmission Timeout. 61
RTT Round Trip Time. xvi, xvii, 7, 10, 11, 12, 13, 16, 17, 18, 23, 59, 61, 70, 87

SABES Statistical Available Bandwidth Estimation. 10
SACK Selective Acknowledgment. 24
SBAS Satellite-based Augmentation Systems. 61, 96
SCTP Stream Control Transmission Protocol. 33
SLA Service Level Agreement. vii, ix, 1
SLoPs Self-Loading Periodic Streams. 13
SNR Signal to Noise Ratio. xvi, 69, 70, 71, 87
SPB Shortest Path Bridging. 32
STP Spanning Tree Protocol. 32
SVR Supported Vector Regression. 20
SYN Synchronize. 10, 11

Tc Traffic control. 38
TCP Transmission Control Protocol. xv, xvi, 7, 10, 11, 15, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30, 31, 33, 34, 38, 41, 45, 50, 51, 52, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 69, 84, 94
TCPW TCP Westwood. 20, 21, 22
TCPWH TCP Westwood with Holt-Winters. 21, 22
TLS Transport Layer Security. 24
ToPP Train of Packet Pairs. 13
TRILL Transparent Interconnection of a Lot of Links. 32
TTL Time to Live. 12, 14, 17
TWAMP Two-Way Active Measurement Protocol. 17, 18

UDP User Datagram Protocol. xvi, 15, 16, 23, 24, 25, 26, 27, 28, 29, 30, 31, 51, 52, 54, 55, 56, 57, 58, 59, 60, 61, 63, 65, 73, 74, 76

VPN Virtual Private Network. xvi, xvii, 3, 23, 24, 25, 26, 27, 28, 29, 30, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 62, 63, 64, 65, 66, 68, 73, 74, 76, 77, 82, 83, 84, 89, 94

VPS Variable Packet Size. 12, 14

WAN Wide Area Network. vii

WCDMA Wide-Band Code-Division Multiple Access. 69

1

Introdução

1.1 Enquadramento

Nos dias que correm existem cada vez mais serviços críticos que estão assentes sobre redes públicas móveis. Os sistemas críticos são sistemas essenciais do ponto de vista da vida humana, podendo a sua falha resultar na perda de vidas. Por exemplo, a comunicação entre corporações de bombeiros no contexto de um incêndio ou a comunicação entre diversas infraestruturas de um sistema ferroviário podem ser considerados sistemas críticos. Os dados associados a estes serviços podem apresentar diversas formas, podendo-se tratar de comunicações de voz, imagem ou até vídeo em tempo real.

Os sistemas de natureza crítica caracterizam-se por terem requisitos de rede muito estritos, necessários para assegurar a sua fiabilidade e segurança. Porém, as comunicações assentes sobre infraestruturas de operadores públicos nem sempre são capazes de oferecer os *Service Level Agreements* (SLA) necessários à criticidade dos serviços. A rede de um determinado operador pode até falhar por completo, colocando em causa os serviços dependentes desta. Assim sendo, estes não podem depender exclusivamente de uma rede, devem ser capazes de utilizar qualquer rede disponível para encaminhar os dados críticos, devendo ser escolhida a melhor rede de entre as que estão disponíveis naquela localização e naquele instante.

1.2 Motivação

Tendo em conta que existem cada vez mais serviços críticos nos quais a comunicação está assente sobre operadores móveis públicos, e sabendo que estes, muitas vezes, não conseguem garantir a qualidade necessária, pretende-se desenvolver um sistema capaz de assegurar a fiabilidade e segurança destes serviços. Este sistema trata-se de uma *gateway* de comunicações para a qual os serviços enviam os seus dados e a *gateway* tratará do seu envio.

1.3 Objetivos

Como objetivos gerais para a construção de uma *gateway* capaz de garantir fiabilidade e segurança aos serviços que dependam deste, identificam-se:

- Investigar diferentes abordagens à análise constante da qualidade das redes disponíveis;
- Definir a integração de uma camada de segurança;
- Identificar todos os componentes necessários à implementação da *gateway*;
- Definir a arquitetura do sistema;
- Desenvolvimento da *gateway* recorrendo aos componentes identificados como necessários e seguindo a arquitetura que se considera mais adequada.

Para o funcionamento da *gateway* identificam-se desde logo alguns objetivos concretos para que este seja o mais fácil possível de integrar com os serviços e de oferecer-lhes a qualidade máxima possível, nomeadamente:

- Capacidade de operar de forma estática ou em movimento, por exemplo, integrando um veículo;
- Sistema assente num sistema de código aberto sem qualquer comprometimento com tecnologias proprietárias;
- Capacidade de realizar uma avaliação constante de todas as redes que estão disponíveis para que assim consiga encaminhar da forma mais eficaz possível os pacotes provenientes dos serviços;

- Capacidade de identificar a *Quality of Service* (QoS) entregue a cada serviço;
- Transparente para os serviços/aplicações, para que estas não necessitem de quaisquer alterações.

1.4 Contributo

A dissertação apresentada foi, em parte, realizada no âmbito do projeto Ferrovia 4.0, ref. LISBOA-01-0247-FEDER-046111 & POCI-01-0247-FEDER-046111 financiado pela Agência Nacional de Inovação, contribuindo para a definição de arquitetura e soluções de comunicação do projeto. As soluções propostas foram apresentadas e debatidas em reuniões internas ao nível dos envolvidos por parte do ISEL e em reuniões com todos os parceiros que integram o consórcio do projeto.

Desta dissertação resultaram duas publicações de artigos científicos, ambos no INFORUM — Simpósio de Informática. No primeiro, publicado em 2021 [1], é avaliado o desempenho do protocolo *Multipath TCP* (MPTCP) incluído no Kernel Oficial, sendo testada a versão 5.11. No segundo artigo, publicado em 2022 [2], é avaliado o desempenho de tecnologias *Virtual Private Network* (VPN) atuais, sendo comparados os protocolos *Internet Protocol Security* (IPsec), OpenVPN e WireGuard.

1.5 Estrutura do Documento

No Capítulo 2 é realizado o estado da arte, sendo apresentados conceitos básicos da avaliação de redes, discutidas técnicas de avaliação de rede, diversas VPN e protocolos *multipath*. No Capítulo 3 são descritas as diversas etapas propostas para a construção de um sistema de encaminhamento resiliente. No Capítulo 4 são retiradas conclusões acerca do trabalho realizado no decorrer da dissertação e é descrito o trabalho futuro para o melhoramento do sistema desenvolvido.

2

Estado da Arte

Ao longo deste capítulo serão abordados conceitos essenciais para a construção de um sistema de encaminhamento resiliente para serviços críticos, assente em redes públicas móveis. Em primeiro lugar serão debatidos os diferentes métodos de avaliação da qualidade das ligações, avaliação essa necessária para entender em cada momento qual a melhor rede para encaminhar os dados dos serviços. De seguida, é abordada a componente de segurança da rede, sendo avaliadas as diversas opções existentes. Em terceiro lugar é realizada uma análise à possibilidade de utilização de um protocolo *multipath* — capacidade de utilizar diversas redes simultaneamente e quais as opções existentes.

2.1 Conceitos Essenciais na Avaliação de Redes

Na avaliação de redes, existem conceitos gerais que devem ser desde logo clarificados para uma melhor compreensão do processo de avaliação. Assim sendo, nesta Secção vão ser abordadas as definições de diversos conceitos.

2.1.1 *Link*

Um *link*, ou ligação, é um canal de comunicação que conecta dois ou mais dispositivos para que exista troca de informação entre os mesmos.

2.1.2 *Path*

Um *path*, ou caminho, é um conjunto de *links* desde um nó de origem até um nó de destino.

2.1.3 Capacidade

A capacidade, muitas vezes confundida com a largura de banda, é a capacidade máxima a que um *link* de rede consegue transferir dados, não dependendo da quantidade de tráfego que atravessa o *link*. A capacidade de um *link* depende ainda da camada do modelo *Open Systems Interconnection* (OSI) que se está a analisar. A capacidade ao nível da camada 2 difere da capacidade ao nível da camada 3. Assim sendo, quando se quer comprar a capacidade de múltiplos *links*, esta deve ser feita com base numa camada específica.

A capacidade de um *path* é dada simplesmente pelo *link* de menor capacidade nesse mesmo *path*. A equação 2.1 reflete esta definição, onde C_i é a capacidade do *link* i e N é o número total de *links* [3].

$$C = \min_{i=0...N} C_i, \tag{2.1}$$

2.1.4 Largura de Banda Disponível

A largura de banda disponível é definida ao nível de um *link*, sendo a capacidade não utilizada pelo tráfego que atravessa esse mesmo *link* , num determinado momento.

A largura de banda disponível num *path*, num determinado instante, é dada pelo *link* que possui o menor valor de capacidade não utilizada nesse instante, isto é, pelo *link* que tenha a maior utilização.

A largura de banda disponível num *link* i, num determinado instante, considerando a sua utilização u_i, é representada por A_i [3] sendo dada pela equação 2.2.

$$A_i = (1 - u_i)C_i. \tag{2.2}$$

A largura de banda disponível num *path*, num determinado instante, é representada por A [3], sendo dada pela equação 2.3.

$$A = \min_{i=0...N}[C_i(1 - u_i)]. \tag{2.3}$$

2.1.5 *Bulk Transfer Capacity*

A *Bulk Transfer Capacity* (BTC) representa a capacidade de uma rede em transferir uma quantidade de dados significativa sobre um determinado intervalo de tempo, numa ligação sobre protocolos de transporte de dados com controlo de congestionamento, como o *Transmission Control Protocol* (TCP) [4]. No fundo, BTC representa um ritmo binário máximo atingível numa única ligação TCP. Esta medida não deve ser confundida com a largura de banda, que é independente do protocolo, enquanto BTC é uma métrica específica de ligações TCP. Formalmente, BTC é definida pela equação 2.4:

$$BTC = \frac{sent\_databits}{elapsed\_time}. \tag{2.4}$$

O valor de BTC depende do comportamento das ligações TCP, comportamento esse que depende diretamente do protocolo de congestionamento utilizado. Os protocolos de congestionamento são mecanismos para controlar o fluxo de dados de uma ligação TCP, procurando não sobrecarregar a rede e procurando partilhar eficazmente a largura de banda com outros fluxos de dados TCP. Um exemplo para diferenciar entre largura de banda e BTC é dado em [3], onde os autores descrevem um cenário em que existe apenas um *link* e existem duas conexões TCP que utilizam esse *link*. A capacidade C é saturada por uma das conexões e, neste caso, a largura de banda é zero, devido ao facto de o *link* ter sido saturado. No entanto, a BTC é cerca de $\frac{C}{2}$, caso a conexão tenha o mesmo *Round Trip Time* (RTT) que a conexão concorrente.

2.1.6 Ritmo Binário

O ritmo binário, também denominado ritmo de transmissão, ou, em inglês, *Throughput*, é a quantidade de dados transmitidos num *path* ou *link* sobre um determinado intervalo de tempo. Contrariamente a BTC, não depende do protocolo de rede utilizado.

2.1.7 Latência

A latência, ou *Delay*, é o tempo que um pacote de dados demora a ser enviado desde um nó de origem até um nó de destino. A latência é dado pela soma de diversos *delays*: *Delay* de Processamento (D_{Proc}), *Delay* de Transmissão (D_{Tran}), *Delay* de Propagação (D_{Prop}) e *Delay* de *Queuing* ($D_{Queuing}$), segundo a equação 2.5 [4].

$$Latncia = D_{Proc} + D_{Tran} + D_{Prop} + D_{Queuing}. \tag{2.5}$$

O *Delay* de Processamento é a soma dos atrasos ocorridos ao longo da rede para processamento dos pacotes enviados. Os *routers* da rede necessitam de analisar os pacotes que recebem para os poderem encaminhar, gerando assim atrasos.

O *Delay* de Transmissão, dado pela equação 2.6, é o tempo despendido a enviar um pacote de tamanho L ao ritmo binário R do *link* respetivo, isto é, o tempo que o *router* demora a colocar a totalidade do pacote no *link*.

$$D_{Tran} = \frac{L}{R}. \tag{2.6}$$

O *Delay* de Propagação, dado pela equação 2.7, é o tempo que o sinal demora a navegar no meio físico, à distância d, desde a origem até ao destino.

$$D_{Prop} = \frac{d}{\eta c}. \tag{2.7}$$

O *Delay* de *Queuing* é o tempo total que um pacote fica nas filas de pacotes do *router* antes de ser encaminhando para o próximo nó. Este tempo depende do tráfego que atravessa o *router* naquele instante, do tamanho dos *buffers* utilizados pelas filas e da capacidade de processamento dos equipamentos.

As medidas de *delay* podem ainda ser unidirecionais, sendo a latência desde o nó de origem até ao nó de destino ou, bidirecionais, do nó de origem até ao destino e do destino até ao nó de origem.

2.1.8 *Jitter*

O *Jitter* é a variação da latência unidirecional entre dois pacotes enviados sucessivamente.

2.1.9 *Tight Link*

O *Tight Link* é o *link* de um *path* que apresenta o menor valor de largura de banda disponível, definindo a largura de banda disponível nesse *path*.

2.1.10 *Narrow Link*

O *narrow link* é o *link* de um *path* que possui o menor valor de capacidade, definindo a capacidade do *path* ao qual pertence. Existem situações nas quais o mesmo *link* é

simultaneamente o que possui a menor capacidade (*narrow*) e o que possui o menor valor de largura de banda disponível (*tight*).

2.2 Métodos de Avaliação de Qualidade de Rede

Ao lidar com serviços críticos que podem colocar em causa o funcionamento de infraestruturas críticas, considerando o envio de dados sobre diversas redes, é crucial avaliar o desempenho de cada uma delas em cada momento, para entender qual o QoS que cada rede consegue entregar aos serviços ou aplicações que a irão utilizar. Dada a criticidade dos serviços, a avaliação a realizar deve ser o mais completa possível, tendo em consideração todas as métricas às quais se consiga ter acesso.

2.2.1 Avaliação Passiva

A avaliação passiva, tal como o nome sugere, caracteriza-se pelo facto de não interferir com o tráfego real, e eventualmente crítico, que navega na rede. Não são injetados quaisquer pacotes na rede, não diminuindo a largura de banda disponível nem aumentando a latência das ligações. A rede é avaliada com base nos pacotes que nela circulam. Desta forma, é crucial a escolha de um ponto de rede onde passe tráfego que seja suficientemente representativo do tráfego habitual. A escolha correta do ponto, associada ao facto de serem avaliados apenas pacotes reais da rede, leva a que a avaliação incida sobre dados mais representativos do cenário real. Na realidade, não deve ser escolhido apenas um ponto para medição, devem ser escolhidos diversos pontos para aumentar os dados obtidos da rede, possibilitando o cálculo de mais métricas de qualidade. Por outro lado, há que pensar que quantos mais pontos forem escolhidos para recolha de dados, mais espaço de armazenamento será necessário e maior capacidade de processamento. A Figura 2.1 apresenta a arquitetura geral de avaliação passiva de redes.

Quando se lida com redes móveis públicas, a decisão acerca de que ponto escolher é limitada pelo acesso que se tem a esses mesmos pontos. Ainda, há que ter em consideração que a observação do tráfego, geralmente, implica que este seja armazenado, o que pode gerar grandes volumes de memória, pelo que é habitual serem aplicadas técnicas de *sampling* em oposição à recolha da totalidade do tráfego. Utilizando este tipo de avaliação passiva, é indispensável possuir um mecanismo que, com base nos pacotes obtidos, consiga calcular métricas que descrevam o desempenho da rede.

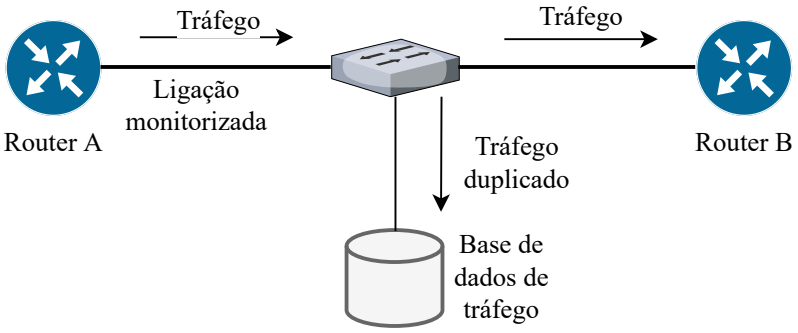


Figura 2.1: Arquitetura geral de avaliação passiva.

Uma forma passiva de estimar a largura de banda, o método *Statistical Available Bandwidth Estimation* (SABES) é apresentada em [5]. Este método utiliza exclusivamente os pacotes trocados numa ligação TCP. Mais precisamente, baseia-se na diferença temporal de chegada dos *Acknowledgment* (ACK). Esta diferença é medida entre a chegada de dois pacotes ACK e é necessário que não exista muito tráfego a chegar entre um pacote e outro, para que, através do intervalo de tempo detetado, se consiga estimar com alguma precisão a largura de banda existente. Assim sendo, esta medição é realizada no início da ligação TCP, enquanto a janela de congestionamento ainda está em processo de crescimento.

Em [6] é apresentada a equação 2.8, que permite o cálculo do ritmo binário, t para uma ligação TCP. Esta equação é baseada no funcionamento dos protocolos de congestionamento TCP e é uma relação entre a capacidade, o *Maximum Transmission Unit* (MTU), o RTT, e uma constante de perda de pacotes p:

$$t = \frac{data\ sent\ per\ cycle}{timepercycle} = \frac{MTU \times C}{RTT \times \sqrt{p}}, \text{ onde } C = \sqrt{\frac{3}{2}}. \tag{2.8}$$

A estimação passiva do RTT de uma ligação pode ser obtida com base no *TCP Handshake* [7], visível na Figura 2.2. Esta estimativa é realizada subtraindo o instante de receção do pacote *Synchronize* (SYN) ao instante de receção do pacote ACK, como demonstrado na equação 2.9.

$$RTT = t_{ACK} - t_{SYN}. \tag{2.9}$$

Em alternativa, esta estimativa também pode ser realizada subtraindo o instante de envio do pacote SYN ao instante de receção do pacote SYN,ACK, como demonstrado na equação 2.10.

$$RTT = t_{SYN,ACK} - t_{SYN}. \tag{2.10}$$

A escolha de uma ou outra equação depende se o ponto de recolha de métricas passivas está mais perto do cliente ou do servidor. Caso esteja mais perto do cliente, deve ser usada a equação 2.10. Se esse ponto for mais próximo do servidor, então deve ser utilizada a equação 2.9.

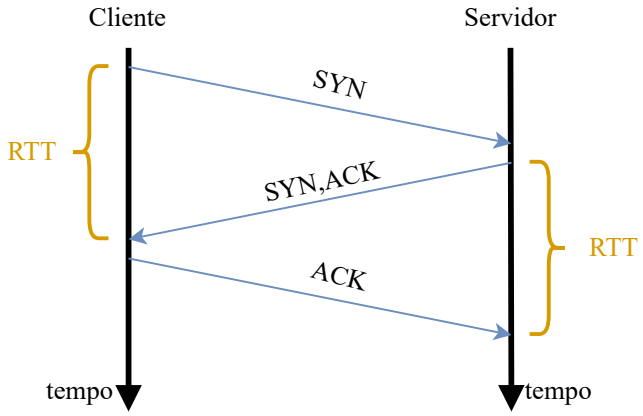


Figura 2.2: TCP *Handshake*.

Com avaliação passiva também é possível estimar a percentagem de perda de pacotes existente, através da contagem dos pacotes enviados e dos pacotes recebidos no recetor [8]. No entanto, não é fácil aferir qual o caminho que os pacotes tomaram, não sendo trivial associar o valor de perda de pacotes a um caminho em concreto. Em [9] é proposta uma solução para este problema, tentando-se, via medidas exclusivamente passivas, encontrar a percentagem de perda de pacotes para cada *link* constituinte de uma rede.

2.2.2 Avaliação Ativa

Uma avaliação ativa consiste em injetar pacotes específicos na rede, variando o conteúdo dos pacotes consoante o que se quer avaliar da rede. A avaliação é realizada com base nos pacotes injetados, sendo calculadas as métricas a partir destes. Dado que existe injeção de pacotes na rede, dá-se uma degradação da largura de banda disponível para o tráfego característico da rede, bem como o possível aumento da latência das comunicações. Por outro lado, os pacotes injetados podem ser manipulados inteiramente dependendo do que se quer avaliar, dando flexibilidade aos responsáveis da rede. Este tipo de avaliação não exige que se tenha acesso a pontos no meio da

rede, as medições podem ser realizadas em qualquer máquina, desde que este cenário seja minimamente representativo do que acontece na rede. A Figura 2.3 apresenta a arquitetura geral de avaliação ativa de redes.

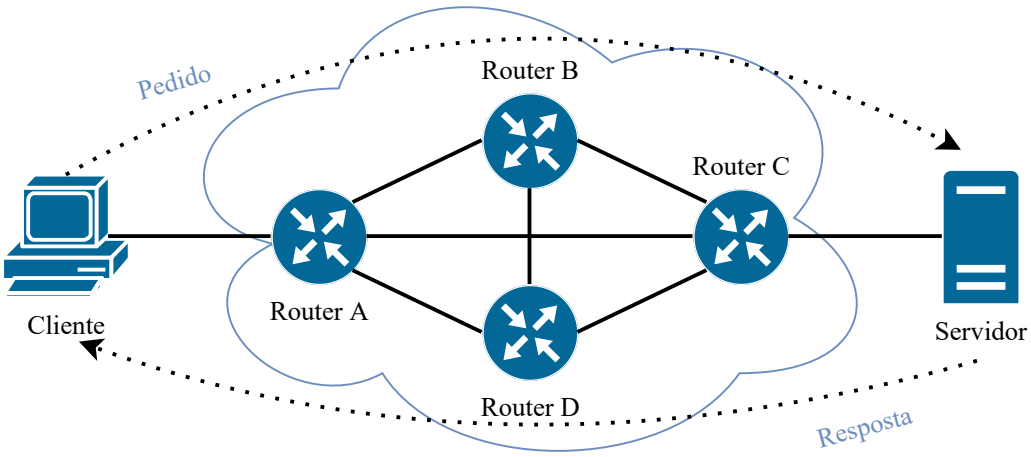


Figura 2.3: Arquitetura geral de avaliação ativa.

Seguidamente são apresentadas diversas técnicas, ferramentas e modelos de estimação de largura de banda, capacidade, latência e *jitter*, de forma ativa.

2.2.2.1 Métodos de Estimação de Largura de Banda e Capacidade

Seguidamente é apresentado um resumo de diversos métodos de estimação de largura de banda e capacidade.

Variable Packet Size

Variable Packet Size (VPS) é uma técnica cujo objetivo é estimar a capacidade existente em cada *link* ao longo de um caminho de comunicação. Foi testada pela primeira vez por Bellovin em 1992 [10], sendo posteriormente utilizada por Jacobson na ferramenta pathchar [11].

A estimação da capacidade é realizada através do envio de *probes* de diversos tamanhos, de um nó para todos os outros no caminho, e medindo o RTT para cada um deles em função do tamanho do pacote enviado. Para isso, é utilizado o campo *Time to Live* (TTL) dos pacotes *Internet Protocol* (IP) para forçar os pacotes a expirarem num determinado nó da rede. No *router* onde o pacote com aquele TTL expira, o pacote é descartado, sendo enviada para o remetente uma mensagem de erro *Internet Control Message Protocol* (ICMP) *time-exceeded*. O remetente utiliza essa mensagem para,

finalmente, calcular o valor do RTT entre ele próprio e o *router* (nó) que enviou essa mensagem. A capacidade do caminho será dada pela capacidade mínima medida.

Um dos problemas apontados a esta técnica é que pode subestimar o valor da capacidade em caminhos nos quais existam diversos *switches* (camada 2 do modelo OSI), que introduzem *delays* de transmissão significativos e não têm capacidade de enviar respostas ICMP, dado que essas são de nível 3 [3]. Para além disso, muitos dispositivos aplicam uma limitação de ritmo binário a pacotes ICMP, dado que estes não são prioritários.

Packet Pair/Train Dispersion

É uma técnica de medição de largura de banda disponível, que se baseia no envio de pacotes do mesmo tamanho ao longo de um caminho para medir a dispersão na chegada dos pacotes ao nó de destino. Na técnica *Packet Pair* são enviados apenas dois pacotes de teste, enquanto na técnica *Packet Train* são utilizados diversos pacotes.

A dispersão na chegada dos pacotes ao destino aumenta consoante o congestionamento que existe na rede e, desta forma, é possível estimar a largura de banda do caminho, dada pela largura de banda do *tight link*. A dispersão de um par de pacotes num determinado *link* é dada pela diferença temporal entre o último *bit* do último pacote e o último *bit* do primeiro pacote [3].

Self-Loading Periodic Streams

A técnica de estimação de largura de banda de um caminho *Self-Loading Periodic Streams* (SLoPs) é baseada no envio de pacotes a partir do nó inicial de um determinado caminho, com destino ao nó final desse caminho. É enviada uma sequência de pacotes de tamanho igual, a um determinado ritmo binário, para aferir o congestionamento criado na rede por parte dos pacotes de teste.

Nesta técnica o remetente pretende encontrar um ritmo binário que seja próximo da largura de banda do caminho, que é dada pela largura de banda do *tight link*. Para isso, vai enviando os pacotes de teste incrementando sucessivamente o ritmo binário, com base num algoritmo iterativo. Se o ritmo binário for maior que a largura de banda, a rede começa a ficar congestionada, existindo um aumento de *delay*. Tenta-se encontrar o ritmo binário ideal, que aproveite as condições da rede, mas sem criar congestionamento na mesma.

Train of Packet Pairs

A técnica *Train of Packet Pairs* (ToPP) [12] é idêntica à SLoPs, mas o incremento do ritmo

binário é linear. Além disso, esta técnica consegue estimar a capacidade do *tight link*. Para a estimação da largura de banda são enviados diversos pares de pacotes.

Packet Tailgating

Packet Tailgating é uma técnica de estimação de capacidade apresentada por Lei e Baker [13], que se baseia nas técnicas VPS e *Packet Pair*. É composta por duas fases. Na primeira fase, denominada por *Sigma*, são medidas as características de todo o caminho, enquanto na segunda, denominada por *tailgating*, são medidas as características de todos os nós de rede de forma individual.

Nesta técnica é enviado um pacote de elevada dimensão, o *tailgated* e, de seguida, é enviado um pacote de pequena dimensão, o *tailgator*. Relativamente ao *tailgated*, o seu TTL é configurado para expirar na ligação (*link*) que está sobre medição. O *tailgator* vai ficando retido nas filas dos *routers* sucessivamente, devido ao pacote anterior ser de elevadas dimensões, até o *tailgated* expirar num nó. A partir desse ponto o *tailgator* continua, não ficando agora retido em nenhuma fila, até ao nó final. Desta forma é captada a informação temporal necessária para estimar a capacidade.

Probe Gap Model

O *Probe Gap Model* (PGM) é um método utilizado por diversas ferramentas de medição de largura de banda tais como Spruce [14], Delphi [15] e IGI [16], dado que o seu impacto na rede é muito reduzido e é rápido na estimação da largura de banda disponível. O PGM assume a existência de apenas um *bottleneck link*, que é tanto o *tight link* como o *narrow link*, algo que nem sempre é verdade.

Este método baseia-se na injeção de pacotes — *probes* na rede para estimar a largura de banda. Os pacotes são enviados aos pares, com um ritmo constante e, por isso, com uma diferença temporal entre eles também constante de Δ_{in}. Em vez de se usarem pares de pacotes podem também ser enviados diversos pacotes a um ritmo constante. O ritmo de envio de pacotes é dado pela capacidade do *tight link*. Assume-se, portanto, que esse valor de capacidade C é conhecido. Os pacotes do par chegam ao destino com uma diferença temporal entre eles dada por Δ_{out}. Entre a transmissão de um e outro pacote, existe transmissão de *cross traffic* que chegou durante Δ_{in}. Assim sendo, o valor Δ_{out} é a diferença temporal entre o instante no qual o *bottleneck* transmite o primeiro pacote do par e o tempo que demora a transmitir o segundo pacote e o *cross traffic* que chegou ao recetor durante Δ_{in}.

Com base na definição de Δ_{in} e Δ_{out}, o tempo de transmissão do *cross traffic* é dado por $\Delta_{out} - \Delta_{in}$, enquanto o ritmo binário deste mesmo tráfego é dado por $\frac{\Delta_{out}-\Delta_{in}}{\Delta_{in}} \times C$, em

que C é a capacidade do *bottleneck* [14]. A largura de banda disponível A é dada pela equação 2.11.

$$A = C \times (1 - \frac{\Delta_{out} - \Delta_{in}}{\Delta_{in}}) \tag{2.11}$$

Um dos problemas apontados a este modelo é que em caminhos *multi-hop* subestima o valor da largura de banda, como demonstrado em [17].

Probe Rate Model

O modelo *Probe Rate Model* (PRM) assume a existência de apenas um *bottleneck*, dado pelo *tight link* e não assume que o valor da capacidade do *bottleneck* é conhecido.

Em vez de pacotes serem enviados a um ritmo binário constante, este vai aumentando com o tempo. Quando o ritmo binário dos pacotes de teste enviados (*probing rate*) é superior ao valor de largura de banda disponível, o *tight link* fica saturado e dá-se a construção de uma fila de pacotes que ficam em espera para serem transmitidos. Nestas circunstâncias, o recetor consegue detetar a saturação, dado que a latência dos pacotes enviados sucessivamente começa a aumentar. A partir desse momento o recetor consegue estimar a largura de banda disponível como sendo o *probing rate* mínimo que não satura o *tight link*. No fundo, o objetivo é encontrar o ponto - *turning point* em que a latência começa a aumentar e nesse ponto a largura de banda disponível é igual ao *probing rate*.

Apesar de ser um modelo simples de implementar e utilizar, apresenta a desvantagem de gerar tráfego de teste em quantidades significativas. Ainda assim, em [17] é mencionado que o modelo PRM é mais preciso a estimar a largura de banda do que o modelo PGM. Por isso, é utilizado por diversas ferramentas de estimação de largura de banda existentes, tais como Pathload [18], pathChirp [19], PTR [16] e TOPP [12].

Iperf

O Iperf [20] é uma das ferramentas mais conhecidas e adotadas para a avaliação de ligações, tendo como objetivo medir a BTC, podendo ser utilizados pacotes *User Datagram Protocol* (UDP) e TCP. Esta ferramenta mede o BTC através do estabelecimento de uma ligação com um dispositivo selecionado e tentando enviar dados para a rede o mais rápido possível [4]. Por exemplo, com as configurações por omissão e utilizando tráfego TCP, são injetadas sequências de 8 KiB ao longo de 10 segundos [21]. É, portanto, uma ferramenta bastante intrusiva, dado que tenta preencher a totalidade da largura de banda da ligação.

2.2.2.2 Métodos de Estimação de Latência e *Jitter*

Nesta Secção vão ser apresentados diversos métodos para estimar a latência e outros para estimar o *jitter*.

Ping

O *Ping* é uma das ferramentas mais conhecidos para obter estimativas acerca de RTT e perda de pacotes, estando presente em praticamente todos os sistemas operativos. Outra utilidade do *Ping*, e a maior razão pela qual esta ferramenta é utilizada, é para verificar se determinado dispositivo está disponível.

Para ambos os casos de utilização, o seu princípio de funcionamento é o mesmo, baseando-se no envio de mensagens ICMP *echo request* até ao nó pretendido, respondendo esse nó com mensagens ICMP *echo response*. A diferença entre o instante no qual a mensagem foi enviada e o instante no qual foi recebida a resposta, dá uma estimativa do RTT. Caso não seja retornada nenhuma mensagem durante um *timeout* definido, assume-se que o nó destino está inacessível.

Apesar da simplicidade e forte adoção desta ferramenta pela generalidade dos sistemas, existem dispositivos que são configuradas para limitar o ritmo binário, filtrar ou rejeitar pacotes ICMP, dada a existência de muitas ferramentas para encontrar dispositivos para posterior realização de ataques [4]. Por esta razão, o *Ping* não é tido como uma ferramenta adequada para obtenção de estimativas fidedignas.

One-Way Active Measurement Protocol

O *One-Way Active Measurement Protocol* (OWAMP) é definido no RFC 4656 [22] como um protocolo para medir características unidirecionais tais como latência unidirecional e perda de pacotes unidirecional.

