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

Hugo Christ Vilela[1], **Keiko Verônica Ono Fonseca**[1], **Mauro Sérgio Pereira Fonseca**[1]

[1] Universidade Tecnológica Federal do Paraná (UTFPR)
Curitiba – PR – Brazil

hchrist85@gmail.com,{keiko,maurofonseca}@utfpr.edu.br

Abstract. *In this work, we present Transmission Control Protocol Best Path (TCPBP), an alternative link-state routing metric aims to optimize 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. In contrast, this function predicts the TCP throughput that should be maximized. As proof of concept, TCP bandwidth tests were executed in an SDN High Speed Lossy Network Linux-emulated topology with ten nodes, employing 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 overall TCP throughput performance obtained with TCPBP was greater than with different routes founded by traditional metrics, showing the potential of our proposal.*

1. Introduction

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 (Transmission Control Protocol) is still the most used transport protocol on the internet and in virtual private networks. So, its performance, especially regarding raw data transfer capacity, i.e., throughput, is a critical subject and primary object of concern to network engineers and researchers in the field of computer networks. When it comes to TCP throughput performance, it can be stated as a function of round-trip delay, packet loss, and available bandwidth (we discuss it in detail in section 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 (interdomain routing) [Kurose and Ross 2017], in which paths are computed 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 (that typically offer both Internet and Virtual Private Network services) often do not provide to 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 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.

In this work, we propose and evaluate the utilization of Transmission Control Best Path (TCPBP), a single mixed metric that maximizes the function of delay and packet loss that predicts the TCP throughput in intradomain *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.

The remainder of the paper is organized as follows. Section 2 summarizes fundamental concepts judged relevant to the presentation of our proposal. Section 3 describes the TCP throughput prediction model used to derive the proposed metric (TCPBP). The proposed metric is detailed in Section 4. The implementation details of TCPBP, performance evaluation, and experiment results and analysis are presented in Section 5. In Section 6 we present the related work. Finally, Section 7 concludes the paper.

2. Background

2.1. 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). Examples 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).

2.2. Routing Metrics Properties

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, a routing metric must satisfy the conditions of loop-freeness, consistency, and optimality [Yang and Wang 2008]. In the general case, it is proven in [Yang and Wang 2008] that isotonicity and monotonicity are sufficient properties that ensure routing protocols satisfy all three aforementioned conditions. The properties of loop-freeness, consistency, optimality, isotonicity, and monotonicity as we adopted are described in [Yang and Wang 2008].

2.3. Network Model and Concepts

(*i*) *Topology*: the network can be represented 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;

(*ii*) *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 $\rightarrow$).;

(*iii*) *Delay*: delay in this work may be: (a) the *round-trip delay* (RTT), or: (b) the *one-way delay* (d). The definition (a) is used whenever the values computed depend on round-trip delays, like in equation (1); shortest path computation, on the other hand, uses definition (b);

(*iv*) *Packet loss*: like delay, in this work, packet loss (p) has two definitions: (a) the round-trip packet loss and: (b) the (*one-way packet loss*). The employment of (a) or (b) is the same as in the case of delay. The usage of definitions (a) and (b) with the same symbol (p) also should be clear from the context.

3. TCP Throughput Predicition

Predictions of *steady-state* TCP throughput can be achieved based on packet loss and delay 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 TCP timeouts mechanism), its formula is also way more complicated. As a consequence of this (i.e., being simple), the work of Mathis et al. often suffices and is preferred. Equation (1) 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]:

$$BW_{max} = \frac{MSS}{RTT} C p^k \tag{1}$$

Realizing that the contributions of the Network Layer are contained in RTT and p, taking exponent k as -0.5 at Equation (1), 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 endpoints, BW_{max} can be stated as in (2):

$$BW_{max} \propto \frac{1}{RTT\sqrt{p}} \tag{2}$$

The proportion stated in (2) 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; also partly AIMD-based) [Ayar et al. 2019]. Furthermore, according to [Baccelli and McDonald 2005], inverse-square root proportionality is also valid with *non-persistent* flows. 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 et al. 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 and Sharma 2016], HSTCP and STCP [Yue et al. 2012].

4. TCPBP

Transmission Control Protocol Best Path (TCPBP) is a single mixed metric such that the $d\sqrt{p}$ per-route is minimized to maximize (2). Considering two nodes A and B in an intradomain 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 bandwidth of the links, suggesting that the adoption of our metric may enhance or even optimize the network *per-route* TCP throughput up to a certain number of TCP flows (n_f). Our proposal is also scalable and easy to deploy: 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 into the metric is that, 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, in particular). As a matter of fact, we consider that bandwidth issues should be treated separately (e.g., through pruning, or with traffic engineering).

4.1. 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 4.1.1 (isotonicity) and 4.1.2 (monotonicity).

