UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

HUGO CHRIST VILELA

TCPBP: Enhancing the TCP Throughput in *High-Speed Lossy Networks* Based on a Single Mixed Routing Metric

CURITIBA

2022

HUGO CHRIST VILELA

TCPBP: ENHANCING THE TCP THROUGHPUT IN HIGH-SPEED LOSSY NETWORKS BASED ON A SINGLE MIXED ROUTING METRIC

TCPBP: Aprimorando a Vazão TCP em Redes de Alta Velocidade com Perdas Através de uma Métrica de Roteamento Mista Simples

Dissertação apresentada como requisito à obtenção do título de Mestre em Ciências do Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial da Universidade Tecnológica Federal do Paraná (UTFPR).

Orientador(a): Prof(a). Dr(a). Keiko Verônica Ono Fonseca

CURITIBA

2022

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Curitiba

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

HUGO CHRIST VILELA

TCPBP: ENHANCING THE TCP THROUGHPUT IN HIGH-SPEED LOSSY NETWORKS BASED ON A SINGLE MIXED ROUTING METRIC

Trabalho de pesquisa de mestrado apresentado como requisito para obtenção do título de Mestre Em Ciências da Universidade Tecnológica Federal do Paraná (UTFPR). Área de concentração: Telecomunicações E Redes.

Data de aprovação: 26 de Setembro de 2022

Dra. Keiko Veronica Ono Fonseca, Doutorado - Universidade Tecnológica Federal do Paraná

Dr. Carlos Marcelo Pedroso, Doutorado - Universidade Federal do Paraná (Ufpr)

Dra. Juliana De Santi, Doutorado - Universidade Tecnológica Federal do Paraná

Documento gerado pelo Sistema Acadêmico da UTFPR a partir dos dados da Ata de Defesa em 22/11/2022.

“The fate of the universe will be decided as it should be: in Mortal Kombat.”

– Elder Gods

Acknowledgments

First and foremost, I would like to thank my family for their continuous support in all aspects of my life, this thesis included. In particular, this goes to my wife, Marina Chiara, and my daughter, Aurora. I would never consider stepping back into the university if it was not for them, to make them proud of me.

My family, besides some names not worth mentioning, also includes (not surprisingly) my mother, Zilca; my father, Nicácio; my sisters, Gabriela and Marina; and my (unfortunately...) deceased grandparents, Sebastião and Paulina, so I have to acknowledge them, too: thank you very much for being such a great inspiration throughout my life since I was a baby[1].

A special acknowledgment, before I forget, also goes to Rocco, my dog, and to Adam Granduciel (*The War On Drugs* frontman) for providing most of the soundtrack of the writing of this thesis.

The rest of the acknowledgments goes to to the following people, not necessarily in order of importance: my friends, in particular, Fernando Albino (roughly 30 years of friendship); my mother-in-law and father-in-law, Marinês and Darci, for the kindness and support in diverse aspects, and also for being exceptional grandparents; my colleagues and managers from old Copel Telecomunicações, especially those from the Network Operation Center (NOC), for the support in this journey and technical advises that helped to shape the way I perceive computer networks; the UTFPR staff (professors, technicians, and employees in general) – you were fundamental; finally, my advisor, professor Keiko, for the kindness and for giving me freedom on most aspects of the research developed.

Kindly,
Hugo.

[1] I would like to acknowledge Reviewer 2 for pointing out that I was born before my sisters, so saying "*...since I was a baby.*" is factually incorrect.

Resumo

Neste trabalho é apresentado o *Transmission Control Protocol Best Path* (TCPBP), uma métrica de roteamento alternativa que visa maximizar a vazão TCP agregada em redes intradomínio de alta velocidade com perdas. Obtém-se desempenho aprimorado ao se combinar atraso e perda de pacotes em uma única métrica mista, como uma função-parâmetro. Em contraste, esta função prevê a vazão TCP por fluxo que deve ser maximizada. Espera-se, com isto, que a maximização da taxa de transferência de cada fluxo TCP que atravessa uma determinada rota deva melhorar a taxa de transferência agregada (*i.e.*, a soma de todos os fluxos TCP).

Para demonstrar a viabilidade da nossa abordagem na obtenção de rotas viáveis em qualquer topologia representada por um grafo direcionado $G(V,E)$, provamos que a métrica satisfaz as propriedades matemáticas de isotonicidade e monotonicidade. Este fato implica sua consistência, otimicidade e ausência de *loops*, *i.e.*, o que fornece rotas viáveis. Para o cômputo do caminho mais curto, desenvolvemos um processo de relaxamento de arestas modificado, adequado para métricas não-aditivas, com custos positivos nas arestas, utilizando o algoritmo de Dijkstra.

Como prova de conceito, executou-se testes de vazão TCP em uma topologia SDN emulada em Linux, com dez nós, empregando-se para tal um protocolo de roteamento intradomínio de estado de enlace (ou seja, tal qual o OSPF), capaz de executar tanto o TCPBP quanto as métricas tradicionais de atraso e número de saltos. Tal topologia foi escolhida devido ao potencial de melhorar sua vazão TCP com seleção de rotas, dadas as características da própria topologia e do modelo de previsão de vazão TCP adotado.

O tempo de execução do cômputo do caminho mais curto no plano de controle não se mostrou significativamente mais lento do que o algoritmo Dijkstra padrão tradicionalmente usado com métricas aditivas padrão (como número de saltos). Além disso, o desempenho geral da vazão TCP obtida com o TCPBP foi maior do que com diferentes rotas calculadas com métricas tradicionais, sem impacto na justiça TCP, mostrando assim o potencial de nossa proposta.

Palavras-chave: Performance TCP, redes de alta velocidade com perdas, QoS, métricas de roteamento, métricas mistas simples.

Abstract

This work presents the *Transmission Control Protocol Best Path* (TCPBP), an alternative routing metric that aims to maximize the aggregated single-path TCP throughput on intradomain *High-Speed Lossy Networks*. The enhanced performance is achieved by combining delay and packet loss on a single mixed metric as a function parameter which predicts the *per-flow* TCP throughput that should be maximized. Maximizing the throughput of each TCP flow traversing a given route is also expected to improve the aggregated throughput (*i.e.*, the sum of all TCP flows).

To demonstrate the feasibility of our approach concerning obtaining viable routes in any topology represented by a $G(V,E)$ directed graph, we proved that the metric satisfies the mathematical properties of isotonicity and monotonicity. This fact implies that it is consistent, optimal, and loop-less, *i.e.*, it is feasible regarding the shortest path computation, thus providing viable routes. For the shortest path computation, we devised a modified edge relaxation process suitable for non-additive metrics with positive edge costs in the Dijkstra algorithm.

As proof of concept, TCP bandwidth tests were executed in an SDN *High-Speed Lossy Network* Linux-emulated topology with ten nodes. For the routing algorithm, we employed an intradomain link-state routing protocol (*i.e.*, OSPF-like) capable of running both TCPBP and traditional metrics delay and *hop-count*. Such topology was chosen because of the potential to improve its TCP throughput with route selection, given the characteristics of the topology itself and the TCP throughput prediction model adopted.

The shortest path computation running time in the control plane proved not significantly slower than the default Dijkstra algorithm traditionally used with standard additive metrics (such as *hop-count*). Also, the overall TCP throughput performance obtained with TCPBP was greater than with different routes founded by traditional metrics, without impact on the TCP fairness, thereby showing the potential of our proposal.

Keywords: TCP performance, *High-Speed Lossy Networks*, QoS, routing metrics, single mixed metrics.

List of Figures

List of Tables

List of Algorithms

List of Symbols and Acronyms

BW_{max} Maximum Bandwidth

n_f Number of TCP Flows

RTT Round-Trip Time

$cwnd$ Congestion Window

d One-Way Delay

p Packet Loss

ABW Available Bandwidth

ACK Acknowledgment

AIMD Additive Increase/Multiplicative Decrease

AQM Active Queue Management

ARGAQ Adaptive Routing Based on Genetic Algorithm for QoS

BGP Border Gateway Protocol

CPU Central Processing Unit

CSMA/CA Carrier Sense Multiple Access with Collision Avoidance

CSMA/CD Carrier Sense Multiple Access with Collision Detection

DBLCLH Delay, Bandwidth, and Loss Probability Constrained Least Hop

DCCP Datagram Congestion Control Protocol

DT Delay Time

DV Distance Vector

DVPTCP Dynamic Virtual Parallel TCP

ECMP Equal-Cost Multi-Path Routing

ECTCP Explicit Centralized Congestion Avoidance Mechanism for TCP

EWMA Exponentially Weighted Moving Average

FMM Fuzzy Mixed Metric

Gbps Gigabits per Second

GW Network IP Gateway

HS-TCP High-Speed TCP

HTE Hypothesis Testing Estimator

ICMP Internet Control Message Protocol

IoT Internet of Things

IP Internet Protocol

IP/MPLS Internet Protocol/Multi Protocol Label Switching

IPv4 Internet Protocol Version 4

IPv6 Internet Protocol Version 6

IS-IS Intermediate System to Intermediate System

ISP Internet Service Provider

LS Link-State

LSA Link-State Advertisement

LUR Link Utilization Ratio

MAC Media Access Control

MANETs Mobile Ad Hoc NETworks

MCR Multi-Constrained Routing

MPTCP MultiPath TCP

MSS Maximum Segment Size

OSPF Open Shortest Path First

OSPFv2 Open Shortest Path First Version 2

OSPFv3 Open Shortest Path First Version 3

OTCP Omniscient TCP

OVS Open vSwitch

PLMBSF Packet Loss Measurement Based on Sampled Flow

QoS Quality of Service

RACC Receiver Assistant Congestion Control

RED Random Early Detection

RFC Request for Comments

RMON Remote MONitoring

SACK Selective Acknowlegment

SARA Simulated Annealing

SCTP Stream Control Transmission Protocol

SDN Software-Defined Networking

SDTCP SDN-Based TCP Congestion Control

SLA Service Level Agreement

SMA Simple Moving Average

SMM Single Mixed Metric

SMM-DV Single Mixed Metric - Distance Vector

SMM-LR Single Mixed Metric with Lagrange Relaxation

SMM-LS Single Mixed Metric - Link-State

SNMP Simple Network Management Protocol

SSSP Single-Source Shortest Paths

TCP Transmission Control Protocol

TE Traffic Engineering

TP Expected Throughput Per-Flow

TSR Transmission Success Rate

UDP User Datagram Protocol

WLAN Wireless Local Area Network

WMN Wireless Mesh Networks

Contents

1 | Introduction

This Chapter introduces our proposal and is divided into the categories of motivation (contextualization of the problem), contribution (the proposal itself and a summary of the proof of concept), and text outline.

1.1 Motivation

Transmission Control Protocol (TCP) is a Transport Layer protocol (TCP/IP layer 4) that runs on top of IP (Internet Protocol; TCP/IP layer 3). It provides reliable communication between endpoints in a best-effort infrastructure computer network, given that the paths in the network are not presumed as reliable and do not provide a guarantee of the delivery of a packet to its destination, nor provide flow control, end-to-end congestion control, acknowledgment, re-transmission, error recovery and other correlated functionalities as detailed initially in RFC 793 and its subsequent updates (KUROSE; ROSS, 2010).

Despite the recent Google QUIC rapid expansion (which runs on top of UDP) and the emergence of new transport protocols such as SCTP and DCCP (POLESE et al., 2019), TCP is still the most used transport protocol on the internet. So, its performance, especially regarding raw data transfer capacity, *i.e.*, throughput, is a critical subject and primary object of concern to a lot of researchers in the field of computer networks. When it comes to TCP throughput performance, the maximum bandwidth achieved by a single TCP stream (hereby denoted as "BW_{max}") can be stated as a function of round-trip delay, packet loss, and available bandwidth (this holds even if the network is not lossy; we discuss it in detail in Section 2.5.3: TCP Throughput Prediction). On the other hand, in charge of defining paths between nodes in the network, there are routing protocols such as OSPF (intradomain routing) and BGP (Border Gateway Protocol; interdomain routing) (KUROSE; ROSS, 2010), in which routes are selected based on static metrics such as *hop-count*, delay and reference bandwidth (as in OSPF), or *as-path* length (the case of BGP).

As a result of this rigidness, for example, Internet Service Providers (ISPs) backbones often do not provide their customers routes that optimize TCP throughput; *e.g.*, even if packet loss and delay as Service Level Agreement (SLA) parameters are met, some of the routes do not have the best possible combinations available of delay and packet loss for TCP, which limits performance and possibly impairs backbone capacity of throughput delivery to the applications. In fact, even sophisticated QoS routing metrics may fail to provide the best paths for TCP throughput simply because they were not designed to do so. This effect is especially true in the case of *High-Speed*

Lossy Networks because the possible routes between any pair of nodes in such topology may differ significantly in their combined values of delay and packet loss.

1.2 Thesis Contribution

In this thesis, we propose and analyze the utilization of Transmission Control Best Path (hereby denoted as "TCPBP"), a single mixed routing metric designed to be an alternative to the usually implemented metrics in classic intradomain routing protocols, such as OSPF (MOY, 1998) and IS-IS (SHAND; GINSBERG, 2014), that maximizes the function of delay and packet loss that predicts the TCP throughput in *High-Speed Lossy Networks*: the basic intuition of such approach is to simply choose the route that is expected to provide the highest *per-flow* throughput according to the TCP prediction formula.

In a nutshell, the use of the TCPBP metric aims to maximize single-path TCP throughput on IP networks with the occurrence of a sufficiently large TCP transfer (*steady-state* scenario), *i.e.*, a data transfer large enough so that slow start phase throughput of *additive increase/multiplicative decrease* (AIMD) TCP congestion control can be neglected in the average throughput computation. With this approach, we hypothesize that ruling out possible bandwidth constraints and limitations of the TCP throughput prediction model, the aggregated TCP performance (*i.e.*, the aggregated throughput of all TCP flows in the route) can be drastically improved in some of the routes of some network topologies (such as *High-Speed Lossy Networks*) using a single mixed routing metric based on a particular proportion of delay and packet loss such that aforementioned BW_{max} is maximized.

Even though TCPBP, because of its name, suggests it is based on a cross-layer design approach, it is not a cross-layer metric. This is so because there is no direct interaction between the OSI layers (Transport Layer and the Network Layer, in this case), which is a fundamental feature to classify any approach as cross-layer one (KAWADIA; KUMAR, 2005). That is to say that the metric does not use any information specifically provided by the TCP functionalities (TCP control information like **ACK** packets, or congestion window size, for example).

To evaluate TCPBP throughput improvement, we have opted for an emulated topology in the Linux kernel that resembles a *High-Speed Lossy Network*. We justify this choice based on the recent emerging scenario where the crescent diversification of network technologies and application requirements results in sub-par performance of standard TCP implementations on large high speed and lossy networks (*e.g.*, heterogeneous topologies with 4G/5G and transcontinental optical high speed links) (SU et al., 2019; SHI et al., 2009). From the chosen test scenarios,

result analysis was discussed, as well as our proposal's contributions and possible limitations.

1.2.1 List of Publications

The research conducted that gave birth to this thesis also produced the paper entitled "TCPBP: Enhancing the TCP Throughput in *High-Speed Lossy Networks* Based on a Single Mixed Routing Metric" (VILELA; V.O FONSECA; FONSECA, 2022) accepted at the Main Track on SBRC 2022.

1.3 Text Outline

The remainder of this thesis is organized as follows. Chapter 2 presents the underlying theory and network concepts necessary to understand our proposal. This includes mainly the concept of routing metrics and their fundamental mathematical properties (Sections 2.2 and 2.3), Single-Source Shortest Paths (SSSP) computation principles and algorithms (Section 2.4), Transmission Control Protocol (TCP) fundamentals, with a focus on the TCP throughput prediction model employed (Section 2.5), *High-Speed Lossy Networks* (Section 2.6), and link estimation concepts for the accounting of delay and packet loss on computer networks (Section 2.7).

Chapter 3 presents our proposal, *i.e.*, the TCPBP intuition (Section 3.1), its most relevant aspects regarding the feasibility of the approach, such as the metric mathematical properties (Section 3.2), shortest path computation mechanism (Section 3.3), $f(d,p)$ computation (Section 3.4), route selection mechanisms (Section 3.5), the expected throughput improvement of the metric employment (Section 3.6), and security considerations (Section 3.7).

Chapter 4 presents a proof-of-concept of the TCPBP on a Linux emulated testbed with ten nodes in a scheme that resembles a *High-Speed Lossy Network*. The metric design for such is detailed (Section 4.1), and the parameters of throughput, fairness, and algorithm running time execution of TCPBP are compared to traditional metrics *hop-count* and delay, with the results being discussed in Section 4.4.

Chapter 5 presents the literature related to our proposal in the categories of QoS Routing Metrics (Section 5.1), Multi-Constrained Routing (MCR) (Section 5.2), Link-State Qos Routing (Section 5.3), Single Mixed Metrics (Section 5.4), TCP performance (Section 5.5), and practical uses of the TCP prediction model employed in our proposal (Section 5.6).

Finally, in Chapter 6, we present and summarize the contributions and considerations about our work, including relevant aspects that were not covered or briefly covered, being, thus, ideas

for future work (which are listed in Section 6.1).

2 | Preliminaries

In this Chapter, we present the underlying theory related to and in the context of the development of our proposal. For the sake of readability, the exposition is not exhaustive, being the reader occasionally referred to additional literature. Additionally, we presume that the reader is familiarized with common concepts of computer networks and terminology as stated, for example, in (KUROSE; ROSS, 2010). The presentation is organized as follows:

2.1 Conventions and General Concepts

We begin the preliminaries by introducing conventions and general concepts as follows: Software-Defined Networking (SDN), topology, routes and route terminology, route symmetry, and algorithm complexity.

