IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 9 September 2022, accepted 4 November 2022, date of publication 14 November 2022,
date of current version 18 November 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3222062

RESEARCH ARTICLE

Flow Fairness With Core-Stateless Resource Sharing in Arbitrary Topology

GERGŐ GOMBOS[1], DÁVID KIS[1], LILLA TÓTHMÉRÉSZ[2], TAMÁS KIRÁLY[2], SZILVESZTER NÁDAS[3], AND SÁNDOR LAKI[1], (Member, IEEE)

[1]Department of Information Systems, ELTE Eötvös Loránd University, 1117 Budapest, Hungary
[2]Department of Operations Research, ELTE Eötvös Loránd University, 1117 Budapest, Hungary
[3]Ericsson Research, 1117 Budapest, Hungary

Corresponding author: Sándor Laki (lakis@inf.elte.hu)

This work was supported in part by the National Research, Development and Innovation Fund of Hungary, financed under the Thematic Excellence Program (National Challenges Subprogram) Funding Scheme, through the Project Application Domain Specific Highly Reliable Information Technology (IT) Solutions, under Grant TKP2020-NKA-06.

ABSTRACT Resource sharing is of utmost importance in networking environments where substantial overprovisioning is economically infeasible. Different centralized solutions have recently been proposed for Wide Area Networks, Access Aggregation Networks and other closed networking domains, relying on different ideas from exploiting the capabilities of Software Defined Networking for dynamically allocating bandwidth among flows or other traffic aggregates, to moving bottlenecks from the network to a single location (e.g., a gateway node). In contrast to centralized solutions, core-stateless resource sharing proposals have also emerged, solving the resource allocation problem in a distributed way. In this paper, we focus on the network-wide behavior of a recent core-stateless resource sharing proposal called Per Packet Value (PPV). In PPV, each traffic aggregate is represented by a distribution of values carried by the packets. The distribution is used to express the resource sharing policy. We provide a theoretical analysis of PPV and show that it solves the generalized max-min fair allocation problem in arbitrary topology. PPV has provable convergence in case of both scalable and non-scalable congestion control behaviors. To validate the theoretical results under various network conditions, thorough simulations have been carried out in networks with real-world topology. The method assumes congestion controlled sources since non-responsive UDP flows can generate dead packets taking bandwidth away from well-behaving flows. To remedy the problem of dead packets, we propose a lightweight core-stateless policer method that can autonomously rule the resource usage of unfriendly flows, reducing the effect of dead packets in the system.

INDEX TERMS Resource sharing, max-min fairness, utility function, core-stateless forwarding.

I. INTRODUCTION

The bandwidth allocation of flows over the Internet is solved by the congestion control mechanism of TCP. This model assumes equal resource share among flows and only results in fair resource allocation if the same or at least compatible congestion control algorithms are used by the competing flows. In many networking scenarios flows may have different priorities or weights. For example, 1) in an Access Aggregation Network different subscribers may have Internet access with different Service-Level Agreements (SLAs) and their traffic mix may contain various number of flows with different importance and throughput requirements; 2) in a private Wide Area Network (WAN) interconnecting cloud computing sites around the globe the service traffic may also have different weights and priorities. These environments require more sophisticated resource sharing than what is provided by existing congestion controls.

Traditional resource sharing solutions like weighted-fair queueing require per-flow states and many queues to be maintained at each potential bottleneck in the network. Though it guarantees isolation and weighted-fair resource sharing,

The associate editor coordinating the review of this manuscript and approving it for publication was Fung Po Tso.

existing hardware routers/switches have limited traffic management capabilities and thus can only maintain 8-16 queues for each egress port that is far less than the number of active flows. To overcome the complexity of stateful solutions, different core-stateless resource sharing approaches have been proposed in the past decade. These solutions tag packets at the edge, while resource sharing at any node of the network solely relies on these tags carried by packets, enabling flow-unaware operation of the network core. The key benefits of such solutions include that they move the computational complexity to the network edge, where per-flow packet marking can be implemented in a distributed way (e.g., each flow is marked by a separate packet marking instance running at the end-host or edge-router, or in a telco cloud). Accordingly, nodes simply use packet tags to decide how to handle the incoming packet in case of congestion. Compared to traditional approaches, this design enables high-scalability and can ensure accurate resource sharing among a large number of flows with ease. With the advent of programmable data planes [1], their real-world deployment has also become possible in different network domains [2], [3], resulting in more flexible and configurable alternatives to traditional methods.