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 $(d_1 + d_2 + ... + d_n + d_{u \to v})(1 - ((1 - p_1)(1 - p_2)...(1 - p_n)(1 - p_{u \to v})))^{1/2}$.

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

4.1.1. 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 $f(d_n, p_n) > 0 \; \forall \; n$.

4.1.2. 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.

4.2. Shortest Path Computation

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 and 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 an per-iteration extra time for the computation of $d\sqrt{p}$, following the computation of d and p, and an 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.

4.3. Expected 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 Eq. (1), 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).

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

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

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

5. Performance Evaluation

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* 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. 1. Specs of the machine used to run all the simulations are in Table 1.

5.1. Methodology and Parameters Evaluated

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. 2,

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

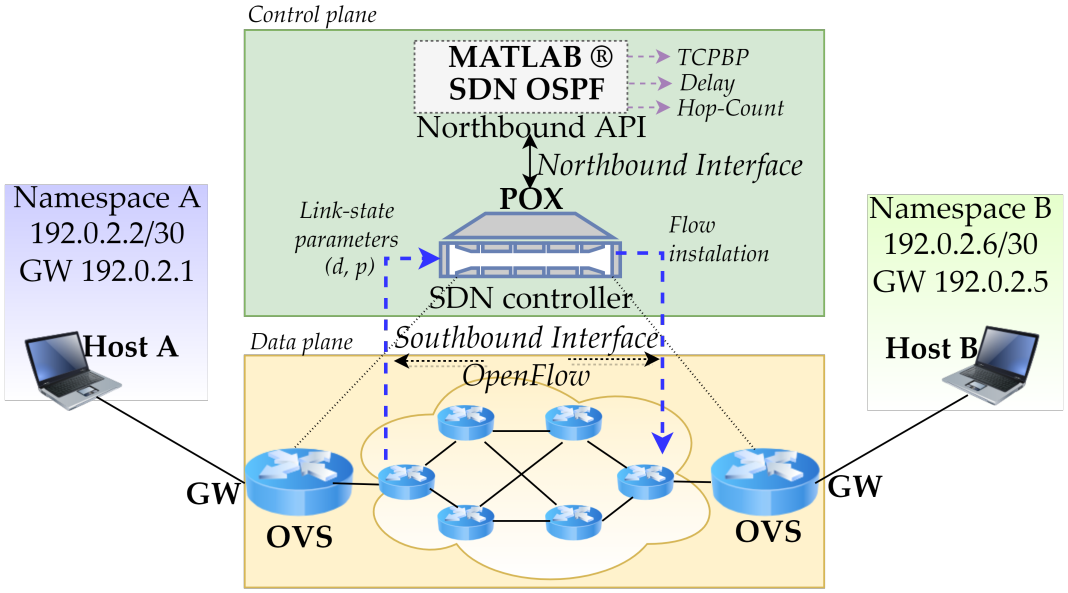

Figure 1. Linux OpenFlow Testbed. Host A communicates with Host B through Open vSwitches in the default Network Namespace. Communication between the POX controller and Open vSwitches uses OpenFlow as the standardized protocol for flow installation and maintenance; once a centralized Northbound API (MATLAB®) 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).

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 3 and 4. 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.

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®, which then calculates the best routes according to the correspondent metric, and instructs back POX to install the flows accordingly in each of the switches. The tests started at $t = 0s$. To gather the first link-state information, a random 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

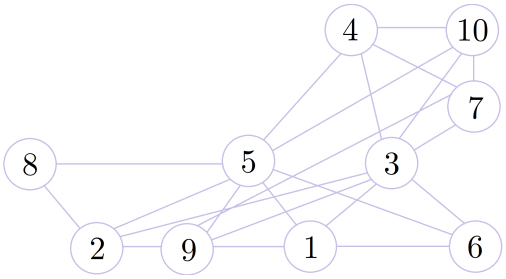

Figure 2. Graph representing the *High Speed Lossy Network* topology with ten nodes used in the proof-of-concept.

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

Figure 3. Delay adjacency matrix. Values in seconds. Each value was emulated with a $\pm 5\%$ uncertainty to provide a rudimentary link-state variability.

rest of the tests, we employed a simple moving average (SMA) of the last $30s$ every $10s$ (i.e., link-state values were updated every $10s$, with each value represented by the mean of the last three collected samples). Throughput, fairness, and TCPBP running time were evaluated from $t = 120s$ as stated below:

(i) **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. 1, 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 address 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;

(ii) **Fairness**: to measure fairness, we resort to the Jain's Index [Jain et al. 1984] 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;

(iii) **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 modi-

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

Figure 4. Packet loss adjacency matrix. Absolute values in the interval [0,1]. Each value was emulated with a $\pm 5\%$ uncertainty to provide a rudimentary link-state variability.

fied edge relaxation is not significantly slower than standard Dijkstra, being capable of running in common off-the-shelf hardware such as laptops.