É um protocolo desenhado com a segurança em mente, utilizando simples pacotes UDP para as medições que se propõe a fazer. Por serem usados pacotes UDP, é difícil identificar este protocolo e, por consequência, manipulá-lo. O tráfego de teste pode ser cifrado, impossibilitando os atacantes de mudarem os *timestamps* dos pacotes sem que essa mudança seja detetada.

A arquitetura deste protocolo divide-se em cinco perfis, tendo cada dispositivo a capacidade de assumir vários:

- *Session-Sender*: O nó que inicia a sessão de testes;
- *Session-Receiver*: O nó recetor da sessão de testes;

- *Server*: Controla as sessões de teste, configura estados para cada sessão e retorna os resultados dos testes;
- *Control-Client*: Inicia pedidos para a realização de sessões de teste, sendo responsável também por decidir acerca do início e do fim das mesmas;
- *Fetch-Client*: Inicia pedidos para obter os resultados de sessões de testes terminadas.

Para medir a latência unidirecional, são enviados pacotes de teste desde um nó remetente até um nó recetor e, quando estes pacotes chegam ao recetor, são apontados os instantes nos quais os pacotes foram enviados e recebidos, números de sequência e os valores TTL. O protocolo calcula a latência comparando os tempos de chegada e de envio dos pacotes. Naturalmente, o tempo de envio é obtido no remetente e o tempo de chegada é obtido no recetor. Assim sendo, os relógios de ambos os nós têm de estar sincronizados com o máximo de rigor. Este é um dos problemas apontados ao protocolo, dado que este requisito não é fácil de cumprir.

Two-Way Active Measurement Protocol

O *Two-Way Active Measurement Protocol* (TWAMP) é definido no RFC 5357 [23] como um protocolo baseado no OWAMP, que pretende adicionar ao mesmo a capacidade de medição de latência bidirecional e medição de RTT. A sua arquitetura é idêntica à do OWAMP, salvo algumas exceções. O papel *Session Receiver* é substituído pelo *Session Reflector*, capaz de criar e enviar pacotes de teste aquando da receção de pacotes de teste enviados pelo *Session Sender*. O *Session Reflector* não regista nenhuma informação do seu lado, a informação da latência bidirecional só está disponível quando os pacotes de teste enviados pelo remetente são refletidos pelo destinatário e chegam novamente ao remetente. O papel *Fetch-Client* não existe na arquitetura do TWAMP, dado que como o *Session Reflector* não faz nenhum registo de dados, o Servidor não tem capacidade para retornar os resultados dos testes.

Para a realização de uma medição de latência ou RTT, o remetente envia pacotes de teste ao nó destinatário, que, por sua vez, envia-os de volta para o remetente. Desta forma o remetente consegue calcular o tempo que o pacote demorou a partir dele próprio até ao destinatário e do destinatário até ele mesmo.

Devido ao facto do nó destinatário não realizar nenhum registo de medidas, não existe necessidade de sincronização entre o relógio do remetente e o relógio do destinatário. Por essa razão, este método é mais fácil de implementar do que o OWAMP, sendo, por isso, mais utilizado. Além disso, como referido anteriormente na análise da ferramenta

Ping, o TWAMP é tido em conta como mais preciso do que o *Ping*, devendo-se dar prioridade à utilização deste [24] [25].

Em [24] é realizada uma comparação entre as medições realizados pelo *Ping* e pelo protocolo TWAMP, numa topologia composta por dois dispositivos e dois *routers*, em que o dispositivo A está ligado ao *router* RA e o dispositivo B ao *router* RB. O dispositivo A é responsável por gerar e enviar os pacotes de teste para o dispositivo B. A porta do *router* RA, *router* por onde vão entrar os pacotes de teste vindos do dispositivo A, é saturada (a largura de banda é limitada) propositadamente para aferir o desempenho do *Ping* e do TWAMP nestas circunstâncias. Para os pacotes de teste do TWAMP foram testados três valores distintos de marcação DSCP, enquanto no caso dos pacotes ICMP (*Ping*), a marcação é *Best Effort*. Os pacotes do *Ping* serem marcados como *Best Effort* serve o propósito de simular a fraca prioridade que os dispositivos de rede podem atribuir a estes pacotes.

Com os testes realizados verifica-se que a medição do RTT entre o TWAMP, para qualquer marcação DSCP, e o *Ping* é idêntica, bem como a do *jitter*. Por outro lado, os pacotes ICMP apresentaram uma elevada taxa de perda de pacotes, dado não serem considerados prioritários. Os pacotes de teste do TWAMP marcados com DSCP 0 apresentaram também uma elevada perda de pacotes visto que *Differentiated Services Code Point* (DSCP) 0 não é uma marcação prioritária. Ainda assim, a perda de pacotes observada é inferior à que ocorreu com os pacotes ICMP. Comprova-se assim que o *Ping* não deve ser considerara uma ferramenta de estimação com alta fiabilidade, sendo ultrapassada pelo TWAMP. As mesmas conclusões são retiradas em [25].

2.3 Avaliação de Qualidade de Ligações em Redes Móveis

Após analisados métodos passivos e ativos genéricos para avaliação de redes, nesta Secção irão ser analisados alguns métodos de estimação de qualidade de ligações no âmbito de redes móveis. Os métodos apresentados a seguir baseiam-se nos métodos genéricos abordados, mas são focados especificamente em redes móveis.

2.3.1 NEXT-FIT

A técnica NEXT, proposta em 2014 por A. K. Paul et al. [26], pretende estimar a largura de banda em redes *Wireless*, incluindo redes móveis. Baseia-se na ferramenta

pathChirp [19] que, por sua vez, se baseia no modelo PRM para estimar a largura de banda disponível.

Tal como em pathChirp, são utilizados conjuntos de pacotes de teste (*packet train* ou *packet chirp*) sendo enviados desde um remetente até um destinatário. Cada *packet train* possui um número de pacotes que depende dos ritmos de transmissão a testar e de um *Spread Factor*. O *Spread Factor* define o espaçamento temporal entre pacotes pertencentes ao mesmo *packet train*. A forma do tráfego injetado, bem como o espaçamento existente entre os pacotes é o que diferencia os dois métodos.

A ferramenta pathChirp permite a configuração de diversos parâmetros para assim encontrar a largura de banda com precisão. Dado que em redes reais as condições mudam constantemente, para se obter uma estimativa precisa com esta ferramenta, é necessário realizar constantes alterações às configurações dos seus parâmetros, o que não é conveniente.

A técnica NEXT-FIT [27], uma extensão de *NEXT*, proposta pelos mesmos autores, surge com o objetivo de realizar estimativas do valor da largura de banda em redes *Wireless*, incluindo redes móveis. Em vez de se utilizar o algoritmo da ferramenta pathChirp, cuja precisão da estimativa depende da correta configuração de todos os seus parâmetros, os autores apresentam uma técnica *curve-fitting* independente de parâmetros de configuração. *Curve-fitting* é o processo de construir uma função matemática que se aproxima o máximo possível de uma série de pontos, neste caso, pretende-se encontrar uma função matemática que vá de encontro à largura de banda, com base nos valores de latência de *queuing* medida, causada pelos pacotes de teste enviados.

2.3.2 PathML

O PathML é um método para estimar a largura de banda apresentado por N. Sato et al. em 2017 [28]. O método proposto utiliza *Machine Learning* para, com base numa grande quantidade de dados recolhidos, obter uma estimativa.

Tal como tantas outras ferramentas, o PathML utiliza o modelo PRM para forçar o congestionamento da rede e encontrar o ponto a partir do qual a latência de pacotes enviados sucessivamente aumenta, significando que o ritmo binário dos pacotes de teste é mais elevado que o valor de largura de banda disponível. Desta forma, a largura de banda pode ser estimada com base na transição entre a não existência de latência, dado que o ritmo binário é mais baixo do que a largura de banda, não levando a que se criem filas de pacotes em espera e o momento em que começa a existir latência com criação de filas de pacotes.

Antes da realização de estimativas, é gerado um preditor de largura de banda utilizando um algoritmo de *Machine Learning*. Este preditor é treinado utilizando conjuntos de valores de latência observados no recetor e um valor de largura de banda real correspondente. O preditor utiliza os valores de latência para estimar um valor de largura de banda. O valor de largura de banda fornecido ao modelo é utilizado por este para este aprender, alterando os seus parâmetros para conseguir retornar uma boa estimativa de largura de banda com base em valores de latência recebidos. Após o processo, podem-se fornecer novos conjuntos de dados de latência ao preditor e este vai produzir uma estimativa da largura de banda.

Especificamente, foram tidos em conta quatro algoritmos de *Machine Learning* para a estimação: *Supported Vector Regression* (SVR), *Kernel Ridge Regression* (KRR), *Random Forests* (RF) e *Convolutional Neural Networks* (CNN). O que estes têm em comum é serem algoritmos de Aprendizagem Supervisionada, uma subsecção dos algoritmos de *Machine Learning*. Os algoritmos de aprendizagem supervisionada caracterizam-se por pretenderem estimar um valor de saída correspondente aos valores de entrada que lhe são fornecidos. A aprendizagem dá-se pelo processo de treino do algoritmo, em que são fornecidos dados de treino, que consistem em pares de valores de entrada e um valor de saída correspondente. Ou seja, os valores de entrada são utilizados pelo algoritmo para este adaptar os seus parâmetros para que o resultado que este produz seja próximo do valor de saída que lhe foi fornecido para treino.

A precisão da estimação dos quatro algoritmos foi comparada entre si e ainda comparada com o método PathQuick3, cuja estrutura de envio de pacotes de teste é igual à adotada pelo PathML. Os algoritmos foram testados numa rede *Long Term Evolution* (LTE) de um operador do Japão. A maior precisão na estimação da largura de banda foi obtida através da utilização do algoritmo *Convolutional Neural Networks*, sendo 83,7% do *ground truth*. A precisão do PathQuick3 foi a mais baixa de todas (74,1%), ficando assim demonstrado que o método PathML, sobre qualquer dos quatro algoritmos testados, apresenta uma melhor estimativa da largura de banda.

2.3.3 TCPW

O TCP *Westwood* (TCPW) [29] é um algoritmo utilizado para controlar o congestionamento de uma ligação TCP. Este é, na realidade, uma modificação de outro algoritmo de congestionamento, o TCP *Reno*. Difere de todos os outros algoritmos devido à forma como ajusta a janela de congestionamento em caso de deteção de perdas. A janela de congestionamento é uma variável que limita a quantidade de dados que podem ser

enviados para a rede antes de ser recebido um pacote de ACK. O TCPW, em vez de diminuir para metade a janela quando são detetadas perdas, como acontece nos restantes algoritmos, realiza um ajuste consoante a largura de banda detetada.

O TCP *Westwood* calcula a largura de banda do lado do remetente como sendo a porção da largura de banda do *bottleneck* utilizada por todos os fluxos de dados na rede. O remetente regista o tamanho de todos os ACK e o intervalo dos ACK adjacentes para calcular a largura de banda. A largura de banda no instante k (b_k), é dada pela equação 2.12, onde $\Delta_k = t_k - t_{k-1}$ e t_{k-1} é o instante em que foi recebido o anterior ACK e d_k é a quantidade de dados transferida entre t_{k-1} e t_k.

$$b_k = \frac{d_k}{\Delta_k}. \tag{2.12}$$

Comparações realizadas em [29] entre o TCP *Reno* e o TCP *Westwood*, mostram que uma ligação TCP que utiliza TCPW consegue ritmos de transmissão até 550% maiores do que uma ligação com TCP *Reno*. Ainda assim, não significa que o algoritmo de estimação de largura de banda utilizado pelo Westwood consiga sempre ser preciso. Existem situações em que este sobrestima a largura de banda [30], algo que pode levar ao congestionamento da rede. Para além disso, o algoritmo não é adequado para redes nas quais as condições estão em constante mudança, como as redes móveis, às quais os dispositivos móveis estão ligados.

Para tornar o algoritmo mais preciso e adequado a redes móveis, Z. Chen et al. introduziram o método TCP *Westwood* com *Holt-Winters* (TCPWH) [31]. O algoritmo de estimação de largura de banda utilizado no TCPW é um algoritmo simples de suavização exponencial que falha em lidar com séries temporais complexas, levando a que exista uma histerese, fazendo com que a precisão da previsão do valor de largura de banda não seja preciso. Por sua vez, *Holt-Winters* é um método de suavização exponencial tripla utilizado em séries temporais, permitindo lidar com padrões complexos que apresentem sazonalidade, de forma a realizar previsões de valores futuros com precisão.

Existem três componentes tidas em conta na previsão realizada pelo *Holt-Winters*: valor médio, tendência e sazonalidade. Estas três componentes levam a que a previsão seja realiza a três níveis e a sua combinação retorna uma previsão mais precisa do que aquela retornada por TCPW, que é composta apenas por um componente. Esta é a grande diferença entre o TCPW e o TCPWH, enquanto os restantes mecanismos no TCP para ambos os métodos permanecem os mesmos.

As simulações realizadas tiveram em conta cenários comuns em redes LTE. Verificou-se que o TCPWH apresentou uma estimativa mais precisa e mais rápida a reagir a alterações. Ainda, o TCPWH não sobrestima a largura de banda, como acontece com o TCPW.

2.3.4 D-TCP

O D-TCP, introduzido em 2018 por M. R. Kanagaranthinam et al. [32], é um algoritmo de controlo de congestionamento TCP pensado para ser utilizado em redes móveis como o LTE e 5G-*New Radio* (5G-NR). Este algoritmo aprende dinamicamente a largura de banda disponível e, a partir dessa, deriva um factor de controlo de congestionamento N. Este fator é utilizado para ajustar o tamanho da janela de congestionamento dinamicamente, incrementando-a ou decrementando-o a adaptativamente, diferindo dos algoritmos de controlo de congestionamento tradicionais.

Quando são detetadas perdas, o algoritmo D-TCP não vai deixar que a janela de congestionamento diminua para metade como acontece nos restantes algoritmos. Depois desta baixar para um determinado valor, o algoritmo faz com que o tamanho da janela atual tenda para o tamanho da janela no estado anterior, com a ajuda da largura de banda calculada com base em aprendizagem.

Neste algoritmo, para estimar a largura de banda no instante i (B_I), apresentada na equação 2.13, é utilizado um filtro utilizando uma aproximação Tustin, que depende da quantidade de dados transferidos num intervalo $Delta_i = t_i - t_{i-1}$ (B_i), e da estimativa anterior de largura de banda (B_{i-1}). À largura de banda (B_{i-1}) é atribuído um peso de 90% e à largura de banda (B_i) um peso de 10%.

$$B_I = 0,90B_{i-1} + 0,10B_i. \tag{2.13}$$

Para evitar o potencial *aliasing* criado pelo filtro, B_I é normalizada, dando origem ao valor de largura de banda estimada BW_E, observável na equação 2.14, onde g é a granularidade do relógio utilizado.

$$BW_E = B_I + g \times (B_i - B_I). \tag{2.14}$$

O fator N que controla o tamanho da janela de congestionamento é calculado com base na largura de banda estimada BW_E, da largura de banda corrente BW_C e do tamanho da janela TCP W_i. Assim, o fator N no instante i+1 é dado pela equação 2.15.

$$N_{i+1} \leftarrow max(1, N_i + 1 - \frac{BW_E - BW_C}{\alpha \times BW_E} W_i). \tag{2.15}$$

Para colocar à prova o D-TCP, este e o algoritmo TCP *Cubic* foram testados recorrendo a dois terminais móveis iguais, um utilizando D-TCP e o outro o TCP *Cubic*. Ambos ligam-se, via *routers* Wi-Fi, a servidores localizados em diversas cidades e países distintos, sendo assim testados cenários com valores de RTT variados. Os resultados mostraram que o *goodput* alcançável com D-TCP é superior ao do TCP Cubic. Para um servidor cujo RTT associado é 10 ms, o *goodput* do D-TCP foi 70,98% superior ao do TCP Cubic. Para um servidor com RTT de 100 ms o *goodput* do D-TCP foi 118,99% superior ao do TCP Cubic e para um servidor com RTT de 220 ms o *goodput* do D-TCP foi 62,83% superior ao do TCP Cubic.

O algoritmo D-TCP foi também testado numa rede LTE, sendo comparado com o algoritmo *Congestion Level TCP* (CLTCP) [33], TCP *Cubic* e TCP *Reno*. Verificou-se que no CLTCP as alterações do tamanho da janela de congestionamento são muito mais agressivas e em caso de perdas demora mais tempo a voltar ao estado anterior do que o D-TCP. Num cenário em que existe um elevado número de perdas, o D-TCP é o que consegue tirar melhor partido da largura de banda disponível, já que consegue evitar a descida desnecessária do tamanho da janela de congestionamento.

2.4 Segurança da Rede

Numa arquitetura de um sistema de suporte a serviços críticos, a segurança dos dados trocados nas comunicações é de tremenda importância. Uma forma de garantir a segurança destes dados passa pela utilização de um túnel VPN. Um túnel VPN é um mecanismo de comunicação capaz de, através de uma rede pública, estabelecer ligações entre múltiplas redes separadas entre si. As VPN transportam dados de qualquer tipo em túneis, sobre um protocolo de transporte como UDP ou TCP, realizando a sua encriptação, o que faz com que a informação transportada dentro do túnel não seja passível de análise, contribuindo para a sua segurança. As VPN podem funcionar ao nível de diferentes camadas do modelo OSI, tais como camada 2 ou camada 3, sendo mais comum ao nível desta última.

Geralmente é preferível que as VPN utilizem como protocolo de transporte UDP em vez de TCP, dado que, caso os dados a navegar sobre a VPN sejam TCP, estando a VPN a utilizar TCP como protocolo de transporte, tem-se um problema bastante conhecido do TCP sobre TCP, conhecido como TCP *Meltdown*, que é, no fundo, a diminuição do

ritmo binário alcançável na ligação VPN. Como é sabido, o TCP possui mecanismos de adaptação às condições da rede, o protocolo de congestionamento, visando garantir que os pacotes enviados chegam ao seu destino. Ora, quando se têm pacotes TCP sobre uma VPN TCP, quando os mecanismos do TCP da VPN detetam um problema, tentam encontrar uma solução, porém, os mecanismos dos pacotes TCP transportados na VPN tentam também encontrar uma solução. Existindo duas tentativas de encontrar uma solução, sendo que a segunda já incide sobre a primeira, existe um aumento de *delay* e um aumento dos problemas na transferência de dados. A existência de dois protocolos de congestionamento e a operação de ambos em simultâneo faz com que estes interfiram entre si, fazendo com que, em determinados cenários, exista uma degradação do desempenho das comunicações, não sendo, por isso, recomendada a utilização de TCP como protocolo de transporte de uma VPN, na generalidade das situações.

O desempenho de uma VPN com TCP depende também dos parâmetros configurados tanto para o protocolo TCP dos dados que nela circulam como do TCP da própria VPN. Em [34] é realizada uma análise do impacto que parâmetros como o *delay* de propagação, *Selective Acknowledgment* (SACK) ou o tamanho do *socket buffer* têm no desempenho de uma ligação TCP sobre um túnel VPN TCP.

O desempenho da rede não é o único fator a ter em conta na escolha entre UDP e TCP. Estes são protocolos bem distintos que devem ser escolhidos consoante o cenário no qual vão ser aplicados. O TCP é um protocolo *Connection-Oriented*, garante uma comunicação fiável, assegurando a entrega de pacotes ao destinatário e preservando a ordem correta de chegada dos pacotes, possuindo, como mencionado anteriormente, mecanismos de adaptação às condições de rede. Por outro lado, o UDP é *Connectionless*, não garante entrega de pacotes (muito menos a sua correta ordem de chegada) nem possui mecanismos de adaptação, sendo, por isso, as ligações sobre UDP, geralmente, mais rápidas.

2.4.1 OpenVPN

A OpenVPN [35] é uma VPN *open source* que permite a criação de canais seguros entre *endpoints*, através de *Transport Layer Security* (TLS). Esta é uma das VPN mais populares dada a sua facilidade de configuração face a outras VPN. Para além disso é possível utilizá-la em praticamente qualquer dispositivo, incluindo dispositivos Android e IOS. Utiliza a biblioteca OpenSSL para as tarefas de encriptação e autenticação, suportando todas as cifras suportadas pela biblioteca. Por omissão, desde a versão 2.5 da OpenVPN, é utilizada a cifra AES-256-GCM, uma cifra autenticada que fornece também integridade, através da função GHASH, eliminando a necessidade de

um algoritmo adicional para esse fim. Para além de AES-256-GCM, é recomendada a utilização de AES-128-GCM. Em versões anteriores a cifra utilizada por omissão era AES-256-CBC e era recomendada também a cifra AES-128-CBC.

Contrariamente à maioria das VPN existentes, que permitem a criação de túneis apenas através do protocolo de transporte UDP, a OpenVPN permite a criação de túneis através de TCP. Cada um destes tipos tem as suas vantagens e casos de utilização. Um túnel UDP apresenta maiores ritmos binários e menor latência, mas é menos estável do que um túnel TCP [36]. O tempo necessário para início de conexão é bastante elevado se comparado com outras VPN existentes no mercado [37].

Outro aspeto importante é que pode atuar tanto no nível 2 como no nível 3 da camada OSI. No nível 2 é utilizada para criar uma ponte entre redes e no nível 3 é utilizada para criar um túnel de rede.

A OpenVPN é *single-threaded*, não sendo por isso muito escalável em ambientes compostos por diversos CPU [38].

2.4.2 *Internet Protocol Security*

Internet Protocol Security (IPsec) é um grupo de protocolos utilizados em conjunto para adicionar a capacidade de encriptação (segurança) ao protocolo IP, dando-se a encriptação dos dados de uma conexão entre dois ou mais dispositivos, sendo por isso utilizado em VPN.

IPsec garante segurança dos dados tendo mecanismos de autenticação, verificação de integridade dos pacotes e encapsulamento dos mesmos num túnel seguro. Tratando-se de um grupo de protocolos normalizado, é possível estabelecer um túnel VPN entre equipamentos de diferentes fabricantes. Este grupo de protocolos atua diretamente sobre o protocolo IP e não sobre a camada de transporte, como é habitual em outros protocolos VPN.

Um dos grandes problemas apontados a este protocolo é a dificuldade de configuração. Ainda assim, é praticamente suportado por todos os tipos de sistemas.

Uma das soluções mais conhecidas que utiliza IPsec é a strongSwan [39] e, dado que IPsec é considerado um protocolo muito escalável, são suportadas diversas cifras, tais como *Advanced Encryption Standard* (AES) com modo *Cipher Block Chaining* (CBC) e AES com modo *Galois/Counter Mode* (GCM). strongSwan também possui suporte a *multithreading*, no entanto, a implementação do IPsec no Kernel Linux não suporta.

Em IPsec existem dois protocolos para garantir a proteção dos dados: *Authentication*

Header (AH) e *Encapsulating Security Payload* (ESP). Cada um destes suporta dois modos de funcionamento: **modo transporte** e **modo túnel**. A principal diferença entre estes é que o modo de transporte utiliza o cabeçalho do pacote IP original, enquanto no modo túnel utiliza-se um novo cabeçalho IP. O modo transporte é habitualmente utilizado para proteger os pacotes trocados entre dois dispositivos. Já o modo túnel é habitualmente utilizado nas VPN *site-to-site* onde é necessário encapsular os pacotes IP originais, dado que os pacotes, na sua grande maioria, têm origem em redes privadas, não podendo navegar na internet. Neste modo, os pacotes IP originais que navegam no túnel são encriptados desde o início até ao fim da VPN, sendo desencriptados no fim do túnel para serem entregues ao seu destinatário.

2.4.3 WireGuard

A WireGuard, segundo descrição do próprio autor e criador Jason Donenfeld [40], define-se como uma implementação moderna, rápida e segura de uma VPN, pretendendo ser mais simples, rápida e de mais fácil utilização do que IPsec. Assume-se ainda como consideravelmente mais rápida do que a OpenVPN.

Este protocolo foi desenvolvido de raiz, procurando utilizar as cifras mais atuais e seguras, atuando no nível 3 do modelo OSI. Tem ganho notoriedade principalmente desde o momento em que foi incluído no Kernel 5.6 e também pelo desempenho anunciado, que é bastante promissor. Porém, o autor reconhece que os *benchmarks* apresentados [40] não estão atualizados e não foram muito bem conduzidos, apresentando-se demasiado otimistas. Diversos artigos abordados na Secção 2.4.4 revelam qual o real desempenho desta VPN. Em alguns cenários é efetivamente muito mais rápida do que as restantes, mas existem também cenários em que tal não acontece.

Como protocolo de transporte apenas é possível a utilização de UDP. Ao nível das cifras é utilizado ChaCha20 para encriptar os dados, combinada com Poly1305 para garantir autenticação, não existindo nenhuma outra opção. Desta forma o código da implementação permanece curto e mitigam-se potenciais erros de configuração, que poderiam levar ao estabelecimento de ligações não seguras. Não são usados quaisquer certificados para o estabelecimento de ligações, fazendo com que sejam necessários menos cálculos, aumentando o desempenho da VPN.

Outro aspeto importante é que esta VPN suporta *multithreading*, sendo, por isso, possível escalar em ambientes compostos por múltiplos CPU. No entanto, para redes de elevada capacidade esta não consegue escalar eficazmente [38].

2.4.4 Desempenho das VPN

Ao nível de desempenho, existem diversos artigos que realizam comparações entre os diferentes protocolos para estabelecimento de VPN apresentados. Devido ao surgimento da WireGuard ser recente, e ainda mais recente é a sua inclusão no Kernel Linux, existem múltiplos artigos em que a WireGuard é comparada, mas numa versão anterior à sua inclusão no Kernel. Outro aspeto importante é o facto da WireGuard possuir apenas um protocolo de criptografia, o ChaCha20, contrariamente aos restantes protocolos VPN que possuem múltiplos protocolos para esse fim, não sendo, por isso, possível realizar uma comparação entre todos os tipos de criptografia suportada por cada VPN.

Começando com um artigo publicado em 2015 por Conjaah et al. [36], neste é avaliada a diferença de desempenho entre OpenVPN com túneis TCP e UDP. Quando é utilizado um túnel TCP a latência é maior (e o ritmo binário menor) do que quando é utilizado um túnel UDP, sendo que o pior cenário ocorre quando é transportado tráfego TCP sobre um túnel TCP. Quando o transporte de dados TCP é realizado sobre um túnel UDP, a latência é mais baixa e o ritmo binário alcançado é superior. Conclui-se que um túnel UDP utiliza as ligações de forma mais eficaz, tornando possível obter ritmos de transmissão muito superiores aos alcançados com um túnel TCP. Ainda, ficou demonstrado que transportar TCP num túnel UDP é mais eficaz do que transportar TCP sobre TCP.

Apesar da comparação entre túneis TCP e UDP da OpenVPN, não é realizada nenhuma comparação face a outras VPN.

Em 2018, num artigo publicado por Pudelko et al. [38], é realizada uma comparação do desempenho de *gateways* VPN sobre ligações de 10 e 40 Gbps para entender se existe alguma VPN preparada para ligações de elevada capacidade. Foram testadas implementações de IPsec com ESP em modo túnel e cifra AES-GCM, que se trata de uma cifra com aceleramento *Advanced Encryption Standard New Instruction* (AES-NI), OpenVPN com AES-CBC sem autenticação com e sem AES-NI e, por último, WireGuard, numa versão anterior à inclusão no *Mainline* Kernel.

IPsec falha em escalar quando são utilizados diversos CPU e a OpenVPN sendo *single-threaded*, só consegue utilizar um CPU, ficando o seu desempenho muito aquém. Nota-se ainda uma diferença significativa quando AES-NI está ativo, alcançando-se um desempenho superior, demonstrando que a ativação de aceleração ao nível de *hardware* é relevante. A WireGuard foi a que apresentou o pior desempenho, não conseguindo

escalar tão bem como as restantes, dado que perde muito tempo a bloquear recursos do sistema, fazendo com que o processo criptográfico se torne lento. Concluiu-se que nenhuma das VPN está preparada para lidar com a quantidade de pacotes em ligações de 10 ou 40 Gbps, estando limitadas pelo Kernel.

Apesar de serem comparadas três soluções VPN, a WireGuard testada não é a versão incluída no Kernel e os autores também não referem se foi utilizado UDP ou TCP para os testes OpenVPN. Ainda, para esta VPN, não foram testadas as cifras AES-GCM e ChaCha20Poly1305, mais recentemente disponibilizadas e para IPsec também não foi testada a cifra ChaCha20Poly1305.

Outro artigo relevante foi publicado em 2020 por Osswald et al. [41], onde é realizada uma análise do desempenho da OpenVPN, WireGuard e IPsec (através de strongSwan), numa ambiente de testes composto por dois servidores virtuais e uma ligação de 10 Gbps entre estes. Foi utilizada uma versão da WireGuard anterior a esta ter sido incluída no Kernel. Foram testadas todas as configurações de cifras tanto para strongSwan como para OpenVPN (exceto ChaCha20 que não era suportada pela OpenVPN na data do teste) e para as cifras AES foi testada a utilização de AES-NI. Foram ainda testados dois cenários, sendo que num deles é utilizado CPU turbo *boost* e noutro é utilizado CPU *pinning*, isto é, colocam-se as VPN a correr sobre um CPU dedicado às mesmas, em conjunto com a não utilização de CPU turbo *boost*.

Sem CPU *pinning*, com CPU turbo *boost* ativo e com AES-NI ativo, todas as implementações de IPsec apresentaram melhores resultados que a WireGuard. A OpenVPN apresenta ritmos de transmissão muito inferiores às restantes. Sem AES-NI a WireGuard apresenta melhores resultados.

Quando utilizado CPU *pinning* em conjunto com a não utilização de *boost*, a WireGuard consegue um incremento de desempenho na ordem dos 40%, levando a concluir que é a implementação mais eficaz na utilização do CPU. Para além disso foi a que apresentou melhor desempenho entre todas as implementações, quando utilizado CPU *pinning*. Quanto às restantes implementações, a não utilização de *boost* é visível no decremento de desempenho tanto da OpenVPN como de IPsec.

Concluindo, IPsec com a utilização de AES-NI é a que apresenta melhores resultados quando não se utiliza CPU *pinning*. Caso seja possível utilizar esta última técnica, então a WireGuard consegue um desempenho superior.

Mesmo sendo comparadas três soluções de VPN, não são apresentados resultados acerca da latência nem foram testadas as cifras AES-GCM. Para a OpenVPN a cifra ChaCha20Poly1305 também não foi testada.

O artigo publicado por Dekker et al. [37] demonstra que o desempenho alcançado com WireGuard nem sempre é tão superior às restantes VPN. Foram comparadas as VPN WireGuard, OpenVPN e strongSwan (IPsec). Como topologia de testes foram utilizados dois servidores, estando o cliente VPN num deles e o servidor VPN no outro. Num deles foi instalado o cliente VPN e no outro o servidor VPN. Os dois servidores foram ligados um ao outro com um *switch* no meio e com ligações de 1 Gbps. Ambos os servidores utilizados tinham 16 GiB de memória RAM, o cliente VPN um processador Intel Xeon E3-1240L e o servidor VPN um processador Intel Xeon E-2124. Ambas as máquinas tinham instalado o Ubuntu 18.04 com o Kernel 5.6.17-050617-generic.

Neste artigo foram avaliados os parâmetros *goodput*, latência, utilização de CPU e tempo de início para cada VPN, considerando tráfego TCP e UDP. Foram também realizados diversos testes com base no tamanho dos pacotes enviados sobre as VPN, sendo a comparação mais relevante para um tamanho de 64 *bytes* e tamanho máximo suportado por cada VPN. O tamanho máximo depende da própria VPN, dos dados serem TCP ou UDP e também do algoritmo criptográfico utilizado. Outro aspeto importante é que este é, dos artigos encontrados à data, o único que testa o desempenho da OpenVPN e strongSwan quando utilizados os algoritmos AES-GCM.

Os melhores resultados foram obtidos quando utilizados pacotes de tamanho máximo e tráfego UDP com a OpenVPN e strongSwan, ambas utilizando AES-GCM. O ritmo binário médio alcançado com OpenVPN foi de 922 Mbps e para strongSwan foi de 921 Mbps. Quanto à WireGuard, o ritmo obtido foi de 917 Mbps.

As cifras AES-CBC têm pior desempenho do que as AES-GCM, excetuando o cenário de tráfego TCP com pacotes de 64 *bytes*.