In this paper, we focus on a recent core-stateless proposal called Per Packet Value (PPV) [4]. Though it has been shown in the past years [3], [4], [5], [6], [7] that the key concepts of PPV can provide a flexible and congestion control-independent solution for resource sharing and active queue management problems, these studies have only focused on the analysis of a single bottleneck scenario. The main aim of this paper is to examine the network-wide behavior of the PPV framework with both theoretical and simulation tools in arbitrary network topology. The key contributions of this paper include:

- We introduce a mathematical model to describe and analyze the throughput allocation of the PPV framework in steady state. We have shown the existence and uniqueness of the equilibrium allocation. As a corollary, applying the same resource sharing policy to all flows the PPV method solves the max-min fair allocation problem.
- We have extended the mathematical model to emulate the behavior of both scalable (Data Center TCP/DCTCP-like) and classic (Additive Increase Multiplicative Decrease/AIMD-like) congestion controls and shown that for scalable flows the throughput allocation converges to the equilibrium, while in case of AIMD-like behavior there is no convergence (because of the throughput oscillations) but guaranteed lower bound for the assigned flow capacities can be proved.
- We also show that non-congestion controlled flows can violate the expected fairness between flows in multi-bottleneck cases. To remedy this issue also known as dead packet problem [8], we introduce a lightweight method called Core-Stateless Policer that can automatically regulate misbehaving, unfriendly flows by applying adaptive source suppression in the edge marker. Similarly to other components of the PPV framework, the proposed policer runs in a flow-unaware way and do not require global coordination.
- The theoretical results hold on an arbitrary topology, but they rely on simplified models. To examine the performance in more realistic environments, we carried out large number of packet-level simulations on both a simple parking lot and real-world topologies from the TopologyZoo data set [9]. We consider both congestion controlled flows with DCTCP and NewReno, and unresponsive UDP traffic. The simulation results are in accordance with the theoretical analysis, showing that PPV method can effectively solve resource sharing between congestion controlled flows and it also ensures that non-responsive UDP flows cannot gain advantages over well-behaving flows. The effect of unresponsive flows can almost fully be eliminated with the proposed core-stateless policer.

The remaining part of the paper is organized as follows: In Sec. II, we summarize the state-of-the-art in resource sharing field. In Sec. III, we present the concept of the PPV resource sharing framework. In Sec. IV, we introduce a mathematical model for analysing of the PPV method in steady state. In Sec. V, we extend the model with dynamic flow behaviour, emulating both scalable and classic congestion control algorithms. Sec. VI investigates the effect of TCP-unfriendly flows, resulting in unnecessary packet drops caused by dead packets of the non-congestion controlled flows. We also propose a lightweight core-stateless policer to eliminate the dead packet problem by regulating misbehaving flows. In Sec. VII, we confirm the mathematical results with simulations, using various settings and real-world topologies. In Sec. VIII we briefly discuss the results, comparing the achievements to the state-of-the-art. Finally, Sec. IX concludes our results.

II. RELATED WORK

The area of sharing network resources have a rich literature with fundamentally different approaches. In this section, we aim to give an overview of the most important related work.

A. STATEFUL RESOURCE SHARING

The majority of existing resource sharing solutions deployed in production networks rely on the common paradigm of weighted queueing. These methods use flow states at the bottleneck nodes for packet scheduling. A well-known realization of weighted fair queueing discipline is the Deficit Round Robin (DRR) scheduling [10]. It maintains a separate queue for each flow (or at least for the active ones). Its buffer management scheme assumes that when the buffer is full one or more packets from the longest queue are dropped. DRR is a sophisticated per-flow queueing algorithm that can achieve high degree of fairness. Fair queueing with DRR can be handled by specific hardware solutions consisting of hardware queues, however it results in scalability issues if the number of flows are large. For example, programmable switches only consist of a limited number of queues per egress port that is not enough for per-flow queueing.