5.2. 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.

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 we were able to determine (perhaps due to the *Central Limit Theorem* applied to the statistical distributions 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.

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 $\neq$ Hop-Count: 19
	TCPBP = Delay: 39	TCPBP $\neq$ 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 provide 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 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

6. Related Work

To the best of our knowledge, our approach is the only work that tackles the problem of throughput delivered by the Transport Layer by means of route selection (e.g., routing

metric); i.e., there is not any related work with this regard. Therefore, due to its relative proximity to our proposal, we classify related work in the following categories: *TCP-Based Solutions* (i.e., implementations on Transport Layer), *Single Mixed Metrics*, and *TCP Throughput Prediction: Practical Uses*.

6.1. TCP-Based Solutions

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).

6.2. Single Mixed Metrics

A single mixed metric is based on a function of multiple metrics, i.e., a metric that mixes different parameters into a single measure and employs it in routing decisions [Wang and Crowcroft 1996]. Despite being deemed as an approach that can, at best, serve as an indicator in path selection [Wang and Crowcroft 1996], we argue that this is valid in the context of Multi-Constrained Routing, but not in the general case: as shown in Section 5, a single mixed metric may be feasible even if it is the only criteria for path selection. Furthermore, subsequent work like [Costa et al. 2000], back in the early 2000's, already demonstrated the feasibility of the single mixed metric in terms of processing requirements, which is a relevant concern. Other work in the field includes mostly solutions to the Multi-Constrained Routing problem based on single mixed metrics (e.g., [Tantisarkhornkhet and Werapun 2016], [Badis and Agha 2003]).

6.3. TCP Throughput Prediction: Practical Uses

[Basso et al. 2012] 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 reported 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.

7. Conclusion

In this work, we have presented TCPBP, a link-state routing metric aiming to optimize single-path TCP throughput in the very particular case of *High Speed Lossy Networks*. From 5.2, it seems to be a promising approach, as the proof-of-concept results show considerable 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 (e.g., as a practical scenario, in private *Wide Area Networks* based on wireless backbones with one of the nodes connected to an *Internet Exchange Point – IX*).

TCPBP proved feasible in the sense of providing viable routes if link-state information can be adequately measured and reliably, fast enough retrieved by the Control Plane, to ensure good convergence behavior, regardless of the network topology; however, in some networks (e.g., lossy, but without the high speed feature; or high speed, but not lossy), TCPBP may not present relevant benefits in the face of the increased complexity of the protocol (e.g., because routes found with classic protocols already satisfy QoS criteria or because TCP throughput performance is not critical). Furthermore, depending on violations of the assumptions in which the *inverse-square root law* is based, as may be the case of arbitrarily short non-persistent TCP flows, and high packet loss ($p \gg 5\%$), the TCPBP metric may not produce the best paths for TCP performance.

To present a more detailed account on the TCPBP 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, in more sophisticated simulated environments or real networks, and asymmetric routing. Given a flexible SDN environment, for example, TCPBP on top of the default OSPF metric can be used to evaluate TCPBP with minimal intrusion on network traffic, which is possible, for example, with flow discrimination in the flow table rules (e.g., installing TCPBP to provide route metric to specific flows). Other opportunities for research in future works are stated below:

(i) **Route selection**: further research on route selection, stability and hysteresis is needed to avoid possible undesired re-convergences resulting from the adoption of TCPBP. E.g., 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 (i.e., 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. In this case, both routes with a close value of $d\sqrt{p}$ could be used to balance traffic in the network, or a threshold value to classify the routes may be desirable;

(ii) **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 $d\sqrt{p}$;

(*iii*) **Link 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. 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.

References

[Ayar et al. 2019] Ayar, T., Altilar, D. T., Budzisz, Ł., and Rathke, B. (2019). Emulation and Performance Evaluation of a Transparent Reordering Robust TCP Proxy. In *2019 IEEE 8th International Conference on Cloud Networking (CloudNet)*, pages 1–3. IEEE.

[Baccelli and McDonald 2005] Baccelli, F. and McDonald, D. R. (2005). A square root formula for the rate of non-persistent TCP flows. In *Next Generation Internet Networks, 2005*, pages 247–254. IEEE.

[Badis and Agha 2003] Badis, H. and Agha, K. A. (2003). A distributed algorithm for multiple-metric link state QoS routing problem. In *Mobile And Wireless Communications Networks: (With CD-ROM)*, pages 141–144. World Scientific.

[Basso et al. 2012] Basso, S., Meo, M., Servetti, A., and De Martin, J. C. (2012). Estimating packet loss rate in the access through application-level measurements. In *Proceedings of the 2012 ACM SIGCOMM workshop on Measurements up the stack*, pages 7–12.