Quando utilizados pacotes de 64 *bytes* em tráfego UDP, o melhor desempenho, 117 Mbps, foi obtido com strongSwan utilizando a cifra AES-GCM. Segue-se a WireGuard com 109 Mbps e strongSwan utilizando ChaCha20 com 90 Mbps. Por último surge a OpenVPN, em que o seu melhor resultado foi de 48 Mbps para a cifra AES-GCM.

Quanto a tráfego TCP, com pacotes de dimensão máxima, o melhor desempenho em termos de ritmo binário foi alcançado pela strongSwan com AES-GCM, sendo de 906 Mbps. Logo a seguir surge a WireGuard com 901 Mbps e, por fim, a OpenVPN que, com qualquer das cifras, foi a que alcançou o menor ritmo binário, ainda assim, com AES-GCM, não ficou longe das restantes.

Ainda para tráfego TCP, mas com pacotes de 64 *bytes*, os resultados obtidos para todas as VPN com as diversas cifras são bastante idênticos. OpenVPN e strongSwan,

ambas com AES-GCM, atingiram 179 e 178 Mbps respetivamente. As restantes combinações de cifra tanto para OpenVPN como para strongSwan também ultrapassam o desempenho da WireGuard, sendo que esta foi a pior, com apenas 156 Mbps.

Abordando agora o desempenho em termos de latência, todas as cifras da VPN strongSwan apresentam valores inferiores aos obtidos com OpenVPN (com qualquer cifra) e WireGuard. A latência média da strongSwan, com CBC e ChaCha20 é de 0,22 ms e, para AES-GCM é de 0,21 ms. Quanto à OpenVPN, a latência média medida com AES-CBC foi de 0,40 ms e com AES-GCM foi de 0,39. Em último lugar surge a WireGuard, com a qual se registou uma latência de 0,42 ms.

Abordando outro aspeto importante, o tempo de início de ligação, a WireGuard apresenta o melhor resultado, levando apenas 6,9 ms para estabelecer ligação. Dado que strongSwan e OpenVPN suportam autenticação com e sem certificados, enquanto WireGuard não suporta certificados, foram analisados os valores de tempo de início de ligação para ambos os cenários. strongSwan com certificados apresenta um valor médio de 33,6 ms, sem certificados este valor é de 31,8 ms. Quanto à OpenVPN, esta é, de longe, a VPN que apresenta os piores tempos de início de ligação. Com certificados o valor é de 1152,7 ms e sem certificados é de 954,9 ms.

Quanto à utilização de CPU, a maior eficiência foi demonstrada pela strongSwan com AES-GCM, tanto para TCP como para UDP. WireGuard apresenta menor utilização de CPU do que OpenVPN para tráfego UDP e TCP, ambos com utilização de pacotes de dimensão máxima. O mesmo acontece ainda para TCP com pacotes de 64 *bytes*. No caso de UDP com pacotes de 64 *bytes*, OpenVPN apresenta menor utilização de CPU do que WireGuard, transmitindo, no entanto, menos de metade dos pacotes transmitidos pela WireGuard.

Em conclusão dos resultados deste artigo, strongSwan com AES-GCM apresentou-se como a implementação com melhor desempenho em termos de *goodput*, latência e eficiência na utilização de CPU. A OpenVPN apresenta um desempenho similar à strongSwan em termos de *goodput* para o caso de tráfego UDP com pacotes de dimensão máxima. A WireGuard é a implementação que apresenta os melhores resultados em termos de tempo necessário para iniciar uma ligação.

Este foi o artigo encontrado mais completo. No entanto, não é referido qual o protocolo usado no transporte dos dados da OpenVPN apesar de, geralmente, ser utilizado UDP. Assim sendo, não são apresentados resultados para tráfego transportado sobre TCP. Outra limitação é o facto de não ter sido testada a OpenVPN com a cifra ChaCha20Poly1305.

2.5 Protocolos *Multipath*

Numa arquitetura de suporte a sistemas críticos, tendo, idealmente, diversas redes disponíveis para o encaminhamento de pacotes, faz sentido pensar em formas de utilizar simultaneamente as múltiplas redes, otimizando a utilização dos recursos disponíveis. Por isso mesmo, pensou-se a utilização de protocolos *multipath*.

Inicialmente, quando a internet foi projetada, a arquitetura das conexões que se tinha em mente era entre dois dispositivos e essa comunicação aconteceria utilizando apenas uma interface de cada dispositivo. Apenas dispositivos como os *routers* ou *switches* possuíam diversas interfaces de comunicação. Ao longo dos anos a internet e as redes foram evoluindo, levando ao aparecimento de dispositivos com múltiplas interfaces de comunicação, capazes de estabelecer ligações com o exterior através dessas múltiplas interfaces. Exemplos de dispositivos com estas capacidades são os *smartphones*, que podem estabelecer ligações via Wi-Fi ou através de redes móveis tais como LTE e 5G-NR. Outros dispositivos que possuem múltiplas interfaces são os servidores. Considerando as novas capacidades de comunicação dos dispositivos, surge a ideia de aproveitar de forma eficaz todas as redes aos quais os dispositivos modernos se conseguem ligar em simultâneo, levando ao aparecimento de protocolos *multipath* que, em alguns casos, são extensões dos protocolos de transporte já existentes, tais como o TCP e UDP.

Os protocolos *multipath* apresentam vantagens tais como maior fiabilidade, dado que uma ligação *multipath* utiliza diversos caminhos e, na ocorrência de um problema num dos caminhos, os restantes continuam a manter a ligação de dados ativa, sem que as máquinas se apercebam. Oferecem agregação da largura de banda, dada pela soma da largura de banda individual de cada caminho que compõe a ligação. Oferecem também maior segurança, dado que com os dados a serem distribuídos por múltiplos caminhos, torna-se mais complexa a sua interceção. Por outro lado, existem alguns problemas associados a estes protocolos, como a compatibilidade com protocolos já existentes ou a possibilidade de reordenação de pacotes, sendo difícil controlar o encaminhamento de pacotes através de caminhos que possuem características diferentes. Desta forma, é essencial que os protocolos *multipath* possuam mecanismos para controlar como os pacotes são encaminhados consoante as características de cada caminho.

Ao longo dos anos têm surgido diversas propostas de protocolos *multipath* capazes de aproveitar de forma fiável e eficaz a existência de múltiplos caminhos para realizar comunicações. Foram surgindo soluções baseadas nas diversas camadas do modelo OSI, não sendo consensual qual a melhor camada para este tipo de protocolo. De seguida serão apresentadas as características e alguns protocolos *multipath* tanto da

camada de dados como da camada de transporte. Protocolos *multipath* ao nível da camada de rede e da camada aplicacional não foram tidos em conta neste estudo.

2.5.1 Camada de ligação de Dados

Geralmente a transmissão *multipath* de dados ao nível da camada de ligação de dados é denominada de *bonding* ou *link aggregation*, dado que múltiplos canais físicos são unidos para criar um único canal de transmissão lógico. O principal objetivo da agregação de diversos canais físicos é a sua coordenação, dado que são independentes, de forma que se crie um canal lógico cuja largura de banda é maior que a largura de banda individual de cada canal físico.

O protocolo mais conhecido desta camada é o *Link Aggregation Control Protocol* (LACP) [42], que disponibiliza uma forma de controlar a criação de um *link* lógico com base na agregação de diversos *links* físicos - interfaces *Ethernet*. Pode-se, por exemplo, agregar portas de diversos *switches*, utilizando assim múltiplas ligações de rede em paralelo, aumentando a largura de banda disponível e a capacidade, bem como disponibiliza redundância em caso de falhas.

O *Shortest Path Bridging* (SPB) [43] é outro protocolo *multipath* que atua na camada de dados, sendo visto como um dos sucessores do *Spanning Tree Protocol* (STP). Suporta o encaminhamento de dados através das estratégias de encaminhamento *Equal Cost Multipath* (ECMP) [44] e *Equal Cost Tree* (ECT) [45], ou seja, se estiverem disponíveis diversos caminhos até ao destino, os pacotes vão ser distribuídos pelos mesmos. Para evitar a reordenação dos pacotes, ambos os protocolos utilizam uma estratégia de distribuição dos dados com base no *flow* ao qual estes pertencem — *per flow multipath forwarding*, isto é, todos os pacotes pertencentes ao mesmo *flow* são encaminhados pelo mesmo caminho. Um protocolo similar a este, visto também como um sucessor do STP é o *Transparent Interconnection of a Lot of Links* (TRILL) [46]. Suporta o encaminhamento de dados através de ECMP apenas e, tal como o SPB, utiliza uma política de distribuição do tráfego *per flow multipath forwarding*.

2.5.2 Camada de Transporte

A utilização de um protocolo ao nível da camada de transporte caracteriza-se por ser transparente para a camada de rede, bem como para as aplicações, algo bastante importante dado que, modificar camadas já há muito estabelecidas, é um processo de elevada complexidade. Por exemplo, no caso de protocolos *multipath* baseados nos

protocolos de transporte clássicos como o TCP, não existe a necessidade de modificar as aplicações para estas usarem o novo protocolo, sendo transparente para estas.

Um dos primeiros protocolos *multipath* a surgir ao nível da camada de transporte foi o *Stream Control Transmission Protocol* (SCTP) [47], pensado inicialmente para o envio de mensagens de sinalização *Public Switched Telephone Network* (PSTN), mas utilizável para outro tipo de aplicações. Caracteriza-se por conseguir enviar múltiplos *streams* de mensagens ou dados através de diversos caminhos, assumindo que pelo menos um dos *endpoints* da ligação é *multi-homed* —- existe mais do que um endereço IP que pode ser usado para atingir o *endpoint*. Apesar da capacidade *multi-homed* deste protocolo, não é possível utilizar os diversos caminhos em simultâneo para o envio de dados. Os restantes caminhos disponíveis são utilizados apenas em caso de falha do caminho a ser utilizado para o envio de dados — *primary path*. A grande desvantagem do SCTP, que leva a que não seja muito utilizado nas redes atuais, é a dificuldade de implementação na internet, dado que a maioria dos dispositivos existentes não reconhece este protocolo.

O *Concurrent Multipath Transfer* (CMT) é um mecanismo de transmissão *multipath* que foi proposto como uma extensão ao protocolo *Stream Control Transmission Protocol* (SCTP-CMT) [48] e utiliza a capacidade de *multi-home* do SCTP, adicionando a este protocolo a capacidade de transferência de dados simultânea através de diversos caminhos.

O *parallel TCP* (pTCP) [49] é um protocolo proposto em 2002 por Hung-Yun Hsieh e Sivakumar, R. que atua ao nível da camada de transporte e pretende dar suporte à utilização de diversos caminhos sobre uma ligação comum, quando existem diversos caminhos entre o remetente e o destinatário da ligação. Foi pensado principalmente para dispositivos móveis que possuem múltiplas interfaces de ligação à rede. Define-se *Micro-Flow* como uma porção de cada ligação que atravessa um único caminho. Posto isto, uma ligação pode possuir múltiplos *Micro-Flows*, um por cada caminho disponível. Assume-se que cada um destes irá ter o mesmo comportamento que uma ligação TCP comum. O protocolo proposto introduz diversos mecanismos para possibilitar a agregação da largura de banda de cada caminho, reconhecido não ser uma tarefa trivial, dado que cada caminho pode apresentar características muito distintas entre si, seja em termos de largura de banda, latência ou *jitter*. Pense-se na existência de dois caminhos com larguras de banda muito distintas, se os dados da ligação forem distribuídos simetricamente entre ambos, a largura de banda alcançável não será a melhor possível.

O protocolo MPTCP, normalizado no RFC 8684 [50], tem sindo desenvolvido há vários anos e é o protocolo *multipath* que tem despertado maior interesse, parecendo o mais

promissor. Foi proposto como sendo uma extensão ao protocolo TCP, capaz de utilizar diversos caminhos entre dois *endpoints* — *subflows*, agregando-os numa única ligação – ligação MPTCP. Foi desenhado desde o início para ser de fácil utilização e adoção, sendo desde logo consideradas as seguintes premissas:

- Funcionar com as *middleboxes*, dispositivos *Network Address Translation* (NAT) e *firewalls* existentes;
- Funcionar sem que as aplicações precisem de ser alteradas;
- Capacidade de passar para TCP quando necessário, por exemplo, não ocorrência de um erro, sem intervenção humana.

Para aplicações que antes já usavam TCP, a utilização de MPTCP é completamente transparente. As aplicações não precisam de ter conhecimento que utilizam MPTCP. Para que se consiga tirar vantagem das características que o protocolo veio oferecer, assume-se que pelo menos um dos dispositivos interveniente numa ligação MPTCP é *multi-homed* e *multi-addressed* — possui diversos endereços. Para tornar possível o estabelecimento de uma ligação MPTCP entre dois dispositivos, ambos têm de suportar o protocolo. Caso isso não aconteça, a ligação estabelecida acaba por ser uma ligação TCP comum. O mesmo acontece quando ambos suportam MPTCP, mas não a mesma versão do protocolo.

Apresentando semelhanças com o TCP, o início do estabelecimento de uma ligação MPTCP é idêntico ao início do estabelecimento de uma ligação TCP, como se pode observar na Figura 2.4. De início apenas é estabelecido um *subflow* na ligação MPTCP. Posteriormente, caso um dos intervenientes seja *multi-homed*, passa a ser possível o estabelecimento de novos *subflows*.

Para que um dispositivo *multi-homed* consiga utilizar outra interface de comunicação, há que estabelecer um novo *subflow*, iniciando-se um novo *handshake*. Qualquer um dos intervenientes pode solicitar o estabelecimento de um novo *subflow*, mas este comportamento está normalmente associado ao cliente. Na Figura 2.5 pode-se observar uma ligação MPTCP composta por dois *subflows*. Tendo em consideração que ao longo de uma ligação o conjunto de endereços associados a um dispositivo pode sofrer alterações, é suportada a adição e remoção de endereços nos dispositivos que participam na mesma.

Uma ligação MPTCP sendo constituída por diversos *subflows*, suporta que se decida qual o comportamento que deve estar associado a cada um, recaindo a decisão sobre dois tipos: i) comum —- participa ativamente na ligação; e, ii) *backup* —- utilizado apenas quando os caminhos comuns não estão disponíveis.

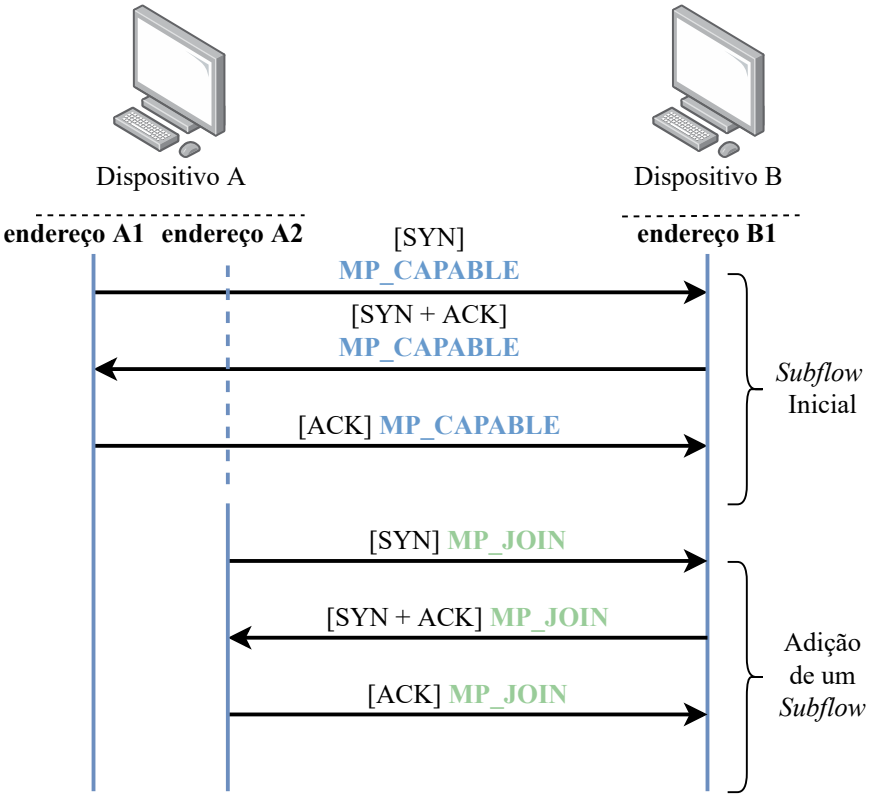


Figura 2.4: Estabelecimento do *subflow* inicial e um segundo *subflow* da ligação MPTCP.

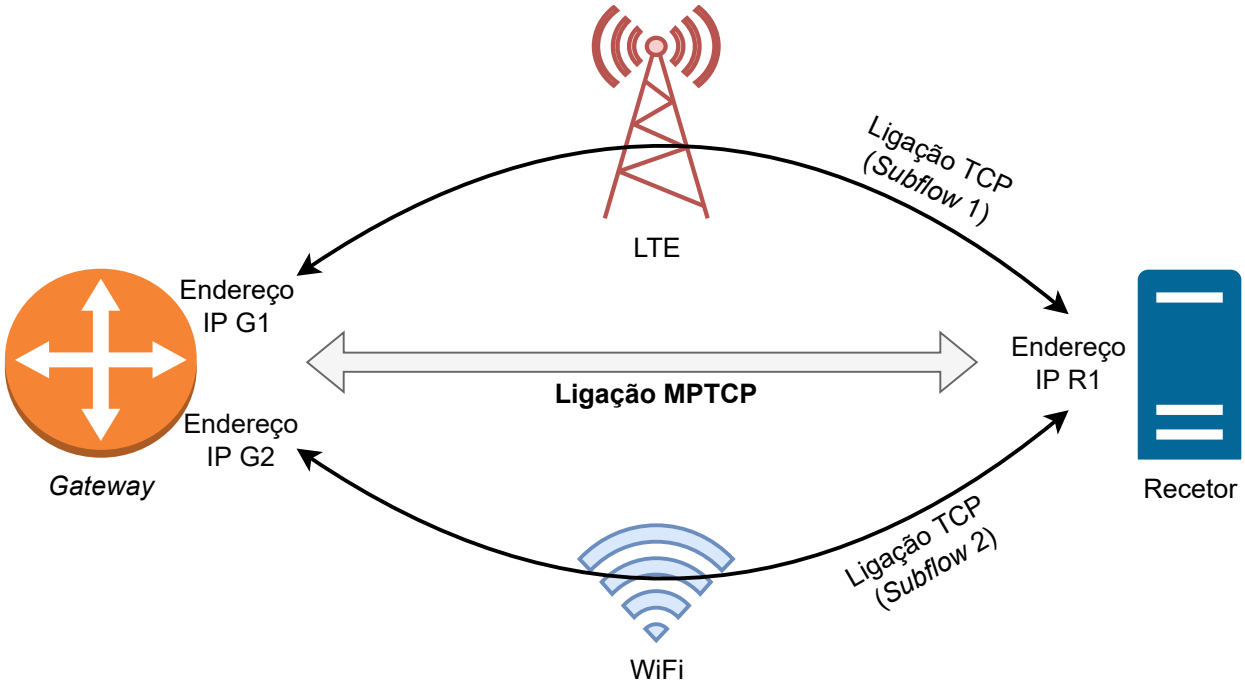


Figura 2.5: Ligação MPTCP com dois *subflows*

3

Trabalho Realizado

Neste capítulo são descritas várias etapas da solução proposta para a construção de um sistema de encaminhamento resiliente para serviços críticos sobre redes públicas móveis, tendo em conta o que foi abordado no estado da arte.

3.1 MPTCP

O que se pretende avaliar com a realização dos testes é a resposta do MPTCP a diversas alterações que podem ocorrer na topologia de rede ao longo do tempo, nomeadamente, diversos valores de largura de banda bem como de latência. Assim sendo, foram realizados os seguintes testes:

- Limitação simétrica de largura de banda;
- Limitação assimétrica de largura de banda;
- Aplicação simétrica de latência;
- Aplicação assimétrica de latência.

Como topologia foram utilizadas três máquinas virtuais: uma atua como cliente, outra como *router* e outra como servidor. A máquina cliente possui duas interfaces que se conectam com o *router*. Entre o *router* e o servidor existe apenas um caminho. Todos

os caminhos da topologia apresentam uma largura de banda de 1 Gbps ao nível do *hardware*. A topologia de teste encontra-se na Figura 3.1.

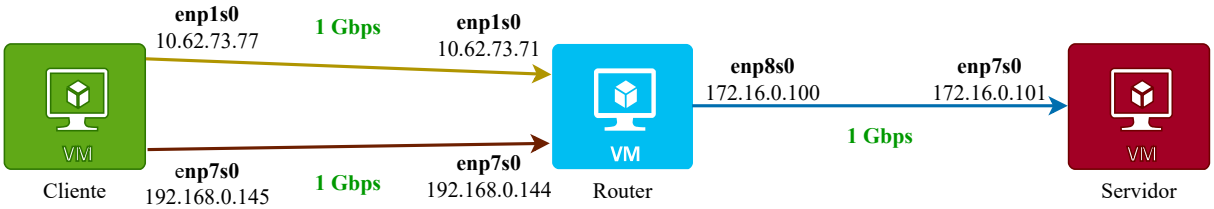


Figura 3.1: Topologia de Testes MPTCP

A interface enp1s0 é denominada de interface primária, já que é a interface usada para iniciar o primeiro *subflow*, e a interface enp7s0 é a interface secundária. Todas as máquinas correm o sistema operativo Ubuntu. A utilização de MPTCP no Kernel oficial está disponível desde a versão 5.6 e a versão utilizada nos testes foi a 5.11.

A maioria dos artigos onde são descritos testes realizados com MPTCP, tais como em [51], onde é avaliado o desempenho dos diferentes algoritmos de congestionamento disponíveis, ou em [52], onde é avaliada a utilização do MPTCP para acesso à internet em dispositivos móveis, é utilizado um Kernel desenvolvido especificamente para suportar o protocolo [53]. Não é o caso destes testes, onde se utilizou o Kernel oficial integrado na distribuição Ubuntu Server 21.04. Esta implementação, que ainda se encontra em desenvolvimento, não tem as mesmas características que a versão do Kernel específico, não tendo, por exemplo, algoritmos de congestionamento desenvolvidos propositadamente para o MPTCP, que estão presentes na implementação específica.

Para testar alterações de largura de banda, bem como de latência, optou-se por limitar artificialmente estes valores para cada interface da máquina cliente, através da ferramenta *Traffic control* (`Tc`). Para a medição de largura de banda utilizou-se a ferramenta Iperf, mais precisamente, Iperf3, sendo retiradas amostras a cada segundo. Para que esta utilize *sockets* MPTCP é utilizado um *wrapper*, carregado com LD_PRELOAD, que através da função `setsockopt` força o seu uso. Para além disso, o código do Iperf3 foi ligeiramente alterado para remover a opção TCP_NODELAY, que criava conflito com a ligação MPTCP.

3.1.1 Kernel 5.11

Os testes realizados dividem-se em dois grupos: testes de assimetria variando a largura de banda e a latência. De seguida serão apresentados os testes e os resultados obtidos, bem como a sua análise e comparação.

3.1.1.1 Limitação Simétrica de Largura de Banda

Para este cenário, foram realizados testes com limitação de 250 Mbps e 500 Mbps aplicados a ambas as interfaces que ligam o cliente ao *router*.

Na Figura 3.2 é apresentada a variação do ritmo binário em função do tempo, quando ambas as interfaces possuem uma largura de banda de 250 Mbps. Inicialmente nota-se a contribuição da largura de banda disponível de ambas as interfaces, ficando o ritmo binário agregado na ordem dos 350/400 Mbps. Depois de alguns segundos, este começa a descer, estabilizando nos 260 Mbps, ainda assim, ficando acima da largura de banda individual de cada interface (caminho).

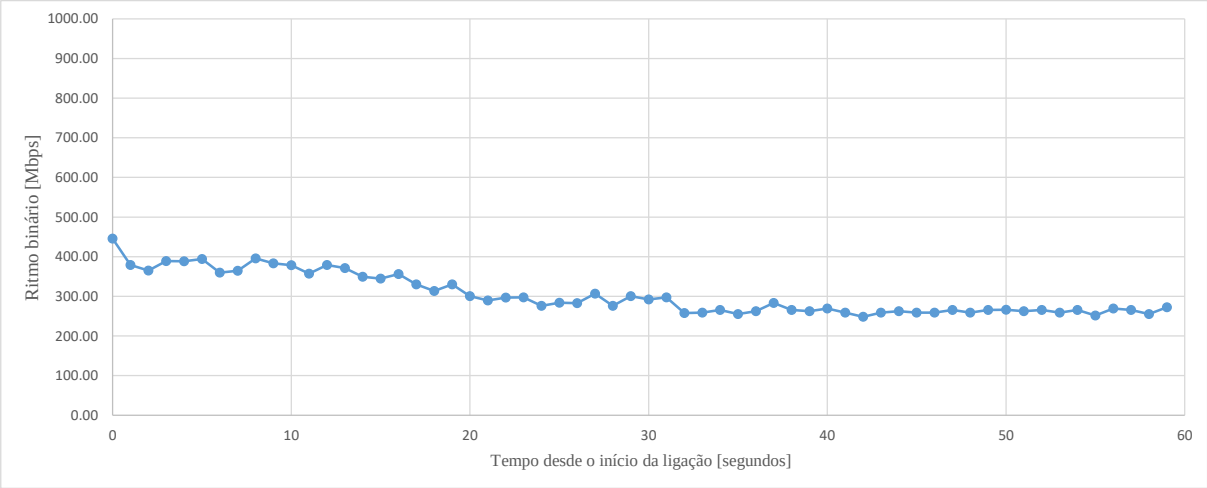


Figura 3.2: Limitação simétrica de 250 Mbps em cada interface.

Na Figura 3.3 é apresentada a variação do ritmo binário em função do tempo quando ambas as interfaces possuem uma largura de banda de 500 Mbps. O ritmo binário da ligação MPTCP manteve-se estável nos 550 Mbps, sendo superior à largura de banda de cada interface, ficando, no entanto, bastante distante da agregação da largura de banda individual de cada interface.

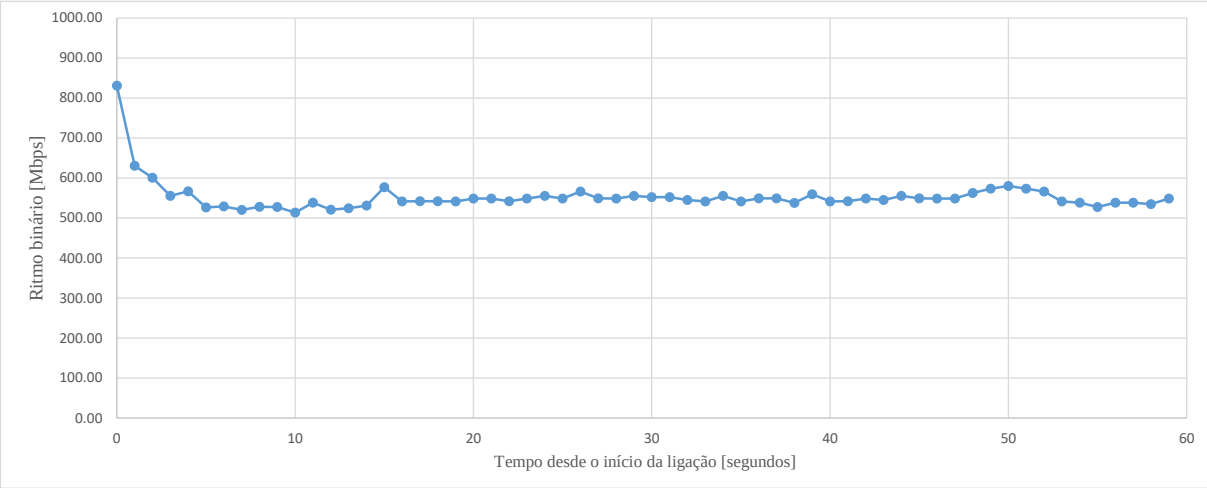


Figura 3.3: Limitação simétrica de 500 Mbps em cada interface.

3.1.1.2 Limitação Assimétrica de Largura de Banda

Estes testes pretendem avaliar qual o comportamento do MPTCP na presença de caminhos com larguras de banda distintas.

A Figura 3.4 apresenta a variação do ritmo binário em função do tempo, a azul, para um cenário no qual a interface primária possui uma largura de banda de 250 Mbps e a interface secundária não foi limitada e, a laranja, para um cenário no qual a interface primária não se encontra limitada e a interface secundária tem 250 Mbps de largura de banda. No primeiro cenário, verifica-se que o ritmo binário alcançada está na ordem dos 440 Mbps, enquanto no segundo o ritmo binário alcançado é inferior. Em qualquer um dos cenários, nota-se a contribuição do ritmo binário alcançável através de cada interface.

A Figura 3.5 apresenta a variação do ritmo binário em função do tempo, a azul, para um cenário no qual a interface primária possui uma largura de banda de 500 Mbps e a interface secundária não foi limitada e, a laranja, para um cenário no qual a interface primária não se encontra limitada e a interface secundária tem uma largura de banda de 500 Mbps. Verifica-se que o ritmo binário resultante da ligação usufruiu dos dois caminhos disponíveis. Ainda assim, fica longe da agregação da largura de banda de ambas as interfaces. O ritmo binário resultante do segundo cenário é superior, dando a entender que os resultados obtidos dependem não só dos limites aplicados, mas também de qual a interface mais limitada.

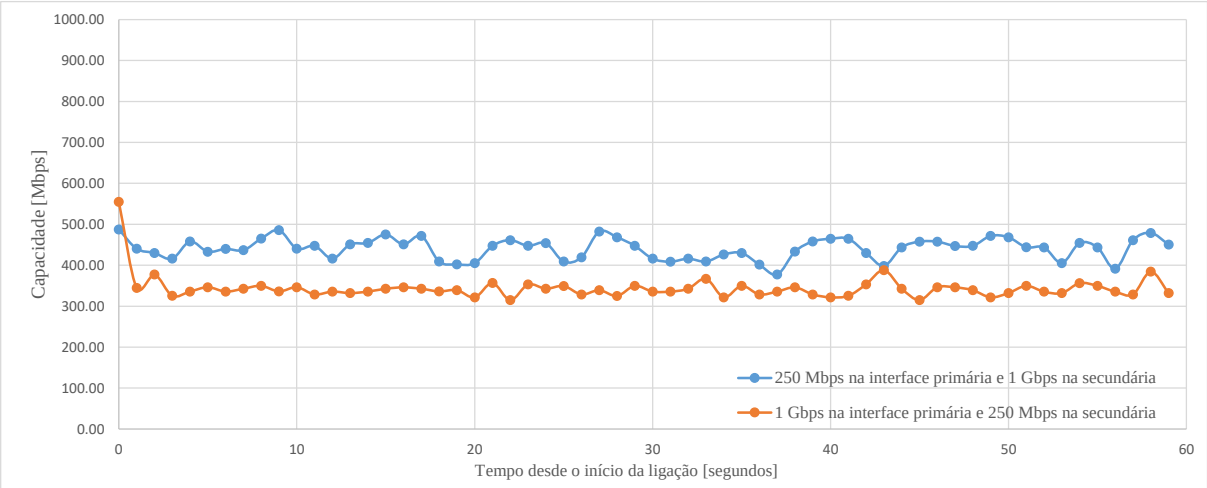


Figura 3.4: Limitação assimétrica de 250 Mbps.

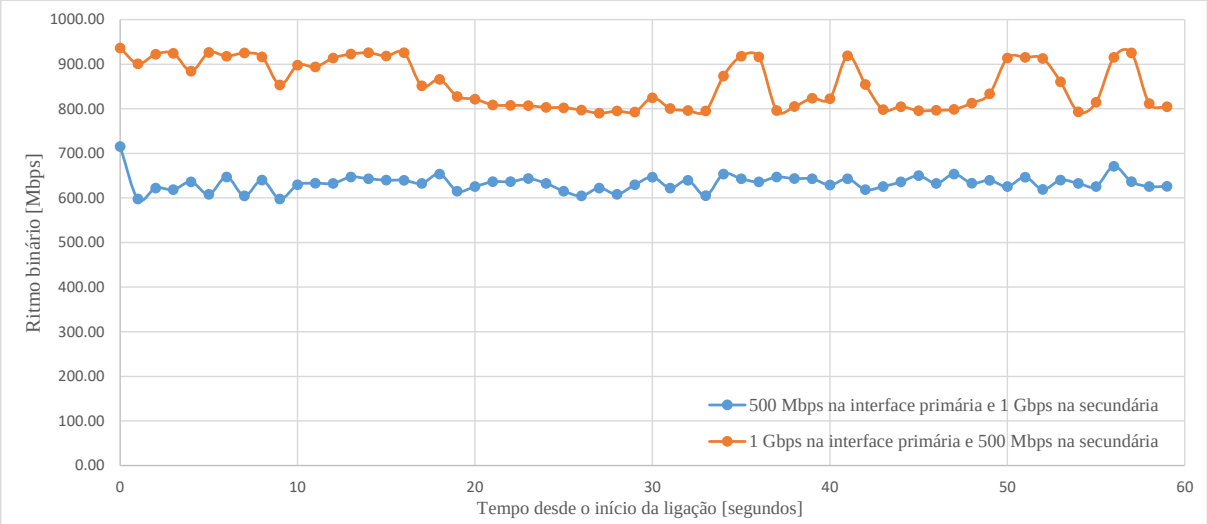


Figura 3.5: Limitação assimétrica de 500 Mbps.