2.1.1 Software-Defined Networking

Software-Defined Networking (SDN) is a network paradigm that aims to centralize network control decisions in one main component (*i.e.*, the SDN controller) by decoupling the forwarding devices of network datagrams (Data Plane) from the routing process control decision (Control Plane). Terminology and concepts regarding SDN are as described in (KREUTZ et al., 2014).

2.1.2 Topology

The network can be described by a directed graph $G(V,E)$, where V is the set of nodes in the network and E is the set of links interconnecting these nodes. For mathematical representation, we have adopted the adjacency matrix format as in *AdjMatrix* (below):

$$AdjMatrix = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1j} \\ a_{21} & a_{22} & \cdots & a_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} \end{bmatrix}$$

Where matrix element a_{ij} corresponds to the link-state value of two interconnected forwarding devices i and j. If there is no connection between i and j, or $i = j$, we set a_{ij} with inf (infinite)[2].

[2]In fact, if $i = j$, the correct value in the matrix diagonal would be zero, which represent the route cost from one router to itself; but, since we are not calculating paths when $i = j$, the representation stated suffices.

This representation may be an avoidable waste of memory, especially in large and sparse networks (as put in (BARABÁSI, 2013): "real networks are sparse"), but as memory consumption is not an issue in our simulations, and being this also a rather didactic and convenient way to represent a graph, the choice is justified. In production environments, if necessary, one can consider other types of representation, such as an adjacency list. It is not our intent to discuss data structures in algorithms (in special, representation of graphs), but, for instance, a comparison between adjacency matrix and adjacency list can be found in (CORMEN et al., 2009).

2.1.3 Routes

Given two nodes, like A and D, of a $G(V,E)$ graph, we denote as $A \Rightarrow D$ the route from A to D (routes are indicated with the symbol $\Rightarrow$). If the set of hops of the same route is $[A,B,C,D]$ (*i.e.*, A to B to C..., and so go on), we denote it as $A \rightarrow B \rightarrow C \rightarrow D$ (links are indicated with the symbol $\rightarrow$).

2.1.4 Route Symmetry

In general, we consider the possibility of asymmetric routing: that is, the route from source to destination possibly does not consist of the same hops as in the route from destination to source, which implies an asymmetric matrix in the topology representation. Given two adjacent nodes (name it "A", and "B"), link parameters may be different in each direction ($A \rightarrow B$ or $B \rightarrow A$); this way, for each particular topology, one asymmetric adjacency matrix is needed per parameter. Also, because of this asymmetry, we have a total of $(n^2 - n)$ routes to compute in a network with n nodes.

If there is symmetry, in the case of an SDN routing protocol, we need to compute $(n^2 - n)/2$ one-way paths only, given that the path in the way back is the same and all path computations are performed in a logically centralized Control Plane, but RTT and p still must be computed considering round-trip paths to permit a direct comparison of the transport layer protocol performance in the case of TCP.

2.1.5 Algorithm Complexity

To present algorithm complexity throughout this work, we have adopted the *Big O* notation as defined by Donald Knuth (KNUTH, 1976). It is basically a mathematical jargon that refers to the asymptotic behavior of a function when its input size grows to a particular value or infinity.

Formally, it is represented as $f(n) = O(g(n))$ if $\exists k > 0\, \exists n_0\, \forall n > n_0 \colon |f(n)| \leq k \cdot g(n)$. Since $O(g(n))$ does not mean that the complexity is proportional to $g(n)$ for all values of n (especially small ones), rather only putting an upper bound on the growth rate of the function, it is not used to directly and unrestrictedly compare algorithm performance, which is particularly clear in the case of shortest path computation with traditional metrics in relatively small graphs that represent typical intradomain networks (see, for example, footnote 4 on page 22).

Nevertheless, quite often it is correctly employed to show that some algorithm is scalable in the face of other alternatives. For example: (1) an $O(n!)$ complexity grows extremely fast in comparison to $O(n^2)$, rendering the former unusable for relatively small values of n, or (2) the asymptotic complexity of the routing algorithm also depends on a high number of flows in the network, as is the case of (PASSOS, 2013).

2.2 Routing Metrics

Routing metrics are parameters used by the Control Plane of a network to determine the best (or shortest[3]) path between two given nodes in a network. In this work, specifically, we define routing metric as the parameter used to compute the shortest path in a network represented by a directed graph $G(V,E)$ (WHITE; SLICE; RETANA, 2005). Perhaps, the simplest example of a routing metric is *hop-count*, in which the best path between two nodes in a network is the one with the lowest possible number of hops. Of course, since *hop-count* does not take into consideration other key factors of the network, quite often it selects poor paths regarding, for example, delay or bandwidth capacity.

Therefore, considering technical requirements of the applications exchanging data in the network, the nature of network (*e.g.*, there is a whole research field dedicated to routing metrics for *Wireless Mesh Networks* (WMNs), due to peculiarities of the wireless media in such cases (WANG; YAN, et al., 2020)), *Service Level Agreement* (SLA) parameters, or simply bearing in mind that end user *Quality of Experience* (QoE) may be improved, other metrics may be desirable, such as (but not limited to): *bandwidth* (the available capacity at a given instant, or the total capacity of the path), *delay* (latency/jitter of the path), *reliability* (factors like bit-error rate, or link availability), *load* (*e.g.*, traffic load over a specific link, or router CPU utilization), and *packet loss* (the probability of a packet getting lost in the path) (KENYON, 2002). Such metrics are also called "QoS metrics", a topic discussed in Section 5.1.

Due to distinct characteristics that determine how the shortest path should be computed,

[3]We make the distinction between "best" and "shortest" explicit only when it might otherwise be unclear from the context.

metrics can be formally put into three categories: *additive* (i), *multiplicative* (ii), and *concave* (iii) (KENYON, 2002). Given that $[c_1,c_2,...,c_n]$ is the set of metric values per-hop across any path, the path cost is calculated as below:

i. *additive*: path cost is based on the sum of the costs across the path, *i.e.*, it is given by $c_1 + c_2 + ... + c_n$. *Delay* and *hop-count* are examples of additive metrics;

ii. *multiplicative*: path cost is based on the product $c_1 \times c_2 \times ... \times c_n$. *Packet loss* is an example of a multiplicative metric;

iii. *concave*: path cost is the minimum value of the links in the path: $min[c_1,c_2,...,c_n]$. *Available bandwidth* is the most common concave metric.

Having a number of possibilities regarding metrics, as stated above, the control plane can determine the path using one specific metric or a combination of metrics: in either case, usually, the metric (or combination of metrics) is treated like a cost, and the path with the best cost is selected as the best path. In this sense, if appropriate, the task of finding the best paths in a computer network may be interpreted as an *optimization problem* that can be tackled with a myriad of techniques, such as dynamic programming or greedy methods.

To find the shortest paths, control plane employs a *shortest path calculation algorithm* such as the Dijkstra algorithm (or specialized variations of it), or the Bellman-Ford algorithm. The choice, here, may depend on a set of factors as described in (YANG; WANG, 2008). In the context of our work, because the proposed metric in Chapter 3 is showcased with a link-state routing protocol running in a central hardware, we have opted for the Dijkstra algorithm.

2.3 Properties of Routing Metrics

To be considered feasible, *i.e.*, to ensure that the requirements for the proper operation of the routing protocol regarding shortest path computation are met, the routing protocols and metrics must satisfy the conditions of loop-freeness, consistency, and optimality. In the general case, it is proven in (SOBRINHO, 2001) that isotonicity and monotonicity, though not always necessary (YANG; WANG, 2008), are sufficient properties that ensure routing protocols satisfy all three aforementioned conditions. Isotonicity, monotonicity, consistency, loop-freeness, and optimality are described in the following Subections as defined in (YANG; WANG, 2008).

2.3.1 Isotonicity

Left-isotonicity basically means that the order relation between the weights of any two paths is preserved if both of them are prefixed by a common third path; right-isotonicity, conversely, states that the order relation between the weights of any two paths is preserved if both of them are appended by a common third path. Therefore, isotonicity implies that both left-isotonicity and right-isotonicity apply to the network.

2.3.2 Monotonicity

Left-monotonicity means that the weight of a path does not decrease when prefixed by another path; right-monotonicity means that the weight of a path does not decrease when appended by another path. Thus, monotonicity implies that both left-monotonicity and right-monotonicity apply to the network.

2.3.3 Consistency

Consistency property states that a consistent routing requires that the packet forwarding decisions made by all nodes along the route must be consistent with each other. Simply put, considering any route $[h_1,...,h_n]$ in a network, where $[h_1,...,h_n]$ represents the vector containing the hops in the path, any sub-sequence of the vector is also a shortest path. For example, in the sequence $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E$, the route from A to C is $A \rightarrow B \rightarrow C$, from B to D is $B \rightarrow C \rightarrow D$, and so on.

2.3.4 Loop-Freeness

A routing protocol is defined as loop-free if, after its convergence, and for any directed graph $G(V,E)$, it does not forward any packet along a cycle. Consequently, a metric *per se* must not ensue the computation of paths containing loops, and, given any metric, a proper shortest path algorithm must be employed to find loop-free paths (for example, see footnote 3 on Section 2.4.2).

2.3.5 Optimality

A routing protocol is said to be optimal if packet forwarding between all pairs of nodes in a network traverses the shortest (lightest) routes regarding the metric itself.

2.4 Single-Source Shortest Paths (SSSP)

A *shortest path calculation algorithm* is the element in charge of finding the paths between every distinct pair of nodes in a graph such that the cost of the found paths is optimal, *i.e.*, each path has the lowest (or best) cost. A Single-Source Shortest Path (SSSP) is an algorithm that finds the shortest paths from a single node to all other nodes in a graph.

In the remainder of this Section, we present the concept of edge relaxation (Section 2.4.1), because it is the main element that we propose a modification to suit our purposes; Dijkstra algorithm (Section 2.4.2), because it is the SSSP algorithm of choice to showcase our metric; and Yen's algorithm for k-shortest path calculation (Section 2.4.3).

2.4.1 Edge Relaxation

We adopt the concept of edge relaxation as described in (CORMEN et al., 2009). Putting it simply, in the process of shortest path calculation, edge relaxation is checking if the current cost estimate of $a \rightarrow b$, two adjacent nodes, can be lowered if we divert the path via a third node (call it c, also adjacent to a and b), where this cost would be the aggregate cost of $a \rightarrow c$ and $c \rightarrow b$: if the cost of $a \rightarrow c \rightarrow b$ is lower than of $a \rightarrow b$, the cost estimate is updated. The concept is based on the triangle inequality principle of the euclidean geometry (HEATH et al., 1956). If isotonicity and monotonicity apply, it follows that the shortest path between all nodes in a graph can be found with a finite number of successive edge relaxation operations.

2.4.2 Dijkstra Algorithm

Dijkstra's algorithm (DIJKSTRA et al., 1959), in its original form (default Dijkstra), is an SSSP *algorithm* based on a greedy approach of the triangle inequality principle, by means of edge relaxation, and used in a wide range of applications, provided that the path cost is the sum of the costs of the edges in the path (additive metric) and there is no negative[4] values as edge costs.

[4]This ensures that the shortest path is loopless; if negative costs are allowed, the shortest path may not be loopless, and Dijkstra is not guaranteed to find the shortest path. In such a case, algorithms such as the Bellmann-

Nevertheless, Dijkstra principle, by means of a modification in the edge relaxation process, can be extended to compute shortest paths without additive metrics, as will be shown in Section 3.3.

Dijkstra is a popular choice in general because it is *optimal* (*i.e.*, guaranteed to find the shortest paths), *uninformed* (used when there is no prior knowledge of the network topology), and has *polynomial running time complexity*, being, in fact, the fastest known shortest path calculation algorithm for arbitrary graphs with positive-only edge costs (time complexity of Fibonacci heap min-priority queue implementation of Dijkstra to a $G(V,E)$ graph is $O(E + VlogV)$)[5]. The pseudocode and other details of default Dijkstra, such as the proof of correctness, can be easily found, as in (CORMEN et al., 2009).

2.4.3 Yen's k-Shortest Path Algorithm

A k-shortest path algorithm does not stop after finding the shortest route, but keep searching until it finds k different routes succeeding each other in cost. Yen's k-shortest Path Algorithm (or simply "Yen's algorithm") (YEN, 1971) was created by Jin Y. Yen in 1971, and employs any shortest path algorithm to find the shortest loopless path, then proceeds to find the other $k-1$ shortest loopless paths. We have opted for Yen's work, firstly, for its simplicity: a relatively short and comprehensive pseudocode whose implementation in a wide range of programming languages can be easily found on the Web. Also, without loss of generality, if the complexity of the shortest path algorithm is $O(f(V,E))$, the complexity of the k-shortest path algorithm is $O(kV(f(V,E)))$. Given that its time complexity grows linearly to the proportion of k, it is a fast and stable choice for intradomain routing protocols. Since Dijkstra algorithm Fibonacci heap time complexity to a $G(V,E)$ graph is $O(E + VlogV)$, Yen's complexity with Dijkstra is $O(kV(E + VlogV))$.

2.5 Transmission Control Protocol (TCP)

2.5.1 TCP Basics

Transmission Control Protocol (TCP; RFC 793 (POSTEL et al., 1981) and its subsequent updates) is a connection-oriented Transport Layer protocol that defines a set of rules to deal with unreliable IP packet switching networks infrastructure. Opposed to UDP, TCP provides a reli-

Ford should be used.

[5]Despite having the best asymptotic worst-case complexity, the Fibonacci heap implementation T_n has big constants for lower order terms, thus implying a slower time execution for small networks; therefore, it is more common to employ a Binary-heap $O(ElogV)$ instead.

able, connection-oriented service: by employing a logical connection (with a process commonly known as *three-way handshake*), acknowledgment of received packets, and other mechanisms such as *congestion control* and flow control, the delivery of a packet is ensured by an end-to-end process communication between the two remote endpoints that are exchanging application-related information in an IP-based network, even if packet re-transmission due to packet loss is needed. Thus, in this sense, TCP makes reliable communication possible over unreliable infrastructure.

To prevent congestion, a feedback control mechanism with the general principle of *additive-increase/multiplicative-decrease* (AIMD) is employed by standard TCP, as follows: upon *three-way handshake*, data transfer starts and its transmission rate is increased in an additive manner (with an additive factor), probing for available end-to-end data rate capacity until loss (caused by congestion, or not) happens; in the face of loss, the transmission rate is decreased by a multiplicative factor. Therefore, *congestion control* regarding an additive parameter α (usually, a received acknowledgement – **ACK**), and a multiplicative parameter β (loss indicator – **LOSS**) can be summarized in (i) and (ii), where *congestion window* ($cwnd$) is a TCP parameter that states the amount of data in a given instant that can be sent to the destination before receiving an acknowledgement:

$$\textbf{ACK} : cwnd \leftarrow cwnd + \alpha \tag{i}$$

$$\textbf{LOSS} : cwnd \leftarrow cwnd \times \beta \tag{ii}$$

Putting it simply, for standard TCP, $cwnd$ is reduced in the face of a **LOSS** event ($0 < \beta < 1$), and increased in the face of an **ACK** event ($\alpha > 0$). Although more recent TCP implementations, such as CUBIC (HA; RHEE; XU, 2008), employ different mechanisms to increase and decrease the $cwnd$, the principle of reacting under **ACK** and **LOSS** events is the same. A detailed account on how TCP works is out of the scope of this work and can be found, for example, in (FALL; STEVENS, 2011) and (COMER, 2014); in relation to our work, because of its direct relationship with the throughput delivery capacity in the face of delay and (especially) packet loss, *congestion control* mechanism of TCP is a key factor.

2.5.2 TCP Fairness

In this work, we adopt the concept of fairness which refers to the distribution of bandwidth utilization per-flow of each TCP connection (*i.e.*, fairness between competing TCP flows) (FLOYD et al., 2008). Basically, in the case of a best effort IP network, one main concern is that bandwidth as a resource is shared with fairness between all TCP flows. Considering the simplest

case where TF_n continuous TCP flows of n TCP connections share the same bottleneck link, fairness states that each flow should get $1/n$ of the link capacity, which is equivalent to say that the standard deviation of the set of values representing the throughputs should be very close to zero. The scenario, of course, can get more complicated if we consider that multiple flows with different bandwidth constraints and *RTT* not only exist but are common.

To evaluate the fairness of TCP flows quantitatively, we can employ Jain's Fairness Index (JAIN; CHIU; HAWE, et al., 1984), which is a common standard for evaluating resource allocation fairness in computer systems. The formula for the computation of the index is stated in Equation (2.1), where TF represents a TCP flow, and n is the number of flows:

$$J(TF_1,TF_2,\ldots,TF_n) = \frac{(\sum_{i=1}^{n} TF_i)^2}{n \cdot \sum_{i=1}^{n} {TF_i}^2} \tag{2.1}$$

In our work, the topic is important because the adoption of different routing metrics can impact the fairness of reliable data transfer transport protocols such as TCP.