[Cohen 2020] Cohen, D. (2020). The challenge of modern wireless backbone networks. Available at: https://www.ceragon.com/blog/challenge-of-modern-wireless-backbone-networks. Accessed in: 02-23-2021. Technical report.

[Costa et al. 2000] Costa, L. H. M., Fdida, S., and Duarte, O. C. M. (2000). Distance-vector QoS-based Routing with Three Metrics. In *International Conference on Research in Networking*, pages 847–858. Springer.

[Dong et al. 2019] Dong, E., Xu, M., Fu, X., and Cao, Y. (2019). A Loss Aware MPTCP Scheduler for Highly Lossy Networks. *Computer Networks*, 157:146–158.

[Ginsberg et al. 2019] Ginsberg, L., Previdi, S., Wu, Q., Tantsura, J., and Filsfils, C. (2019). BGP-Link State (BGP-LS) Advertisement of IGP Traffic Engineering Performance Metric Extensions. Technical report.

[Jain et al. 1984] Jain, R. K., Chiu, D.-M. W., Hawe, W. R., et al. (1984). A quantitative measure of fairness and discrimination. *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*.

[Kurose and Ross 2017] Kurose, J. and Ross, K. (2017). *Computer Networks: A Top Down Approach Featuring The Internet*. Pearson Addison Wesley.

[Li et al. 2021] Li, J., Li, D., Wu, W., Ramakrishnan, K., Geng, J., Wang, F., and Zheng, K. (2021). Sphinx: A transport protocol for high-speed and lossy mobile networks. *Computer Networks*, page 108193.

[Lukaseder et al. 2016] Lukaseder, T., Bradatsch, L., Erb, B., Van Der Heijden, R. W., and Kargl, F. (2016). A comparison of TCP congestion control algorithms in 10G networks. In *2016 IEEE 41st Conference on Local Computer Networks (LCN)*, pages 706–714. IEEE.

[Mathis et al. 1997] Mathis, M., Semke, J., Mahdavi, J., and Ott, T. (1997). The macroscopic behavior of the TCP congestion avoidance algorithm. *ACM SIGCOMM Computer Communication Review*, 27(3):67–82.

[Padhye et al. 1998] Padhye, J., Firoiu, V., Towsley, D., and Kurose, J. (1998). Modeling TCP throughput: A simple model and its empirical validation. In *Proceedings of the ACM SIGCOMM'98 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 303–314.

[Polese et al. 2019] Polese, M., Chiariotti, F., Bonetto, E., Rigotto, F., Zanella, A., and Zorzi, M. (2019). A survey on recent advances in transport layer protocols. *IEEE Communications Surveys & Tutorials*, 21(4):3584–3608.

[Poojary and Sharma 2016] Poojary, S. and Sharma, V. (2016). Analysis of multiple flows using different high speed TCP protocols on a general network. *Performance Evaluation*, 104:42–62.

[Shi et al. 2009] Shi, K., Shu, Y., Yang, O., and Luo, J. (2009). Receiver Assistant Congestion Control in High Speed and Lossy Networks. In *2009 16th IEEE-NPSS Real Time Conference*, pages 91–95. IEEE.

[Su et al. 2019] Su, B., Jiang, X., Jin, G., and Ma, A. (2019). DVPTCP: A Delay-Driven Virtual Parallel TCP for High-Speed and Lossy Networks. *IEEE Access*, 7:99746–99753.

[Tantisarkhornkhet and Werapun 2016] Tantisarkhornkhet, P. and Werapun, W. (2016). QLB: QoS Routing Algorithm for Software-Defined Networking. In *2016 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, pages 1–6. IEEE.

[Wang and Crowcroft 1996] Wang, Z. and Crowcroft, J. (1996). Quality-of-Service Routing for Supporting Multimedia Applications. *IEEE Journal on selected areas in communications*, 14(7):1228–1234.

[Xu et al. 2004] Xu, L., Harfoush, K., and Rhee, I. (2004). Binary increase congestion control (BIC) for fast long-distance networks. In *IEEE INFOCOM 2004*, volume 4, pages 2514–2524. IEEE.

[Yang and Wang 2008] Yang, Y. and Wang, J. (2008). Design guidelines for routing metrics in multihop wireless networks. In *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*, pages 1615–1623. IEEE.

[Yue et al. 2012] Yue, Z., Zhang, X., Ren, Y., Li, J., and Zhong, Q. (2012). The performance evaluation and comparison of TCP-based high-speed transport protocols. In *2012 IEEE International Conference on Computer Science and Automation Engineering*, pages 509–512. IEEE.

[Zaragoza 2006] Zaragoza, D. (2006). Challenging the square-root law for TCP send-rate. *Electronics Letters*, 42(24):1430–1431.