Os resultados obtidos demonstram que o ritmo binário resultante da ligação MPTCP é fortemente limitado pelo caminho que apresenta o valor mais baixo de largura de banda. O algoritmo de congestionamento utilizado não é um algoritmo específico para MPTCP, sendo utilizado o algoritmo *Cubic*. O Kernel utilizado, apesar de possuir já uma implementação de MPTCP, não inclui um algoritmo específico para o mesmo. Assim sendo, os resultados obtidos quando se têm múltiplas ligações não são ideais, que seria a agregação do ritmo binário alcançável em cada caminho ou, num cenário em que o TCP conseguisse utilizar a totalidade da largura de banda disponível, a agregação da largura de banda de cada interface. Estes resultados foram também observados pelos autores em [51].

Quando existem caminhos com larguras de banda distintas, sem um algoritmo adequado, os pacotes não chegam pela mesma ordem pela qual foram enviados, dado que os pacotes enviados pelo caminho que apresenta uma menor largura de banda, demoram mais a chegar ao destino do que os pacotes enviados pelo caminho com maior largura de banda disponível. Como tal não pode acontecer, e não tendo o algoritmo capacidade de adequar a quantidade de pacotes enviados por cada caminho para que estes sejam entregues pela ordem correta, dá-se um alinhamento por baixo do ritmo binário total da ligação. O algoritmo vai-se adaptar com base no caminho mais limitado. Para equilibrar o envio dos pacotes, o algoritmo envia pacotes pelo caminho mais capacitado a uma velocidade similar à do caminho mais limitado. No caso em que existe uma interface com 100 Mbps de largura de banda e outra de 1 Gbps, esta última passa a ser usada, na realidade, apenas como se tivesse 100 Mbps ou perto disso. Assim, os pacotes passam a ser enviados ao mesmo ritmo por ambas as interfaces.

Algo que também é notável nos resultados demonstrados é uma diferença do ritmo binário obtido quando é limitado o caminho associado à interface primária e o caminho associado à interface secundária, nas mesmas condições. Nota-se que quando o caminho primário possui uma determinada largura de banda associada, o ritmo binário obtido na ligação MPTCP é mais baixo se comparado com o cenário em que é o caminho associado à interface secundária a ter a mesma largura de banda que tinha sido aplicada à interface primária.

3.1.1.3 Limitação Simétrica de Latência

Nestes testes pretende-se avaliar o impacto que a latência tem na ligação MPTCP, sendo aplicadas latências iguais — 10 e 100 ms nas duas interfaces.

A Figura 3.6 apresenta o ritmo binário medido quando é aplicada uma latência de 10 ms em cada interface. O ritmo binário máximo atingido foi superior a 900 Mbps, ficando próximo da largura de banda de cada interface. Nota-se, no entanto, que o ritmo binário oscila de forma considerável ao longo do tempo.

A Figura 3.7 mostra o ritmo binário medido quando é aplicada uma latência de 100 ms em cada interface. Contrariamente ao teste anterior, o ritmo binário apresenta-se estável ao longo de toda a ligação, situando-se na ordem dos 200 Mbps.

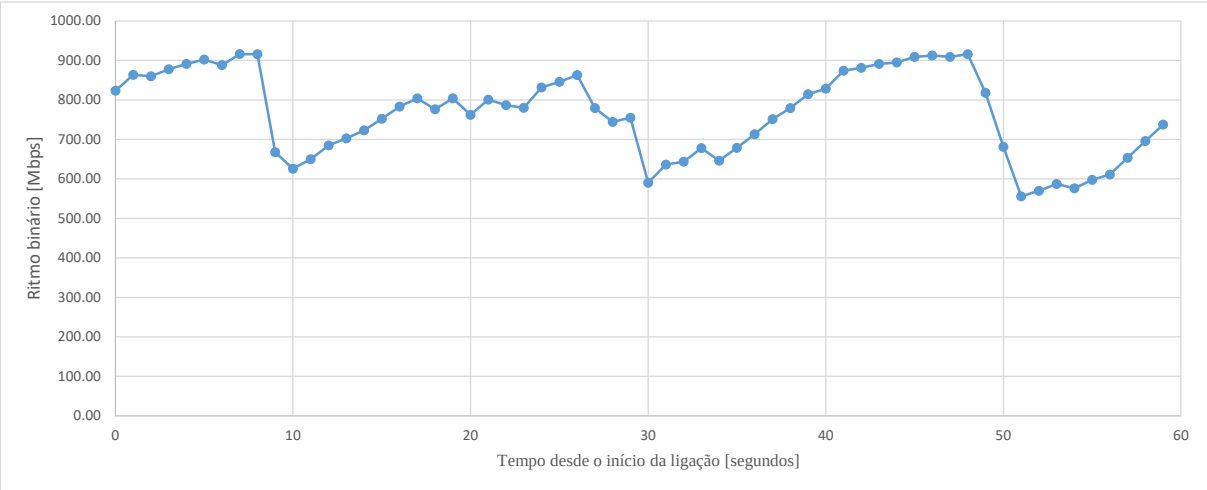


Figura 3.6: Limitação simétrica de 10 ms em cada interface.

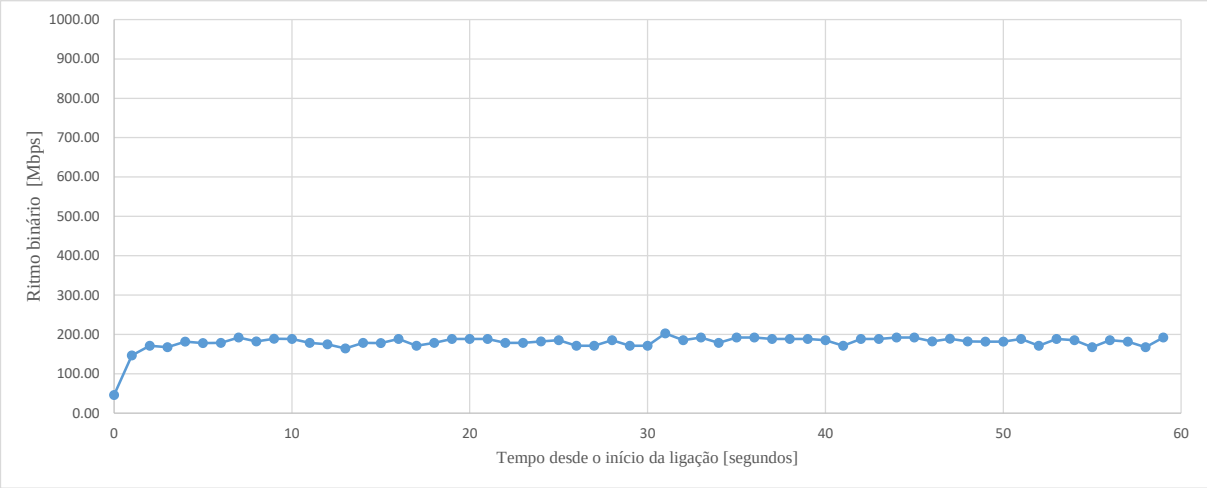


Figura 3.7: Limitação simétrica de 100 ms em cada interface.

3.1.1.4 Limitação Assimétrica de Latência

Nestes testes pretende-se avaliar o impacto que a latência tem na ligação MPTCP, sendo aplicadas latências de forma assimétrica nas duas interfaces. Os valores aplicados foram de 10 e 100 ms.

A Figura 3.8 apresenta a variação do ritmo binário em função do tempo, a azul, para um cenário no qual a interface primária possui uma latência de 10 ms e a secundária uma latência de 100 ms e, a laranja, para um cenário no qual a interface primária possui uma latência de 100 ms e a secundária 10ms. Para o primeiro cenário, existe

uma grande oscilação do ritmo binário ao longo da ligação, situando-se este, maioritariamente, entre os 100 e os 240 Mbps, com uma média de 170 Mbps sensivelmente. O resultado para o segundo cenário não apresenta grande oscilação de ritmo binário, sendo estável ao longo da ligação, ficando na ordem dos 170 Mbps.

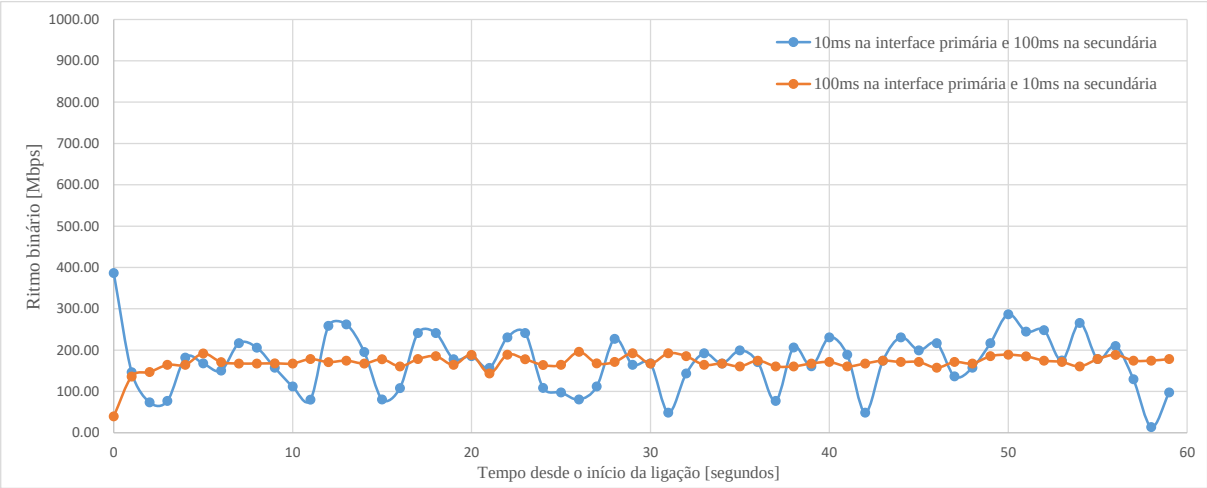


Figura 3.8: Limitação assimétrica de 10 ms e 100 ms.

Comparando os dois testes de aplicação de latência assimétrica, verifica-se que existe uma diferença de comportamento relativamente à aplicação da latência mais elevada na interface primária ou na secundária. Quando a latência mais elevada está associada à interface primária o ritmo binário apresenta-se menos oscilante ao longo da ligação. Apesar desta diferença, em termos de valor médio de ritmo binário, a diferença é mínima, ficando ambos na ordem dos 175 Mbps, no caso dos testes apresentados.

Se compararmos os resultados dos testes assimétricos de latência com os testes simétricos, verifica-se que o resultado obtido aquando da existência de uma latência de 100 ms em cada interface e o resultado obtido quando existe uma latência de 100 ms na interface primária e 10ms na secundária é bastante similar. O primeiro cenário referido leva uma ligeira vantagem, dado que o ritmo binário medido situa-se, em média, na ordem dos 180 Mbps, enquanto para o segundo teste mencionado esta é mais próxima dos 170 Mbps, como se pode verificar na Figura 3.8. Tendo em conta, novamente, o resultado obtido quando existe uma latência de 100 ms em cada interface e comparando com o resultado obtido quando existe uma latência de 10 ms na primária e 100 ms na secundária, em termos de ritmo binário, são também idênticos. A diferença entre estes reside na ocorrência de picos, recorrentes no último teste referido.

Relativamente aos testes de latência realizados, conclui-se que, na existência de caminhos com valores de latência distintos, o ritmo binário da ligação é fortemente limitado

pelo caminho com maior latência. A ligação MPTCP não consegue tirar vantagem do caminho que apresenta menor latência, não dando prioridade a este para o encaminhamento de pacotes. Nota-se ainda uma diferença de comportamento quando a latência mais elevada está associada ao caminho por onde é iniciada a ligação (o *subflow* principal). Quando isso acontece, a ligação MPTCP não apresenta picos de ritmo binário, este mantém-se estável ao longo da ligação. Os resultados obtidos para estes testes corroboram o que já foi mencionado para os testes de largura de banda, a não utilização de um algoritmo adequado para o MPTCP faz com que a existência de múltiplos caminhos não seja aproveitada por parte do MPTCP. Mesmo quando se nota a contribuição dos diversos caminhos, nem sempre existe consistência no comportamento apresentado.

3.1.1.5 Conclusões

Com os testes realizados, verificou-se que na maioria dos cenários, a vantagem de utilização de MPTCP é praticamente inexistente. Em situações nas quais estejam disponíveis caminhos com largura de banda disponíveis equivalentes, nota-se que ambos são utilizados, sendo o ritmo binário resultante superior ao ritmo binário capaz de ser atingido individualmente em cada caminho. Ainda assim, este fica longe da soma do ritmo binário alcançável em cada interface ou, considerando um cenário ideal em que é possível utilizar a totalidade da largura de banda existente, da largura de banda de cada um. Em situações nas quais uma ligação MPTCP tenha ao seu dispor diversos caminhos com valores de largura de banda bastante distintos ou latências bastante distintas, o protocolo não consegue tirar proveito do melhor caminho, não dando prioridade a este. Nestas situações, o que ocorre é que o ritmo binário alcançado é inferior ao ritmo binário do caminho de maior largura de banda ou menor latência, fazendo com que o uso de uma ligação TCP usual, utilizando esse mesmo caminho, seja mais vantajosa. Para além do ritmo binário ficar aquém do esperado, algo que se pode observar nesta implementação do MPTCP é que os resultados diferem em função de qual o caminho mais limitado em termos de largura de banda ou latência, não dependendo apenas dos valores de largura de banda e latência dos mesmos.

Considerando os resultados obtidos, não se considera viável a utilização de MPTCP num contexto de aumentar o ritmo binário de uma ligação. Ainda assim, a utilização do MPTCP é relevante se o objetivo for aumentar a resiliência de uma ligação, dado que a ligação só é quebrada caso todos os caminhos disponíveis fiquem indisponíveis.

3.1.2 Kernel 5.13

Após testes realizados sobre a versão 5.11 do Kernel e dos resultados aquém do esperado, resolveu-se testar uma versão mais recente, o Kernel 5.13, na esperança de terem sido realizadas alterações que melhorem o desempenho do MPTCP. Os testes realizados foram os mesmos que foram executados para o Kernel 5.11, para ser realizada uma comparação fidedigna.

3.1.2.1 Limitação Simétrica de Largura de Banda

Com uma limitação de 250 Mbps aplicada a ambas as interfaces — Figura 3.9, o ritmo binário total alcançado na ligação revela a vantagem de utilização de MPTCP face a uma ligação TCP, ficando este na ordem dos 450 Mbps, perto da soma da largura de banda de cada interface. Com uma limitação de 500 Mbps — Figura 3.10 o resultado revela igualmente vantagens na utilização de MPTCP.

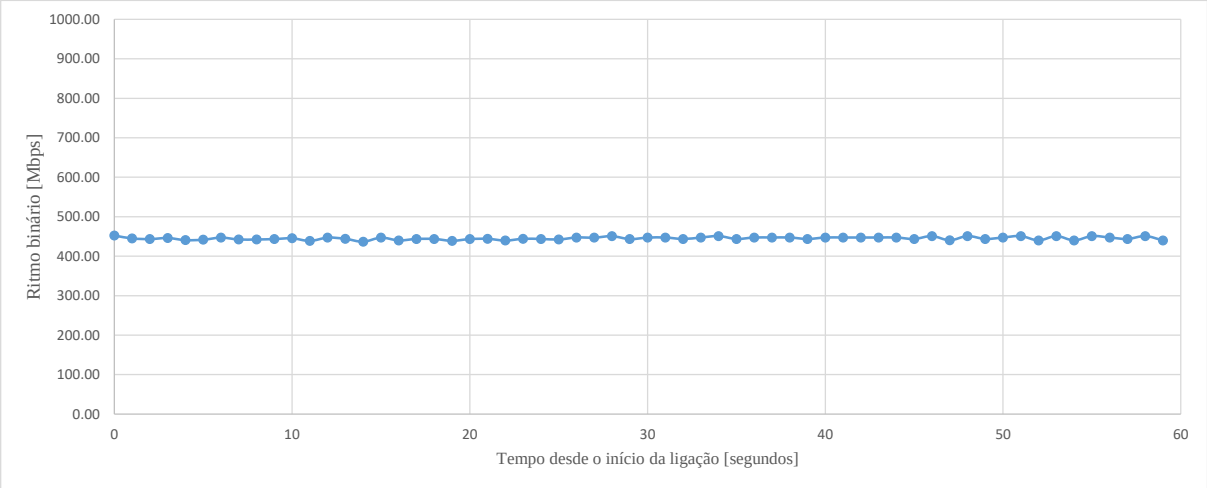


Figura 3.9: Limitação simétrica de 250 Mbps em cada interface.

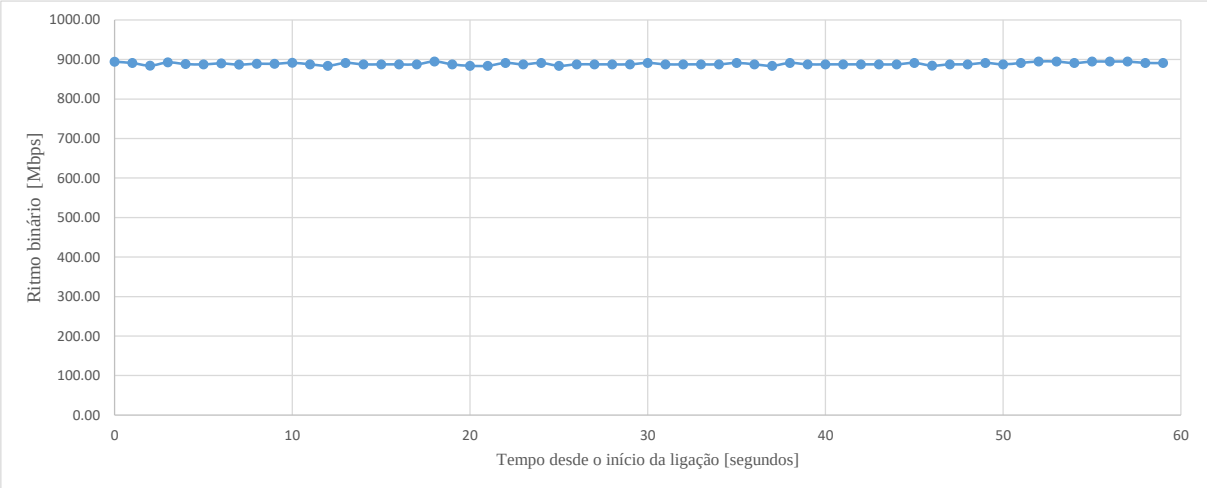


Figura 3.10: Limitação simétrica de 500 Mbps em cada interface.

3.1.2.2 Limitação Assimétrica de Largura de Banda

Quando aplicados 250 Mbps às interfaces — Figura 3.11, seja à primária ou à secundária, o ritmo binário alcançado é bastante superior aos testes equivalentes realizados no Kernel 5.11. De forma igual aos testes realizados na versão 5.11, quando os 250 Mbps são aplicados à interface secundária, o desempenho é inferior ao cenário no qual esse valor é aplicado à interface primária.

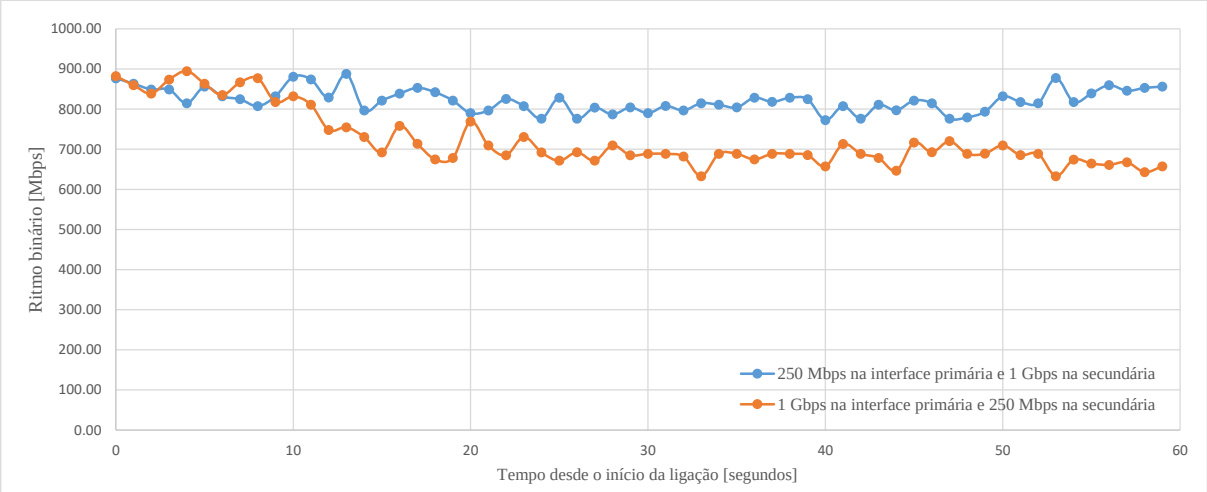


Figura 3.11: Limitação assimétrica de 250 Mbps.

Quando aplicados 500 Mbps — Figura 3.12 seja na primária ou na secundária, o resultado obtido é muito similar, na ordem dos 920 Mbps e muito estável ao longo do tempo.

Comparando com os resultados do Kernel 5.11, o resultado para os 500 Mbps na interface primária na versão 5.13 é muito superior. Já quando temos os 500 Mbps aplicados à interface secundária o ritmo binário é idêntico, apesar de neste último Kernel se apresentar mais estável.

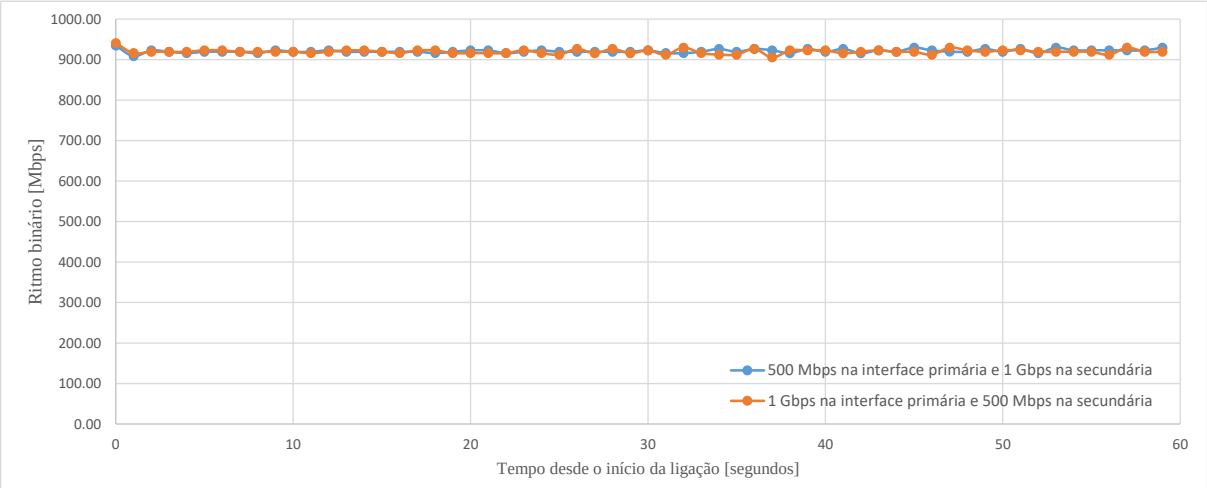


Figura 3.12: Limitação assimétrica de 500 Mbps.

3.1.2.3 Limitação Simétrica de Latência

Tanto para uma limitação de 10 ms — Figura 3.13 como para os 100 ms — Figura 3.14, o ritmo binário manteve-se estável o tempo todo e em termos de ritmo binário total os valores são idênticos entre versões. Comparando com os resultados do Kernel 5.11, o teste de 10 ms provou-se muito mais estável nesta versão, enquanto para os 100 ms é idêntico.

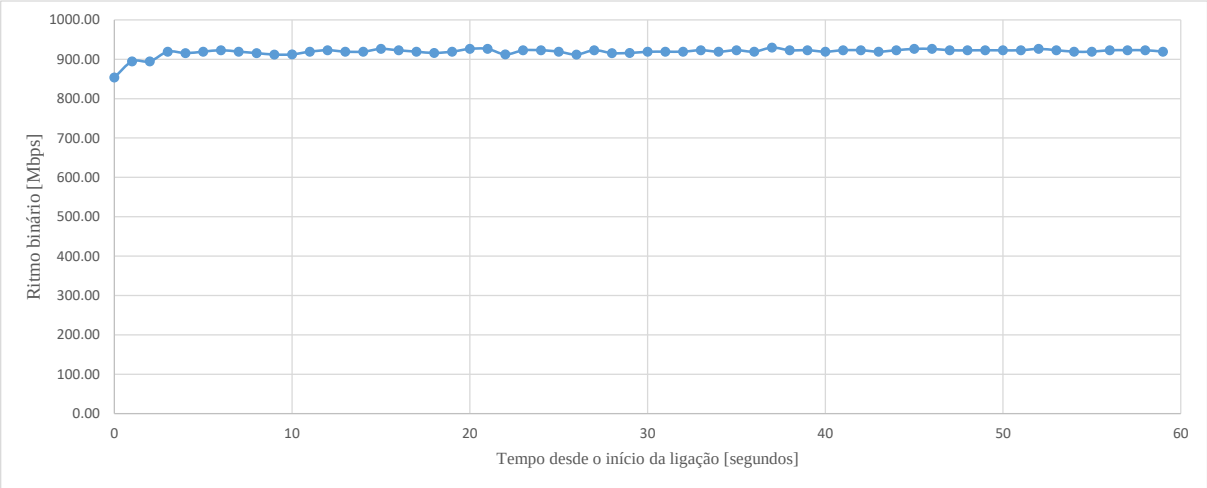


Figura 3.13: Limitação simétrica de 10 ms em cada interface.

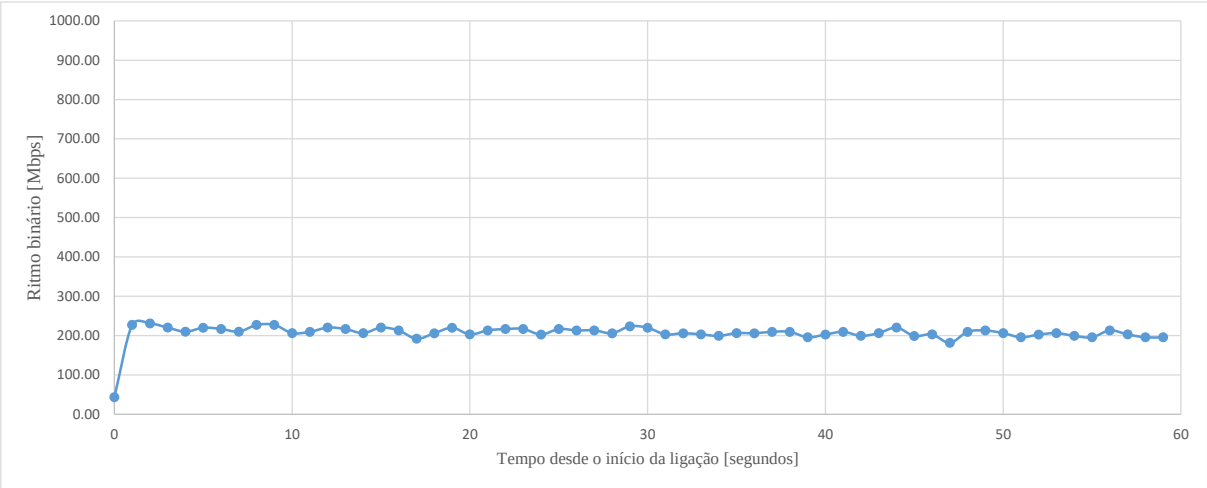


Figura 3.14: Limitação simétrica de 100 ms em cada interface.

3.1.2.4 Limitação Assimétrica de Latência

O ritmo binário ao longo de toda a ligação apresenta enorme estabilidade para este teste — Figura 3.15. O cenário de 10 ms na interface primária, nesta versão, é muito mais estável que o mesmo teste na versão 5.11, apesar de o ritmo binário ser inferior.

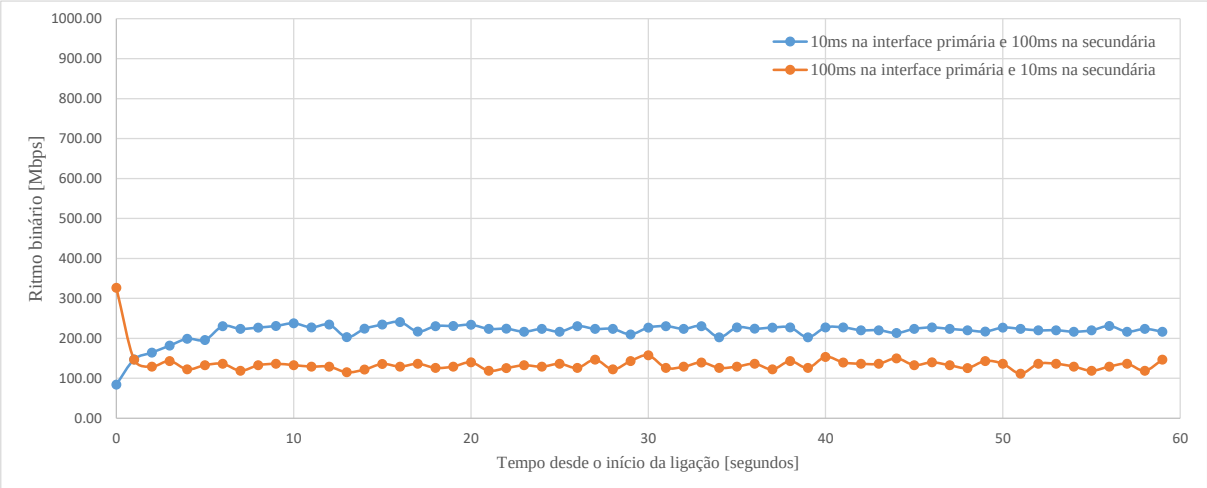


Figura 3.15: Limitação assimétrica de 10 ms e 100 ms.

3.1.2.5 Conclusões

Realizando uma comparação geral entre os resultados obtidos para a implementação do MPTCP no Kernel 5.11 e os resultados para o Kernel 5.13, é notória a diferença comportamental do protocolo, sendo este mais favorável e compreensível no Kernel 5.13.

Enquanto na versão 5.11, na maioria dos cenários de testes, a vantagem da utilização de MPTCP face ao TCP era praticamente nula ou mesmo inexistente, na versão 5.13 é notório o seu impacto positivo, apresentando um comportamento muito previsível conforme as vantagens teóricas do protocolo. Conclui-se que existiram evoluções na implementação do MPTCP, tornando viável a utilização do protocolo, o que não era possível afirmar para o Kernel 5.11. Pelo que se pode aferir, existia um *bug* na implementação que fazia com que os pacotes não fossem corretamente distribuídos pelos diversos caminhos disponíveis, levando a que o desempenho do MPTCP fosse fraco, algo que foi resolvido na versão 5.13.

3.2 VPN

Um dos objetivos do projeto é garantir a segurança dos dados trocados nas comunicações realizadas e, tal pode ser alcançado com a utilização de uma VPN. Tendo em conta que se pensa a utilização de MPTCP, pretende-se verificar se é possível conjugar as VPN com este protocolo. Para tal, foram realizados testes com o intuito de aferir que arquiteturas podem ser adotadas. Seguidamente, foram realizados testes para aferir o desempenho de cada VPN.