2.5.3 TCP Throughput Prediction

Predictions of *steady-state* TCP throughput can be achieved based on a function of packet loss and delay (in the general case, $f(d,p)$) for most AIMD-based TCP implementations (such as Tahoe, Reno, New Reno etc.), with accurate experimental results with packet loss from $p \approx 10^{-5}$ up to $p \approx 0.05$ (5%) (MATHIS et al., 1997; PADHYE et al., 1998). Though the prediction model presented in (PADHYE et al., 1998) is more accurate, extending Mathis et al. work by taking into consideration more factors (such as the TCP timeouts mechanism), its formula is also way more complicated. As a consequence of this, the work of Mathis *et al.*, despite being more than twenty years old, often suffices and is preferred due to its simplicity, as can be seen in recent papers such as (ABDELMONIEM; BENSAOU, 2019; KFOURY et al., 2020). Equation (2.2) states that the maximum average throughput of *steady-state* TCP data transfer (BW_{max}) is a function of TCP Maximum Segment Size (*MSS*), round-trip delay (*RTT*) and packet loss (p); C is a constant that put together diverse attributes of a specific TCP implementation such as ACK policy and loss recovery procedures, and k can be approximated as -0.5 (MATHIS et al., 1997):

$$BW_{max} = \frac{MSS}{RTT} C p^k \tag{2.2}$$

Realizing that the contributions of the Network Layer are contained in *RTT* and p, taking exponent k as -0.5 at Equation (2.2), putting aside all constant factors not related to the network infrastructure, *e.g.*, switches and routers, and disregarding delay and packet loss due to the end-

points, BW_{max} can be stated as in Equation (2.3):

$$BW_{max} \propto \frac{1}{RTT\sqrt{p}} \tag{2.3}$$

The proportion stated in Equation (2.3) can be referred to as the *inverse-square root law*, and, provided the same packet loss conditions stated above (from $p \approx 10^{-5}$ up to $p \approx 0.05$), is also valid in more recent TCP implementations such as CUBIC (Linux default TCP congestion control protocol) and Compound (MS Windows default up to earlier builds of Windows 10; also partly AIMD-based) (AYAR et al., 2019). Furthermore, according to (BACCELLI; MCDONALD, 2005), inverse-square root proportionality is also valid with *non-persistent* flows. The effects of packet loss vs. RTT in the resulting TCP throughput according to the *inverse-square root law* can be seen on Fig. 1.

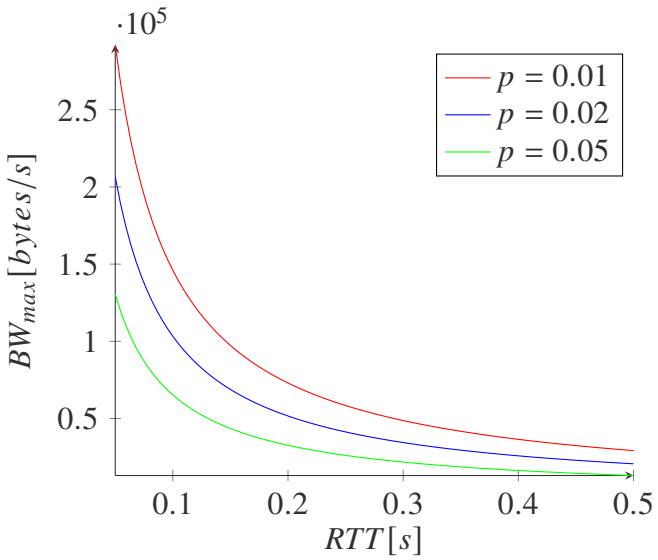

Figure 1: Eq. (2.2) stating theoretical BW_{max} (measured in bytes per second) for three values of p (0.01, 0.02 and 0.05: 1%, 2% and 5% of packet loss, respectively) versus RTT (measured in seconds). $C = 1$ and $MSS = 1460$ bytes.

Nevertheless, the *inverse-square root law* for TCP throughput prediction is not universal; if packet loss is sufficiently small, the *steady-state* TCP throughput prediction can be obtained with the proportion $1/p$ (linear regime) instead of $1/\sqrt{p}$ if p is not greater than a value of p_{max}, as stated in (ZARAGOZA, 2006). Just as an additional example, regarding any TCP loss-based congestion control protocol, a more general formula is stated by (XU; HARFOUSH; RHEE, 2004) in $R(p) = \frac{1}{RTT}\frac{c}{p^y}$; $R(p)$ is the response function representing the *steady-state* sending rate of the protocol in the unit of packets per RTT, stated as a function of packet loss (p), being c and y constants depending on the implementation. The value of d is 0.5 for AIMD-based congestion protocols, which, in this case, is compliant with the *inverse-square root law*. In

contrast, for HS-TCP and STCP, the values of d are 0.82 and 1, respectively (not compliant with the *inverse-square root law*).

In general, although not universally compliant with the *inverse-square root law*, most of the TCP congestion control protocols employed nowadays (such as the CUBIC as mentioned above and Compound) are compliant if packet loss ranges from $p \approx 10^{-5}$ up to $p \approx 0.05$. This occurs because, in the face of a more severe packet loss, those implementations operate in fallback mode and behave very similarly to Reno with an AIMD-like scheme; namely, it is valid with Reno, New Reno (both AIMD-based), CUBIC (LUKASEDER et al., 2016), Compound (POOJARY; SHARMA, 2016), HSTCP and STCP (YUE et al., 2012). As will be detailed in Chapter 3, our proposal relies on this fact to develop the novel metric TCPBP.

2.5.4 Assumptions of the Inverse-Square Root Law Model

The work (BASSO et al., 2012) presents an application-level approach for the measurement of packet loss rate in a TCP data transfer. This approach is built on eight assumptions of the model in which we based our formula, as stated below:

i. the receive window is large enough;

ii. the sender is always sending data;

iii. persistent TCP connection;

iv. use of fast re-transmit and fast recovery mechanisms;

v. selective acknowledgment (SACKs) are implemented, and multiple subsequent losses indicate a single congestion event;

vi. *steady-state* data transfer, *i.e.*, a data transfer large enough so that *slow start* phase can be neglected in the throughput computation;

vii. constant RTT;

viii. random packet loss.

Items *i-vi*, with the exception of *v* that depends on both network and TCP implementation (the protocol behavior before multiple subsequent losses which can occur due to network), refers to TCP and user behavior on the endpoints, *i.e.*, factors due not to network, but to design characteristics of the transport protocol and traffic profile. Items *vii-viii* refers to characteristics heavily

determined by network infrastructure, such as packet loss (item *viii*), bandwidth-delay product (item *vii*), and delay/jitter (also item *vii*). An assumption not listed, because it is not relevant to (BASSO et al., 2012), but one which we take into account, also due to network, because we are interested in express TCP throughput as a function of delay and packet loss, is available bandwidth. This is so because, regardless of how large BW_{max} can be, the actual throughput can be much lower due to the physical limitations of the path.

Therefore, our main concern with respect to the assumptions of the model focus on *vii* and *viii*: round-trip delay, jitter (the statistical variation of delay, presumed to be small), and random packet loss, *e.g.*, loss due to the nature of the links, or even loss from mild to moderate congestion under RED or Drop-Tail (MITZENMACHER; RAJARAMAN, 2001), but not the case of packet loss due to severe congestion; interestingly, no prior specific statistical distribution is described.

2.6 *High-Speed Lossy Networks*

A *High-Speed Lossy Network* is defined as a network topology that simultaneously exhibits the features of high speed (raw capacity in bits per second) and lossy links (*e.g.*, wireless links subject to propagation and media access control factors). An example of such are heterogeneous networks comprising 4G/LTE and transcontinental optical links that have both lossy and *high-speed* characteristics (DONG et al., 2019; SU et al., 2019)(interdomain scenario), and redundant wireless backbone networks in remote areas where optical fiber coverage is not feasible (COHEN, 2020) (intradomain scenario).

An example of *High-Speed Lossy Network* is shown on Fig. 2. It consists of three separate intradomain networks located far away from big centers, where optical fiber coverage is not feasible, and topography favors the installation of omnidirectional point-to-point radios: *(i)* a rural area backbone interconnected with omnidirectional high-speed gigabit radios; *(ii)* a small city gigabit cabled network, and *(iii)*, a community wireless mesh network (WMN) similar to the one employed on the proof-of-concept of (MANZOOR et al., 2020). All three networks are interconnected with both omnidirectional *high-speed* radios and low-earth orbit satellites. As such, the network as a whole is heterogeneous (due to the diversity of link technologies) and *high-speed* (mostly gigabit interconnections). It is also lossy since the omnidirectional radios, satellite interconnections, and wireless mesh links are subject to conditions such as weather or crowded spaces that either affect signal propagation or degrade network performance due to great media sharing effects, resulting in a packet loss rate much greater than of provided by cable such as optical fiber networks.

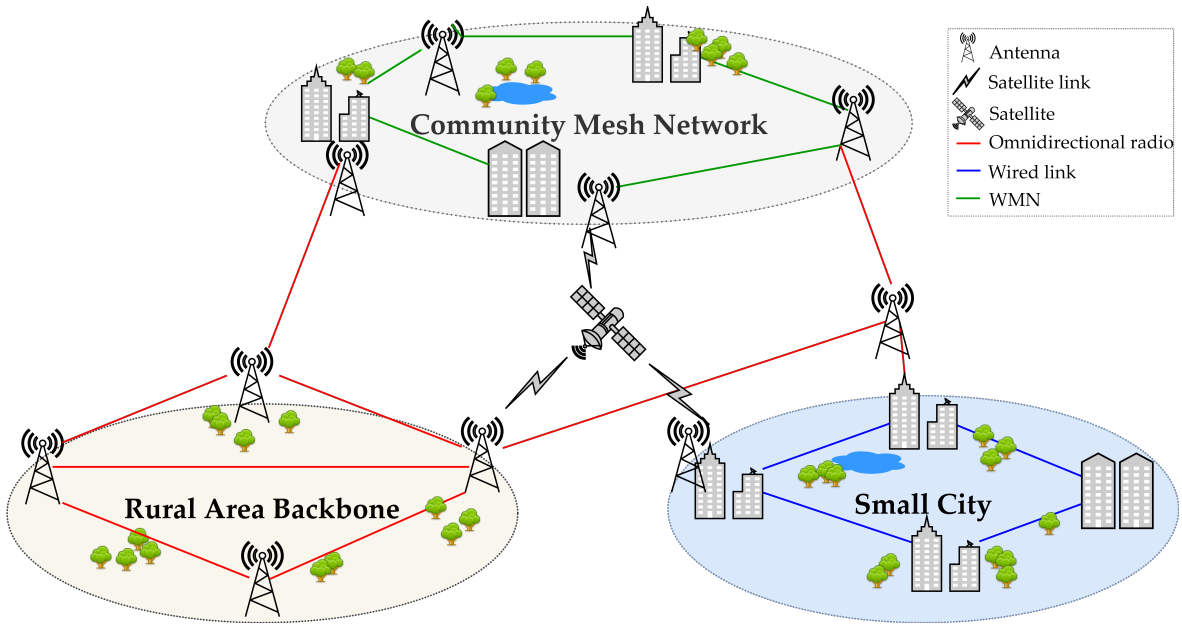

Figure 2: Example of a *High-Speed Lossy Network*.

2.6.1 High-Speed Definition

The concept of "*high-speed*" changes over time, *e.g.*, an 1 Mbps internet connection was indeed deemed as *high-speed* twenty years ago. However, it is not the case anymore, given advances in communications technology and the crescent end-user applications requirements concerning bandwidth. Therefore, since there is no strict definition of "*high-speed*", in this work, we arbitrarily consider *high-speed* links with over 1 Gbps capacity, which are used in the proof-of-concept.

2.6.2 Lossy Definition

In this work, we consider a link "lossy" if its packet loss rate ranges from $p \approx 10^{-5}$ to $p \approx 0.05$, *i.e.*. In such networks, the *inverse-square root law* applies almost universally to the TCP behavior, as stated in Section 2.5.3, and, due to the high-speed feature, we may guess that it takes quite a number of parallel TCP streams to fill the links' capacity before congestion occurs. For this very reason (*i.e.*, the combination of lossy links and *high-speed* capacity), selecting the routes in such networks based on the *inverse-square root law* seems to be an alternative to enhancing overall TCP throughput worth investigating.

2.7 Link Estimation

TCPBP, as will be shown in Chapter 3, does not enforce any particular mean of link parameter estimation or measurement, nor specifies the eventual format of messages (*e.g.*, LSA, in the case of a link-state routing protocol) between forwarding devices (*e.g.*, routers). Thus, passive or active measurements can be performed in the network to estimate the parameters: the approach works as long as minimizing the function of one-way delay (d) and packet loss (p) actually represents an improvement to network traffic and links' information can be correctly and periodically retrieved by the Control Plane before shortest path calculation, regardless of how it is done.

Since our approach is based on the assumption that end-to-end parameters can be calculated based on the quality of links (hops) in the path, it is also important to account for the implications of Link Layer on the delay and packet loss behavior and measurement. We discuss it briefly on Section 2.7.4, focusing on the differences between the MAC (Media Access Control) sublayer of Ethernet (IEEE 802.3 and its amendments), and WLAN (IEEE 802.11 and its amendments).

2.7.1 General Assumptions

Though in the general case we recognize that delay and packet loss may change with respect to traffic load (*e.g.*, bandwidth or packets per second), router CPU utilization, and packet size, for the sake of simplicity, in most simulations, unless stated otherwise, we assume it will not, or, even if it does, it is not a relevant factor. This is equivalent to say that end-to-end parameters can be properly estimated based on wire-to-wire traffic samples of the links which depend mostly on propagation and media sharing effects.

Additionally, we presume that delay and packet loss are uncorrelated or at least have a very low statistical correlation, which is true if the network is not collapsing with respect to the bandwidth consumption (*i.e.*, self-inflicted packet loss due to congestion).

2.7.2 Delay

Delay in this work may be: (1) the time taken by a given packet sent by a source node to reach its destination node plus the time taken by a packet on the way back (destination to source), *i.e.*, *round-trip delay* (*RTT*), or: (2) the time taken by a given packet sent by a source node to reach its destination, *i.e.*, *one-way delay* (d). The definition (1) is used whenever the values computed in fact depend on round-trip delays, like in Equation (2.2); shortest path computation, on the

other hand, uses definition (2).

As a link-state parameter, it is an additive metric (total delay is the sum of delays in the hops along the path), so the computation of shortest paths, in this case, is straightforward with Dijkstra shortest path algorithm, as the delay in the one-way route, being $[d_1,d_2,...,d_n]$ the set of delay values of the hops in the route, is $d_1 + d_2 + ... + d_n$.

Though end-to-end delay is composed of the contributions of processing delay (d_{proc}, in the order of μs), queuing delay (d_{queue}, theoretically zero if the queue is not full, that is, if there is no bandwidth consumption very close to congestion), transmission delay (d_{trans}, also in the order of μs) and propagation delay (d_{prop}, ranging from ms to a fraction of a second under normal circumstances), where $d = d_{proc} + d_{queue} + d_{trans} + d_{prop}$, for means of path computation, we consider the propagation delay (d_{prop}) as the preponderant factor, but values of d can be refined *a posteriori* with end-to-end evaluation that also take into account the effects of processing, queuing, and transmission.

2.7.3 Packet Loss

Like delay, in this work, packet loss (p) has two definitions: (1) the probability of a packet getting lost in the path – *e.g.*, source to destination and back –, and: (2) the probability of a packet getting lost from source to destination (*one-way packet loss*). The employment of (1) or (2) is the same as in the case of delay. The usage of definitions (1) and (2) using the same symbol (p) also should be clear from the context.

As a link-state parameter, it is a multiplicative metric, making the shortest path computation possible with the default Dijkstra shortest path algorithm transforming the multiplicative metric into an additive metric with a logarithm. Since one-way packet loss is the probability of a packet getting lost *en route* to the destination, the shortest path is the one with the greatest $(1-p_1)(1-p_2)...(1-p_n)$ product representing the probability of a packet successfully traversing the network from source to destination in n hops, being $[p_1,p_2,...,p_n]$ the set of p values of the hops in the route. With the aid of the log function, which is a group homomorphism that translates multiplication of positive real numbers into the addition of real numbers ($log : \mathbb{R}^+ \rightarrow \mathbb{R}$ with $log(ab) = log(a) + log(b)$), we can turn the multiplication into a sum and the shortest path can be found in additive fashion taking $abs(log((1-p_1)(1-p_2)...(1-p_n))) = abs(log(1-p_1) + log(1-p_2) + ... + log(1-p_n))$.

2.7.4 Considerations on Data Link Layer

Considering that end-to-end parameters delay and packet loss are composed of the contributions of each hop in the path, thus yielding the possibility of a hop-by-hop shortest path computation, one interesting point is to ponder whether links' load (*e.g.*, bandwidth utilization) impact the aforementioned parameters. This is a crucial fact in the design of TCPBP because if bandwidth utilization impacts delay and packet loss, both parameters may change a lot in a short-term perspective, perhaps in unpredictable ways. Such behavior may complicate link estimation and even render TCPBP unusable.

Apart from the cases where bandwidth utilization of any link in the network is at (or close to) maximum, where packet loss obviously increases due to packet discarding (*i.e.*, congestion) and delay may be impacted by the queue buffering of the forwarding devices (increasing the queuing delay d_{queue}), regarding the bandwidth utilization vs. packet loss and delay interplay, it is convenient presenting two main categories of networks classified with respect to the Data Link Layer:

i. networks where all links share the same collision domain;

ii. networks where each link is at an exclusive collision domain.

Examples of networks described in *i* are wireless networks based on IEEE 802.11, such as Wireless Mesh Networks (WMNs) that employ *CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance)* in the medium access control (MAC) sub-layer. In such networks, due to the shared essence of the links, performance depends not only on the characteristics of the physical layer (such as modulation schemes or signal propagation), which is a complex matter by itself, but also on how many devices try to access the medium, the relative position of the nodes in the network (with occasional emergence of the *hidden-terminal* problem), and when they attempt to do so (PASSOS, 2013). Needless to say, in such networks, end-to-end parameters p and d as random variables are not *i.i.d.*[6], and are prone to change from time to time concerning the bandwidth utilization.

As an example of *ii*, we have modern Ethernet-based (802.3) networks will Full-Duplex interconnections that occasionally resort on *CSMA/CD (Carrier Sense Multiple Access with Collision Detection)*. Although IEEE 802.3 standard and its amendments specify *CSMA/CA* as MAC

[6] *i.i.d.* stands for *independent and identically distributed*. In the case of exclusive collision domains, we expect at least that variables are independent, though they are not necessarily identically distributed. For the shared collision domain case, an interesting discussion about the practical aspects of the dependency between packet reception events in different nodes of wireless mesh networks can be found in Appendix B of (PASSOS, 2013).

mechanism, in practice, nowadays, it is employed in a few cases only, mostly for retro compatibility and in the occurrence of Half-Duplex interconnections. Consequently, under normal conditions, end-to-end parameters of such networks are not prone to change with respect to the bandwidth utilization of their links.

2.7.5 Passive vs. Active Measurement

The adoption of any QoS metric requires the proper measurement of the parameters of interest, which can be done in many ways. Two of the main approaches are broadly defined as "passive" and "active", so one can resort to one (or a combination) of the approaches whenever designing a QoS routing protocol.

The passive approach is the one that relies on the observation of the data flowing in the network, *i.e.*, watching the network traffic as it passes by. It can be done with the periodic polling of statistics of the forwarding devices in the network, usually with the aid of protocols such as SNMP[7], *NetFlow* or RMON (Remote Monitoring). The active approach, on the other hand, is based on the injection of specific probe packets in the network. A classic example of this category is *ping* employed in the measurement of round-trip delay and packet loss, but more refined techniques can be used, such as with the *iPerf* tool with injection of UDP probe packets.

In general, we consider that the measurement techniques should be adopted according to the specificities of the underlying network whenever it is possible; *e.g.*, take into consideration if the network is lossy or not (convergence of packet loss estimation may be faster in such cases). Some guidelines can be found in the IETF IP Performance Metrics (IPPM) Working Group website, like RFC 7679 (one-way delay measurement) (ALMES et al., 2016a) and RFC 7680 (one-way packet loss) (ALMES et al., 2016b).

2.7.6 Time-Variant Estimates

Regardless of the nature of the measurement (*e.g.*, passive vs. active), it is important to establish a coherent way to present d and p information to the Control Plane, so it can provide values for $f(d,p)$ and use it to make routing decisions, *i.e.*, use it as a metric. It is so because we wish to obtain not only sufficiently accurate values that represent the parameters in a given instant or time span but reference values that change smoothly over time: if we fail to fulfill this requirement, the stability of the packet forwarding in the network may be compromised, as well as sub-optimal

[7]Simple Network Management Protocol - RFC 2570 and its subsequent updates.

choices may be taken by the routing protocol.

Considering a scenario in which d and p may change abruptly over time[8], even if the parameters can be properly measured, it may trigger a lot of undesired route changes in a short time (via re-convergence of the underlying routing protocol), possibly triggering forwarding loops and compromising stability in general. Therefore, it is clear that a problem at the very core of the employment of TCPBP routing metric is adjusting the accuracy versus smoothness of d and p, thus $f(d,p)$, over time.

In such a case, a mechanism to prevent route instability may be necessary, and the curve of the parameter $f(d,p)$ representing the metric versus time should be smoothed: to do so, we propose the use of time-tested moving averages. The basic idea of employing the moving averages is to present the value of a given parameter in a given instant based on the average of the last k measurements. Formally, the simple moving average (SMA) is the simple mean over the last k entries of a data set with n entries, where the vector representing the measurements is $[x_1,x_2,...,x_n]$. The mean over the last k data-points is stated as $\bar{x}_k$ in Equation (2.4):

$$\bar{x}_k = \frac{1}{k}\sum_{i=1}^{k} x_i, \quad k = 1,\ldots,n \tag{2.4}$$

It is also possible to employ a weighted moving average, *i.e.*, an average where data within the same sample at different positions are given different weights, instead of a simple one; *e.g.*, in a time series, we may attribute greater weight to more recent measurements. An example of the employment of the weighted moving average can be found in (WOO; TONG; CULLER, 2003), where the authors use an exponentially weighted moving average (EWMA) for the link estimation technique of packet loss in multi-hop sensor networks.

Moving averages processes are computationally inexpensive and can be easily deployed on the Control Plane of the network. In the particular case of a centralized Control Plane, for the purposes of research and prototyping, it can be calculated with built-in functions of software like MATLAB. The effects of the employment of the moving average on a time series can be seen in Fig. 3.

Nevertheless, although we advocate for the use of moving averages, it is not the only choice documented in the literature. It is also possible to employ link estimation techniques based on linear regression, Kalman filters (WOO; CULLER, 2003), or statistical hypothesis tests like

[8]Under normal circumstances, such behavior is not expected to occur in wired networks such as IEEE 802.3 based networks, but it can occur in typical wireless networks like ones based on IEEE 802.11, as discussed in Section 2.7.4.

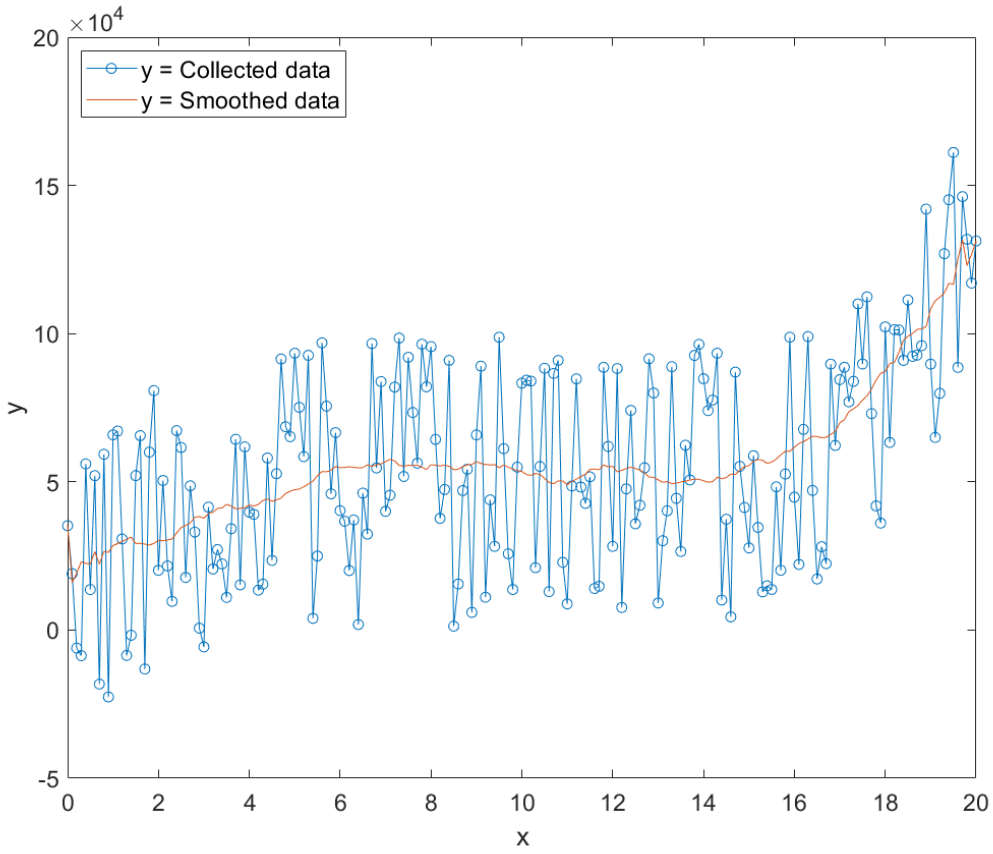

Figure 3: Time series of the hypothetical collected data (blue curve) and the curve of the moving average employed to smooth the original curve (red curve). Intuitively, the red curve can be interpreted as a *"smoothed"* approximation of the original blue curve, *i.e.*, an approximation that sacrifices the correctness of individual data for the sake of obtaining a curve that changes more slowly over time ($x = t(min)$); like it was a part of a hysteresis mechanism. If a simple moving average is employed, the *"smoothness"* vs. accuracy trade-off of the approach can be adjusted with the proper choice of k and n in Equation (2.4).

Hypothesis Testing Estimator (HTE) (SILVA; PASSOS; ALBUQUERQUE, 2018). HTE is particularly interesting because it is a frame[9] link delivery probability estimator built in such a way that the estimate remains constant unless the link behavior in the recent past significantly drifts from what is expected, *i.e.*, it per-design also focuses on stability.

[9]Very appropriate to distinct "*frame*" and "*packet*" here. It is so because there is a possibility of re-transmission of frames at Link Layer in wireless networks such as in IEEE 802.11. Therefore, in this context, "*frame*" and "*packet*" are not synonyms.

3 | Transmission Control Protocol Best Path (TCPBP)

In this Chapter we present our proposal in details (metric intuition – Section 3.1 –, feasibility regarding shortest path computation – Section 3.2 –, a shortest path computation method based on a modified edge relaxation process – Section 3.3 –, $f(d,p)$ computation – Section 3.4), the main aspects to be considered in the route selection (route selection mechanisms, initial routes and re-convergence – Section 3.5), a discussion on the expected throughput improvement – Section 3.6 –, and security considerations – Section 3.7.

3.1 Metric Intuition

Transmission Control Protocol Best Path (TCPBP) is a single mixed metric such that the $d\sqrt{p}$ per-route is minimized to maximize Equation (2.3) (*i.e.*, $BW_{max} \propto \frac{1}{RTT\sqrt{p}}$). Considering two nodes A and B in the network, the minimization of $RTT\sqrt{p}$ follows from the minimization of $d\sqrt{p}_{A\Rightarrow B}$ and $d\sqrt{p}_{B\Rightarrow A}$ because of metric isotonicity. The key idea of this approach is to be simple and allow the maximum per-flow TCP throughput as possible, under the assumption that the TCP throughput prediction model is valid in the network in which it is employed and there is enough available bandwidth to accommodate the flow during its duration.

Concerning *High-Speed Lossy Networks*, the *inverse-square root law* applies, and it is expected that the *per-flow* TCP throughput is much lower than the total bandwidth of the links, suggesting that the adoption of our metric may optimize the network *per-route* TCP throughput up to a certain number of TCP flows (denoted as n_f). Our proposal is also scalable and easy to deploy: for example, considering that it can be implemented with a simple modification of the default Dijkstra as the shortest path algorithm, suitable to *hop-by-hop* routing, and that d and p can be passively or actively measured in the network with the aid of SNMP and ICMP, it can be employed as an extension of a link-state routing protocol such as OSPF or IS-IS.

Ideally, the metric should also consider available bandwidth at a given time. However, one reason we did not incorporate it to the metric is that, as opposed to end-to-end delay and packet loss – parameters which we assume do not change with time, or that it changes slowly enough such that we can get accurate and consistent measurements –, and due to traffic dynamics, it most probably changes its value faster than the convergence time of the underlying routing protocol, possibly affecting the stability of the packet forwarding in the network and requiring additional mechanisms to maintain stability. Another issue with incorporating bandwidth in a single mixed metric is that it is a concave metric; mixing it with additive and multiplicative metrics breaks the

set of assumptions that guarantee the metric feasibility (isotonicity and consistency, in particular). As a matter of fact, we consider that bandwidth issues with respect to route selection should be treated separately from the shortest path computation algorithm (*e.g.*, through pruning or with traffic engineering).

3.2 Metric Feasibility

In this Section, we prove that the single mixed metric we proposed is both isotonic and monotonic, thus satisfying the properties of loop-freeness, consistency, and optimality. Considering that the metric is particular case of a function of delay and packet loss denoted as $f(d,p)$, we proceed to Section 3.2.1 (isotonicity) and Section 3.2.2 (monotonicity).

3.2.1 Isotonicity

Considering $[d_1,d_2,...,d_n]$ and $[p_1,p_2,...,p_n]$ the set of values of delay and packet loss per-route, respectively, and that d and p values are strictly positive, where each metric is isotonic by itself and shortest path computation of each metric is possible by means of an additive process, isotonicity of $f(d,p)$ follows immediately if $f(d,p) > 0 \; \forall \; d > 0, 0 \leq p \leq 1$, and $f(d_n,p_n) > 0 \; \forall \; n$, which can be proved with contradiction, but it is more simple to show that the commutative rule for sum and associative rule for multiplication implies the isotonicity of $f(d,p)$: in the case of $f(d,p) = d \times p$, $d = d_1 + d_2 + ... + d_n$ and $p = 1 - ((1 - p_1)(1 - p_2)...(1 - p_n))$, because of the commutative and associative rules of mathematics, swapping the positions of the hops will preserve the values of d and p, thus preserving the value of $d \times p$; appending or prefixing with hops such that $f(d_n,p_n) > 0$ will not provoke any changes in the original path, because of the same properties. Without loss of generality, the reasoning applies to any function of d and p, if $f(d_n,p_n) > 0 \; \forall \; n$.

3.2.2 Monotonicity

Monotonicity of $f(d,p)$ also follows immediately if $f(d_n,p_n) > 0 \; \forall \; n$; this is so because the path cost will never decrease when adding new hops (by appending or prefixing) with positive costs.

Algorithm 1: Modified edge relaxation. Because the path structure of TCPBP is not additive, since $d(u)$ is $(d_1 + d_2 + ... + d_n)(1 - ((1 - p_1)(1 - p_2)...(1 - p_n)))^{1/2}$, $d(u)+w(u,v)$ is $(d_1+d_2+...+d_n+d_{u\rightarrow v})(1-((1-p_1)(1-p_2)...(1-p_n)(1-p_{u\rightarrow v})))^{1/2}$.

```
if d(u) + w(u,v) < d(v) then
    d(v) = d(u) + w(u,v)
else
    do not update d(v)
end
```

3.3 Shortest Path Computation

This Section describes the shortest path computation mechanism employed to find the shortest path according to our metric. For so, we devised a modified edge relaxation process able to work with non-additive positive edge costs in a $G(V,E)$ directed graph based on the default Dijkstra algorithm.

3.3.1 Modified Dijkstra Algorithm

While the default implementation of the Dijkstra algorithm assumes that the cost of the path is the sum of the weights of the edges across the path, $d\sqrt{p}$ is not additive. Therefore, we propose a slight modification of the default algorithm suitable to the computation of $d\sqrt{p}$ (which we call "modified Dijkstra"). Such modification works because being additive is not a requirement to employ Dijkstra in the broader sense: monotonicity and isotonicity are sufficient conditions (YANG; WANG, 2008). The modification we propose is summarized below:

i. create an adjacency matrix with all values of $d\sqrt{p}$ per-hop. These values are used to determine the first edge to be considered by a node according to the greedy approach of the algorithm, in consonance with the triangle inequality principle that guides Dijkstra's execution in the edge relaxation process;

ii. the updated value of $d\sqrt{p}$ is stored alongside the individual values of d and p, in order to allow the proper computation of multi-hop $d\sqrt{p}$.

The modification proposed basically introduces a per-iteration extra time for the computation of $d\sqrt{p}$, following the computation of d and p, and extra space for storing d and p separately. Thus, the execution of the modified algorithm to a $G(V,E)$ graph is expected to be slightly greater (at least in theory, due to the additional operations required to store the updated costs of the non-additive metric), but the algorithm complexity regarding time execution in function of V and

E remains the same since V and E values and its respective number of operations (like edge selection and path cost update) that determine the time complexity also remain. Considering that the relaxation process is part of a single source shortest path algorithm with source S, $d(u)$ is the cost $S \Rightarrow u$, $d(v)$ is the cost $S \Rightarrow v$, $w(u,v)$ the cost $u \rightarrow v$, $d_{u \rightarrow v}$ the *one-way delay* cost of $u \rightarrow v$, and $p_{u \rightarrow v}$ the *one-way packet loss* of $u \rightarrow v$, the modified edge relaxation process is shown in Algorithm 1.

3.4 Computation of *f(d,p)*

The main aspects of the computation of $f(d,p)$ are granularity (Section 3.4.1) and, as proposed in Section 2.7.6, the employment of moving averages (Section 3.4.2).

3.4.1 Metric Granularity

Granularity in the context of the computation of $f(d,p)$ refers to the number of significant digits (or precision) used to present the values of d, p, and of the function itself. On the one hand, an extremely precise value may yield the computation of paths with different values that in practice are roughly equal, perhaps unnecessarily leading to the use of additional hysteresis mechanisms. On the other hand, a low precision value may fail to capture the desired behavior of the paths (*e.g.*, the case of TCP throughput prediction). Therefore, the granularity must be considered carefully, as it is an important object of concern to the metric design.

3.4.2 Simple Moving Averages

The value of $f(d,p)$ is based on a simple moving average (SMA). The values of k and n of Equation (2.4) are adjustable according to peculiarities of the network, such as link-state variability expectation or other factors listed in Section 2.7.4.

3.5 Route Selection Mechanism

3.5.1 Initial Routes

Considering the instant where TCPBP is configured in the network, there are no statistics of d and p to calculate $f(d,p)$. Until the first measurement of link-state parameters, *hop-count* can be used as a metric to initial route selection.

3.5.2 Route Candidates

We define the *Route Candidates* as an arbitrary-sized set of shortest paths according to a metric sorted in the crescent order of cost. In the context of our work, it can be specifically defined as the set of k-shortest paths with respect to $f(d,p)$, where the value of k_{TCPBP} (as we denote k for TCPBP) depends on particularities of the metric design; therefore, *e.g.*, it may be obtained with Yen's algorithm based on the modified Dijkstra proposed in Section 3.3.1.

3.5.3 Re-Convergence Criteria

Re-convergence basically refers to the Control Plane prerogative of deciding whether, and how fast, a new route should be put into use in the face of new information received concerning the routing metric. If a link goes down, for example, the Control Plane must command the forwarding devices to establish a new route as soon as possible, *e.g.*, by calculating a new path or selecting a precomputed backup path with approaches like the one presented in (FRANSSON; CARR-MOTYČKOVÁ, 2004).