3.2.1 Arquiteturas

Para testar as arquiteturas, utilizou-se uma topologia simplificada com três máquinas virtuais a correr o sistema operativo Ubuntu. Têm-se uma máquina cliente, com duas ligações à máquina que faz de *router* e do *router* para o servidor existe apenas uma ligação — Figura 3.16. Todas as ligações possuem uma largura de banda de 1 Gbps, ao nível do *hardware*.

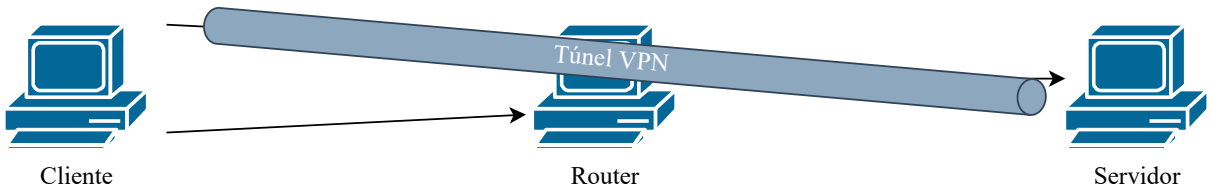


Figura 3.16: Topologia geral de teste das VPN.

Antes de se testarem as VPN, testou-se qual o ritmo binário alcançado sem qualquer VPN, na mesma topologia, para se verificar o impacto da VPN e do MPTCP. Em média, o ritmo binário alcançado foi de 950 Mbps.

A OpenVPN foi a primeira VPN a ser testada. Com esta, é possível usar UDP ou TCP para o transporte dos dados. Sendo que se pretende utilizar MPTCP, realizaram-se testes para verificar se tal era possível, e de facto, é possível fazê-lo. Para isso basta configurar a OpenVPN para utilizar TCP e assegurarmos que o MPTCP está ativo no Kernel, para além de ser necessária a utilização de um *wrapper* para forçar as aplicações, como a própria VPN, a utilizarem uma *socket* MPTCP. Este *wrapper* é necessário dado que na implementação do Kernel Oficial o MPTCP não é forçado automaticamente.

O ritmo binário obtido com UDP não foi testado, dado que a possibilidade de se utilizar a OpenVPN prende-se precisamente com o facto desta usar TCP e isso permitir que seja utilizado MPTCP. Recorrendo à topologia de teste, e utilizando apenas uma das ligações entre a máquina cliente e o *router*, testou-se o desempenho da VPN utilizando TCP.

Apesar de ser sabido que a introdução de uma VPN leva a atrasos e a uma redução do ritmo binário obtido, não é possível utilizar determinada VPN para suporte a serviços críticos se esta apresentar uma degradação elevada da largura de banda disponível. Foi exatamente isso que se notou ao testar a OpenVPN. Com esta, e considerando as configurações de cifra por omissão, a média de ritmo binário obtido foi de apenas 130 Mbps, muito abaixo dos 950 Mbps alcançados sem qualquer VPN. Para além do ritmo binário, testou-se se a OpenVPN era rápida a reagir a alterações na rede. Para isso, a interface física foi desligada por alguns segundos e após alguns segundos era ligada novamente, sendo este processo repetido diversas vezes. O que se notou foi que a OpenVPN demorava muito tempo a reestabelecer ligação, ficando diversos segundos em baixo mesmo quando a interface física já estava ligada.

Depois de testada a OpenVPN sobre TCP testou-se a mesma utilizando MPTCP no Kernel 5.13, sendo utilizadas as duas ligações entre o cliente e o *router* — Figura 3.17. Notou-se, no entanto, que o ritmo binário alcançado foi exatamente igual ao alcançado com TCP. O problema não foi identificado, mas pode ser tanto um problema na ferramenta Iperf3 que não contemplou a utilização dos dois caminhos e fez a medição do ritmo binário alcançado contemplando apenas um ou, do *wrapper*, utilizado. De qualquer forma, foi possível verificar que ambos os caminhos foram utilizados.

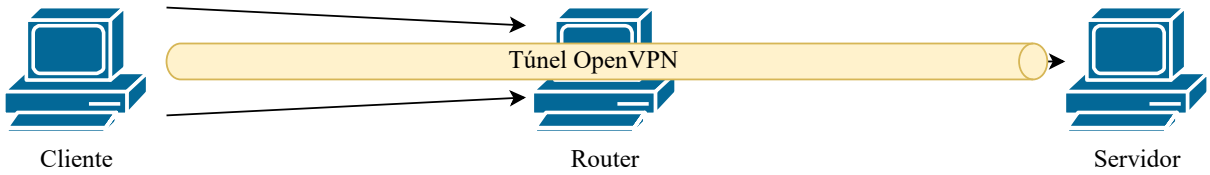


Figura 3.17: Topologia de teste de OpenVPN com MPTCP.

Na mesma topologia, foram realizados testes com a WireGuard. Esta suporta apenas UDP e o ritmo binário alcançado foi, em média, de 300 Mbps, muito superior ao obtido com a OpenVPN. Para além disso, notou-se que a WireGuard é bastante rápida a reagir a alterações na rede. Quando uma interface era desligada e novamente ligada, o reestabelecimento de ligação era quase imediato.

Querendo utilizar MPTCP, e visto que a WireGuard só suporta UDP, impossibilitando o uso de MPTCP como protocolo de transporte, há que repensar a arquitetura de rede a implementar com esta VPN. A arquitetura idealizada foi a utilização de duas instâncias de WireGuard, ou seja, uma por cada interface presente no cliente. A partir dos dois endereços virtuais que vão existir no router, estabelecidos pelas VPN, passa a ser possível forçar a utilização de MPTCP, que vai utilizar estes endereços para encaminhar os dados até ao servidor, ou seja, nesta arquitetura, o MPTCP vai correr sobre as múltiplas instâncias VPN. Esta topologia é apresentada na Figura 3.18.

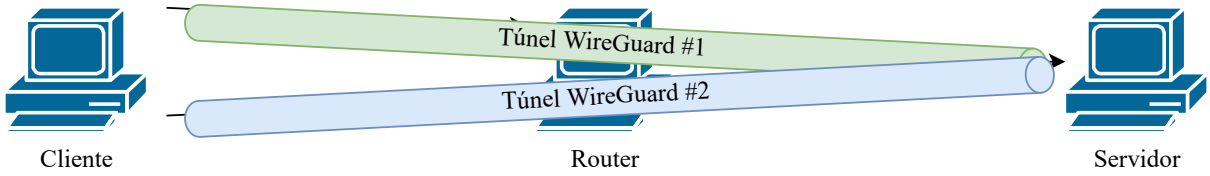


Figura 3.18: Topologia de teste de WireGuard com MPTCP.

A nova abordagem descrita foi testada. Porém, os resultados foram idênticos aos obtidos ao testar MPTCP com a OpenVPN, isto é, o ritmo binário obtido com duas instâncias de WireGuard foi o mesmo que foi alcançado com uma instância apenas. As razões pelas quais tal situação ocorreu serão certamente as mesmas que levaram a semelhante resultado com a OpenVPN.

Considerando as abordagens apresentadas, caso se opte pela que inclui a OpenVPN, há um problema a destacar. Utilizando esta VPN, iremos ter tráfego TCP transportado sobre TCP (ou MPTCP), algo que degrada o ritmo binário atingível.

3.2.2 Desempenho

Aquando dos testes de arquiteturas, foram realizados testes simples de ritmo binário para entender o impacto das VPN e MPTCP em conjunto. Pretende-se, portanto, realizar testes mais extensivos acerca do desempenho de cada VPN.

Concretamente, foi avaliado o ritmo binário alcançado, latência, utilização de CPU e perda de pacotes para as diversas cifras suportadas pela OpenVPN, WireGuard e

IPsec, esta última recorrendo à solução strongSwan. Através das métricas avaliadas consegue-se aferir a viabilidade de utilização de cada VPN no sistema que se pretende desenvolver.

Como topologia de rede foram utilizadas duas máquinas virtuais Ubuntu, instaladas em servidores físicos diferentes. A plataforma de virtualização utilizada foi a oVirt. Os dois servidores estão conectados a um *switch*, com ligações físicas de 1 Gbps. Estas ligações são utilizadas pelas máquinas virtuais para comunicarem. A topologia descrita é visível na Figura 3.19.

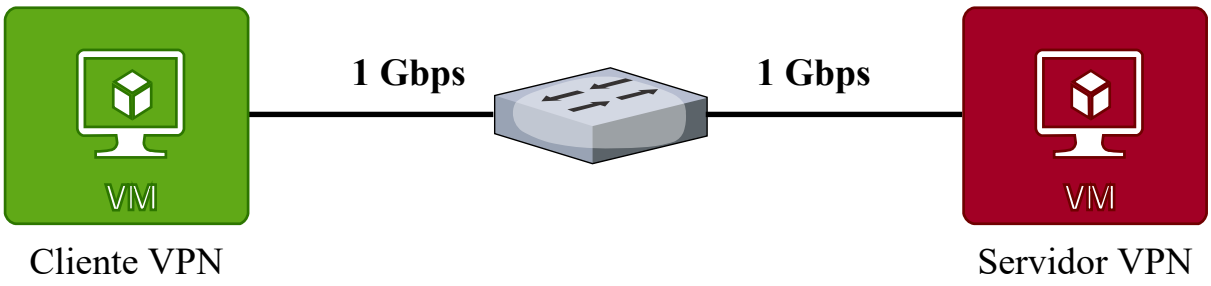


Figura 3.19: Topologia de rede utilizada nos testes.

Na tabela 3.1 são apresentadas as características de *hardware* e *software*, bem como a informação das versões das VPN utilizadas.

Tabela 3.1: Características de *hardware* e *software*

Sistema Operativo	Ubuntu Server 22.04 LTS
Kernel	5.15.0-39-generic
Processador	Intel Xeon E5-2609 v4 (1 core)
Memória	1 GB
Plataforma de Virtualização	oVirt
OpenVPN	2.5.6
strongSwan	5.9.5
WireGuard	1.0.20210914
OpenSSL	3.0.2

Os testes foram realizados tendo em conta apenas as cifras recomendadas para cada VPN [54] [55].

Em termos de ritmo binário, foram realizados 3 testes de 60 segundos para cada cifra de cada VPN, recorrendo ao Iperf3, sendo retiradas amostras a cada segundo. Além

disso, para cada VPN e cifra, foram realizados testes UDP e TCP. No caso dos testes UDP, em que o Iperf3 requer como critério a largura de banda disponível, foi utilizado um valor de 1000 Mbps, o valor disponibilizado pela interface de rede física utilizada.

Ao nível dos testes de latência, foi utilizada a ferramenta Ping, sendo enviados 50 000 pacotes ICMP a um ritmo de 1 000 por segundo.

A medição da utilização de CPU foi realizada em simultâneo com os testes de ritmo binário, a partir dos quais foram também retirados os valores da perda de pacotes. Estes valores foram retirados apenas para os testes UDP, dado que o Iperf3 não disponibiliza estes dados quando utilizados pacotes TCP.

Para ter uma base de comparação, foram realizados testes sem qualquer VPN, utilizando tráfego UDP e tráfego TCP — testes *Baseline*.

Os resultados apresentados foram agrupados pelo protocolo utilizado no envio dos pacotes. Em primeiro lugar serão abordados todos os resultados retirados dos testes Iperf3 e, seguidamente, são apresentados os resultados dos testes de latência.

Dado que a OpenVPN suporta a utilização de UDP e TCP como protocolo de transporte, para esta VPN, todas as cifras foram testadas a correr sobre estes dois protocolos.

3.2.2.1 Ritmo binário

Na Figura 3.20 são apresentados os valores de ritmo binário alcançados utilizando cada VPN, em Mbps, para tráfego UDP. Em primeiro lugar, é evidente que a OpenVPN, tanto a utilizar UDP como TCP como transporte para o tráfego da VPN, é a que apresenta os valores mais baixos de ritmo binário. Em segundo lugar, IPsec com todas as suas cifras, supera as restantes VPN, com uma margem considerável. Por fim, verifica-se também que as cifras AES-GCM e ChaCha20Poly1305 conseguem melhores resultados que as cifras AES-CBC.

Na Figura 3.21 são apresentados os valores de ritmo binário alcançados para cada VPN, em Mbps, para tráfego TCP. Tal como para UDP, a OpenVPN é a que apresenta os piores resultados. Neste caso, para as cifras AES-GCM, o ritmo binário obtido sobre UDP é superior. Para as restantes, o maior ritmo binário é alcançado sobre TCP. A WireGuard é a VPN com melhor desempenho com uma larga vantagem face a IPsec, mais 300 Mbps do que a sua melhor cifra — ChaCha20Poly1305.

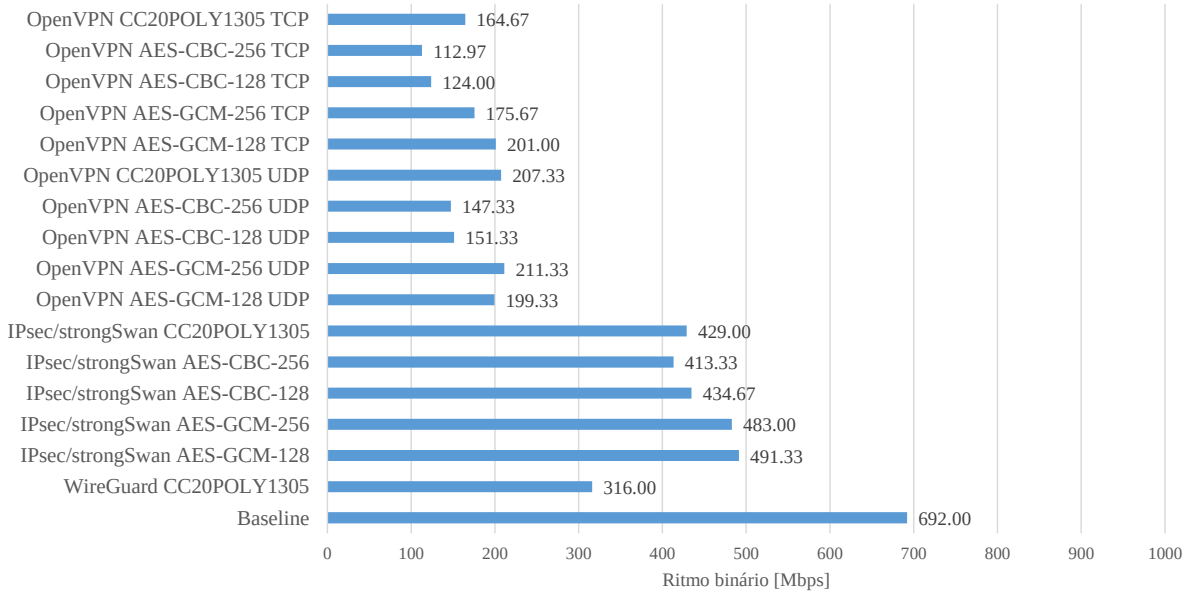


Figura 3.20: Ritmo binário das VPN para tráfego UDP.

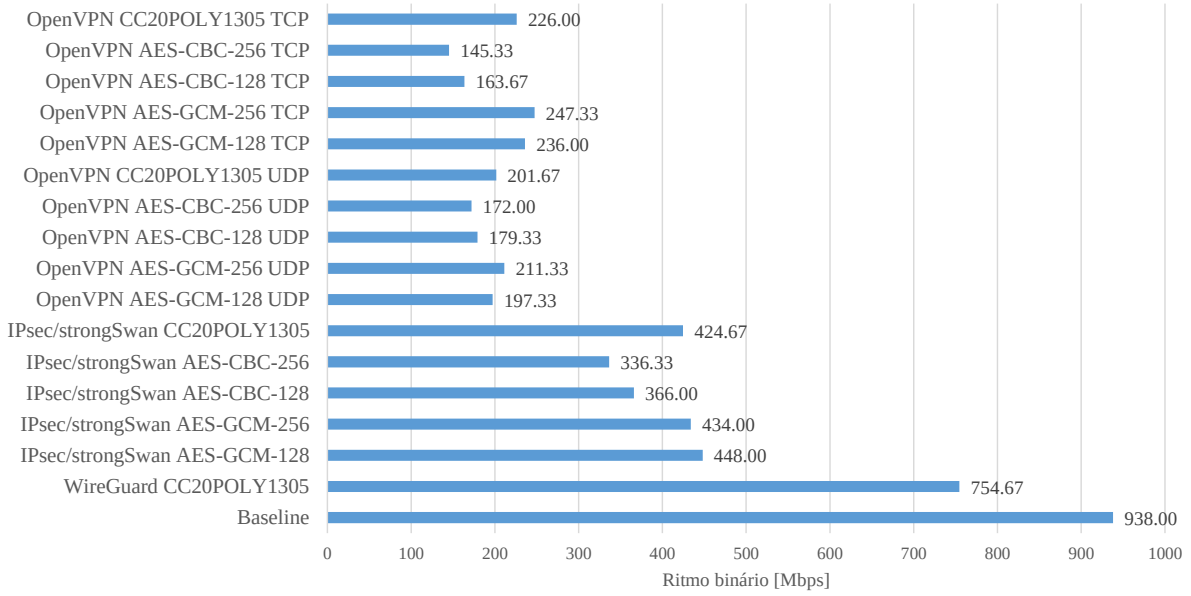


Figura 3.21: Ritmo binário das VPN para tráfego TCP.

Comparando os resultados do tráfego UDP e TCP, a OpenVPN foi a pior em ambos os cenários. Com tráfego TCP apresenta resultados superiores aos obtidos quando transporta pacotes UDP, à exceção da cifras ChaCha20Poly1305 e AES-GCM-128 sobre UDP. No caso de IPsec, para TCP, os resultados foram inferiores aos obtidos com UDP, sendo

as cifras AES-CBC as que apresentam uma maior diferença. Quando à WireGuard, esta é a melhor para tráfego TCP. Nos dois cenários apresentados, outro aspeto interessante é que foi notória a superioridade das cifras AES-GCM e ChaCha20Poly1305.

3.2.2.2 Utilização de CPU

Na Figura 3.22 são apresentados os valores de utilização de CPU para cada VPN, em percentagem, para tráfego UDP. De notar, desde logo, que não é apresentado qualquer valor para strongSwan, dado que não foi possível detetar qualquer processo a correr no sistema associada à mesma. Mesmo no caso deste existir, não foi observável qualquer processo a correr no sistema que apresentasse um valor significativo de utilização de CPU. Para os testes *Baseline* também não existe nenhum processo, dado não ser usada qualquer VPN.

A OpenVPN apresenta valores de utilização de CPU muito similares entre si, não se notando grande influência tanto da cifra como do protocolo de transporte utilizado. Estes são consideravelmente elevados, principalmente considerando a utilização de CPU não significativa por parte do IPsec strongSwan. A WireGuard apresenta sensivelmente metade da utilização de CPU da OpenVPN, que, em ambientes mais limitados ou de natureza crítica, é um fator de elevada importância.

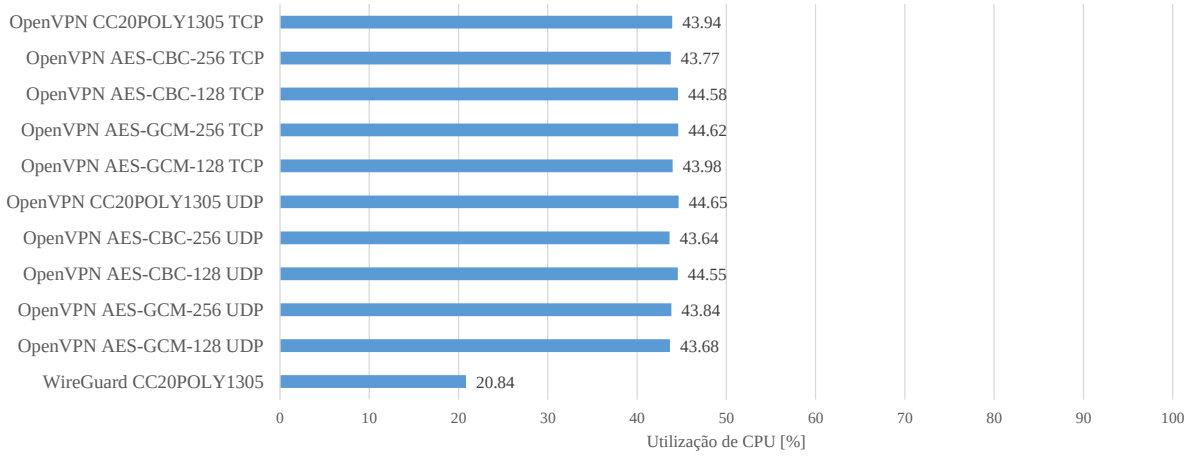


Figura 3.22: Utilização de CPU das VPN para tráfego UDP.

Na Figura 3.23 são apresentados os valores de utilização de CPU para cada VPN, em percentagem, para tráfego TCP. Tal como para o tráfego UDP, não são apresentados quaisquer valores para strongSwan e *Baseline*.

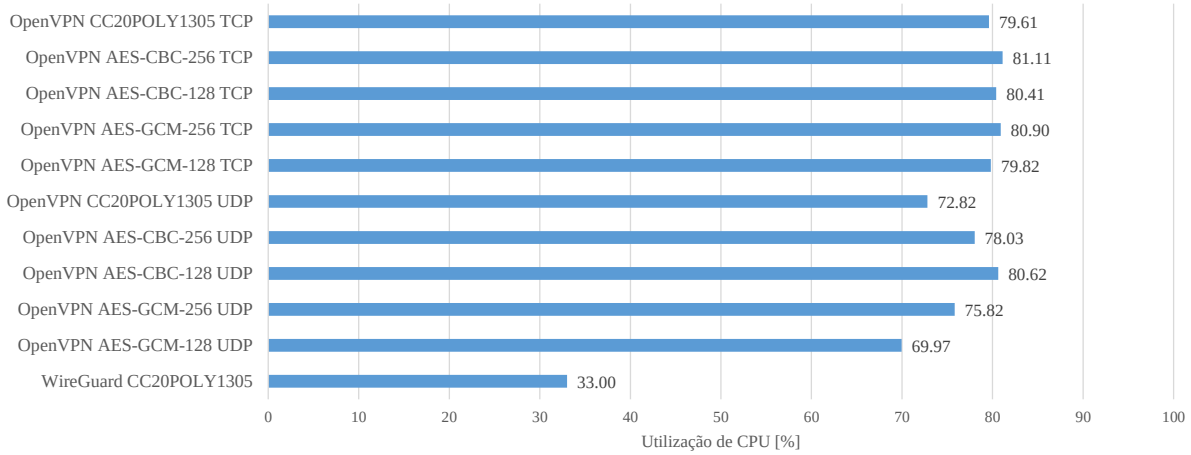


Figura 3.23: Utilização de CPU das VPN para tráfego TCP.

A OpenVPN, novamente, é que apresenta a maior utilização de CPU. Neste cenário, no caso do transporte ser sobre TCP, os valores são muito idênticos entre si. Quando o tráfego é transportado sobre UDP, a carga sobre o CPU é inferior e esta varia mais consoante a cifra. Nota-se que a cifras AES-GCM consomem menos CPU que as AES-CBC, enquanto a ChaCha20Poly1305 fica entre estas duas. Quanto à WireGuard, esta apresenta uma carga sobre o CPU que é menos de metade do melhor resultado alcançado com OpenVPN.

Comparando os resultados de tráfego UDP e TCP, verifica-se que o processamento de pacotes TCP é mais pesado do que o processamento de tráfego UDP. Isto nota-se tanto para a OpenVPN como para a WireGuard. A diferença entre os dois cenários é bastante grande, pelo que, se a utilização de CPU for fulcral, deve ser utilizado tráfego UDP. Utilizar UDP leva a que seja possível correr mais processos sem que exista uma sobrecarga, como seria provável com tráfego TCP. Quanto à WireGuard, em ambos os cenários, esta ganha com larga vantagem, sempre com menos de metade da carga sobre o CPU do que a OpenVPN.

3.2.2.3 Perda de pacotes para tráfego UDP

Na Figura 3.24 são apresentados os valores da perda de pacotes para cada VPN, em termos de percentagem da totalidade dos pacotes enviados, para tráfego UDP.

Se por um lado a perda de pacotes para as VPN strongSwan e WireGuard é praticamente inexistente, a perda de pacotes para todas as combinações de cifra e protocolo de transporte da OpenVPN é muito grande, sendo que o pior caso é quando os pacotes

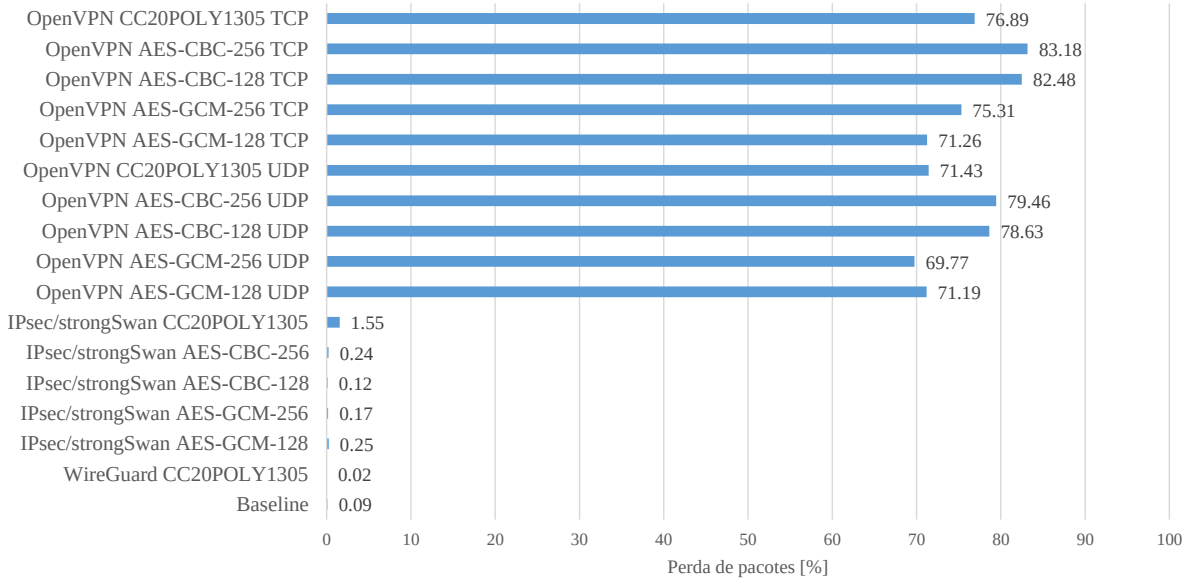


Figura 3.24: Perda de pacotes das VPN para tráfego UDP.

são transportados sobre TCP. Observa-se ainda que, para ambos os protocolos, os melhores resultados são obtidos com as cifras AES-GCM. Resultados bastante elevados de perda de pacotes foram também observados em [56]. Em todos os casos, à exceção da cifra AES-GCM-128 sobre UDP, existe uma perda superior a 70%. Estes valores tão elevados levam a que a escolha desta VPN não seja viável na maioria dos cenários.

A perda de pacotes para a VPN WireGuard até foi inferior aos testes *Baseline*. Relembrar, no entanto, que, tratando-se de máquinas virtuais, existe sempre algum grau de instabilidade, que pode ter feito com que os testes *Baseline* apresentassem uma latência superior. Quando a IPsec, a sua perda de pacotes também é bastante baixa, mas superior à da WireGuard. A única exceção, na qual a perda de pacotes não é assim tão similar aos valores de WireGuard, é a cifra ChaCha20Poly1305.

3.2.2.4 Latência

Por último, foram realizados testes de latência e, na Figura 3.25, são apresentados os valores do RTT médio, em milissegundos, para todas as VPN. A latência da strongSwan e WireGuard é pouco superior aos resultados *Baseline*, algo bastante positivo.

A OpenVPN apresenta os maiores valores de latência, demonstrando mais uma vez que é a VPN com pior desempenho. Para esta VPN nota-se bastante a influência da escolha do protocolo de transporte a ser usado. Em todos os casos, quando comparada a mesma cifra, sobre UDP e TCP, o melhor desempenho é obtido com a utilização de

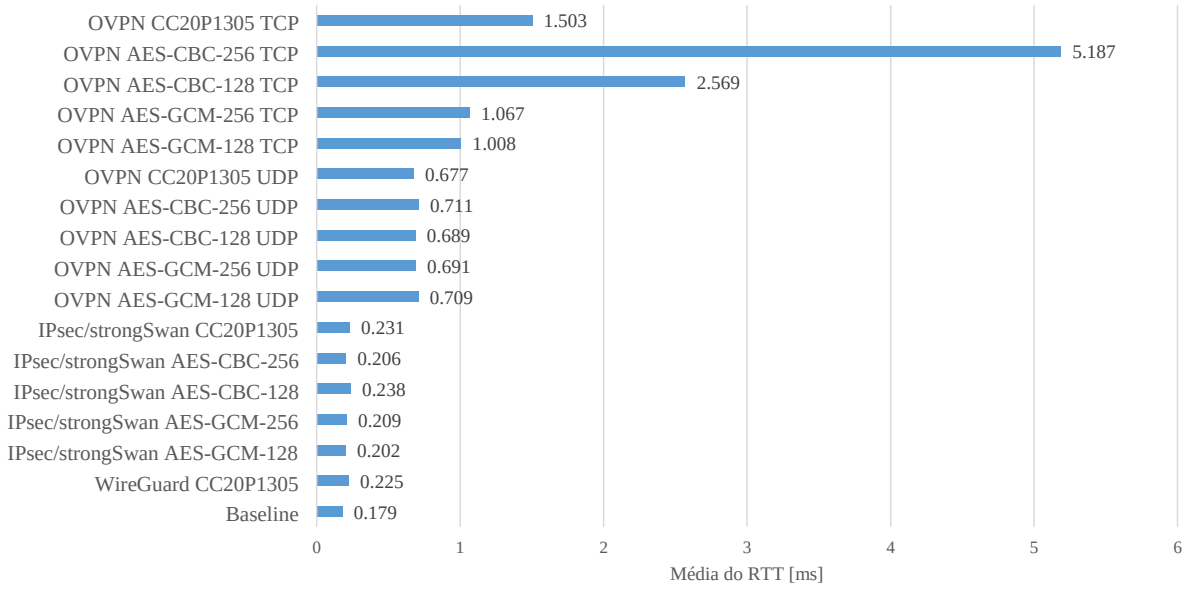


Figura 3.25: RTT médio em milissegundos das VPN.

UDP. O facto da OpenVPN correr em *User Space* acaba por ter um papel fundamental nos valores obtidos, dado que introduz atrasos que acabam por impactar os mesmos [37]. Comparações entre a latência obtida sobre túneis UDP e TCP foram também realizadas em [36], onde as conclusões são similares.

3.2.3 Conclusão

Os resultados obtidos com os testes realizados demonstram que a OpenVPN apresenta os piores resultados em todos os testes, sem exceção. A sua utilização de CPU é muito elevada, e o mesmo acontece com a perda de pacotes. A latência é também, de longe, a mais elevada.

Em termos de ritmo binário, a strongSwan apresenta os melhores valores para tráfego UDP, já em relação a TCP a WireGuard revela resultados substancialmente superiores.

Em termos de utilização de CPU, como afirmando anteriormente, não foi possível observar processos associados a IPsec com utilização significativa de CPU. Considerando a comparação entre a WireGuard e OpenVPN, a WireGuard leva a melhor, apresentando sensivelmente metade da utilização de CPU da OpenVPN, para qualquer protocolo e cifra. Foi também possível verificar que o processamento de tráfego TCP é mais pesado para o CPU.

Quanto à perda de pacotes, existe praticamente um empate entre WireGuard e strongSwan

(IPsec). Por fim, como afirmando anteriormente, não foi possível observar processos associados a IPsec com utilização significativa de CPU, pelo que consideramos que o processamento introduzido é diminuto.

No que toca à latência, IPsec e WireGuard apresentam resultados idênticos, pouco mais elevados que os resultados *Baseline*. A OpenVPN apresenta a maior latência. Para UDP, cerca de três vezes a latência de IPsec e para TCP superior a cinco vezes.

O desempenho da OpenVPN ser, em todos os testes, pior do que as restantes VPN testadas era algo esperado, pelo facto desta correr no *User Space*. Desta forma, é gasto algum tempo de execução na comunicação entre o *User Space* e o *Kernel Space*, o que não acontece para a strongSwan e WireGuard.