Thus, such event (*i.e.*, re-convergence) is of utmost importance in the network, and re-convergence criteria is mandatory. Considering the end-user perspective, the main concern is that re-convergence should be as seamless as possible, *i.e.*, occur with minimal traffic disruption[10]. In the context of TCPBP, we focus on the Control Plane decision-making whenever values of $f(d,p)$ are updated. In this respect, particularly interesting is the case if there are two (or even more) routes with a close value of $z = f(d,p)$ with overlapping values considering a Δz variation of z: if this happens, the preferred route may begin to alternate over time between the two best choices, perhaps unnecessarily, urging the adoption of a hysteresis mechanism.

3.6 TCP Throughput Improvement

3.6.1 Case Study

To showcase the TCP throughput improvement that can be achieved with the adoption of TCPBP, we start by presenting a case study in which two remote endpoints communicate with TCP using a dedicated 1 Gbps "pipe", where the parameters p and RTT, as well as the number of parallel TCP streams sending data, can be adjusted. The tests of this Subsection were carried out in a

[10]To network engineers and administrators, there are other concerns not relevant to this work; see, for instance, (SILVA; MOTA, 2017).

Linux testbed where the endpoints are represented by two different Network Namespaces interconnected with an Open vSwitch; network parameters p, RTT and bandwidth are emulated with native traffic control (*tc*) module *NetEm* as detailed in Fig. 4.

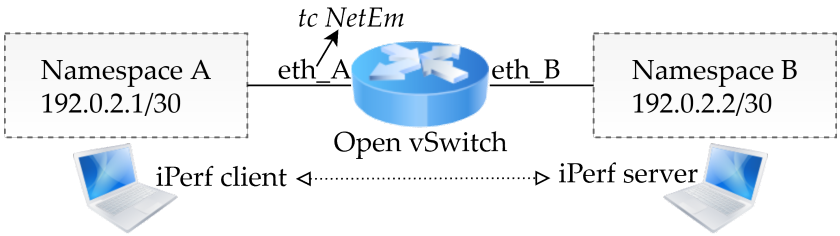

Figure 4: Linux testbed employed in Section 3.6.1. Network parameters of each Namespace are: TCP Cubic, MSS = 64 KB; *NetEm* is applied to Open vSwitch interface eth_A which interconnects to Namespace A, and emulates p, RTT, and bandwidth.

Firstly (first test), we set $p = 0.001\%$ and $RTT = 10ms$, and executed a 300s iPerf single connection test that presented 348 Mbps as a result; no TCP tuning was performed, and default TCP window was used. This is close to the theoretical maximum bandwidth that can be achieved according to the *inverse-square root law* formula: $MSS \times C/RTT\sqrt{p}$, with $C = 1$, $\approx$ 369.4 Mbps. Since a bandwidth of a value close to 650 Mbps was left unused, it was hinted that the data transfer between the endpoints can be doubled if two TCP connections[11] are used instead of one. Hence, we proceeded to check the total throughput in a test with two connections (second test), once again employing iPerf for 300s, and obtaining as a result a total of 692 Mbps (the sum of two TCP streams of 350 Mbps and 342 Mbps, respectively; both values are very close due to TCP fairness). Nevertheless, in this case, the link is still underused. An interesting fact is that real-time monitoring showed that the total capacity of the link was never achieved during both tests. Thus, packet loss that triggered TCP congestion control in the previous tests were not caused by congestion.

With three parallel connections (third test), we obtained a total of 900 Mbps (per stream: 281 Mbps, 296 Mbps, and 323 Mbps). The scenario ensued in this case, however, stumbled upon the physical limitation of 1 Gbps, with the occurrence of loss not only due to characteristics of the link but also because of congestion due to the small buffer of the Open vSwitch interface (no particular AQM[12] or packet scheduling scheme was employed in the OVS interfaces; Linux kernel default was used). Anyway, considering a 1 Gbps[13] interconnection and three TCP streams, in a scenario with perfect fairness, the average throughput expected per stream is at maximum

[11] At the expense of opening another pair of TCP sockets between client and server, thus increasing CPU and memory consumption in both endpoints, though.

[12] *Active Queue Management*.

[13] Not to be confused with IEEE Gigabit Ethernet (GbE) standard that actually does not provide 1000 Mbps capability, but rather $\approx$ 950 Mbps. The setup we employed in fact emulates a close-to-perfect 1 Gbps bandwidth.

$\approx$ 333 Mbps (1 Gbps divided by three streams), which is lower than the values obtained in the first two tests, and even lower than the expected throughput according to the *inverse-square root law* disregarding bandwidth limitations.

Supposing that we are able to improve the values of p and RTT of the link (which is true, since we control the testbed with *tc NetEm*), it is clear that the *steady-state* throughput with one and two connections can be improved: if RTT or p is properly reduced, with one connection, a single TCP stream can achieve a value close to 1 Gbps, and, with two connections, each TCP stream can achieve up to 500 Mbps, totaling 1 Gbps (*i.e.*, 1000 > 369.4 and 500 > 369.4); and it is clear that it cannot be in the case of three parallel connections, because fairness, in this case, ensues a roughly equal division of bandwidth per stream, since 333 < 369.4. Therefore, we conclude that, in this example, reducing $RTT\sqrt{p}$ does not improve TCP throughput if there are more than three flows.

3.6.2 Theoretical Improvement

Considering any route $A \Rightarrow B$ obtained with single metrics such as delay or *hop-count*, TCPBP provides an improvement if its route provides better individual TCP throughput performance for several flows up to $n_f \geq 1$. Denoting $R_{SingleMetric}$ as the original route, R_{TCPBP} as the route obtained with TCPBP, BW the average TCP throughput *per-flow*, TP the expected throughput per-flow according to Equation (2.2), and ABW the available bandwidth in the path, throughput improvement occurs if conditions in (i) and (ii) are met; such conditions stem mainly from the available bandwidth factor. The expected value for n_f is given in (iii).

$$min[TP_{TCPBP}, ABW_{TCPBP}] > min[TP_{SingleMetric}, ABW_{SingleMetric}] \quad \text{(i)}$$

$$R_{TCPBP}BW > R_{SingleMetric}BW \quad \text{(ii)}$$

$$(ABW_{TCPBP})/n_f > R_{SingleMetric}BW \quad \text{(iii)}$$

3.7 Security Considerations

Procedures and protocol extensions described in this thesis do not affect inter or intradomain routing security models. This is so because the adoption of a single mixed routing metric based on a function of packet loss and delay does not relate directly to network security issues, nor does it affect the communication security between the Control Plane and the Data Plane in the network (GAO et al., 2018; XIE et al., 2015).

Regarding specifically the intradomain case, which is the scope of this work, the security is mostly ensued by cryptographic authenticated messages of the underlying routing protocol employing a given routing metric. See, for example, with respect to OSPF, the "*Security Considerations*" topic of RFC 2328 (MOY, 1998), and RFCs 6863 – "*Analysis of OSPF Security According to the Keying and Authentication for Routing Protocols (KARP) Design Guide*" (HARTMAN; ZHANG, 2013) and 7474 – "*Security Extension for OSPFv2 When Using Manual Key Management*" (BHATIA et al., 2015), and note that routing metrics are not even mentioned.

4 | Simulation and Results

In this Chapter, we present an experimental evaluation of the TCPBP, comparing it to delay and *hop-count* metrics (because of the prevalence of those metrics in most intradomain networks nowadays), followed by a discussion of the results obtained and other insights. The evaluation is not extensive, focusing on the throughput improvement and shortest path computation running time. However, we consider it suffices to showcase the soundness of our proposal and point out possible drawbacks of employing the metric as we did.

4.1 Metric Design

This Section presents the metric design characteristics employed in the TCPBP proof of concept. We depict the design taking into consideration the SDN environment and the nature of the Data Link Layer, as stated in Section 2.7.4 (network is IEEE 802.3-based) and considerations following on Chapter 3.

4.1.1 Shortest Path Computation

The shortest path computation for TCPBP is performed with modified Dijkstra, as depicted in Section 3.3.1. Delay and *hop-count* are computed with standard Dijkstra. All processes are executed in a MATLAB SDN Northbound API running an OSPF-like program.

4.1.2 $f(d,p)$

The $f(d,p)$ is the $d\sqrt{p}$ expression derived from the formula $\frac{1}{RTT\sqrt{p}}$ that predicts the maximum average TCP throughput according to (MATHIS et al., 1997) (*i.e.*, TCPBP). Its value is obtained from the measurements of d and p as described in the following Sections.

4.1.2 Link-State Measurement

One-way packet loss is measured passively by polling the statistics of inbound and outbound interfaces of the two adjacent forwarding devices. That is, counting how many packets arrive at a device interface, dividing by the number of packets sent by the adjacent interface on the other device, and subtracting the value from one. One-way delay is measured by checking the

timestamps of the packets, which is possible because all devices' clocks are synchronized.

The passive approach makes sense because link-state data provided to the Control Plane is based on the average packet size generated by the endpoints; with an active approach, such accounting for different levels of packet loss for different packet sizes may introduce unnecessary complexity to the process. One possible drawback of the passive measurement for such means (p in special), in the general case, is that we obtain different sample sizes for the same interval in different links, but in this proof of concept, we do not expect it to be a problem if the minimal sample size is exceeded, for the lossy nature of the links provides good estimation with much smaller samples than what we would expect in not lossy links[14].

4.1.2 Moving Averages

We employed a simple moving average (SMA) of the last $30s$ every $10s$ (*i.e.*, link-state values were updated every $10s$, with MATLAB built-in function $filter$ and each value represented by the mean of the last three collected samples). Such values were chosen based on the expectation of an average link-state variation not greater than ±10%.

4.1.2 Granularity

Delay is an integer number converted from the millisecond rounded unit obtained with one significant digit after the decimal (*e.g.*, 2.5ms, 10.6ms, etc.). Packet loss is presented as multiples of 0.01% converted to absolute values.

4.1.3 Route Selection

Routes are selected based on the shortest path obtained with the metric, based on the average values of $f(d,p)$. In other words, we employ the Route Candidates with $k_{TCPBP} = 1$. In the absence of packet loss and delay statistics to compute $f(d,p)$, initial routes for TCPBP and delay are based on *hop-count*.

[14]For example: if we have a packet loss rate of 1%, the sample size (and mostly the time, too) to estimate it with the passive approach is much smaller than of the case in which $p = 10^{-12}$.

4.1.4 Re-Convergence Criteria

Re-convergence occurs upon the resurgence of a new route with the lowest value of $f(d,p)$, without additional mechanisms. It is a "greedy" approach that does not account for the route stability of the underlying routing protocol, but one which we resorted to due to its simplicity.

4.2 Testbed Outline

As a proof-of-concept, we implemented SDN OSPF with TCPBP metric and default metrics (*hop-count* and delay) on a Linux *OpenFlow* testbed with IPv4 forwarding enabled in the kernel, where each particular topology tested consisted on a set of *L3-Routing-Enabled* Open vSwitches (OVS) interconnected with virtual ethernet pairs (*veth)* in the Default Network Namespace according to its adjacency matrix representation, using additional Network Namespaces as endpoints.

Link-state parameters, *i.e.*, delay and packet loss are emulated with Linux traffic control (*tc*) module *NetEm*[15] such that the scheme resembles a *High-Speed Lossy Network*. As a Northbound API, we used MATLAB R2019, and, as Southbound API, OpenFlow 1.5.1. SDN controller of choice is POX 0.2.9. The architecture of the scheme employed is shown in Fig. 5. Specs of the machine used to run all the simulations are in Table 1.

Table 1: Specifications of the machine used to run all simulations.

Processor	Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz
RAM Memory	16 GB (2x8GB DDR4 Dual-Channel 1066Mhz)
HD	Samsung SSD 970 EVO Plus 250GB
Operational System	Linux Mint 18.04

4.3 Parameters Evaluated and Methodology

To measure TCPBP throughput performance and fairness in the face of OSPF default metrics delay and *hop-count*, we have opted for a topology with ten nodes. Simulations were executed in a symmetric topology with 10 Open vSwitches, as detailed in Fig. 6, and *Full Duplex* interconnections with a rate limit of 1 Gbps. Emulated link-state parameters of the topology (delay and packet loss) are in the adjacency matrices of Figures 7 and 8; the values of the matrices

[15]Usage and caveats of NetEm can be found in (FOUNDATION, 2021).

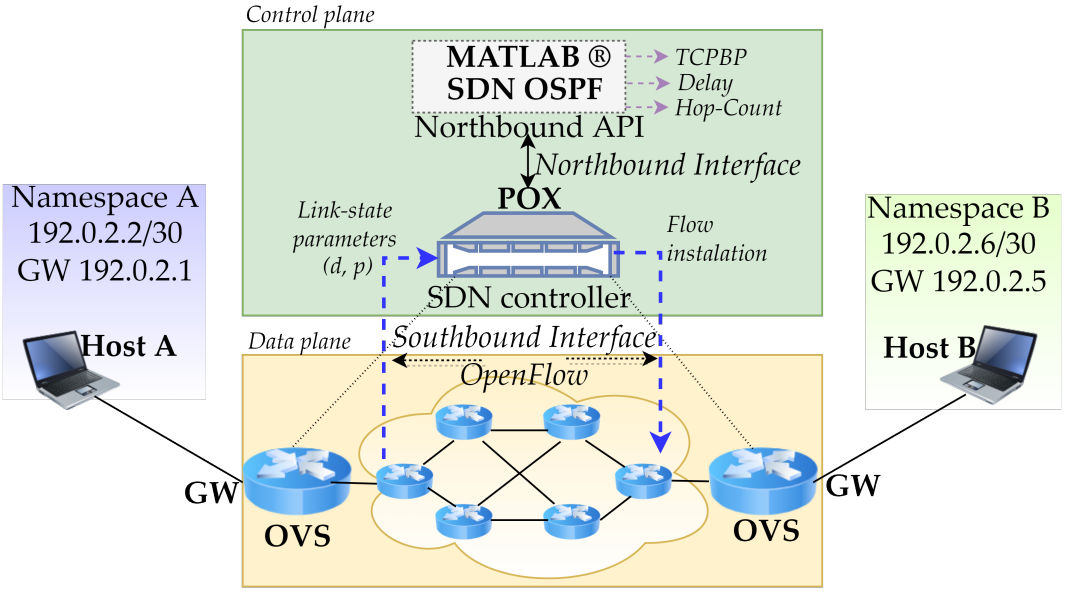

Figure 5: Linux OpenFlow Testbed. Host A communicates with Host B through Open vSwitches in the default Network Namespace. Communication between the POX controller and Open vSwitches uses OpenFlow as the standardized protocol for flow installation and maintenance; once a centralized Northbound API (MATLAB) after retrieving topology information from POX defines the desired forwarding behavior of the network, proper flow instructions are installed in each of the Open vSwitches (OVS).

were randomly chosen from an interval compatible with a lossy network. TCPBP running time was evaluated in symmetric topologies, with 10, 25, 50, and 100 Open vSwitches and uniformly distributed pseudo-random numbers as link-state parameters.

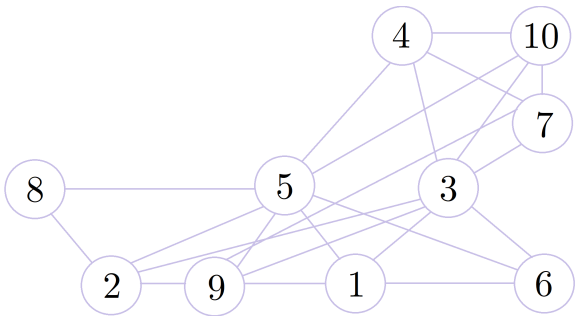

Figure 6: Graph representing the *High-Speed Lossy Network* topology with ten nodes used in the proof-of-concept.

To generate all necessary topologies, we used MATLAB. Once a topology was created, it was exported to a text file as input into a Bash Script, which then generated the topology with the interconnection of Open vSwitches in the default Linux Network Namespace. POX SDN controller also runs in default Network Namespace and, to receive link-state information from each of the switches, communicates with the Data Plane via 127.0.0.1 IPv4 loopback address. Upon receiving link-state information of the topology, POX forwards information to MATLAB,

$$\begin{bmatrix}
\infty & \infty & 0.0085 & \infty & 0.004 & 0.0015 & \infty & \infty & 0.0032 & \infty \\
\infty & \infty & 0.004 & \infty & 0.0055 & \infty & \infty & 0.0019 & 0.007 & \infty \\
0.0085 & 0.0044 & \infty & 0.0018 & \infty & 0.0096 & 0.0082 & \infty & \infty & 0.0044 \\
\infty & \infty & 0.0018 & \infty & 0.0005 & \infty & 0.0073 & \infty & \infty & 0.003 \\
0.004 & 0.0055 & \infty & 0.0005 & \infty & 0.0094 & \infty & 10^{-5} & 0.0063 & 0.005 \\
0.0015 & \infty & 0.0096 & \infty & 0.0094 & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & 0.0082 & 0.0073 & \infty & \infty & \infty & \infty & 0.0054 & 0.0076 \\
\infty & 0.0019 & \infty & \infty & 10^{-5} & \infty & \infty & \infty & \infty & \infty \\
0.0032 & 0.007 & \infty & \infty & 0.0063 & \infty & 0.0054 & \infty & \infty & \infty \\
\infty & \infty & 0.0044 & 0.003 & 0.005 & \infty & 0.0076 & \infty & \infty & \infty
\end{bmatrix}$$

Figure 7: Delay adjacency matrix. Values in seconds. Each value was emulated with a ±5% uncertainty to provide a rudimentary link-state variability.

$$\begin{bmatrix}
\infty & \infty & 0.003 & \infty & 0.0181 & 0.009 & \infty & \infty & 0.0012 & \infty \\
\infty & \infty & 0.004 & \infty & 0.0079 & \infty & \infty & 0.0116 & 0.0196 & \infty \\
0.003 & 0.004 & \infty & 0.0002 & \infty & 0.0116 & 0.0144 & \infty & \infty & 0.0192 \\
\infty & \infty & 0.0002 & \infty & 0.0005 & \infty & 0.013 & \infty & \infty & 0.0037 \\
0.0181 & 0.0079 & \infty & 0.0005 & \infty & 0.0137 & \infty & 0.0023 & 0.0057 & 0.0039 \\
0.009 & \infty & 0.0116 & \infty & 0.0137 & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & 0.0144 & 0.013 & \infty & \infty & \infty & \infty & 0.0119 & 0.0068 \\
\infty & 0.0116 & \infty & \infty & 0.0023 & \infty & \infty & \infty & \infty & \infty \\
0.0012 & 0.0196 & \infty & \infty & 0.0057 & \infty & 0.0119 & \infty & \infty & \infty \\
\infty & \infty & 0.0192 & 0.0037 & 0.0039 & \infty & 0.0068 & \infty & \infty & \infty
\end{bmatrix}$$

Figure 8: Packet loss adjacency matrix. Absolute values in the interval [0,1]. Each value was emulated with a ±5% uncertainty to provide a rudimentary link-state variability.

which then calculates the best routes according to the correspondent metric, and instructs POX back to install the flows accordingly in each of the switches.

The tests started at $t = 0s$. To gather the first link-state information, random unicast traffic of 50 Mbps between all adjacent pairs was generated up to $t = 120s$, and data was collected in $10s$ intervals by means of passive measurement; then, during the rest of the tests, we employed a simple moving average (SMA) as described in the Metric Design Section. Throughput, fairness, and TCPBP running time were evaluated from $t = 120s$ as stated below (in Sections 4.3.1, 4.3.2 and 4.3.3):

4.3.1 Throughput

TCP persistent flow throughput was measured for all $(n^2 - n)/2$ possible distinct routes in the topology with $n = 10$ nodes, with TCPBP, delay, and *hop-count* metrics, using Host A (IPv4 192.0.2.2/30, GW 192.0.2.1) and Host B (IPv4 192.0.2.6/30, GW 192.0.2.5) as "seeds" to replicate other hosts in the 192.2.0.x/24 subnet, each of them in a different Network Namespace and

different subnet, as shown in Fig. 5, with TCP CUBIC, MSS=1460 bytes, and *iPerf* tool in TCP client vs. server mode. During the tests, Host A, Host B, and 6 (six) other hosts cloned from Hosts A and B (*i.e.*, identical configuration, but with different /30 IPv4 addresses in the 192.0.2.0/24 subnet) were randomly put into each node of the distinct pairs in the topology, and each metric was enforced according to the client TCP source port in the OVS flow tables. The resulting throughput for each topology is the average of n_f = 10, 25, and 50 TCP parallel flows per-route and per-metric for 60 seconds.

4.3.2 Fairness

To measure fairness, we resort to the Jain's Index and present the average fairness per scenario evaluated in the throughput test, *i.e.*, for all metrics evaluated (TCPBP, delay, *hop-count*) and for n_f = 10, 25 and 50.

4.3.3 Running Time

TCPBP with modified Dijkstra and standard Dijkstra running times at application layer (*i.e.*, Control Plane) were measured 100 times per topology, for random topologies with n = 10, 25, 50 and 100 nodes, with MATLAB *tic* and *toc* functions; Dijkstra implementation is "naive", with vertex set stored in an array and edges in an adjacency matrix, $O(V^2)$ complexity. We did so not to achieve the best running time as possible or measure running times with deep precision, but to present a value of reference that demonstrates TCPBP shortest path computation with modified edge relaxation is not significantly slower than standard Dijkstra, being capable of running in common off-the-shelf hardware such as laptops.

4.4 Results and Discussion

Table 2 shows a route summary per topology tested. As expected, some of the routes provided are the same in TCPBP, delay, and *hop-count* metrics; in fact, in our case, the majority of the routes are the same (57.8% for TCPBP and *hop-count*, and 86.7% for TCPBP and delay). Globally, *i.e.*, considering all routes, the TCP throughput improvement ratio (Table 3) exhibits a slight improvement (less than 1%) with TCPBP in respect to *hop-count* and delay, for n_f = 10, n_f = 25, and n_f = 50. However, when TCPBP provides different routes, the average TCP throughput ratio shows much greater performance (greater than 65% in comparison to *hop-count* and 40% in comparison to delay), even though it diminishes considerably from n_f = 25 to n_f = 50, mostly

due to the fact that, for $n_f = 50$, in average, the available bandwidth divided by the number of flows (n_f) is smaller than the predicted by the *inverse-square root law*, implying additional packet loss and delay due to congestion itself that may reduce the expected throughput *per-flow*. Also, since the topology tested is symmetric, we believe that with asymmetric routing, a common feature of IP networks, there would be a greater number of different round-trip paths produced with TCPBP, thus presenting a greater throughput ratio in favor of our approach (*e.g.*, a rough estimation for the TCPBP versus delay in the case of our simulations, where 86.7% of the routes are the same, is we should have $0.867 \times 0.867 \approx 0.751$ of the same round-trip paths in an asymmetric topology, *i.e.*, 75%).

Though we have opted for presenting ratios instead of absolute values for throughput, for our analysis focus is the global improvement and not the measurement of specific TCP flows, we consider it worth presenting the specific results obtained in the route from node 1 to node 10, for didactic reasons. A summary of the results obtained in the $1 \Rightarrow 10$ route is presented in Table 4, in which we can note that the *per-flow* throughput to $\frac{1}{RTT\sqrt{p}}$ ratios are quite close, showcasing the adequacy of the *inverse-square root law* in predicting at least the TCP throughput proportion between available routes: with respect to route selection with a metric, it is the fundamental concern. Though our simulated environment is obviously simple in comparison to real networks, it is interesting to note that there was not any occurrence of re-convergence, and we were able to determine (perhaps due to the *Law of Large Numbers* applied to the long-term behavior of d and p) that TCPBP is better than delay, even though their values of $\frac{1}{RTT\sqrt{p}}$ are very close (324.09 versus 321.99); in real networks, or with short-spanned flows, such difference may not exist, implying that the best TCPBP route may change from time to time, possibly demanding the employment of a mechanism to prevent route instability.

With respect to fairness, results are presented in Table 5; for the values do not differ significantly across the metrics, we have not observed any negative effects on fairness that stem from the adoption of TCPBP. Finally, TCPBP shortest path calculation running time with the modified edge relaxation showed good performance, consonant with the expected increment to the default Dijkstra implementation; *i.e.*, it is slower than default Dijkstra, but the difference is not something to raise concern. The results of shortest path computation times are summarized in Table 6.

Table 2: Route summary of the topology tested (10 Open vSwitches, 21 links); TCPBP versus *hop-count* and Delay.

Route summary: TCPBP vs. *hop-count* and Delay		
Total distinct routes: 45	TCPBP = *hop-count*: 26	TCPBP ≠ *hop-count*: 19
	TCPBP = Delay: 39	TCPBP ≠ Delay: 6

Table 3: Throughput performance comparison in the tested: TCPBP versus *hop-count* and Delay (10 Open vSwitches, 21 links); n_f = 10, 25 and 50 parallel TCP Flows. AR = all routes; DR = when TCPBP provides different routes.

	Average Throughput			
	TCPBP/*hop-count* ratio (AR)	TCPBP/*hop-count* ratio (DR)	TCPBP/Delay ratio (AR)	TCPBP/Delay ratio (DR)
$n_f = 10$	1.0095	1.8222	1.0025	1.5523
$n_f = 25$	1.0093	1.8362	1.0027	1.5321
$n_f = 50$	1.0092	1.650	1.0022	1.412

Table 4: Values of Eq. 2.3 and its respective TCP throughput *per-flow per-metric* obtained in the $1 \Rightarrow 10$ route. Both RTT and p refers to round-trip values, and symmetry implies $1 \Rightarrow 10 = 10 \Rightarrow 1$.

	TCPBP	*hop-count*	Delay
	$1 \rightarrow 3 \rightarrow 4 \rightarrow 10$	$1 \rightarrow 5 \rightarrow 10$	$1 \rightarrow 5 \rightarrow 4 \rightarrow 10$
$\frac{1}{RTT\sqrt{p}}$	324.09	265.18	321.99
Throughput (Mbps)	17.1	13.3	16.8

Table 5: Jain's index (J) per scenario evaluated in the throughput test. Each value presented is an average.

	Jain's index (J)		
	TCPBP	Delay	*hop-count*
$n_f = 10$	0.88	0.91	0.90
$n_f = 25$	0.89	0.93	0.92
$n_f = 50$	0.95	0.94	0.94

Table 6: Time execution (t_e) of shortest path computation with Standard Dijkstra and Modified Dijkstra. Each value is the result of 100 experiments that represent the average time for calculating all routes in a central hardware, with n parallel instances running each turn.

$t_e(s)$	$n = 10$	$n = 25$	$n = 50$	$n = 100$
Standard Dijkstra	0.0022	0.0304	0.1716	2.18564
Modified Dijkstra	0.00260	0.0362	0.1906	2.2241

5 | Background and Related Work

Since the performance of TCP in the face of the QoS parameters of the network (like delay and packet loss) also depends on the protocol behavior at endpoints (*i.e.*, the performance depends not only on QoS parameters but on how the transport protocol reacts before these parameters), the TCPBP is not a QoS routing metric in the strict sense. Nevertheless, the topic of QoS routing metrics is closely related to our proposal, particularly because of the path computation based on QoS metrics and the impact on users' Quality of Experience (QoE) (which can be improved with both TCPBP and QoS metrics).

Because of this, we consider the topics of QoS Routing Metrics (Section 5.1), MCR (Section 5.2), Link-State QoS Routing (Section 5.3), and Single Mixed Metrics (Section 5.4) as related subjects to our work. Furthermore, to the best of our knowledge, our approach is the only work that tackles the problem of throughput delivered by the Transport Layer employing route selection (*e.g.*, routing metric); *i.e.*, there is not any related work with this regard. So, due to its relative proximity to our proposal, we also classify related work in the following categories: TCP Performance (*i.e.*, implementations on Transport Layer; Section 5.5), and Practical Uses of the *Inverse-Square Root Law* (Section 5.6).

5.1 QoS Routing Metrics

Quality-of-Service (QoS) is a broad term in computer networks, and its precise meaning pretty much depends on the context. Regarding our work, in specific, QoS metrics can be defined as metrics used to quantitatively measure the quality of service of a network and/or network service, whereas typical QoS parameters considered are error rates (or packet loss, in the context of IP computer networks), bandwidth, throughput, transmission and propagation delay, availability, jitter etc. (DAS; CHAUDHURI, 2015).

In the pioneer (yet mostly up-to-date) work of (WANG; CROWCROFT, 1996), it is stated a couple of requirements for QoS routing, such as complexity analysis, metric selection criteria, and path computation algorithms. The key concepts of the paper are described below:

i. path computation algorithms must be efficient and scalable, *i.e.*, routes must be calculated in a feasible time considering the increasing number of nodes in the network, with time complexity comparable to current shortest path calculation algorithms;

ii. metrics should reflect basic characteristics of the network, whereas each QoS requirement

translates into a routing metric;

iii. whenever possible, metrics should be orthogonal to each other, *i.e.*, no redundant information should be presented among the metrics of the network. It is so because redundant information can introduce interdependence that makes impossible the individual evaluation of each considered metric.

We accept items *i* and *ii* above as a principle (*ii* is certainly valid at least in the case of Multi-Constrained Routing), but item *iii* should be analyzed carefully. In fact, a certain degree of redundancy (in the sense of a positive correlation) between the metrics may be desirable, as it could help evaluate one metric only instead of two, *i.e.*, the individual evaluation of each metric can be neglected. Furthermore, if a considered metric is compound, stated as a function of other metrics, the individual evaluation of each metric is not only possible, but probably mandatory, even if the metrics are not orthogonal[16].

5.2 Multi-Constrained Routing

Because end-user applications may require that more than one QoS metric is satisfied, one interesting problem at the very core of QoS routing is finding paths that simultaneously satisfy multiple constraints, whereas each constraint translates into a routing metric: *e.g.*, finding paths in which both delay (*additive*) and packet loss (*multiplicative*) are better than certain established thresholds. For this reason, such a problem is called Multi-Constrained Routing (MCR).

Regarding the aforementioned case in which a desirable path in a network is the one satisfying a set of different metrics (*i.e.*, multiple constraints). The theoretical basis for such cases is established, as can be seen, for example, in the works of (WANG, 1999) and (KUIPERS et al., 2002). In (WANG, 1999), it is shown that finding a path satisfying multiple metrics with additive (*e.g.*, delay and *hop-count*) and multiplicative metrics (*e.g.*, packet loss) is NP-complete; in practice[17], this means that shortest path computation in such cases is not be feasible for network topologies with more than a couple of nodes, because the algorithms that provide exact solutions are way too slow. By the way, such path may not exist at all.

This fact led to the proposal of heuristic approaches to deal with the problem. (KUIPERS et al., 2002) provides an overview of heuristic algorithms with the goal of finding the shortest paths subject to multiple constraints (or metrics), and a handful of subsequent papers also present

[16]Saying that two metrics are not orthogonal means there is some degree of redundant information between metrics, but it does not state how much; therefore, in such cases, it is probably necessary to evaluate both metrics.

[17]Given the current status of the P versus NP problem.

approaches to tackle the problem in both traditional and SDN environments, such as in (GUCK et al., 2017).

One case worth noting is that the combination of one concave metric plus any other metric (be it additive or multiplicative) is the only straight feasible combination when it comes to providing exact solutions. Not surprisingly, a lot of papers deal with the combination of bandwidth (concave) and delay (TANTISARKHORNKHET; WERAPUN, 2016) (TOMOVIC; RADUSINOVIC, 2018) (BADIS; AGHA, 2003).

5.3 Link-State QoS Routing

We choose to present this Section because our simulations were executed with a link-state routing protocol. Open Shortest Path First (OSPF) (RFC 1583; RFC 2328 – OSPFv2 (MOY, 1998), and subsequent updates) is, perhaps, the most used Link-State routing protocol, employing the Dijkstra algorithm as its shortest path calculation algorithm. Consequently, it was developed such that routes are calculated based on the concept of cost, being the cost of the route the sum of the costs *per-hop* across the path, and primarily taking reference bandwidth (default behavior of the protocol), delay and load as possible metrics; it is also possible to manually adjust the costs of the links with arbitrary weights.

Even though the original OSPF already offered the possibility of rudimentary QoS routing (with delay and load as possible metrics), subsequent research, as in RFC 2676 – "*QoS Routing Mechanisms and OSPF Extensions*" (APOSTOLOPOULOS et al., 1999), proposed extensions to OSPF to support QoS mechanisms. In addition to these efforts, there are also Traffic Engineering (TE) extensions to OSPF that can be employed to provide and maintain QoS policies in the network, such as (but not limited to) described in RFCs 3630, 4973, and 8330.

5.4 Single Mixed Metrics

The work (WANG; CROWCROFT, 1996) introduces the concept of a single mixed metric and questions whether such an approach can support end-user QoS requirements, as follows:

> "*One possible approach might be to define a function and generate a single metric from multiple parameters. The idea is to mix various pieces of information into a single measure and use it as the basis for routing decisions.*" (WANG; CROWCROFT, 1996).

Still, in (WANG; CROWCROFT, 1996), the authors make two appointments:

i. single mixed metric does not permit the individual assessment of whether QoS requirements can be met, and can be used only as an indicator;

ii. a problem of metric composition may arise when mixing some classes of metrics, such as with concave and additive metrics. For example, considering a path with two hops ab and bc, if metric $f(pt)$ (pt stands for "path") is delay, the composition rule is additive ($f(ab+bc) = f(ab)+f(bc)$); if it is bandwidth, the composition rule is concave ($f(ab+bc) = min[f(ab), f(bc)]$). But if $f(pt)$ is a product of bandwidth and delay, none of the composition rules apply, and, in fact, a composition rule may not exist.

The first part of the statement in *i* is true because the value of a simple composition does not have precise information about any individual QoS requirement, but we disagree with the dismissal of a single mixed metric as a mere "indicator": first, because as exposed throughout this work, a single mixed metric based on delay and packet loss may be employed to enhance transport layer protocols performance, in particular, TCP throughput performance (and most probably other connection-oriented transport protocols); and, second, there is longstanding work in the literature that present good results with single mixed metrics, such as in (COSTA; FDIDA; DUARTE, 2000), who in the early 2000's already demonstrated the feasibility of the single mixed metric in terms of processing requirements, which is a relevant concern.

Furthermore, although the exposed in *ii* is entirely true, the absence of a composition rule *per se* does not rule out the possibility of the feasibility of shortest path calculation mechanisms based on the single mixed metric, like Dijkstra algorithm, as we demonstrated in Chapter 3: simply put, the existence of a coherent composition rule is not a pre-requisite to the adoption of a routing metric. In Section 5.4.1, we detail some of the papers in the research field. Those include mostly solutions to the Multi-Constrained Routing problem based on single mixed metrics.