Outro aspeto importante que se confirmou foi a superioridade geral das cifras AEAD. Estas apresentaram um desempenho superior às cifras AES-CBC em todos os testes. Assim sendo, quando as cifras AEAD estiverem disponíveis, devem ser escolhidas em detrimento das cifras clássicas.

Em suma, apesar de tanto a WireGuard como a strongSwan terem pontos de destaque, revelam-se como sendo VPN bastante equilibradas, apresentando resultados similares de perda de pacotes e latência, e adequadas à maioria dos cenários onde seja necessária a utilização de uma VPN.

Sendo que o sistema a desenvolver têm o propósito de servir serviços críticos, a utilização da OpenVPN é impensável dado os resultados obtidos. Entre strongSwan e WireGuard, apesar destas apresentarem algumas métricas de desempenho similares, e apesar de não terem sido realizado testes de tempo necessário para iniciar ligações, segundo a literatura, a WireGuard é a que apresenta os tempos mais reduzidos, crucial para o tipo de sistema que se pretende desenvolver. Este fator leva a que consideremos esta VPN como a mais adequada de entre as testadas, dado que, se pensarmos numa *gateway* de comunicações móveis, esta vai estar constantemente a mudar de rede, pelo que é de extrema importância que a VPN seja rápida a restabelecer ligação.

3.3 Estratégias de Avaliação Passiva da Qualidade das Ligações

No ambiente de comunicações em que vai atuar o sistema de encaminhamento, é essencial aferir a qualidade das redes móveis disponíveis em cada momento, para que se consigam tomar decisões relativamente à forma como os dados vão ser enviados. Certo tipo de dados deve ter prioridade face a outro tipo de dados e, alguns deles,

podem ser descartados ou o seu envio atrasado, caso se verifique que a rede não tem capacidade suficiente para processar todos eles.

Dando suporte a serviços críticos, o sistema de encaminhamento resiliente deve interferir o menos possível no tráfego real, pretendendo-se, por isso, utilizar uma avaliação passiva. Tendo em conta que do lado da *gateway* vai ser usado um sistema operativo Linux, procuraram-se mecanismos que estes sistemas possuam que possam ajudar nesta tarefa. Depois de alguma pesquisa foi possível verificar que o Kernel armazena informações acerca de todas as ligações TCP que se encontram a decorrer, sendo preenchida uma estrutura de dados com múltiplas métricas, denominada de `tcp_info`. Esta inclui métricas como o RTT, *Retransmission Timeout* (RTO), *Maximum Segment Size* (MSS), *Congestion Window* (cwnd), entre outros. A partir das métricas disponíveis, o objetivo é encontrar formas de gerar diversos *Key Performance Indicator* (KPI) que contribuam para a tomada de decisão.

Considerando *gateways* que estejam constantemente em movimento, e que tenham um percurso bem definido como, por exemplo, uma *gateway* integrada num comboio, pode ser pertinente armazenar as suas coordenadas geográficas. A informação das coordenadas pode ser cruzada com as métricas das ligações medidas e ser traçado um mapa de qualidade da rede consoante a localização. Naturalmente, a qualidade da rede está em constante mudança, mas este pode ser um bom indicador da qualidade de rede esperada. Para o armazenamento das coordenadas, é mandatório que a *gateway* possua um recetor *Global Navigation Satellite System* (GNSS), que suporte mais que um sistema de posicionamento, para que não se dependa apenas de um e, que suporte a *Satellite-based Augmentation Systems* (SBAS), para aumentar a precisão do posicionamento.

Através de um *script* Python e recorrendo à biblioteca `netlink` [57], foi possível ler as estruturas `tcp_info` de cada ligação TCP estabelecida num determinado momento. No *script* realizado, a informação contida na estrutura `tcp_info` de cada ligação TCP é armazenada numa tabela de uma base de dados PostgreSQL. Nessa tabela são também armazenadas as coordenadas geográficas da *gateway* no momento em que foi realizada a leitura. A informação `tcp_info` e a informação geográfica são colocadas simultaneamente na tabela a cada segundo.

Um aspeto a ter em conta é que esta abordagem de leitura dos dados armazenados pelo Kernel só é possível para ligações TCP. As ligações UDP são *Stateless*, não são armazenados quaisquer indicadores da sua qualidade, pelo que devem ser procuradas outras formas de estimar a qualidade destas ligações.

Para testar o *script* realizado, é necessário existirem ligações TCP ativas. Assim sendo, e

tendo acesso a uma *gateway* real, foram estabelecidas ligações *Message Queuing Telemetry Transport* (MQTT) com um *broker* remoto, atuando a *gateway* como cliente. A *gateway* possuía diversos modems disponíveis, sendo utilizados dois modems e, a cada um deles, foi associado um cliente MQTT, existindo duas ligações com o *broker*. A topologia de testes é apresentada na Figura 3.26.

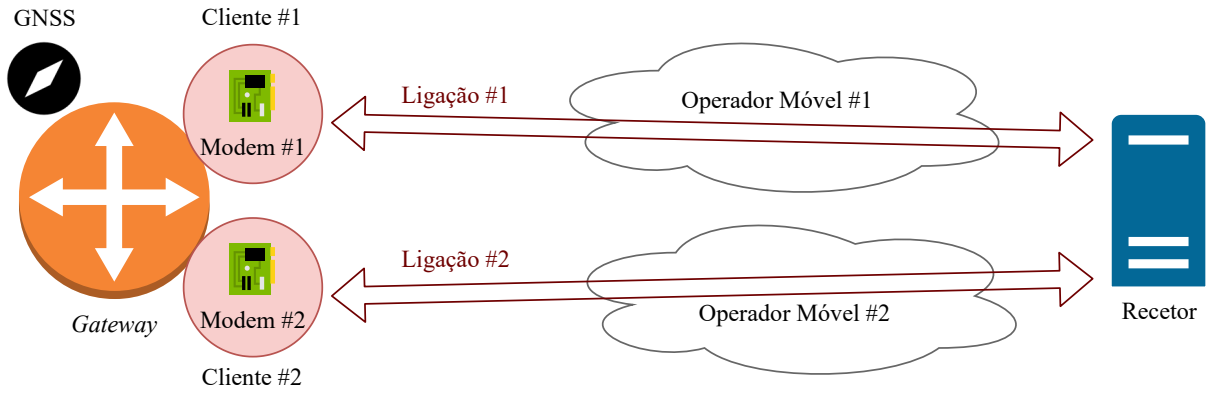


Figura 3.26: Topologia de testes para medição passiva de ligações TCP.

3.4 Propostas de Arquiteturas de Rede

Nesta Secção vão ser apresentadas as arquiteturas de suporte a sistemas críticos que se propõem, tendo sido estudadas as vantagens e desvantagens de cada uma. Para a idealização de cada arquitetura, foi tida em conta o desempenho das VPN, os protocolos *multipath* estudados, a possibilidade de agregação e as técnicas de avaliação da qualidade das ligações.

3.4.1 VPN sobre Agregado de Ligações

A *gateway* que se pretende desenvolver deve possuir diversos modems e, através destes, conseguir estabelecer comunicação via diferentes operadores e diversas tecnologias como o 4G e 5G. Com base nestas características, uma dos objetivos é encontrar uma forma de agregar as múltiplas ligações disponíveis, para maximizar o ritmo binário possível de alcançar. Desta forma, o ritmo binário obtido para o envio de dados será, teoricamente, a soma do ritmo binário alcançável através de cada ligação móvel individual disponível num determinado momento.

Geralmente quando se aborda o tema de agregação de ligações, pensa-se numa agregação ao nível *ethernet*, nível 2 da camada OSI. No entanto, sendo que recorremos a redes

móveis públicas, uma agregação a esse nível não é possível, sendo apenas possível realizar agregações de nível 3 (camada IP) para cima.

Considerando que a agregação a realizar só é possível a partir da camada 3, pode-se optar pela utilização do protocolo MPTCP, que atua na camada 4, dado que é considerado o protocolo mais promissor do seu género, tendo sido alvo de desenvolvimentos importantes nos últimos tempos. Outro aspeto importante é a sua inclusão no Kernel Linux, facilitando a sua adoção. Para além da agregação de ligações, outra temática a ter em conta é a segurança dos dados, podendo ser utilizada uma VPN para o efeito.

Tendo em mente os requisitos mencionados, como primeira arquitetura de comunicação, estudou-se uma arquitetura em que se tem uma VPN, o componente de segurança, a correr sobre a agregação das diversas ligações móveis disponíveis, eventualmente utilizando o protocolo MPTCP descrito. O MPTCP vai ser o protocolo de transporte utilizado pela VPN em vez do tradicional UDP ou TCP, sendo necessária a adoção de uma VPN que suporte TCP e em vez disso utilizar MPTCP, que será transparente para a VPN.

A Figura 3.27 representa o cenário da arquitetura proposta. Podem-se observar duas ligações móveis, podendo, naturalmente, ser mais. O MPTCP realizará a agregação de todas as ligações disponíveis. Tendo em conta que os dados a enviar podem ser TCP, há que ter em consideração o problema do TCP sobre MPTCP. Sobre a agregação mencionada temos a VPN estabelecida, que é inicializada na *gateway* e termina num concentrador de VPN. O concentrador de VPN atua simplesmente como um *router*, o recetor não tem conhecimento dos diversos IP das VPN.

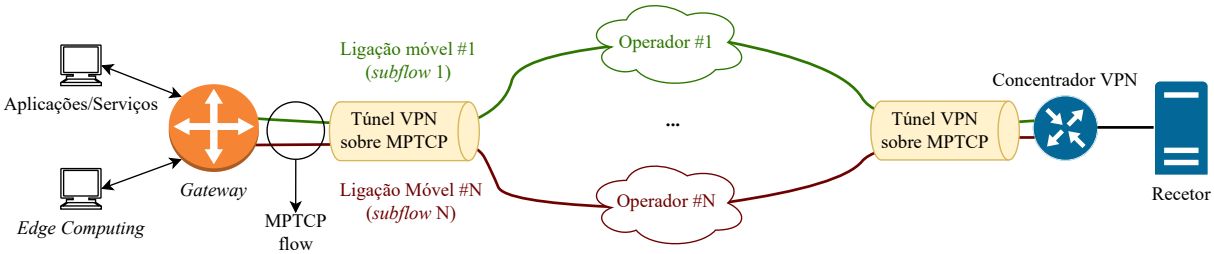


Figura 3.27: VPN sobre agregado de ligações.

A Figura 3.28 descreve a arquitetura lógica da arquitetura de comunicação proposta, em que se têm as diversas ligações móveis, a camada de agregação por cima, que realiza a agregação das diversas ligações e por cima desta temos a camada de segurança, representada pela VPN. Sobre esta última está a camada aplicacional.

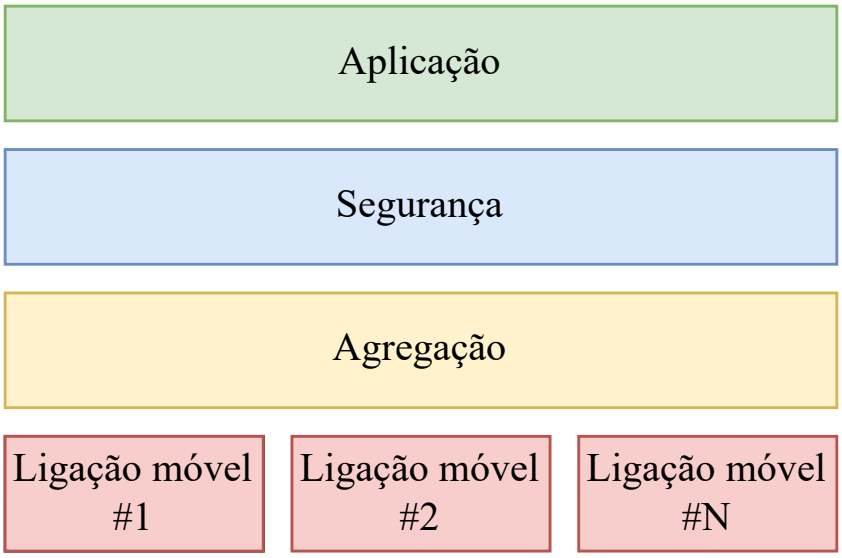


Figura 3.28: Arquitetura lógica da arquitetura VPN sobre agregado de ligações.

A tabela 3.2 resume as principais vantagens e desvantagens da arquitetura apresentada.

Tabela 3.2: Vantagens e desvantagens da arquitetura VPN sobre agregado de ligações.

Vantagens	Desvantagens
Utilização de MPTCP como protocolo de transporte da VPN	TCP sobre MPTCP introduz atrasos significativos
Agregação das diversas ligações permite o aproveitamento da largura de banda disponível em cada ligação simultaneamente	VPN exige processamento significativo
Aumento de tolerância a falhas	Ritmo binário obtido pode ser bastante afetado

3.4.2 Agregação de Ligações VPN

Apresentando outra proposta de arquitetura, inverte-se a posição do componente de agregação e do componente de segurança face à anterior. Se na arquitetura anterior se utilizava uma única VPN sobre a agregação de diversas redes móveis, nesta é realizada uma agregação sobre diversas VPN. Nesta situação, teríamos uma VPN por cada ligação móvel, para assegurar o componente de segurança, e para agregação utiliza-se MPTCP sobre as diversas VPN estabelecidas.

A Figura 3.29 descreve a arquitetura proposta. Repare-se que na Figura da arquitetura anterior tínhamos apenas uma VPN, e nesta existe uma VPN por cada ligação móvel. Quanto aos *subflows MPTCP*, estes são estabelecidos sobre as VPN. Mais uma vez, as VPN começam na gateway e terminam num concentrador de VPN próximo da plataforma de gestão.

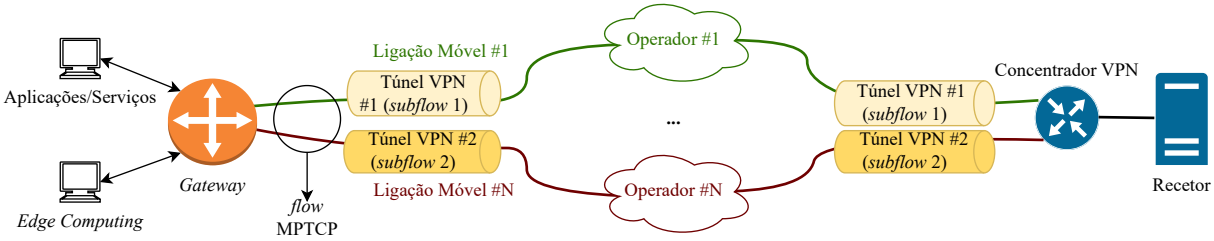


Figura 3.29: Agregação de ligações VPN.

É importante referir ainda que, para implementar esta arquitetura, é necessário que a ligação com o/os destinatário/destinatários seja inicializada na *gateway* e não em aplicações que utilizem a mesma para encaminhar o seu tráfego, para que se possa usar o protocolo MPTCP. Se é a *gateway* que tem acesso a diversas ligações, se se pretende utilizar MPTCP, então a ligação tem de ser iniciada na *gateway*. Face ao requisito mencionado, é necessário possuir um *broker* na *gateway*, para onde os dados das aplicações são enviados e, posteriormente, é a *gateway* a enviar os dados para os destinatários, iniciando a ligação com o mesmos.

Utilizando esta arquitetura, em vez de termos TCP (ou UDP) sobre MPTCP como acontecia na anterior, temos o oposto, MPTCP sobre TCP ou UDP, consoante o protocolo de transporte utilizado nas VPN. Podendo utilizar UDP, esta é a escolha mais acertada, levando à inexistência do problema de TCP sobre TCP, mais precisamente, de MPTCP sobre TCP. Considerando as características já abordadas sobre a WireGuard, considera-se que é a escolha acertada para VPN a utilizar nesta arquitetura.

A Figura 3.30 descreve a arquitetura lógica da arquitetura de comunicação proposta, em que se têm as diversas ligações móveis, uma camada de segurança (VPN) sobre cada ligação e a camada de agregação por cima, finalizando com a camada aplicacional. Portanto, face à anterior, uma inversão da camada de agregação e da camada de segurança, diferindo ainda na quantidade de componentes de segurança.

A tabela 3.3 resume as principais vantagens e desvantagens da arquitetura apresentada.

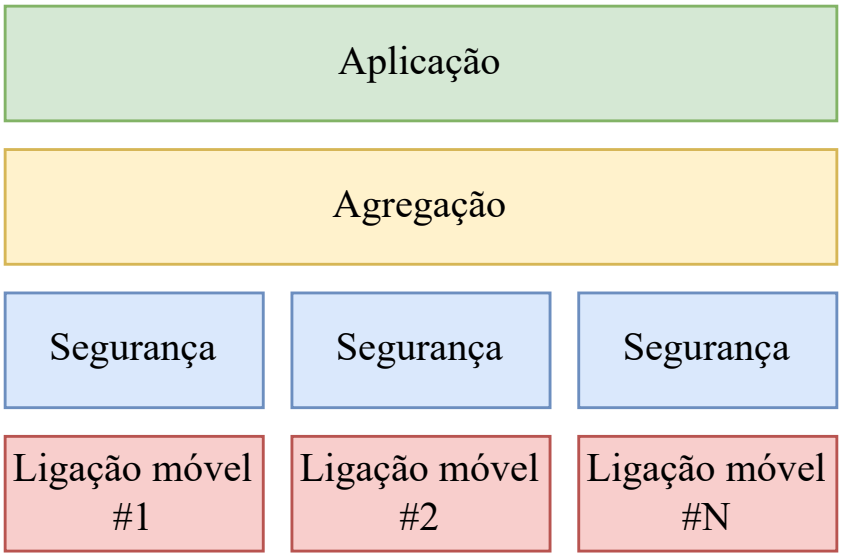


Figura 3.30: Arquitetura lógica da arquitetura agregação de ligações VPN.

Tabela 3.3: Vantagens e desvantagens da arquitetura agregação de ligações VPN.

Vantagens	Desvantagens
Utilização de MPTCP como protocolo de transporte da VPN	Necessidade de estabelecer uma VPN por ligação móvel
Agregação das diversas ligações permite o aproveitamento da largura de banda disponível em cada ligação simultaneamente	Utilização de diversas VPN introduz processamento significativo
Aumento de tolerância a falhas	Apenas utilizável caso se adote uma arquitetura em que existe um *broker* na *gateway*

3.4.3 *Routing* Dinâmico

Na terceira e última arquitetura de comunicação, defende-se uma abordagem de comunicação utilizando *routing* dinâmico. Para isso, elimina-se por completo o componente de agregação, permitindo assim escolher dinamicamente qual a ligação que deve ser usada para o encaminhamento dos dados em cada momento, em vez de serem utilizadas todas as ligações simultaneamente, como acontecia nas outras duas arquiteturas. O componente de segurança continua presente, sendo estabelecida uma VPN por cada ligação.

A Figura 3.31 descreve esta arquitetura e podemos observar que não existe agregação, tendo apenas diversas VPN, uma por cada ligação móvel. Nesta arquitetura, tal como na anterior, tem de existir um *broker* na *gateway* para onde as aplicações vão enviar os

seus dados. O *broker* é necessário, dado que, para decidir como se vai realizar o encaminhamento dos pacotes, considerando uma avaliação passiva com base na estrutura `tcp_info` disponível no Kernel, é necessário acesso às *sockets* das ligações. Sendo que é a *gateway* que tem acesso às diversas redes, é esta que vai realizar a sua avaliação. Para tal ser possível, tem de ser a *gateway* a iniciar as ligações com o exterior, recebendo os dados das aplicações através do *broker* e enviando-os para os destinatários, garantindo assim o acesso às *sockets* das ligações.

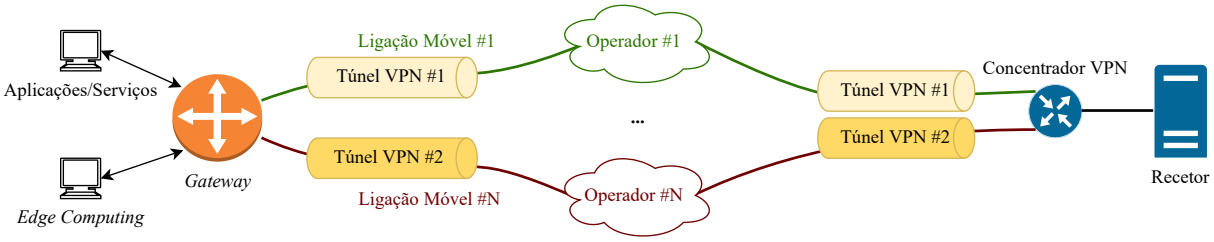


Figura 3.31: *Routing* dinâmico.

A Figura 3.32 descreve a arquitetura lógica da arquitetura de comunicação proposta e é de salientar a ausência da camada de agregação.

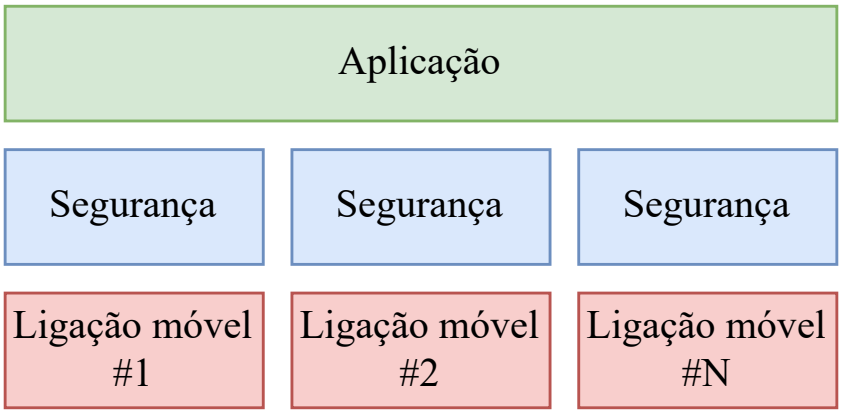


Figura 3.32: Arquitetura lógica da arquitetura de *routing* dinâmico.

A tabela 3.4 resume as principais vantagens e desvantagens da arquitetura apresentada.

Tabela 3.4: Vantagens e desvantagens da arquitetura de *routing* dinâmico.

Vantagens	Desvantagens
Não existe necessidade de MPTCP	Não aproveitamento simultâneo da largura de banda disponível em cada ligação móvel
Possibilidade de escolher a ligação móvel que apresenta melhor qualidade num determinado momento	Necessidade de estabelecer uma VPN por ligação móvel
	Utilização de diversas VPN introduz processamento significativo
	Necessidade de recolha passiva de métricas das ligações para influenciar o encaminhamento de pacotes - utilização de um *broker*

3.4.4 Arquitetura Escolhida e Conclusões

Das três arquiteturas apresentadas, foi escolhida a terceira — arquitetura de *routing* dinâmico, segundo diversas características que se consideram as mais acertadas para o sistema de encaminhamento resiliente a desenvolver.

A arquitetura selecionada utiliza a VPN WireGuard, que, como visto anteriormente, foi a VPN que consideramos como mais adequada, principalmente por apresentar, segundo a literatura, o tempo necessário mais reduzido para iniciar ligações.

Apesar de ter sido pensada a utilização de protocolos *multipath*, mais precisamente o MPTCP, a arquitetura selecionada não o utiliza. Este é ainda um protocolo em desenvolvimento, cujo desempenho pode colocar em causa o funcionamento do sistema. Adicionaria também uma camada de complexidade à arquitetura. Outro aspeto importante é que, sem a utilização de um protocolo *multipath*, é possível escolher a melhor rede em cada momento para o encaminhamento do tráfego. As restantes redes disponíveis podem também ser utilizados com total controlo por parte de uma eventual API que tomará decisões acerca da forma como os pacotes vão ser encaminhados. Caso se utilizasse um protocolo *multipath*, o controlo sobre o encaminhamento dos pacotes seria mais limitado, tendo que se utilizar todas as ligações simultaneamente.

Para decidir de que forma será realizado o encaminhamento dos pacotes com base no seu nível de prioridade, é elementar a existência de uma forma de avaliação das múltiplas redes. Dos métodos abordados na Secção 2.2, os mais adequados seriam os métodos passivos, dado que não injetam tráfego nas redes, não degradando a adequação das mesmas face ao potencial encaminhamento dos dados. Por essa razão, pretende-se

utilizar a estrutura `tcp_info` disponível no Kernel. Ainda assim, não é fácil realizar uma avaliação exclusivamente passiva. As avaliações ativas providenciam dados que não são possíveis de obter apenas com métodos passivos, sendo, portanto, necessária a injeção de pacotes nas redes a analisar, mas, sempre tendo em conta que estes devem interferir o menos possível com as mesmas.

3.5 Qualidade de Serviços de Redes Móveis

Para além dos parâmetros das ligações TCP recolhidos através do Kernel dos sistemas UNIX, poderá ser pertinente ter em conta também os indicadores de qualidade da rede que os *modems* de redes móveis conseguem retornar. Esses indicadores podem ser uma ajuda preciosa para tornar o processo de decisão acerca do encaminhamento mais preciso.

Assim sendo, recorrendo a uma *gateway* real com diversos modems ligados, e utilizado o `Modem Manager` disponível no Linux, é possível interagir com os modems e obter indicadores que estes consigam retornar. Os indicadores retornados dependem da tecnologia sobre a qual a rede opera, podendo ser WCDMA (3G), LTE 4G ou NR (5G). Na tabela 3.5 são apresentados os indicadores retornados consoante a tecnologia utilizada.

Tabela 3.5: Parâmetros de rede retornados pelos modems, por tecnologia.

Parâmetros	WCDMA	LTE	NR
Cell ID	X	X	X
Received Signal Strength Indicator (RSSI)	X	X	X
TX *Power*	X	X	X
LAC/TAC	X	X	X
Received Signal Code Power (RSCP)	X	-	-
Energy per chip to Interference power ratio (ECIO)	X	-	-
Reference Signal Received Power (RSRP)	-	X	X
Reference Signal Received Quality (RSRQ)	-	X	X
Signal to Noise Ratio (SNR)	-	X	X

Para a obtenção dos parâmetros retornados pelos modems foi utilizado um *script* Bash que interage com o `Modem Manager`. Este *script* é evocado dentro de um *script* Python, responsável por retirar os dados relevantes e armazená-los numa base de dados PostgreSQL.

Para testar a obtenção de indicadores dos modems e podermos aferir a relação que estes têm com a qualidade das ligações, é necessário ter ligações a decorrer, sendo utilizada a ferramenta Iperf3 para estabelecer uma ligação com um servidor remoto

e medir o ritmo binário e a latência. A topologia utilizada nestes testes é apresentada na Figura 3.26. Dado que, na realidade, os resultados de ritmo binário e latência apresentados pelo Iperf3 são obtidos pela ferramenta através dos dados da estrutura `tcp_info`, pode-se, em alternativa, utilizar apenas uma ligação MQTT e obter as métricas da ligação através da estrutura.

Segundo alguns testes realizados, verificou-se que existe uma determinada correlação entre o valor da SNR e o valor do ritmo binário medido. Quando a correlação é muito elevada, significa que o ritmo binário é bastante dependente do valor de SNR, o que significa que o *bottleneck* da ligação é a comunicação entre a *gateway* e a estação base do operador móvel a que este se conecta. É possível retirar esta conclusão dado que a SNR está relacionada apenas com a comunicação entre a *gateway* e a estação base do operador. Se o *bottleneck* que afeta negativamente o ritmo binário obtido estiver na rede do operador, ou no próprio processamento por parte do servidor com o qual se está a estabelecer ligação associada ao Iperf3, então a correlação entre SNR e ritmo binário será mais baixa.

Na Figura 3.33 são apresentados os valores de SNR e ritmo binário medidos para uma determinada ligação de teste, e na tabela 3.6 as correlações entre parâmetros. A correlação entre as duas medidas foi de 75.6%, revelando então que é a comunicação entre a *gateway* e a estação base do operador que influencia fortemente o ritmo binário alcançado. A correlação entre o RSRQ e o ritmo binário também foi elevada, mas menor quando comparada com a SNR. Relativamente às restantes medidas, as correlações entre estas e o ritmo binário foram abaixo dos 55% na generalidade. As correlações com o RTT não fornecem grandes indícios de relação. Note-se, no entanto, que o RTT e o ritmo binário têm uma correlação de -67.75%, revelando que existe uma relação significativa entre estas medidas. A correlação tem um valor negativo dado que à medida que o ritmo binário aumenta, o valor de RTT diminui, e vice-versa.

Tabela 3.6: Correlação entre índices de qualidade de rede, ritmo binário e RTT.

	Ritmo binário	RTT
Ritmo binário	100%	-67.75%
RTT	-67.75%	100%
RSSI	39.45%	-47.23%
RSRP	54.52%	-57.66%
RSRQ	72.22%	-56.60%
SNR	75.60%	-53.60%

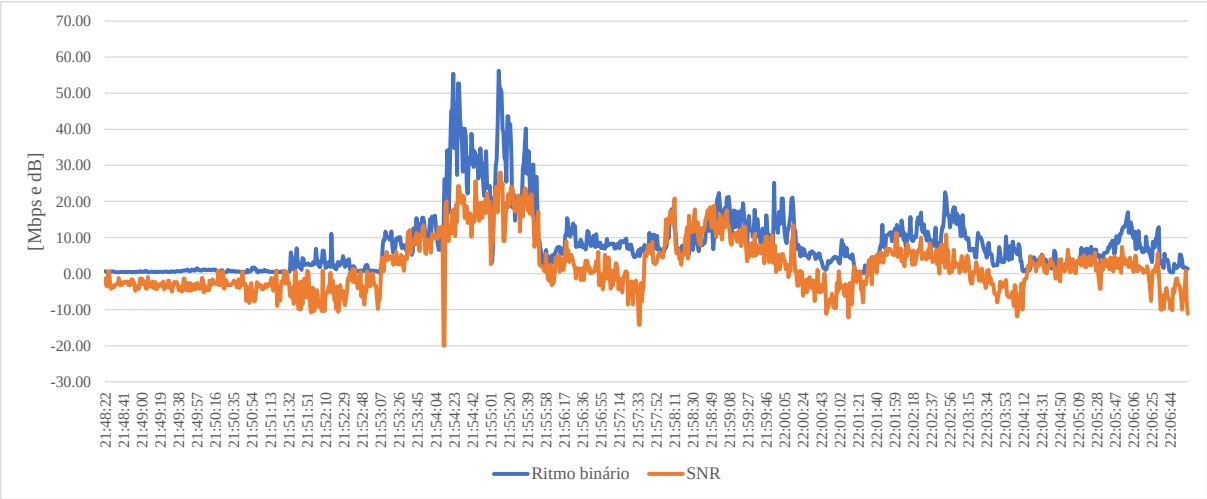


Figura 3.33: SNR e ritmo binário em função do tempo para uma ligação móvel de teste.

Para se poderem tirar maiores conclusões, é necessário ter em conta mais ligações realizadas e não apenas uma, pelo que existe necessidade de realizar mais testes. De qualquer das formas, métodos de avaliação de redes baseadas em dados obtidos das redes móveis são uma adição de elevada importância às técnicas de avaliação mais comuns. Estes dados fornecem informação acerca da qualidade da ligação entre a *gateway* e a entrada da rede do operador. Através destes é possível entender, por exemplo, se o *bottleneck* da ligação é precisamente a ligação entre a *gateway* e a estação base do operador ou se será outra parte da rede.

3.6 Encaminhamento com base em Marcação

Um problema já apontado múltiplas vezes é a forma como vai ser distribuído o tráfego pelas múltiplas ligações. Apesar de a arquitetura de rede escolhida não contemplar a utilização de MPTCP, existem diversos caminhos disponíveis para o encaminhamento dos dados, pelo que devem ser todos utilizados. A distribuição do tráfego pelas diversas ligações pode ser feita com base numa marcação. Esta marcação, por exemplo DSCP, é atribuída por alguma máquina que se encontre na rede, antes do tráfego chegar à *gateway*.

A marcação DSCP apesar de ser amplamente adotada por dispositivos de rede, especificamente no Kernel, não é possível, através de comandos ip-route, estabelecer regras de encaminhamento com base em marcação DSCP. Com este tipo de comandos só é possível estabelecer regras com base em marcação ToS, que não é tão utilizada atualmente, ou, por exemplo, com base em `FWMARK`.

Para atingir o dinamismo em termos de escolher que rede deve ser usada para que tipo de tráfego, é necessário utilizar uma tabela de encaminhamento por cada rede disponível. Isto acontece porque na tabela de encaminhamento principal não é possível ter diversas rotas para o mesmo destino.