5.4.1 Selected Papers

In (COSTA; FDIDA; DUARTE, 2000), *"Developing scalable protocols for three-metric QoS routing"*, it is presented SMM, a single mixed metric based on propagation delay (d_p), available bandwidth, and logarithmic transmission success ($slog$), where the cost is $slog+d_p$. The metric is evaluated on link-state (LS) and distance-vector (DV) routing protocols (called SMM-LS and SMM-DV, respectively). To showcase the metric feasibility, an heuristic based on residual loss probability and metric combination is employed to achieve a scalable and polynomial running

time algorithm. Furthermore, as the title suggests, it goes beyond the metric scope, analyzing other aspects of the routing processes involved, such as the count-to-infinite problem of DV routing protocols, processing requirements, and link failure procedures.

In the second part of the work (BADIS; AGHA, 2003), the authors employ the single mixed metric approach to tackle the Delay, Bandwidth, and Loss probability Constrained Least Hop path problem (DBLCLH). Specifically, as a part of the heuristic solution proposed, they make use of the combination of logarithmic transmission success function ($slog$) and delay (d_p), where the cost is $slog + d_p$, just like in (COSTA; FDIDA; DUARTE, 2000).

"A New Method for Multi-Constraint QoS Routing" (KHADIVI; SAMAVI; SAIDI, 2003) presents a heuristic solution to the MCR problem based on the single mixed metric approach. For the case of two constraints (e.g, $w(e) = d_1 w_1 + d_1 w_1$, where $w(e)$ is the mixed metric, w_1 and w_2 are the single metrics, and d_1 and d_2 are constants), the mixed metric is a linear combination of the two base metrics, and is based on Jaffe's algorithm (JAFFE, 1984). The novelty introduced in the work is that extending Jaffe's consideration of link weights, they also consider the variation of link weights.

A handful of Genetic Algorithm approaches are found in the literature with respect to QoS routing (KOYAMA et al., 2004). Of those, ARGAQ (Adaptive Routing based on Genetic Algorithm for QoS) (BAROLLI et al., 2003), yet another multi-objective optimization scheme based on heuristics, aiming to improve on previous work, employs the single mixed metric: $T = \frac{\sum_{i=1}^{n} DT_i}{\prod_{i=1}^{n} TSR_i}$, where T is metric, DT is the delay time, and TSR is the transmission success rate. The gene coding scheme is based on Tree Junctions.

The work of Majd and Yaghmaee (MAJD; YAGHMAEE, 2006) presents FMM (Fuzzy Mixed Metric), a single mixed metric built upon a fuzzy-based heuristic based on a modified Dijkstra algorithm. The goal of the metric, which relies on available bandwidth, propagation delay, and the logarithmic transmission success function, is to select routes in IP/MPLS networks, aiming at fulfilling QoS requirements. Though the mechanism employed is not described in sufficient detail, the simulations carried out by the authors suggest an incremental bandwidth performance improvement upon previous related work.

In (ZHANG et al., 2006), it is presented an OSPFv3 single metric to IPv6 networks called LUR (Link Utilization Ratio, which is the ratio between the link utilized bandwidth and the link capacity). Apart from the explicit QoS concern, an important aspect of the metric that distinguishes it from the previously work described in this Section is its architecture designed to integrate with OSPFv3 with ease (*e.g.*, LSA flooding uses only one metric field of the control packets). Although the authors here explicitly state the metric is not single mixed, but rather a

single metric, we argue that this is disputable: since available bandwidth and reference bandwidth are widely considered as single metrics, from another point of wiew, the LUR can also be classified as a single mixed metric.

In (LIU; FENG, 2007), a single mixed metric based on Simulated Annealing (SARA) is proposed to tackle the MCR problem in MANETs (Mobile Ad-Hoc Networks), because prior-to-the-work existing methods for dealing with the problem in wired networks are deemed by the authors not able to solve it effectively. The scheme employed to do so is a heuristic based on an energy function that translates multiple QoS metrics into a single mixed metric and Markov chain. In terms of performance, it is described as especially good for networks not very large in scale.

Abramkina and Shuvalov (ABRAMKINA; SHUVALOV, 2016) propose another solution to the MCR problem with the employment of a single mixed metric. It is stated that their solution is based on Lagrangian Relaxation (specifically, SMM-LR, an acronym for single mixed Metric with Lagrangian Relaxation), but, in our opinion, the text overall is very confusing, as well as the results.

In (TANTISARKHORNKHET; WERAPUN, 2016), it is presented a QoS routing algorithm for SDN environments based on a single mixed metric. With the use of the function $f(p) = \frac{b(p)}{d(p) \times l(p)}$, where $b(p)$, $d(p)$ and $l(p)$ represents bandwidth, delay and packet loss, respectively, the metric employs a modified version of Dijkstra algorithm and aims to select QoS parameters depending on the requirements of the applications (throughput, delay, jitter, packet loss) according to quantized levels of service (low, medium, high).

5.5 TCP Performance

TCP performance is a vast research field. Therefore, we restrained ourselves to present the categories of SDN and TCP (Section 5.5.1) and TCP in *High-Speed Lossy Networks* (Section 5.5.2) due to its proximity to our proposal, and described some of the related literature.

5.5.1 SDN and TCP

It is shown that MPTCP performance could be enhanced in SDN environments aided by SDN controllers which can dynamically control TCP subflows based on the available capacity of the routes (NAM; CALIN; SCHULZRINNE, 2016). Another MPTCP scheme for IoT networks in SDN (IWATA; ITO, 2017) showed that throughput of proposed method was higher than normal

TCP up to a certain delay difference between the available paths but lower if this difference was greater than a certain threshold. The authors proposed the switching of forwarding methods triggered by the delay difference between the paths, or, alternatively, delay outgoing packets to reduce the delay difference between the paths.

Motivated by two specific drawbacks of current MPTCP implementation (fixed number of subflows and high dependency on ECMP random hashing) (DUAN; WANG; WU, 2015), a responsive MPTCP system was proposed for simultaneously solving both limitations in SDN-based Data centers. The solution consists on a centralized SDN controller for smart subflow route calculation and a monitor running on each server, aimed at actively adjust the number of subflows.

The paper of Singh et al. (SINGH et al., 2019) addresses TCP performance in Wi-Fi networks with SDN-based mechanisms which avoid TCP sender from entering its congestion control mode, without violation of end-to-end semantics in the protocol. Per definition, TCP assumes occurrence of congestion if packets are dropped in the end-to-end communication process between hosts. Although a reasonable approach in wired networks, in wireless environments packets are more likely to be dropped due to transmission factors rather than traffic congestion.

An SDN method in (BINDER; BOROS; KOTULIAK, 2015) performs TCP handoff that achieves faster handoff times in comparison to classic redirection methods based on application-level mechanisms. TCP handoff is a mechanism to non-disruptively handoff one TCP socket in an endpoint from one host to the other, that is, to migrate the established TCP state from the original host to a new one, in such a way the new host can directly communicate with the other TCP endpoint.

In (SONG et al., 2016), it is presented an SDN congestion avoidance algorithm. The algorithm checks the bandwidth utilization of each switch port in a given route and diverts flow traffic to another path based on a threshold utilization is higher than an established threshold (70% as default). Whenever the utilization of the original path is reduced to 50% or less, the original path is put into use again. According to the authors, their approach brought improvements in jitter, delay, and throughput in high congestion-prone networks.

For IoT applications in Data Centers, an SDN-based TCP congestion control mechanism (SDTCP) is presented in (LU et al., 2017). The main idea is to classify flows as burst or background flows and reduce the sending rate of established (background) flows to guarantee traffic of burst flows, which are usually presented as urgent ones. The reduction was implemented on the advertised receiving window of TCP ACK packets.

Omniscient TCP (OTCP), a congestion avoidance mechanism in SDN Data Center networks that configures TCP congestion parameters based on route information such as delay, buffering, throughput and topology, is presented in (JOUET; PERKINS; PEZAROS, 2016), as well as a survey on detection and mitigation of congestion in SDN-based Data Center networks, including, among other mechanisms, the aforementioned SDTCP and OTCP.

In (BAO et al., 2018), it is presented Explicit Centralized congestion avoidance mechanism (ECTCP). It works by keeping track of TCP flow information in the controller and proper allocation of bandwidth to flows in the network.

Finally, in (LAI et al., 2017) it is presented an analytical model for performance analysis of TCP flows in SDN networks, that is, a model which takes TCP flows into consideration, instead of a packet-level model of performance analysis.

5.5.2 TCP in High-Speed Lossy Networks

Besides the proposal of schedulers for Multipath TCP (MPTCP) in highly lossy networks (like LAMPS (DONG et al., 2019)) and the development of a couple of multipurpose TCP variants with the goal of addressing longstanding issues of TCP in specific scenarios (notably, the case of wireless networks), which we consider as a related subject, literature is scarce regarding the problem of TCP performance specifically in *High-Speed Lossy Networks* (*i.e.*, both high speed and lossy).

Of the few work found in the literature that addresses the aforementioned problem, there are mechanisms such as DVPTCP (Dynamic Virtual Parallel TCP) (SU et al., 2019), a delay-based TCP congestion control algorithm that adjusts the number of virtual parallel streams dynamically; this, to some extent, is analogous to the employment of multiple standard TCP streams to increase throughput, but taking multiple round-trip times (*RTT*) of multiple routes into consideration. In (SHI et al., 2009), to tackle the same problem, it is presented RACC (Receiver Assistant Congestion Control mechanism), a TCP congestion control mechanism that combines loss-based and delay-based features. (LI et al., 2021) proposes Sphinx, a novel transport protocol alternative to TCP that addresses the problem of throughput performance in mobile *High-Speed Lossy Networks*, *i.e.*, mobility is also taken into consideration.

In a sense, we consider that our work is complementary to such efforts because regardless of the scheme employed in the Transport Layer, the chosen routes will allow higher throughput to the individual TCP sub-flows (*e.g.*, for DVPTCP, with TCPBP, a lower number of sub-flows may be sufficient to obtain the same throughput of DVPTCP with routes computed with traditional

metrics).

5.6 Practical Uses of the *Inverse-Square Root Law*

The work of (BASSO et al., 2012), described in Section 2.5.4, presents an application-level approach for the measurement of packet loss rate in a TCP data transfer that is based on the *inverse-square root law*, with the results deemed as accurate. Regarding this work, it is particularly interesting to note that it needs a more accurate TCP throughput prediction model than ours. This is so because we do not need to present specific and accurate values for d and p, only provide the route that achieves the best TCP performance according to the TCP throughput prediction, *i.e.*, the route with the best $d\sqrt{p}$ value. Here, an analogy can be made as follows: if one wants to measure the actual speed of a car on a highway (for means of possibly fining an offending driver), it is desirable to obtain a value closer to the actual speed in *km/h*, but if we have two or more cars competing against each other at an illegal car racing competition, measuring the speed of the cars is not necessary for one to say who is going faster (in fact, such task can be done with the eyesight, unless speeds are very close).

In (LAN et al., 2021), it is presented "*Packet Loss Measurement Based on Sampled Flow*" (PLMBSF), a model for packet loss rate (PLR) measurement based on the *inverse-square root law*, TCP network flow sampling (that can be obtained, for example, with *NetFlow*), and statistical regression analysis. Based on the experimental validation, it is argued that the method is accurate compared with active trace-based loss measurement. Because the authors also claim their proposal suits to "real-time requirements of current high-speed backbone performance monitoring", we believe their method can be improved with the usage of the linear regime for TCP throughput instead of the *inverse-square root law*.

6 | Conclusion

In this thesis, we examined the problem of route selection taking into account the throughput provided by the Transport Layer with TCP in *High-Speed Lossy Networks* in the face of delay and packet loss. Specifically, we proposed the utilization of a paradigm that relies on a single mixed metric based on end-to-end delay (d) and packet loss (p) (TCPBP), which aims to optimize single-path TCP throughput *per-route*. The basic intuition of this approach is that the *per-flow* throughput performance in such cases is proportional to $1/f(d,p)$, so, finding the adequate model for $f(d,p)$ and routes that minimize it may yield optimal (or at least better than of existent metrics) throughput-wise paths if the available bandwidth is not an issue and other conditions stated in Section 3.6.2 are met. In the specific case of TCPBP, we have shown that the *inverse-square root model* applies, therefore $f(d,p) = \frac{1}{d\sqrt{p}}$.

We also demonstrate that, in an emulated SDN *High-Speed Lossy Network*, TCPBP provides better TCP throughput performance in comparison to pure delay and *hop-count* metrics. Thus, TCPBP proved feasible in the sense of providing routes whenever link-state information can be properly measured and reliably fast enough retrieved by the Control Plane (in the case of our proof-of-concept, the SDN controller), to ensure good convergence behavior, regardless of the network topology.

In some networks, however, TCPBP may not present relevant benefits in the face of the increased complexity of the protocol (*e.g.*, because routes found with classic metrics: (1) already provide maximum TCP throughput, as could be the case of delay; (2) satisfy QoS criteria; or (3) because TCP throughput performance is not critical), or even provide worse routes with less available bandwidth. Since the single mixed metric proposed does not take bandwidth (either available or consumption) into consideration, such issues should be treated separately, as is the case of, for example, *Capacity Based Routing* (HAN, 2011). Furthermore, depending on violations of the assumptions in which the *inverse-square root model* is based, as is the case of high packet loss ($p \gg 5\%$), or low packet loss (in which the linear regime applies, and the estimation of packet loss with a passive approach may be more complicated to perform), the TCPBP metric may not produce the best paths for TCP performance, as discussed in Section 2.5.3.

As for the proof-of-concept, we evaluated the TCPBP on SDN environments for their prospects in the future of computer networks: SDN can provide the advantages of centralized programming of a routing protocol running as an API in a centralized or distributed hardware with far greater capabilities (*e.g.*, CPU and memory) than ordinary forwarding devices, thus allowing the usage of sophisticated and tailor-made QoS routing protocols that are more space and time-wise

expensive algorithms than usual, including the possibility of parallelism. For so, it is suitable for the adoption of routing metrics and protocols that can account for the peculiarities of a given network topology, and we conclude that TCPBP matches very well with SDN.

Though we presented a novel shortest path calculation mechanisms (modified Dijkstra) for link-state protocols, *i.e.*, based on algorithms that work with a complete view of the network topology, we should emphasize that TCPBP is a metric that satisfies the conditions of monotonicity and isotonicity, not a routing protocol. Therefore, it is suitable for any routing scheme (proactive or reactive routing; link-state or vector-distance protocols; and source routing or hop-by-hop routing) if delay and packet loss estimation of the links in the network can be periodically retrieved with sufficient precision and diffused to the Control Plane.

From the theoretical exposition and experimental evaluation, TCPBP seems to be a feasible approach, as the proof-of-concept results show throughput improvement in *steady-state* flows consonant with the TCP throughput prediction model itself, suggesting that TCPBP use should be considered whenever it is possible, and its effects are desirable. Given a flexible SDN environment, for example, TCPBP on top of default OSPF metric can be evaluated with minimal intrusion on network traffic, which is possible in many ways, such as with flow discrimination in the flow table rules (*e.g.*, installing TCPBP to provide route metric to specific flows, and evaluate the effects on throughput performance).

Nevertheless, to present a more detailed account of TCPBP performance and its possibilities of usage, further investigation and future works regarding specific cases are needed, like throughput and convergence performance evaluation with a greater number of forwarding devices and links, on specific topologies – *Wireless Mesh Networks*, *Low Power and Lossy Networks*, or ISP backbones, for example –, and asymmetric routing.

6.1 Future Work

Opportunities for research in future works are described below:

i. **practical aspects of TCPBP in real networks**: our work provided guidelines for TCPBP adoption, presenting only a basic proof-of-concept based on a simulated environment. Therefore, the implementation of TCPBP in real networks is necessary to assess its performance and discuss practical aspects. For example, although d and p are prone to change with respect to bandwidth utilization in the case of WMNs, in practice, it is not clear how much impact it has on the TCPBP utilization in such networks;

ii. **bandwidth issues in TCPBP**: the adoption of a metric that is based on quality aspects like delay or packet loss may repel traffic from links deemed "poor", or, conversely, attract traffic to links deemed "good". This is a good thing if congestion can be prevented with other measures, but, in any case, TCPBP raises a concern involving bandwidth consumption;

iii. **TCPBP performance with *non-persistent* flows**: HTTP is one of the most commonly used protocols on the Internet and mainly uses non-persistent TCP flows to retrieve WEB pages, so we consider it an important subject of research. Therefore, another research focus could be TCPBP performance evaluation in the transient state (*non-persistent* flows). Even though such flows, according to the literature, comply with the *inverse-square root model* assumptions, it was not evaluated in the present work;

iv. **route stability**: further research on route stability mechanisms (like dampening or hysteresis algorithms) is needed to avoid unnecessary re-convergences resulting from the adoption of TCPBP. For example, TCPBP may produce one or more routes with an arbitrarily close value of $d\sqrt{p}$ relative to the best route. In contrast, this difference may not be statistically significant (meaning that there are two or more best routes); this may trigger re-convergence not because of changes in the network dynamics but by measurements that may differ from time to time, possibly resulting in routing instability. In this case, both routes with a close value of $d\sqrt{p}$ can be used to balance traffic in the network, or a threshold value to classify the routes may be desirable;

v. **interdomain routing**: TCPBP is a routing metric, not a traffic engineering solution. Since a lot (if not most) TCP flows on the Internet occur between different network administrative domains (*e.g.*, interdomain routing between different autonomous systems) with routes defined not only by intradomain protocols, but also by BGP, where most networks are not lossy, it is possible to evaluate the usage of BGP extensions similar to BGP-LS (GINSBERG et al., 2019) to propagate key values of $d\sqrt{p}$;

vi. **link-state measurement and estimation**: although in this work we have used passive measurement and simple moving averages to perform link estimation of the emulated parameters, active measurement alongside other more sophisticated link estimation techniques can be considered to account for link quality variability over time and one-way delay proper computation in the context of the TCPBP. Since the appropriate link-state measurement is critical to ensure the proper calculation of $d\sqrt{p}$, thus critical to the TCPBP metric operation, we consider it crucial;

vii. **energy consumption due to re-transmissions**: in some networks, energy consumption

is a critical factor. If TCP or other reliable connection-oriented transport protocol is used in such environments, avoiding re-transmissions may be interesting to reduce power consumption; thus, there may be the possibility of the employment of TCPBP alongside other mechanisms to save energy;

viii. **performance enhancement of other transport protocols**: the possibility of extending the TCPBP principle to other transport layer protocols, such as QUIC, upon the establishment of a throughput prediction model, seems to be a possibility worth investigating;

ix. **comparison with other metrics**: although we have compared TCPBP to delay and hop-count metrics in the proof-of-concept, because of the prevalence of those metrics in most intradomain networks nowadays, a comparison of TCPBP and some of the metrics shown in Section 5.4.1, such as SMM-LS or SMM-DV, could also be a potential future work.

REFERENCES

ABDELMONIEM, Ahmed M; BENSAOU, Brahim. Hysteresis-based Active Queue Management for TCP Traffic in Data Centers. **IEEE Conference on Computer Communications**, p. 1621–1629, 2019.

ABRAMKINA, Olga A; SHUVALOV, Vyacheslav P. About One of the Methods for Dolving Problems in QoS Routing. **2016 International Siberian Conference on Control and Communications (SIBCON)**, p. 1–4, 2016.

ALMES, Guy et al. A One-way Delay Metric for IP Performance Metrics (IPPM). **IETF, January**, v. 10, 2016.

ALMES, Guy et al. A One-way Loss Metric for IP Performance Metrics (IPPM). **RFC-Internet Standard, RFC 7680**, 2016.

APOSTOLOPOULOS, George et al. QoS Routing Mechanisms and OSPF Extensions. **RFC 2676, August**, 1999.

AYAR, Tacettin et al. Emulation and Performance Evaluation of a Transparent Reordering Robust TCP Proxy. **2019 IEEE 8th International Conference on Cloud Networking (CloudNet)**, p. 1–3, 2019.

BACCELLI, François; MCDONALD, David R. A Square Root Formula for the Rate of Non-Persistent TCP Flows. **Next Generation Internet Networks, 2005**, p. 247–254, 2005.

BADIS, Hakim; AGHA, Khaldoun Al. A Distributed Algorithm for Multiple-metric Link State QoS Routing Problem. **Mobile And Wireless Communications Networks: (With CD-ROM)**, World Scientific, p. 141–144, 2003.

BAO, Jiannan et al. ECTCP: An Explicit Centralized Congestion Avoidance for TCP in SDN-based Data Center. **2018 IEEE Symposium on Computers and Communications (ISCC)**, p. 00347–00353, 2018.

BARABÁSI, Albert-László. Network Science. **Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences**, The Royal Society Publishing, v. 371, n. 1987, p. 20120375, 2013.

BAROLLI, Leonard et al. A Genetic Algorithm Based QoS Routing Method for Multimedia Communications Over High-Speed Networks. **IPSJ Journal**, v. 44, n. 2, p. 544–552, 2003.

BASSO, Simone et al. Estimating Packet Loss Rate in the Access Through Application-Level Measurements. **Proceedings of the 2012 ACM SIGCOMM workshop on Measurements up the stack**, p. 7–12, 2012.

BHATIA, Hartman et al. Security Extension for OSPFv2 When Using Manual Key Management. **IETF**, 2015.

BINDER, Andrej; BOROS, Tomas; KOTULIAK, Ivan. A SDN Based Method of TCP Connection Handover. **Information and Communication Technology-EurAsia Conference**, p. 13–19, 2015.

COHEN, Dudy. **The Challenge of Modern Wireless Backbone Networks. Available at: https://www.ceragon.com/blog/challenge-of-modern-wireless-backbone-networks. Accessed in: 02-23-2022**. [S.l.], 2020.

COMER, Douglas E. Internetworking with TCP/IP. **Pearson**, 2014.

CORMEN, Thomas H et al. Introduction to Algorithms. **MIT press**, 2009.

COSTA, Luís Henrique MK; FDIDA, Serge; DUARTE, Otto Carlos MB. Distance-Vector QoS-based Routing with Three Metrics. **International Conference on Research in Networking**, p. 847–858, 2000.

DAS, Abhijit; CHAUDHURI, Atal. Derivation and Simulation of an Efficient QoS Scheme in MANET Through Optimised Messaging Based on ABCO Using QualNet. **Handbook of Research on Swarm Intelligence in Engineering**, IGI Global, p. 507–536, 2015.

DIJKSTRA, Edsger W et al. A Note on Two Problems in Connection with Graphs. **Numerische Mathematik**, v. 1, n. 1, p. 269–271, 1959.

DONG, Enhuan et al. A Loss Aware MPTCP Scheduler for Highly Lossy Networks. **Computer Networks**, Elsevier, v. 157, p. 146–158, 2019.

DUAN, Jingpu; WANG, Zhi; WU, Chuan. Responsive multipath TCP in SDN-based datacenters. **2015 IEEE International Conference on Communications (ICC)**, p. 5296–5301, 2015.

FALL, Kevin R; STEVENS, W Richard. TCP/IP Illustrated, Volume 1: The Protocols. **Addison-Wesley**, 2011.

FLOYD, Sally et al. Metrics for the Evaluation of Congestion Control Mechanisms. **RFC 5166, March**, 2008.

FOUNDATION, Linux. **Netem**. 12 Oct. 2021. Available from: <https://wiki.linuxfoundation.org/networking/netem>. Visited on: 14 Apr. 2022.

FRANSSON, Pierre; CARR-MOTYČKOVÁ, Lenka. Brief Announcement: An Incremental Algorithm for Calculation of Backup-Paths in Link-State Networks. **Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing**, p. 381–381, 2004.

GAO, Shang et al. Security Threats in the Data Plane of Software-Defined Networks. **IEEE network**, IEEE, v. 32, n. 4, p. 108–113, 2018.

GINSBERG, Les et al. **BGP-Link State (BGP-LS) Advertisement of IGP Traffic Engineering Performance Metric Extensions**. [S.l.], 2019.

GUCK, Jochen W et al. Unicast QoS Routing Algorithms for SDN: A Comprehensive Survey and Performance Evaluation. **IEEE Communications Surveys & Tutorials**, IEEE, v. 20, n. 1, p. 388–415, 2017.

HA, Sangtae; RHEE, Injong; XU, Lisong. CUBIC: A New TCP-Friendly High-Speed TCP Variant. **ACM SIGOPS operating systems review**, ACM New York, NY, USA, v. 42, n. 5, p. 64–74, 2008.

HAN, Bo. **Cognitive Routing for Wireless Ad Hoc Networks**. 2011. PhD thesis – University of York.

HARTMAN; ZHANG. RFC 6863 – Analysis of OSPF Security According to the Keying and Authentication for Routing Protocols (KARP) Design Guide. **IETF**, 2013.

HEATH, Thomas Little et al. The Thirteen Books of Euclid's Elements. **Courier Corporation**, 1956.

IWATA, Kaori; ITO, Yoshihiro. Proposal and Development of TCP Multi-Pathization Method With SDN by IoT Devices. **2017 27th International Telecommunication Networks and Applications Conference (ITNAC)**, p. 1–6, 2017.

JAFFE, Jeffrey M. Algorithms for Finding Paths with Multiple Constraints. **Networks**, Wiley Online Library, v. 14, n. 1, p. 95–116, 1984.

JAIN, Rajendra K; CHIU, Dah-Ming W; HAWE, William R, et al. A Quantitative Measure of Fairness and Discrimination. **Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA**, 1984.

JOUET, Simon; PERKINS, Colin; PEZAROS, Dimitrios. OTCP: SDN-managed Congestion Control for Data Center Networks. **NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium**, p. 171–179, 2016.

KAWADIA, Vikas; KUMAR, Panganamala Ramana. A Cautionary Perspective on Cross-Layer Design. **IEEE Wireless communications**, IEEE, v. 12, n. 1, p. 3–11, 2005.

KENYON, Tony. **High Performance Data Network Design: Design Techniques and Tools**. [S.l.]: Digital Press, 2002.

KFOURY, Elie F et al. An Emulation-Based Evaluation of TCP BBRv2 Alpha for Wired Broadband. **Computer Communications**, Elsevier, v. 161, p. 212–224, 2020.

KHADIVI, Pejman; SAMAVI, Shadrokh; SAIDI, Hosein. A New Method for Multi-Constraint QoS Routing. **IEEE Iranian Conference on Electrical Engineering**, IEEE, 2003.

KNUTH, Donald E. Big Omicron and Big Omega and Big Theta. **ACM Sigact News**, ACM New York, NY, USA, v. 8, n. 2, p. 18–24, 1976.

KOYAMA, Akio et al. A GA-Based Multi-Purpose Optimization Algorithm for QoS Routing. **18th International Conference on Advanced Information Networking and Applications, 2004. AINA 2004.**, v. 1, p. 23–28, 2004.

KREUTZ, Diego et al. Software-Defined Networking: A Comprehensive Survey. **Proceedings of the IEEE**, IEEE, v. 103, n. 1, p. 14–76, 2014.

KUIPERS, Fernando et al. An Overview of Constraint-Based Path Selection Algorithms for QoS Routing. **IEEE Communications Magazine**, IEEE, v. 40, n. 12, p. 50–55, 2002.

KUROSE, James; ROSS, Keith. Computer Networks: A Top Down Approach Featuring The Internet. **Pearson Addison Wesley**, 2010.

LAI, Yuan-Cheng et al. Performance Modeling and Analysis of TCP Connections Over Software Defined Networks. **GLOBECOM 2017-2017 IEEE Global Communications Conference**, p. 1–6, 2017.

LAN, Haoliang et al. Packet Loss Measurement Based on Sampled Flow. **Symmetry**, Multidisciplinary Digital Publishing Institute, v. 13, n. 11, p. 2149, 2021.

LI, Junfeng et al. Sphinx: A Transport Protocol for High-Speed and Lossy Mobile Networks. **Computer Networks**, Elsevier, p. 108193, 2021.

LIU, Lianggui; FENG, Guangzeng. Simulated Annealing Based Multi-Constrained QoS Routing in Mobile Ad Hoc Networks. **Wireless Personal Communications**, Springer, v. 41, n. 3, p. 393–405, 2007.

LU, Yifei et al. SDTCP: Towards Datacenter TCP Congestion Control with SDN for IoT Applications. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 17, n. 1, p. 109, 2017.

LUKASEDER, Thomas et al. A Comparison of TCP Congestion Control Algorithms in 10G Networks. **2016 IEEE 41st Conference on Local Computer Networks (LCN)**, p. 706–714, 2016.

MAJD, Nahid Ebrahimi; YAGHMAEE, Mohammad Hossien. A Fuzzy Algorithm for QoS-based Routing in MPLS Network. **2006 Asia-Pacific Conference on Communications**, p. 1–5, 2006.

MANZOOR, Jawad et al. On the Performance of QUIC Over Wireless Mesh Networks. **Journal of Network and Systems Management**, Springer, v. 28, n. 4, p. 1872–1901, 2020.

MATHIS, Matthew et al. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. **ACM SIGCOMM Computer Communication Review**, ACM New York, NY, USA, v. 27, n. 3, p. 67–82, 1997.

MITZENMACHER, Michael; RAJARAMAN, Rajmohan. Towards More Complete Models of TCP Latency and Throughput. **The Journal of Supercomputing**, Springer, v. 20, n. 2, p. 137–160, 2001.

MOY, J. RFC 2328 – OSPF Version 2, 1998. Internet Engineering Task Force. **IETF**, 1998.

NAM, Hyunwoo; CALIN, Doru; SCHULZRINNE, Henning. Towards Dynamic MPTCP Path Control Using SDN. **2016 IEEE NetSoft Conference and Workshops (NetSoft)**, p. 286–294, 2016.

PADHYE, Jitendra et al. Modeling TCP Throughput: A Simple Model and its Empirical Validation. **Proceedings of the ACM SIGCOMM'98 conference on Applications, technologies, architectures, and protocols for computer communication**, p. 303–314, 1998.

PASSOS, Diego. Flow-based Interference-Aware Routing in Multihop Wireless Networks. **Ph. D. dissertation**, 2013.

POLESE, Michele et al. A Survey on Recent Advances in Transport Layer Protocols. **IEEE Communications Surveys & Tutorials**, IEEE, v. 21, n. 4, p. 3584–3608, 2019.

POOJARY, Sudheer; SHARMA, Vinod. Analysis of Multiple Flows Using Different High Speed TCP Protocols on a General Network. **Performance Evaluation**, Elsevier, v. 104, p. 42–62, 2016.

POSTEL, Jon et al. RFC 793 – Transmission Control Protocol. **IETF**, 1981.

SHAND, Mike; GINSBERG, Les. Reclassification of RFC 1142 to Historic. **Relatório técnico**, 2014.

SHI, Kai et al. Receiver Assistant Congestion Control in High Speed and Lossy Networks. **2009 16th IEEE-NPSS Real Time Conference**, p. 91–95, 2009.

SILVA, Bruno; PASSOS, Diego; ALBUQUERQUE, Célio. Reducing the Variability in Routing Decisions in Wireless Mesh Networks. **2018 IEEE Symposium on Computers and Communications (ISCC)**, p. 00504–00507, 2018.

SILVA, Ricardo Bennesby da; MOTA, Edjard Souza. A Survey on Approaches to Reduce BGP Interdomain Routing Convergence Delay on the Internet. **IEEE Communications Surveys & Tutorials**, IEEE, v. 19, n. 4, p. 2949–2984, 2017.

SINGH, Krishna Vijay et al. Improving Performance of TCP for Wireless Network Using SDN. **Proceedings of the 20th International Conference on Distributed Computing and Networking**, p. 267–276, 2019.

SOBRINHO, João L. Algebra and Algorithms for QoS Path Computation and Hop-by-Hop Routing in the Internet. **Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)**, v. 2, p. 727–735, 2001.

SONG, Seungbeom et al. A Congestion Avoidance Algorithm in SDN Environment. **2016 International Conference on Information Networking (ICOIN)**, p. 420–423, 2016.

SU, Bo et al. DVPTCP: A Delay-Driven Virtual Parallel TCP for High-Speed and Lossy Networks. **IEEE Access**, IEEE, v. 7, p. 99746–99753, 2019.

TANTISARKHORNKHET, Piyawit; WERAPUN, Warodom. QLB: QoS Routing Algorithm for Software-Defined Networking. **2016 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)**, p. 1–6, 2016.

TOMOVIC, S; RADUSINOVIC, I. A New Traffic Engineering Approach for QoS Provisioning and Failure Recovery in SDN-based ISP Networks. **2018 23rd International Scientific-Professional Conference on Information Technology (IT)**, p. 1–4, 2018.

VILELA, Hugo; V.O FONSECA, Keiko; FONSECA, Mauro. TCPBP: Enhancing the TCP Throughput in High Speed Lossy Networks With a Single Mixed Routing Metric. **Anais do XL Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos**, SBC, Porto Alegre, RS, Brasil, p. 266–279, 2022. ISSN 2177-9384. DOI: 10.5753/sbrc.2022.222306.

WANG, Yunlong; YAN, Zhongjiang, et al. Survey of Routing Metric in Wireless Mesh Networks. **International Conference on Smart Grid and Internet of Things**, p. 307–325, 2020.

WANG, Zheng. On the Complexity of Quality of Service Routing. **Information Processing Letters**, Elsevier, v. 69, n. 3, p. 111–114, 1999.

WANG, Zheng; CROWCROFT, Jon. Quality-of-Service Routing for Supporting Multimedia Applications. **IEEE Journal on selected areas in communications**, IEEE, v. 14, n. 7, p. 1228–1234, 1996.

WHITE, Russ; SLICE, Don; RETANA, Alvaro. Optimal Routing Design. **Pearson Education India**, 2005.

WOO, Alec; CULLER, David E. Evaluation of Efficient Link Reliability Estimators for Low-Power Wireless Networks. **Computer Science Division, University of California Oakland, Calif, USA**, 2003.

WOO, Alec; TONG, Terence; CULLER, David. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. **Proceedings of the 1st international conference on Embedded networked sensor systems**, p. 14–27, 2003.

XIE, Junjie et al. Control Plane of Software Defined Networks: A survey. **Computer communications**, Elsevier, v. 67, p. 1–10, 2015.

XU, Lisong; HARFOUSH, Khaled; RHEE, Injong. Binary increase congestion control (BIC) for fast long-distance networks. **IEEE INFOCOM 2004**, v. 4, p. 2514–2524, 2004.

YANG, Yaling; WANG, Jun. Design Guidelines for Routing Metrics in Multihop Wireless Networks. **IEEE INFOCOM 2008-The 27th Conference on Computer Communications**, p. 1615–1623, 2008.

YEN, Jin Y. Finding the k-Shortest Loopless Paths in a Network. **Management Science**, Informs, v. 17, n. 11, p. 712–716, 1971.

YUE, Zhaojuan et al. The Performance Evaluation and Comparison of TCP-Based High-Speed Transport Protocols. **2012 IEEE International Conference on Computer Science and Automation Engineering**, p. 509–512, 2012.

ZARAGOZA, D. Challenging the Square-Root Law for TCP Send-Rate. **Electronics Letters**, IET, v. 42, n. 24, p. 1430–1431, 2006.

ZHANG, Jing et al. QoS Routing in IPV6 With Link-Utilization-Ratio Single Metric. **2006 International Conference on Wireless Communications, Networking and Mobile Computing**, p. 1–5, 2006.