A marcação `FWMARK` é utilizada apenas ao nível de máquinas Linux, não sendo suportada por mais dispositivos. Esta pode ser aplicada na *gateway*, quando esta recebe os pacotes, utilizando iptables. Desta forma é possível fazer *match* com o valor da marcação e encaminhar esses dados para uma determinada rede ou para ser avaliada uma tabela de encaminhamento em específico. Ou seja, a marcação `FWMARK` pode ser usada apenas localmente na *gateway* para que esta consiga definir que interface de rede deve ser usada para a saída do tráfego. Através de ip-rule definimos que, com base no valor da marcação dos pacotes, será avaliada uma determinada tabela de encaminhamento. Dentro de cada tabela de encaminhamento, recorrendo a comandos ip-route, é colocada uma regra para que a interface associada à rede correspondente à tabela seja utilizada como *gateway*.

Na Figura 3.34 é possível observar o exemplo de uma regra iptables na qual a *gateway* aplica uma marcação `FWMARK` 0x4 a 30% do tráfego destinado à rede 192.168.200.0/24.

```
root@gateway:/home/fit# iptables -A PREROUTING -t mangle -d 192.168.200.0/24 -m statistic --mode random --probability 0.3 -j MARK --set-mark 4
```

Figura 3.34: Regras para encaminhamento do tráfego.

Na Figura 3.35 é possível observar um exemplo de uma lista de regras que definem, com base na marcação `FWMARK` dos pacotes, aplicada na própria *gateway*, que tabela de encaminhamento deve ser avaliada. Neste caso particular, todos os pacotes que não possuem marcação `FWMARK` vão ser encaminhados com base na tabela principal (*main*) e os pacotes com `FWMARK` 0x4 são encaminhados com base na tabela 1.

```
root@gateway:/home/fit# ip rule show
0:      from all lookup local
32765:  from all fwmark 0x4 lookup 1
32766:  from all lookup main
32767:  from all lookup default
```

Figura 3.35: Regras para encaminhamento do tráfego.

Na eventualidade do tráfego chegar marcado à *gateway*, pode-se optar por realizar uma correspondência entre o valor de DSCP e um valor de `FWMARK`, sendo o encaminhamento feito com base na importância e prioridade dos pacotes. Mesmo que os

pacotes já possuam marcação, esta também pode ser completamente ignorada e ser realizada uma nova avaliação por parte da *gateway* e uma atribuição de um valor `FWMARK` consoante a mesma.

3.7 Múltiplas Instâncias de VPN na *Gateway*

Como visto no capítulo anterior, é possível marcar tráfego para influenciar como é realizado o encaminhamento do mesmo. A marcação é realizada com o intuito de distribuir o tráfego pelos múltiplos caminhos, maximizando os recursos disponíveis.

Existindo uma VPN por cada caminho, pretende-se testar se existe ou não vantagem em realizar distribuição do tráfego por diversos caminhos, considerando que as VPN necessitam de processamento considerável para o processo de cifra dos dados.

A VPN utilizada para os testes foi a WireGuard. Para aferir o impacto da utilização de diversas instâncias, em primeiro lugar, foi realizada distribuição do tráfego para ligações UDP sem qualquer VPN estabelecida sobre as ligações. Seguidamente, foi utilizada a WireGuard e estabelecido tráfego UDP sobre a mesma.

Para os testes utilizou-se uma máquina virtual com ligação à *gateway*, em que esta envia dados para a *gateway*. Nesta, é realizada marcação `FWMARK` nas iptables para distribuir o tráfego pelas duas ligações, com ou sem VPN. Foram aplicados diversos valores de distribuição de pacotes, sendo a marcação realizada com base no módulo de estatística das iptables, com o qual se conseguem definir diversos valores de distribuição do tráfego em termos de percentagem. Por exemplo, um dos cenários testados foi distribuir 30% do tráfego por uma ligação e os restantes 70% por outra, como visível na Figura 3.36. Na realidade, na marcação realizada, só é aplicado um valor de marcação (0x4) e os restantes pacotes não são marcados.

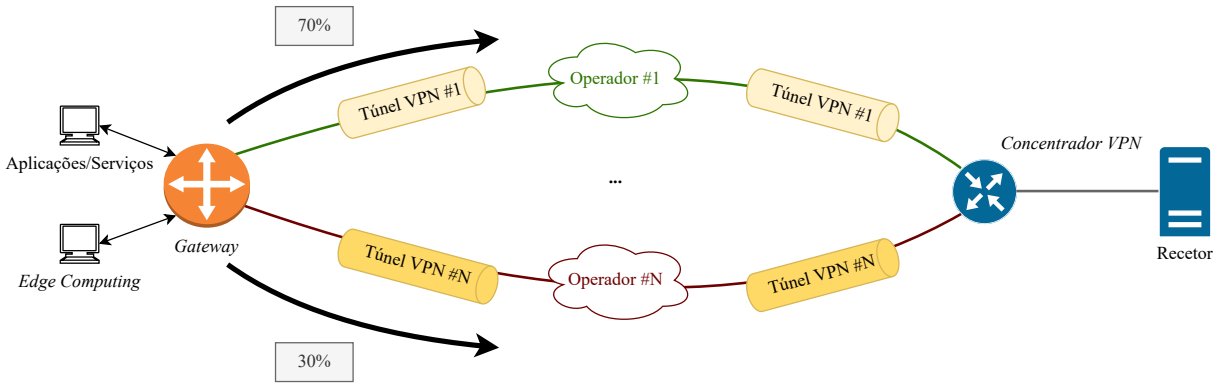


Figura 3.36: Exemplo de distribuição de tráfego por diversas interfaces.

Em termos de tabelas de encaminhamento, como mencionado anteriormente, é utilizada uma por cada ligação, portanto, no cenário testado, duas tabelas. A tabela principal é para tráfego que não possui marcação e está associada à ligação 1 sem VPN (C1) ou VPN 1 (V1). A tabela 1 é utilizada para o tráfego marcado com valor `FWMARK` 0x4 e está associada à ligação 2 sem VPN (C2) ou VPN 2 (V2).

Foram testados diversos valores de largura de banda, sendo estes alterados de forma virtual recorrendo à ferramenta `Wondershaper`. Os valores utilizados foram 100, 200, 300 e 400 Mbps. Os testes realizados contemplam o mesmo valor de largura de banda para cada ligação de cada vez, devendo ser realizados testes com valores de largura de banda diferentes para cada ligação para se entender o seu impacto. Antes de se testar a distribuição, foi também testada cada ligação de forma individual, de forma a verificar o ritmo binário alcançado.

As ligações de teste foram estabelecidas com o Iperf3, UDP e utilizando a opção "-b" para forçar a largura de banda.

3.7.1 UDP sem VPN

As Tabela 3.7, 3.8, 3.9 e 3.10 mostram os resultados de ritmo binário obtidos para ligações UDP sem VPN e para uma largura de banda forçada no Iperf3 de 800 Mbps, tendo sido realizadas 3 séries de testes para cada cenário. Foi utilizada a largura de banda de 800 Mbps dado ser a soma do valor máximo testado, para cada ligação, ou seja, 400 Mbps em cada ligação. Usando este valor para todos os testes faz com que uma eventual influência do valor deste parâmetro nos resultados obtidos seja mitigada.

Os resultados mostram que existe um ganho em distribuir o tráfego pelas diversas ligações. Em todos os cenários de largura de banda e distribuição existiu um ganho na utilização das duas ligações. Observe-se, por exemplo, os resultados obtidos para qualquer valor de largura de banda e com uma distribuição de 50% 50%, estes são muito próximos da soma do ritmo binário de cada ligação.

Tabela 3.7: Ritmo binário médio alcançado com UDP para uma ligação com largura de banda de 100 Mbps.

Distribuição de tráfego	Ritmo binário (Mbps)
C1 100 Mbps	91,43
C2 100 Mbps	91,67
50% C1, 50% C2	182,00
30% C1, 70% C2	181,33
70% C1, 30% C2	181,67
10% C1, 90% C2	164,33
90% C1, 10% C2	164,33

Tabela 3.8: Ritmo binário médio alcançado com UDP para uma ligação com largura de banda de 200 Mbps.

Distribuição de tráfego	Ritmo binário (Mbps)
C1 100 Mbps	181,33
C2 100 Mbps	180,67
50% C1, 50% C2	357,00
30% C1, 70% C2	351,33
70% C1, 30% C2	352,67
10% C1, 90% C2	252,00
90% C1, 10% C2	253,33

Tabela 3.9: Ritmo binário médio alcançado com UDP para uma ligação com largura de banda de 300 Mbps.

Distribuição de tráfego	Ritmo binário (Mbps)
C1 100 Mbps	267,67
C2 100 Mbps	269,00
50% C1, 50% C2	517,33
30% C1, 70% C2	468,67
70% C1, 30% C2	482,00
10% C1, 90% C2	342,67
90% C1, 10% C2	344,67

Tabela 3.10: Ritmo binário médio alcançado com UDP para uma ligação com largura de banda de 400 Mbps.

Distribuição de tráfego	Ritmo binário (Mbps)
C1 100 Mbps	354,00
C2 100 Mbps	354,33
50% C1, 50% C2	639,67
30% C1, 70% C2	555,33
70% C1, 30% C2	562,00
10% C1, 90% C2	425,67
90% C1, 10% C2	425,33

3.7.2 VPN sobre UDP

A Tabela 3.11, 3.12, 3.13 e 3.14 mostram os resultados de ritmo binário obtidos para ligações UDP sobre VPN e para uma largura de banda forçada no Iperf3 de 800 Mbps, tendo sido realizadas 3 séries de testes para cada cenário.

Os resultados para uma largura de banda de 100 e 200 Mbps mostram que existe vantagem na utilização de diversas ligações. Ainda assim, para os 200 Mbps já se nota uma quebra face ao ritmo binário alcançado nas mesmas condições, mas sem VPN. Quando se observam os resultados para os 300 Mbps entende-se que os resultados são similares à utilização de apenas uma VPN, ou que existe uma perda de desempenho, no caso da distribuição 50% 50%. Para os 400 Mbps não existe mesmo nenhum cenário no qual o ritmo binário obtido seja superior à utilização de uma VPN apenas, sendo estes os testes que apresentaram os piores resultados de entre todos.

Quanto maior foi o valor de largura de banda das ligações sobre VPN, piores foram os resultados obtidos. Para os 100 Mbps o desempenho é similar ao alcançado sem quaisquer VPN envolvidas. A partir dos 200 e principalmente para os 300 e 400 Mbps, não existe vantagem na utilização de múltiplas instâncias VPN e distribuição de tráfego sobre as mesmas, é mais benéfico encaminhar a totalidade do tráfego por uma VPN apenas.

Os resultados foram piores quanto maior o valor de largura de banda, dado que quanto maior esta for, maior é o esforço que as VPN têm de realizar para cifrar todos os dados e proceder ao seu envio. Notou-se que a utilização do CPU era maior quanto maior a largura de banda, chegando perto dos 100% de utilização para os 300 e principalmente para os 400 Mbps.

Tendo em conta que o processador utilizado pelas máquinas virtuais é um Intel Xeon E5-2609 v4, um processador de elevada capacidade, e ainda que é bastante provável que uma eventual *gateway* real não tenha um processador mais capaz do que este, espera-se que os resultados obtidos numa *gateway* real sejam próximos dos que foram aqui obtidos. Estes revelam ser necessário gerir muito bem a existência de diversas VPN, devendo ser analisada a largura de banda que está disponível em cada ligação para encontrar a melhor forma de distribuir o tráfego, ou não o distribuir de todo.

Tabela 3.11: Ritmo binário alcançado com UDP sobre VPN para uma ligação com largura de banda de 100 Mbps.

Distribuição de tráfego	Ritmo binário (Mbps)
C1 100 Mbps	92,37
C2 100 Mbps	92,20
50% C1, 50% C2	183,33
30% C1, 70% C2	182,33
70% C1, 30% C2	184,00
10% C1, 90% C2	158,67
90% C1, 10% C2	160,33

Tabela 3.12: Ritmo binário alcançado com UDP sobre VPN para uma ligação com largura de banda de 200 Mbps.

Distribuição de tráfego	Ritmo binário (Mbps)
C1 100 Mbps	181,67
C2 100 Mbps	182,33
50% C1, 50% C2	290,00
30% C1, 70% C2	303,00
70% C1, 30% C2	298,67
10% C1, 90% C2	247,67
90% C1, 10% C2	249,33

Tabela 3.13: Ritmo binário alcançada com UDP sobre VPN para uma ligação com largura de banda de 300 Mbps.

Distribuição de tráfego	Ritmo binário (Mbps)
C1 100 Mbps	230,00
C2 100 Mbps	249,67
50% C1, 50% C2	205,00
30% C1, 70% C2	247,67
70% C1, 30% C2	266,33
10% C1, 90% C2	253,67
90% C1, 10% C2	248,00

Tabela 3.14: Ritmo binário alcançada com UDP sobre VPN para uma ligação com largura de banda de 400 Mbps.

Distribuição de tráfego	Ritmo binário (Mbps)
C1 100 Mbps	271,33
C2 100 Mbps	263,67
50% C1, 50% C2	167,33
30% C1, 70% C2	243,33
70% C1, 30% C2	243,33
10% C1, 90% C2	238,67
90% C1, 10% C2	243,33

3.8 Proposta Final

Chegada esta fase, foram utilizados todos os estudos e testes realizados anteriormente, para começar a desenvolver uma aplicação de avaliação de redes que incorpora todos os componentes analisados e que se reflete na arquitetura final a adotar.

Em primeiro lugar são abordados os perfis de encaminhamento definidos, seguindo-se a análise dos módulos que compõem a aplicação e, por fim, é descrito o ficheiro de configuração utilizado para personalizar o comportamento da aplicação.

3.8.1 Perfis de Encaminhamento

Visto que a aplicação vai encaminhar dados de diversos serviços/aplicações e cada uma tem requisitos definidos acerca da qualidade necessária para o tratamento dos

seus dados, a aplicação define um contrato composto por diversos perfis de encaminhamento. Estes estão disponíveis para as aplicações poderem decidir qual o perfil de encaminhamento que querem que seja aplicado aos seus dados. Esta escolha é realizada por meio de marcação DSCP conforme o especificado no contrato. Existindo diversos requisitos consoante as aplicações e serviços, foram definidos diversos perfis com características variadas, apresentados na Tabela 3.15. Denote-se, no entanto, que não estão definidos nem valores de DSCP, nem de `FWMARK`, marcação utilizada localmente na *gateway*, visto que estes são configuráveis por parte do gestor do sistema.

Todo o tráfego que não possui marcação, ou marcação que não consta nos perfis de encaminhamento, é tratado com uma política *best-effort*.

Tabela 3.15: Perfis de Encaminhamento

Nome do Perfil	Descrição
100% Tráfego por um único caminho	Existe um perfil por cada caminho existente. Estes permitem que os dados de determinada aplicação sejam encaminhados exclusivamente por um determinado caminho.
Tráfego distribuído uniformemente	Percentagem de tráfego por cada caminho é dada por 100 sobre a quantidade de caminhos/interfaces utilizadas.
Caminho com maior ritmo binário	Perfil sempre associado ao caminho com maior valor de ritmo binário. Este é um perfil dinâmico, dado que tem de ser atualizado de tempo a tempo, consoante o desempenho dos caminhos.
Caminho com maior largura de banda	Perfil sempre associado ao caminho com maior largura de banda. Este é um perfil dinâmico, dado que tem de ser atualizado de tempo a tempo, consoante o desempenho dos caminhos.
Caminho de menor latência	Perfil sempre associado ao caminho com menor latência.
Caminho de menor *jitter*	Perfil sempre associado ao caminho com menor valor de *jitter*.

Tráfego distribuído conforme a percentagem de ritmo binário do caminho face ao total	O tráfego é distribuído segundo a percentagem de ritmo binário que um caminho tem face à soma de ritmo binário de cada caminho num determinado instante.
Tráfego distribuído consoante a percentagem de largura de banda do caminho face ao total	O tráfego é distribuído consoante a percentagem de largura de banda que um caminho tem face à soma de largura de banda de cada caminho num determinado instante.
Tráfego distribuído de acordo com o inverso da percentagem de latência do caminho face ao total	O tráfego é distribuído consoante o inverso da percentagem de latência que um caminho tem face à soma de latência de cada caminho num determinado instante.
Tráfego distribuído de acordo com o inverso da percentagem de *jitter* do caminho face ao total	O tráfego é distribuído segundo o inverso da percentagem de *jitter* que um caminho tem face à soma de *jitter* de cada caminho num determinado instante.
Tráfego distribuído de acordo com o ritmo binário e latência	O tráfego é distribuído de acordo com o ritmo binário do caminho face ao total, num determinado instante e consoante o inverso da percentagem de latência que um caminho tem face à soma de latência de cada caminho num determinado instante.

Para facilitar a publicação do contrato e respetivos perfis, definiu-se um modelo JSON composto por tantos objetos quantos os perfis definidos, em que o nome dos objetos é dado pelo nome do perfil, em inglês. Os objetos dos perfis associados a 100% do tráfego por uma única interface/caminho possuem a marcação DSCP a si associada, bem como um campo constituído pelas medidas de ritmo binário, largura de banda disponível, latência e *jitter*, métricas essenciais na avaliação de redes. Os restantes perfis possuem também um campo com a marcação DSCP e, em vez de um campo com as medidas, apresentam um campo com a distribuição do tráfego pelas interfaces. Por exemplo, no perfil utilizado para o caminho com o maior valor de ritmo binário, como

só existe um caminho associado, não existe um campo com o valor da distribuição, mas são incluídas as diversas métricas atuais da rede associada a essa interface. Outro exemplo, o perfil de distribuição consoante o ritmo binário, tendo tantas interfaces associadas como aquelas que são geridas pela aplicação, no campo de distribuição possui o nome das interfaces e os respetivos valores de distribuição atuais, alterados constantemente. O modelo do contrato, com alguns perfis definidos para exemplo, é visível na Listagem 3.1.

Dada a necessidade de dar a conhecer às aplicações os perfis de encaminhamento disponíveis, optou-se por utilizar mensagens MQTT para o efeito, sendo enviadas para um tópico definido no ficheiro de configuração. O *broker* responsável pela sua receção deve ser acessível às aplicações, que subscrevem o tópico respetivo para poderem ter conhecimento do contrato publicado. Assim sendo, decidiu-se colocar o *broker* na *gateway*, pelo que as aplicações conseguem aceder. Para além disso é publicado noutro broker MQTT numa plataforma central, isto porque como no contrato também são incluídas algumas métricas da qualidade das ligações atuais e ainda a distribuição atual de tráfego para cada perfil, um administrador pode aceder à plataforma central para visualizar as métricas e saber qual a qualidade atual de todas as redes.

```
{
  "100_enp1s0": {
    "DSCP": "57",
    "measurements": {
      "enp1s0": {
        "throughput": "29.3",
        "bandwidth": "100.01",
        "latency": "0.23",
        "jitter": "0.54"
      }
    }
  },
  "100_enp7s0": {
    "DSCP": "58",
    "measurements": {
      "enp7s0": {
        "throughput": "28.61",
        "bandwidth": "85.68",
        "latency": "0.29",
        "jitter": "0.32"
      }
    }
  },
```

```
  "100_enp8s0": {
    "DSCP": "59",
    "measurements": {
      "enp8s0": {
        "throughput": "38.12",
        "bandwidth": "136.25",
        "latency": "0.39",
        "jitter": "0.48"
      }
    }
  },
  "higher_throughput": {
    "DSCP": "60",
    "distribution": {
      "enp1s0": "100"
    }
  },
  "throughput_based_traffic_distribution": {
    "DSCP": "61",
    "distribution": {
      "enp1s0": "30.52",
      "enp7s0": "29.79",
      "enp8s0": "39.69"
    }
  }
}
```

Listagem 3.1: Modelo JSON do Contrato

3.8.2 Módulo de Controlo de Redes

A aplicação deve ser dotada da capacidade de gerir a tabela de encaminhamento da máquina com o objetivo de controlar as rotas. Pela mesma razão tem de ser dotada da capacidade de estabelecer regras de encaminhamento. Outro aspeto importante é a capacidade de controlar as nftables, que vieram substituir as iptables, pelo que se decidiu usar esta última. O controlo das nftables é essencial dado que, como discutido na Secção 3.6, através desta *framework*, é possível definir que tráfego será enviado por cada ligação e em que quantidade.

Num nível acima, dada a utilização de um componente de segurança por cada ligação, isto é, uma VPN, a aplicação tem de conseguir iniciar e terminar o estabelecimento de cada VPN, mais precisamente, instâncias WireGuard, visto que foi esta a escolhida.

Quando a aplicação é inicializada, este é o primeiro módulo a atuar, começando por verificar quais as interfaces de rede que estão disponíveis no sistema. Dado que em algumas situações pode não ser necessário que a aplicação faça a gestão de todas as interfaces de rede, é configurável quais as interfaces que se pretende que a aplicação utilize. Desta forma, depois da verificação das interfaces disponíveis no sistema é necessário verificar se estas constam na lista de interfaces especificadas. Caso alguma interface disponível não esteja nessa lista, não é realizada qualquer ação de gestão sobre a mesma.

Tendo em consideração a utilização da VPN WireGuard, a aplicação encarrega-se de iniciar uma instância de WireGuard, conforme a arquitetura de rede adotada, por cada interface sobre as qual a aplicação deve atuar. Portanto, a gestão que a aplicação realiza é sobre as interfaces de rede virtuais que as instâncias WireGuard estabelecem por cima das interfaces do sistema.

Após estabelecidas as VPN, a aplicação necessita de criar uma tabela de encaminhamento por cada interface/instância WireGuard. De seguida, estas tabelas são preenchidas com as rotas adequadas que permitam que o tráfego possa sair por todas as interfaces, isto é, por qualquer instância WireGuard disponível.

Dada a necessidade de marcação para definir como o tráfego é encaminhado, como debatido na Secção 3.6, definiu-se que cada interface/caminho está associado a um valor `FWMARK`. A associação entre a marcação e a interface é estabelecida através de regras ip-rule, com as quais se definem que, consoante o valor `FWMARK`, é avaliada uma determinada tabela de encaminhamento, que possui as regras específicas para que o tráfego seja encaminhado através da interface associada. Os valores de marcação são aplicados com nftables, antes do processo de avaliação das regras ip-rule. É nas regras nftables que, ao longo do tempo e com base na avaliação realizada, os valores `FWMARK` são alterados, para adequar as regras ao estado atual de todas as redes. Isto é, a alteração do valor `FWMARK` nas regras, altera as interfaces utilizadas nas mesmas, regras essas que refletem perfis de encaminhamento. Portanto, as regras ip-rule permanecem iguais ao longo do tempo, estas só são alteradas caso exista uma alteração nos perfis de encaminhamento estabelecidos no contrato, ou se uma interface ficar disponível ou indisponível, já que nestes casos têm de ser adicionadas ou removidas regras. Apenas as regras nftables são alteradas para que, em qualquer instante, seja entregue às aplicações a qualidade estabelecida nos perfis de encaminhamento que estas escolheram para o tratamento dos seus dados.

Como nos instantes iniciais a aplicação não possui quaisquer métricas das redes, a marcação a atribuir pode ser arbitrária, não sendo este um problema, dado que em

poucos instantes a aplicação terá dados para redefinir as regras de encaminhamento.

Para além da marcação que atua localmente na *gateway*, existe a marcação realizada anteriormente à chegada do tráfego à *gateway*, possivelmente, aplicada pelas próprias aplicações. Esta, na maioria das vezes, é marcação DSCP e define quais as características do tráfego, qual a sua prioridade e requisitos em termos de qualidade da rede. A aplicação utiliza a informação definida no contrato para construir as regras nftables, uma por cada perfil de encaminhamento. As regras nftables são um mapeamento entre um valor DSCP e a percentagem de tráfego encaminhado por cada caminho, bem como a definição de quais os caminhos a utilizar, seguindo o que está estabelecido nos perfis de encaminhamento. Nas regras constam os valores `FWMARK` associados aos caminhos a utilizar, que vão sendo alterados consoante a qualidade medida, bem como a própria distribuição. As nftables aplicam localmente os valores `FWMARK` definidas nas regras, segundo a percentagem especificada, aos pacotes que chegam à *gateway*, consoante o valor DSCP que estes contêm.

Após o estabelecimento das rotas e regras de encaminhamento a aplicação prossegue para a instalação das regras nftables.

Concluindo, este módulo abrange a gestão das regras e rotas de encaminhamento, gestão das nftables e gestão da WireGuard. É o primeiro módulo a atuar, preparando a *gateway* para o encaminhamento dos dados. Depois disso é utilizado quando é necessário estabelecer uma nova regra ou rota, ou realizar a sua alteração, bem como iniciar ou terminar alguma instância VPN.

3.8.3 Módulo de Avaliação de Redes

O componente principal deste módulo e da aplicação é a avaliação das redes. Considerando avaliação passiva, existe necessidade de um componente de avaliação que interaja com a estrutura `tcp_info` do Kernel para obter as métricas TCP. Do componente de avaliação passiva também faz parte o componente de obtenção de métricas dos modems, na qual não é avaliada a estrutura `tcp_info`, mas sim os parâmetros de qualidade retornados pelos modems, como descrito em 3.5.

Sendo objetivo interferir o mínimo possível no tráfego real da rede, mas considerando também a dificuldade de medir largura de banda, ritmo binário ou BTC exclusivamente com medidas passivas, considera-se essencial o envio de dados para estimar a BTC, isto é, um componente de avaliação ativa. Na Secção 2.2.1 foram abordadas diversas técnicas de o fazer. Pela sua forte adoção e por ser uma ferramenta muito testada, optou-se pela utilização do Iperf3, lançado periodicamente para o cálculo da

BTC, que, neste cenário, é utilizada como uma estimativa da largura de banda. O intervalo entre cada medição é configurável, bem como a duração de cada uma dessas medições, devendo estes valores ser adaptados consoante o ambiente no qual a *gateway* irá ser utilizada.

Num ambiente no qual a *gateway* se encontra em constante movimento, um intervalo de 5 minutos entre medições não é adequado. Neste espaço de tempo, possivelmente, a *gateway* já alterou por diversas vezes as redes às quais esta ligada. Mesmo considerando uma *gateway* num ambiente estático, dada a natureza de constantes alterações do estado da rede, devido à constante alteração de dispositivos móveis conectados e desconectados, não se deve utilizar um intervalo de tempo muito elevado. Por outro lado, o intervalo de tempo também não deve ser muito baixo, visto que o Iperf3 tenta enviar pacotes ao maior ritmo possível, tentando utilizar a totalidade da largura de banda disponível, algo que impacta significativamente a rede. Se estas medições estiverem sempre no ativo, fazem com que o tráfego crítico que precisa efetivamente dos recursos disponibilizados pela *gateway* seja prejudicado. Concluí-se que tanto o intervalo de tempo como a própria duração de cada teste despoletado devem ser definidos com muita precaução.

Este módulo integra ainda um componente de obtenção de coordenadas geográficas, dado que pode existir uma correlação entre a localização e a qualidade de rede existente. A partir desta correlação pode ser traçado um mapa de qualidade da rede.

Os componentes de avaliação passiva, ativa e o componente de obtenção de coordenadas interagem com o componente repositório para serem armazenadas as métricas e coordenadas na base de dados, respetivamente.

Num nível mais elevado da arquitetura lógica da aplicação, temos um componente de agregação das métricas passivas e ativas. Os componentes de avaliação passiva e avaliação ativa reportam as métricas a este componente. Este, é responsável por fornecer a outro componente, o módulo de cálculo, todas as métricas, para serem realizados cálculos que reflitam a qualidade das ligações disponíveis. O componente de obtenção de coordenadas também reporta as medidas recolhidas ao componente de cálculo.

As métricas de avaliação passiva e ativa, bem como as coordenadas geográficas recolhidas, no componente de cálculo, poderiam ser obtidas diretamente da base de dados, através do repositório. No entanto, é mais eficiente estes serem passados diretamente, evitando-se assim que seja realizado um pedido à base de dados para obter os dados atuais. Ainda assim, o componente de cálculo comunica com o repositório, mas para obter métricas recolhidas anteriormente na mesma localização, para relacionar com as métricas atuais do mesmo local.

Com base nos cálculos realizados e no estado atual do encaminhamento dos pacotes, existe outro componente, o componente de decisão, que verifica se têm de ser estabelecidas novas rotas, regras de encaminhamento e novas regras de nftables, bem como apagar as rotas e regras estabelecidas anteriormente. Portanto, este interage com os componentes de gestão de regras de encaminhamento, gestão de rotas e gestão de nftables do módulo de controlo de redes.

Ao iniciar a aplicação, este módulo segue-se ao módulo de avaliação de redes. Começando pela avaliação passiva, utilizar este tipo de medição de desempenho das redes não significa que não exista necessidade de dados a serem enviados pela *gateway*, significa apenas que os dados utilizados para obtenção de métricas não são injetados propositadamente para esse fim. Assim sendo, a aplicação deve utilizar determinadas ligações de dados para realizar as medições passivas. Possuindo diversas interfaces, e para que os dados de teste sejam os mesmos, existe necessidade de utilizar ligações de dados equivalentes entre todas as interfaces, só assim se consegue uma avaliação fidedigna. Desta forma, podem-se seguir duas vertentes: ou são escolhidas ligações de dados estabelecidas em cada interface em que é garantido que os dados a circular são idênticos na forma e quantidade, ou são injetados pacotes artificialmente. A primeira opção é complexa pelo facto da obrigatoriedade destas ligações existirem e de existir uma por cada interface. A segunda opção é um problema porque a injeção de pacotes degrada a qualidade das ligações e, a avaliação tornar-se-ia ativa e não passiva. Porém, se forem injetados pacotes de tamanho reduzido, de certa forma, pode-se considerar avaliação passiva, visto que pouco impacto terá no desempenho das redes.

A solução encontrada para este problema, considerando o uso de mensagens MQTT para a publicação constante do contrato com os perfis de encaminhamento estabelecidos, é utilizar as mesmas mensagens para a avaliação passiva. A utilização de mensagens MQTT para avaliação passiva foi abordada também na Secção 3.3. Garantindo que todas as interfaces sobre a gestão da aplicação enviam os dados para o *broker* e que estes são iguais, obtém-se uma avaliação fidedigna. Com esta metodologia as medições com MQTT podem ser consideradas passivas, visto que as mensagens enviadas são necessárias ao funcionamento do sistema, os pacotes não são injetadas exclusivamente com o propósito de realizar medições.

Dada a abordagem escolhida para a avaliação passiva, o módulo de avaliação de rede começa por estabelecer um cliente MQTT por cada interface que a aplicação vai gerir. Após o estabelecimento de todos os clientes, começa o envio periódico de mensagens, sendo recolhidos os dados de avaliação passiva, obtidos através da estrutura `tcp_info` do Kernel. Esta estrutura possui diversos parâmetros de rede e nem todos são relevantes, pelo que foi realizada uma filtragem, só sendo armazenados alguns

dos dados. É, no entanto, importante afirmar que mesmo após a filtragem, grande parte dos dados não são utilizados pela aplicação para avaliar as redes. Estes foram mantidos para suportar um possível trabalho futuro, no qual esses parâmetros podem ser relevantes para melhorar o algoritmo de avaliação. Dos parâmetros filtrados, os mais relevantes são a quantidade de *bytes enviados* (para calcular o ritmo binário), RTT (latência), quantidade de pacotes perdidos e quantidade de pacotes retransmitidos.

Para minimizar o risco de instabilidade no encaminhamento dos dados, o módulo de decisão não analisa exclusivamente as medidas recolhidas no último instante. Se assim fosse, num cenário no qual existem duas redes disponíveis, se num determinado segundo apenas, a segunda rede tiver métricas mais favoráveis, caso este módulo realizasse decisões apenas com base nas métricas daquele momento, iria logo alterar as regras estabelecidas. No entanto, no instante seguinte a primeira rede poderia apresentar melhores métricas, pelo que seria necessário estabelecer novas regras mais uma vez. Assim sendo, optou-se por calcular médias deslizantes de todas as medidas recolhidas, fazendo com que as medidas recolhidas anteriormente tenham um peso na decisão realizada no instante atual. A decisão é então realizada com base no último valor calculado da média deslizante. A janela utilizada no cálculo é configurável e, consoante o valor, isso faz com que se altere o peso das medidas recolhidas anteriormente.

Para além das medidas passivas que, como já afirmado, não são necessárias para uma avaliação completa, é estabelecido um cliente Iperf3 por cada interface. A utilidade do Iperf3 é o cálculo da largura de banda disponível em cada caminho. Dado que o Iperf3 aquando da realização de um teste tenta saturar a largura de banda disponível, degrada a qualidade das redes para o tráfego real, pelo que este é lançado em intervalos temporais maiores e com uma duração tendencialmente reduzida. Ambos os parâmetros são configuráveis para um ajuste correto ao cenário de atuação. Visto que a largura de banda não é medida em todos os instantes, não se considerou adequado realizar cálculos de média deslizante para esta métrica. A decisão a tomar é sempre baseada nos últimos valores recolhidos para todas as interfaces.

Outro aspeto referido na Secção 3.5 é a avaliação com base em parâmetros de redes móveis. Apesar de ter sido encontrada uma correlação entre a SNR e o ritmo binário alcançado, são necessários mais testes para realizar conclusões com maiores certezas. Apesar de terem sido realizados testes recorrendo a uma *gateway* física, o desenvolvimento da aplicação em si, foi realizado recorrendo a máquinas virtuais, sem acesso a modems de redes móveis. Desta forma, apesar de a aplicação estar estruturada para a recolha de parâmetros retornados pelos modems, esta recolha, e consequente inclusão na fórmula de cálculo de qualidade não é realizada. Outro componente não disponível

na máquina virtual é um sistema GNSS, pelo que, novamente, apesar de a aplicação estar estruturada para a leitura de dados geográficos, estes não são tidos em consideração.

A partir do momento em que se coloquem em utilização os componentes de obtenção de coordenadas geográficas e de obtenção de parâmetros de redes móveis, dado que estas contribuem para a avaliação das redes, os perfis de encaminhamento do contrato devem ser revistos. Além de revisões, podem ser definidos novos perfis.

Tal como a própria definição dos contratos, os objetos que os descrevem no modelo apresentado na Secção 3.8.1 podem sofrer alterações como, por exemplo, incluir as medidas obtidas a partir dos modems.

Como não são recolhidas as medidas geográficas, não é armazenado qualquer valor na base de dados de coordenadas. Na base de dados das métricas são colocadas todas as métricas recolhidas, associadas ao instante no qual foram recolhidas.

Concluindo, este módulo abrange os componentes de avaliação passiva e ativa, agregação de métricas, cálculo da qualidade da rede, um repositório de acesso às bases de dados e um módulo de decisão, responsável por entregar sempre a qualidade adequada às aplicações.

Na Figura 3.37 é apresentada a arquitetura lógica da aplicação desenvolvida, onde se podem observar os módulos de avaliação de rede e o módulo de controlo de redes, bem como os componentes descritos anteriormente.

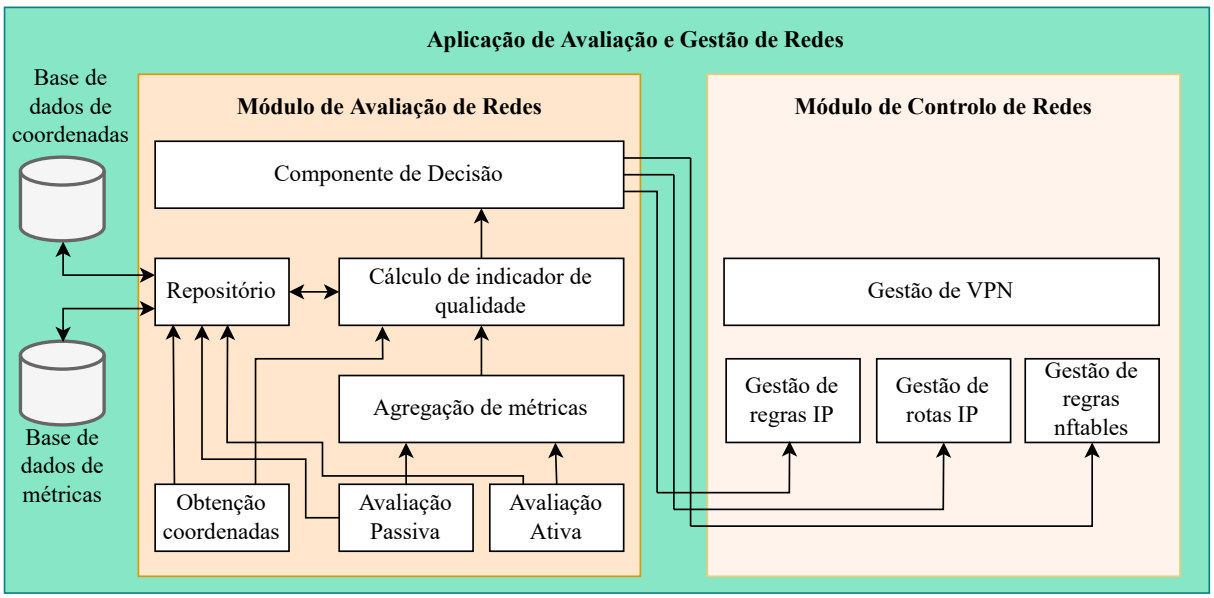


Figura 3.37: Arquitetura lógica Aplicação de Avaliação e Gestão de Redes.

Na Figura 3.38 é apresentada a arquitetura lógica da proposta final para a *gateway*, em que se podem observar as múltiplas ligações móveis, as múltiplas camadas de segurança. Por cima das camadas de segurança temos tanto a aplicação desenvolvida para a avaliação e gestão das redes da *gateway* como as aplicações/serviços ou *Edge Computing* genéricos que usarão a *gateway*.

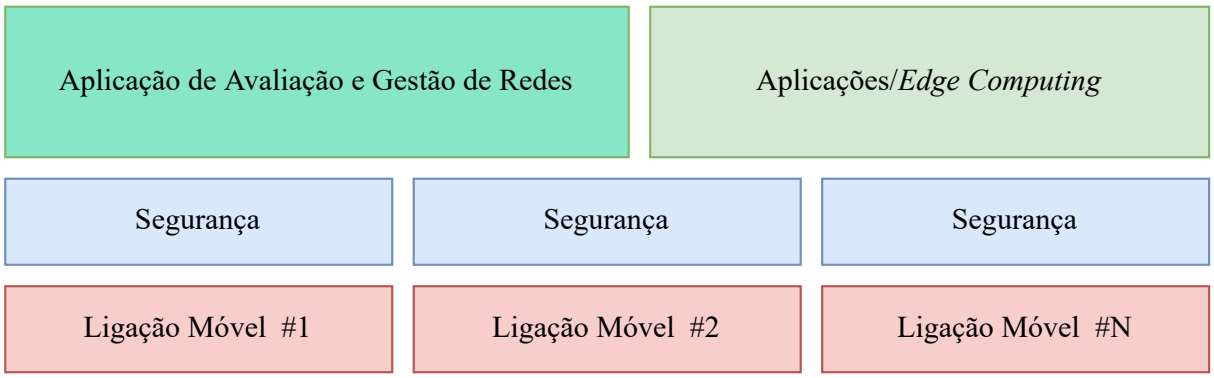


Figura 3.38: Arquitetura lógica *gateway*

Na Figura 3.39 é apresentada a arquitetura final proposta para a *gateway*. Nesta podem se observar os múltiplos modems que irão estar integrados na *gateway* e as diversas ligações móveis associadas, um recetor GNSS e a aplicação de avaliação e gestão de redes que vai integrar a *gateway*. A seguir ao concentrador de VPN temos um recetor e uma plataforma central. O objetivo da plataforma central é incorporar o *broker* MQTT para que as aplicações subscrevam o tópico respetivo e tenham conhecimento do contrato publicado, mensagens MQTT que são usadas também na avaliação passiva. Outro objetivo para a utilização da plataforma central é que esta sirva para visualização das métricas de medição de redes recolhidas, por parte de uma ou diversas *gateways*. Idealmente, as métricas devem ser integradas numa interface gráfica para que os responsáveis de rede consigam tirar conclusões de forma mais simples e eficaz. No entanto, o desenvolvimento desta plataforma ficou estabelecido como trabalho futuro, sendo abordado no Capítulo 4.

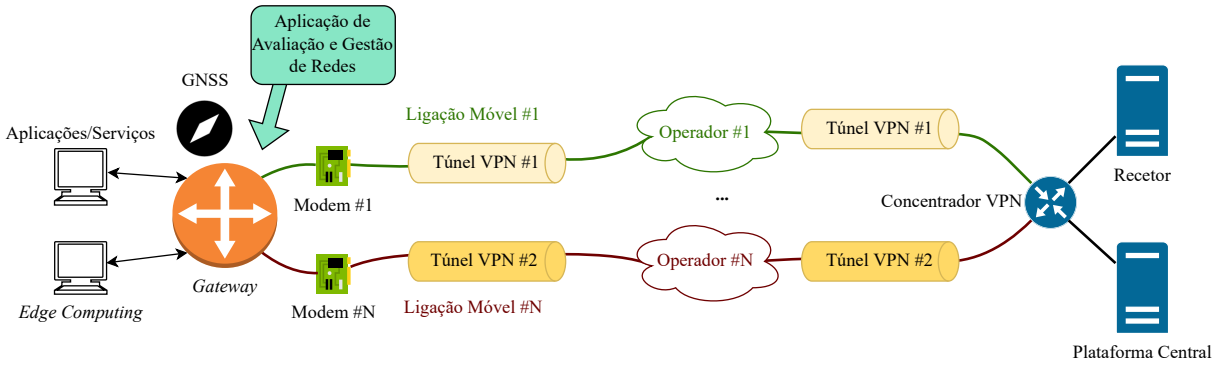


Figura 3.39: Arquitetura final *gateway*.

3.8.4 Ficheiro de Configuração

Para aumentar a flexibilidade de configuração da aplicação, foi elaborado um ficheiro de configuração onde constam parâmetros que podem ser alterados consoante o cenário no qual a aplicação irá atuar. Este ficheiro dá ao gestor de redes/gestor de sistema uma flexibilidade que o permite alterar o funcionamento da aplicação e, consequentemente, alterar como as redes são avaliadas e como o tráfego é encaminhado. Um dos parâmetros configuráveis é a lista de interfaces sobre as quais se pretende que a aplicação atue. Estão presentes também os parâmetros para configurar as ligações MQTT, tais como o *username*, *password*, *broker*, porto do *broker* e o nome do tópico que será usado para publicar as mensagens. Tendo em conta que a aplicação pode atuar num ambiente no qual existem diversas *gateways*, outro parâmetro configurável é o *gateway* ID, usado para identificar a *gateway* na ligação MQTT.

Em termos da marcação `FWMARK`, é definida uma lista com os valores a serem utilizados, devendo ser definidos tantos valores quanto as interfaces a serem tidas em conta na *gateway*. Os valores definidos devem seguir a mesma ordem que as interfaces apresentam na listagem de interfaces que a aplicação deve gerir.

Os valores DSCP associados ao contrato definido também são configuráveis através do ficheiro, sendo necessários definir tantos valores como a quantidade de perfis de encaminhamento definidos.

Relativamente às tabelas de encaminhamento, existe uma lista na qual são definidos os nomes das tabelas de encaminhamento consoante a interface associada. Os valores definidos devem seguir a mesma ordem que as interfaces apresentam na listagem de interfaces.

No que toca aos parâmetros de avaliação das redes, para o Iperf3 é configurável a

duração de cada teste, o intervalo de tempo entre testes, o endereço do servidor e o porto do mesmo.

Por fim, é configurável o tamanho da janela utilizada para o cálculo da média deslizante das métricas recolhidas, tais como a latência e ritmo binário.

4

Conclusões

Neste capítulo será apresentada uma síntese dos resultados obtidos com esta dissertação, bem como o trabalho futuro que se considera relevante realizar.

4.1 Síntese de Resultados

Nesta dissertação foram abordados diversos componentes essenciais para a construção de um sistema de encaminhamento resiliente, de forma a definir uma arquitetura de uma *gateway* capaz de realizar o encaminhamento de tráfego crítico, recorrendo a redes móveis. O primeiro passo foi realizar um estado da arte de todas as componentes relevantes, seguindo-se alguns testes para aferir a viabilidade de utilização de cada um deles. Com base nesses estudos e testes decidimos quais os componentes e tecnologias que deveriam integrar uma proposta final.

Dado que a *gateway* possui múltiplas ligações a redes móveis, realizou-se uma investigação acerca dos métodos disponíveis para avaliação de redes. Os métodos estudados dividem-se em avaliação passiva e ativa. Entre estas duas categorias, a mais adequada a um ambiente crítico é a avaliação passiva, que não injeta tráfego na rede, apenas observa o tráfego real. Evita-se assim a injeção de pacotes específicos e uma consequente degradação da rede, que iria impactar negativamente as aplicações críticas. Apesar da preferência por uma avaliação passiva, verificou-se que nem todas as métricas relevantes, como BTC, são possíveis de obter exclusivamente desta forma, pelo que a *gateway* também precisa de ter um componente de avaliação passiva.

Um dos objetivos iniciais era que o desenvolvimento da *gateway* fosse assente sobre sistemas de código aberto, sendo usado Linux. Como tal, para a avaliação passiva, foi feito um estudo para verificar que parâmetros de rede o Kernel conseguiria retornar. Verificou-se que este armazena métricas para todas as ligações TCP ativas, sendo possível aceder às mesmas. Quanto à avaliação ativa, tendo sido analisados diversos métodos e ferramentas, optou-se pela mais conhecida e adotada, o Iperf3.

Para além de métricas obtidas através do Kernel e do Iperf3, sendo que são utilizadas redes móveis, é oportuno obter métricas de rede que os modems consigam retornar. Recorrendo a uma *gateway* física, foi possível verificar quais as métricas retornadas. Estas são uma adição importante para tornar o processo de avaliação mais preciso.

A componente de segurança é fulcral em ambientes críticos, tendo sido testadas diversas soluções VPN, bem como diversas arquiteturas. Através do estado da arte e de testes realizados na dissertação, concluiu-se que a VPN mais adequada ao nosso cenário é a WireGuard. Esta apresenta baixa latência, elevado ritmo binário e, mais importante, um tempo de estabelecimento de ligação muito inferior às outras soluções.

Outro aspeto amplamente estudado foi o *multipath*, dado que, possuindo diversas ligações, a sua utilização tornaria o aproveitamento de recursos mais eficiente. Dos protocolos disponíveis, o mais promissor e discutido na atualidade é o MPTCP, que se encontra presente no Kernel Oficial desde a versão 5.6. O seu comportamento foi testado no Kernel 5.11 e este protocolo apresentou-se muito instável, não sendo benéfica a sua utilização face à utilização de apenas uma ligação em cada momento. Foi testado também no Kernel 5.13 e desta vez o seu comportamento foi mais linear, verificando-se os benefícios da sua utilização. Ainda assim, este não foi incorporado na proposta final, dado que adiciona uma camada de complexidade à solução e não dá tanta liberdade na manipulação do tráfego como a utilização de diversas ligações geridas separadamente.

Tendo em conta a VPN escolhida e o facto de não ser utilizado MPTCP, das três propostas de arquitetura de comunicações, foi escolhida a de *routing* dinâmico, em que não existe agregação e existe uma camada de segurança por cada ligação móvel, isto é, é estabelecida uma instância de WireGuard por ligação.

Em termos de encaminhamento dos dados ao nível do Linux, foram estudados os seus mecanismos. Verificou-se que não era possível fazer um mapeamento direto entre valores DSCP dos pacotes recebidos e a tabela de encaminhamento que deveria ser analisada para o encaminhamento. Posto isto, recorreu-se às nftables para aplicar uma marcação `FWMARK` no tráfego, marcação essa que depende do DSCP dos pacotes recebidos. Com esta marcação, os mecanismos ip-rule do Linux permitem um mapeamento para uma tabela de encaminhamento definida.

Existindo a possibilidade de a *gateway* ser móvel, é relevante a obtenção das suas coordenadas geográficas, principalmente se a *gateway* for integrada num veículo que realiza percursos bem definidos. Através das coordenadas pode ser possível traçar uma relação com a qualidade de rede detetada e assim prever a qualidade de rede numa determinada localização nas restantes vezes que a *gateway* se encontrar na mesma. Desta forma, considerou-se para a proposta final que a *gateway* deve possuir um recetor GNSS.

Estudados e discutidos todos os componentes relevantes da *gateway* procedeu-se à concretização de uma aplicação capaz de recolher métricas passivas e ativas, de obter a localização atual, manipular o encaminhamento do tráfego e realizar decisões acerca do mesmo, consoante a avaliação realizada. Para a avaliação das redes foi também contemplada uma eventual correlação entre as medidas recolhidas num determinado momento, numa determinada localização e as medidas recolhidas anteriormente no mesmo local.

Para alocar adequadamente os recursos para cada aplicação, são definidos contratos por parte da *gateway*, em que é especificado qual o valor DSCP que a aplicação deve utilizar para ter, por parte da rede, determinadas condições, por exemplo, requisitos de latência. A aplicação, através da componente de decisão, vai verificar se tem de existir alguma alteração ao nível do encaminhamento para serem cumpridos os requisitos mencionados no contrato.

Com a construção da aplicação concluiu-se que é possível desenvolver um sistema capaz de manipular o encaminhamento de dados consoante a sua prioridade, para encaminhar o tráfego de forma resiliente, através de uma correta atribuição de prioridade a determinado tipo de dados. Consegue-se assim uma gestão eficiente dos recursos para atender às necessidades de cada aplicação ou serviço.

4.2 Trabalho Futuro

Em primeiro lugar é relevante realizar uma análise de protocolos e sistemas existentes no mercado que tenham objetivos similares aos aqui apresentados, isto é, que pretendam realizar o encaminhamento de dados de serviços críticos de forma resiliente.

Em segundo lugar será necessária uma maior análise da relação entre os parâmetros de qualidade de rede fornecidos pelos modems e a qualidade das ligações de dados estabelecidas.

Para o cálculo da qualidade das redes pretende-se ter em conta os diversos parâmetros de qualidade de rede retornados pelos modems. Há que encontrar quais os parâmetros

que são relevantes e que estão diretamente relacionados com o ritmo binário, latência e *jitter*. Para isso serão necessários mais testes com uma *gateway* real, estática ou em movimento.

Em terceiro lugar, apesar de ter sido possível obter diversas métricas, quer diretamente do Kernel como através do Iperf3, não foi construída uma fórmula geral de avaliação. Ainda, deve ser realizada uma análise mais aprofundada de quais as métricas recolhidas que são realmente relevantes e que reflitam a qualidade de cada rede. Aos parâmetros relevantes há que atribuir-lhes um peso com base na relação entre estes e a qualidade das ligações. Assim, constrói-se uma fórmula geral da qualidade da rede, composta pelos diversos parâmetros relevantes e pesos associados. Esta fórmula geral pode ser usada também para gerar um índice de qualidade geral.

Em quarto lugar é necessária a realização de testes aos módulos de obtenção de métricas dos modems e de obtenção de coordenadas geográficas. Para tal é necessário ter, novamente, acesso a uma *gateway* real com acesso a modems e com um sistema GNSS que suporte diversos sistemas de posicionamento e SBAS.

Em quinto lugar, apesar de terem sido definidos perfis de encaminhamento, estes devem ser revistos a partir do momento em que se coloquem em funcionamento os componentes de obtenção de métricas dos modems e de obtenção de dados geográficos. Devem ser revistos também cada vez que a fórmula de avaliação de redes é redefinida.

Por último, é necessário desenvolver o código que vai integrar a plataforma central, tanto para a receção das mensagens MQTT enviadas por uma ou mais *gateways*, como para a visualização gráfica das métricas recolhidas. Pode até ser desenvolvido um sistema mais complexo que permita um tratamento dos dados mais completo visando melhorar o suporte de apoio à decisão.

Referências

[1] Carlos Rodrigues & Nuno Cruz, "Avaliação do desempenho do MPTCP *upstream*", *INFORUM—Simpósio de Informática*, 2021.

[2] ——, "Avaliação de desempenho de tecnologias VPN atuais", *INFORUM—Simpósio de Informática*, 2022.

[3] R. Prasad, C. Dovrolis, M. Murray & K. Claffy, "Bandwidth estimation: metrics, measurement techniques, and tools", *IEEE Network*, vol. 17, n.º 6, páginas 27–35, 2003. DOI: 10.1109/MNET.2003.1248658.

[4] Venkat Mohan, Y. R. Janardhan Reddy & K. Kalpana, "Active and Passive Network Measurements : A Survey", 2011.

[5] Francesco Ciaccia, Ivan Romero, Oriol Arcas-Abella, Diego Montero, René Serral-Gracià & Mario Nemirovsky, "SABES: Statistical Available Bandwidth EStimation from passive TCP measurements", em *2020 IFIP Networking Conference (Networking)*, 2020, páginas 743–748.

[6] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi & Teunis Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm", vol. 27, n.º 3, 67–82, jul. de 1997, ISSN: 0146-4833. DOI: 10.1145/263932.264023. URL: https://doi.org/10.1145/263932.264023.

[7] Eduardo Miravalls-Sierra, David Muelas, Jorge López de Vergara Méndez, Javier Ramos & Javier Aracil, "On the Use of Affordable COTS Hardware for Network Measurements: Limits and Good Practices", *Information*, vol. 9, pág. 43, fev. de 2018. DOI: 10.3390/info9020043.

[8] Atsushi Miyamoto, Kazuho Watanabe & Kazushi Ikeda, "Packet loss rate estimation with active and passive measurements", em *Proceedings of The 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference*, 2012, páginas 1–4.

[9] Y. Tsang, M. Coates & R. Nowak, "Passive network tomography using EM algorithms", em *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, vol. 3, 2001, 1469–1472 vol.3. DOI: 10.1109/ICASSP.2001.941208.

[10] Steven M. Bellovin, "A Best-Case Network Performance Model", *ATT Research, Tech. Rep*, 1992.

[11] Allen Downey, "Using pathchar to estimate Internet link characteristics", *Computer Communication Review*, vol. 29, ago. de 1999. DOI: 10.1145/316188.316228.

[12] B. Melander, M. Bjorkman & P. Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks", em *Globecom '00 - IEEE. Global Telecommunications Conference. Conference Record (Cat. No.00CH37137)*, vol. 1, 2000, 415–420 vol.1. DOI: 10.1109/GLOCOM.2000.892039.

[13] Kevin Lai & Mary Baker, "Measuring Link Bandwidths Using a Deterministic Model of Packet Delay", *SIGCOMM Comput. Commun. Rev.*, vol. 30, n.º 4, 283–294, ago. de 2000, ISSN: 0146-4833. DOI: 10.1145/347057.347557. URL: https://doi.org/10.1145/347057.347557.

[14] Jacob Strauss, Dina Katabi & Frans Kaashoek, "A Measurement Study of Available Bandwidth Estimation Tools", *Proc. ACM SIGCOMM Conference on Internet Measurement*, out. de 2003. DOI: 10.1145/948205.948211.

[15] Vinay Ribeiro, Mark Coates, Rudolf Riedi, Shriram Sarvotham, Brent Hendricks & Richard Baraniuk, "Multifractal Cross-Traffic Estimation", nov. de 2000.

[16] Ningning Hu & Peter Steenkiste, "Evaluation and characterization of available bandwidth probing techniques", *Selected Areas in Communications, IEEE Journal on*, vol. 21, páginas 879 –894, set. de 2003. DOI: 10.1109/JSAC.2003.814505.

[17] Li Lao, Constantine Dovrolis & M.Y. Sanadidi, "The Probe Gap Model can underestimate the available bandwidth of multihop paths", *Computer Communication Review*, vol. 36, páginas 29–34, out. de 2006. DOI: 10.1145/1163593.1163599.

[18] Manish Jain & Constantine Dovrolis, "Pathload: A Measurement Tool for End-to-End Available Bandwidth", *Proceedings of Passive and Active Measurement Workshop*, mar. de 2002.

[19] Vinay Ribeiro, Rudolf Riedi, Jiri Navrátil & Les Cottrell, "pathChirp: Efficient Available Bandwidth Estimation for Network Paths", *Proceedings of Passive and Active Measurement Workshop*, abr. de 2003. DOI: 10.2172/813038.

[20] *Iperf - the ultimate speed test tool for TCP, UDP and SCTP*. URL: https://iperf.fr.

[21] Mohammad Fraiwan, "Overlay networks monitoring", jan. de 2008.

[22] S. Shalunov, B. Teitelbaum, A. Karp, J. Boote & M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)", RFC Editor, RFC 4656, set. de 2006.

[23] K. Hedayat, R. Krzanowski, A. Morton, K. Yum & J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC Editor, RFC 5357, out. de 2008.

[24] Cemal Kocak, "Performance Analysis of IP Network Using Two-Way Active Measurement Protocol (TWAMP) and Comparison with ICMP (Ping) Protocol in a Saturated Condition", nov. de 2016.

[25] Cemal Kocak & Kahraman Zaim, "Performance measurement of IP networks using Two-Way Active Measurement Protocol", em *2017 8th International Conference on Information Technology (ICIT)*, 2017, páginas 249–254. DOI: 10.1109/ICITECH.2017.8080008.

[26] Anup Kumar Paul, Atsuo Tachibana & Teruyuki Hasegawa, "NEXT: New enhanced available bandwidth measurement technique, algorithm and evaluation", em *2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*, 2014, páginas 443–447. DOI: 10.1109/PIMRC.2014.7136206.

[27] ——, "NEXT-FIT: Available Bandwidth Measurement over 4G/LTE Networks – A Curve-Fitting Approach", em *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, 2016, páginas 25–32. DOI: 10.1109/AINA.2016.24.

[28] Natsuhiko Sato, Takashi Oshiba, Kousuke Nogami, Anan Sawabe & Kozo Satoda, "Experimental comparison of machine learning-based available bandwidth estimation methods over operational LTE networks", em *2017 IEEE Symposium on Computers and Communications (ISCC)*, 2017, páginas 339–346. DOI: 10.1109/ISCC.2017.8024553.

[29] Claudio Casetti, M. Gerla, M.Y. Sanadidi & Ren Wang, "TCP Westwood: End-to-end Bandwidth Estimation for Efficient Transport over Wired and Wireless Networks", jan. de 2001.

[30] Ren Wang, M. Valla, M.Y. Sanadidi & M. Gerla, "Adaptive bandwidth share estimation in TCP Westwood", em *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, vol. 3, 2002, 2604–2608 vol.3. DOI: 10.1109/GLOCOM.2002.1189101.

[31] Zehang Chen, Yitong Liu, Yameng Duan, Hao Liu, Gang Li, Yami Chen, Junshuai Sun & Xin Zhang, "A novel bandwidth estimation algorithm of TCP westwood in typical LTE scenarios", em *2015 IEEE/CIC International Conference on Communications in China (ICCC)*, 2015, páginas 1–5. DOI: 10.1109/ICCChina.2015.7448600.

[32] Madhan Raj Kanagarathinam, Sukhdeep Singh, Irlanki Sandeep, Abhishek Roy & Navrati Saxena, "D-TCP: Dynamic TCP congestion control algorithm for next generation mobile networks", em *2018 15th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2018, páginas 1–6. DOI: 10.1109/CCNC.2018.8319185.

[33] Xianliang Jiang & Guang Jin, "CLTCP: An Adaptive TCP Congestion Control Algorithm Based on Congestion Level", *IEEE Communications Letters*, vol. 19, n.º 8, páginas 1307–1310, 2015. DOI: 10.1109/LCOMM.2015.2447541.

[34] Osamu Honda, Hiroyuki Ohsaki, Makoto Imase, Mika Ishizuka & Junichi Murayama, "Understanding TCP over TCP: effects of TCP tunneling on end-to-end throughput and latency", em *Performance, Quality of Service, and Control of Next-Generation Communication and Sensor Networks III*, Mohammed Atiquzzaman & Sergey I. Balandin, eds., International Society for Optics & Photonics, vol. 6011, SPIE, 2005, páginas 138 –146. DOI: 10.1117/12.630496. URL: https://doi.org/10.1117/12.630496.

[35] *OpenVPN official website*. URL: https://openvpn.net (acedido em 20/02/2022).

[36] Irfaan Coonjah, Pierre Clarel Catherine & K. M. S. Soyjaudah, "Experimental performance comparison between TCP vs UDP tunnel using OpenVPN", em *2015 International Conference on Computing, Communication and Security (ICCCS)*, 2015, páginas 1–5. DOI: 10.1109/CCCS.2015.7374133.

[37] Erik Dekker & Patrick Spaans, "Performance comparison of VPN implementations WireGuard, strongSwan, and OpenVPN in a 1 Gbit/s environment", 2020.

[38] Maximilian Pudelko, Paul Emmerich, Sebastian Gallenmüller & Georg Carle, "Performance Analysis of VPN Gateways", em *2020 IFIP Networking Conference (Networking)*, 2020, páginas 325–333.

[39] *strongSwan official website*. URL: https://strongswan.org (acedido em 20/02/2022).

[40] Jason Donenfeld, "WireGuard: Next Generation Kernel Network Tunnel", jan. de 2017. DOI: 10.14722/ndss.2017.23160.

[41] Lukas Osswald, Marco Haeberle & Michael Menth, "Performance Comparison of VPN Solutions", 2020.

[42] "IEEE Standard for Local and metropolitan area networks–Link Aggregation", *IEEE Std 802.1AX-2008*, páginas 1–163, 2008. DOI: 10.1109/IEEESTD.2008.4668665.

[43] D. Fedyk, P. Ashwood-Smith, D. Allan, A. Bragg & P. Unbehagen, "IS-IS Extensions Supporting IEEE 802.1aq Shortest Path Bridging", RFC Editor, RFC 6329, abr. de 2012.

[44] C. Hopps, "Analysis of an Equal-Cost Multi-Path Algorithm", RFC Editor, RFC 2992, 2000.

[45] David Allan, Peter Ashwood-Smith, Nigel Bragg, Janos Farkas, Don Fedyk, Michel Ouellete, Mick Seaman & Paul Unbehagen, "Shortest path bridging: Efficient control of larger ethernet networks", *IEEE Communications Magazine*, vol. 48, n.º 10, páginas 128–135, 2010. DOI: 10.1109/MCOM.2010.5594687.

[46] M. Umair, S. Kingston Smiler, D. Eastlake 3rd & L. Yong, "Transparent Interconnection of Lots of Links (TRILL) Transparent Transport over MPLS", RFC Editor, RFC 8385, 2018.

[47] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang & V. Paxson, "Stream Control Transmission Protocol", RFC Editor, RFC 2960, out. de 2000.

[48] J.R. Iyengar, P.D. Amer & R. Stewart, "Concurrent Multipath Transfer Using SCTP Multihoming Over Independent End-to-End Paths", *IEEE/ACM Transactions on Networking*, vol. 14, n.º 5, páginas 951–964, 2006. DOI: 10.1109/TNET.2006.882843.

[49] Hung-yun Hsieh & R. Sivakumar, "A Transport Layer Approach for Achieving Aggregate Bandwidths On Multi-homed Mobile Hosts", *Wireless Networks*, vol. 11, set. de 2002. DOI: 10.1007/s11276-004-4749-6.

[50] A. Ford, C. Raiciu, M. Handley, O. Bonaventure & C. Paasch, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC Editor, RFC 8684, mar. de 2020.

[51] Sunit Kumar Nandi, "MPTCP Performance with Various Configurations, Path Failures and Recovery", em *2018 3rd International Conference for Convergence in Technology (I2CT)*, 2018, páginas 1–6. DOI: 10.1109/I2CT.2018.8529504.

[52] Bo Han, Feng Qian, Shuai Hao & Lusheng Ji, "An anatomy of mobile web performance over multipath TCP", dez. de 2015, páginas 1–7. DOI: 10.1145/2716281.2836090.

[53] S. Barré et al. C. Paasch, *Multipath TCP in the Linux Kernel*. URL: http://www.multipath-tcp.org.

[54] *OpenVPN Cipher Negotiation*, https://community.openvpn.net/openvpn/wiki/CipherNegotiation.

[55] *strongSwan Security Recommendation*, https://docs.strongswan.org/docs/5.9/howtos/securityRecommendations.html.

[56] Si Thu Aung & Thandar Thein, "Comparative Analysis of Site-to-Site Layer 2 Virtual Private Networks", em *2020 IEEE Conference on Computer Applications(ICCA)*, 2020, páginas 1–5. DOI: 10.1109/ICCA49400.2020.9022848.

[57] maliubiao, *python native library for network device*, https://github.com/maliubiao/python_netlink, 2014.