In addition, if we want to use a different resource sharing policy, it is hard to change since it require the reconfiguration of several network devices. One solution for this problem is the OpenQueue [11] that provides a language in which we can easily add or change policies and manage the buffers in run-time at a high abstraction level.

Some recent proposals aim to handle these issues for data center and WAN environments and go further by applying policies described by utility functions to control weighted queues of the bottleneck nodes or by enabling hierarchical resource sharing with flexible policy definition. For example, NUMFabric [12] aims at solving the network utility maximization problem by splitting it between end-hosts and switches, and introducing a weight exchange protocol between the two, promising fast convergence times in data center environments. Though NUMFabric provides very flexible means to describe a rich set of policies, it still applies traditional weighted fair queueing in the switches, and assumes ultra-low delay between hosts and switches.

BwE [13] proposed by Google uses rate limiting instead of weighted queues to manage the resource sharing for a globally-deployed private WAN. It introduces a bandwidth function to define flexible policies. BwE also shares the idea of the PPV framework [4] that routers cannot support the scale and complexity of enforcing per-flow policies. The difference between the two solutions is BwE uses a centralized control while PPV solves the throughput allocation in a fully distributed way.

B. STATELESS RESOURCE SHARING

Several resource sharing architectures have been proposed in the past decade that work without any per-flow states inside the network. Such architectures typically consists of two kind of nodes: 1) stateful Edge Nodes that are located close to the subscribers and responsible for packet marking; 2) stateless Core (or Resource) Nodes deployed in the core of the network that perform simple packet processing solely based on the markings of incoming packets. According to this terminology, all the packet forwarding elements that could potentially be a bottleneck in the network are Core Nodes, including Edge Nodes as well.

The most widely known of such architectures is DiffServ [14] where markings identify a set of pre-defined policies called Per-Hop Behaviors (PHBs) to be applied by the routers. One of the key disadvantages of this approach is that Core Nodes have to be re-programmed whenever a new PHB (a new policy) is introduced, since Edge Nodes only assign packets to traffic classes and mark them accordingly, but the PHB logic is implemented by the Core Nodes.

Another such proposal is Core Stateless Fair Queuing [15] that implements a single policy: proportional fair bandwidth sharing and marks packets of each flow with its estimated rate at the edge. In contrast to DiffServ, the logic of the policy to be applied is implemented by Edge Nodes while Core Nodes solely use the packet markings during dropping and scheduling, resulting in a highly scalable bandwidth allocation approach.

Rainbow Fair Queueing (RFQ) [16] assigns a few drop precedence levels called colors to the packets while Core Nodes drop packets according to the assigned precedence level.

This concept is extended by the PPV method [4] that uses scalar values as packet markings instead of a few colors. PPV method uses a Throughput-Value Function to express operator policies and solves the resource sharing problem by maximizing the total transmitted packet value. It aims at providing a practical, distributed approximate solution for the network utility maximization problem [17]. Different AQM algorithms [3], [5], [6], [7], [18] have also been proposed as extensions for the PPV concept.

A recent core-stateless resource sharing proposal is ABC [19] that measures the activity level of flows and encodes activity information into packets that is solely used at forwarding nodes to enforce fair-bandwidth sharing among users by dropping packets with high activity values more likely. This solution is similar to the PPV method, but is less flexible in defining operator policies.

Core-stateless networking solutions used to be a widely examined research area in the early 2000s, however only a very few proposals [4], [5], [6], [18], [19] have been published in the past few years. With the emerging trend of softwarization in computer networks and with the advent of programmable data planes, their applicability has become possible even in production environments.

III. PER PACKET VALUE

In this section, we briefly overview our core-stateless resource sharing framework called PPV and introduce the key definitions needed for the network-wide performance analysis. Similarly to other core-stateless proposals [2], [14], [15], [16], [19], the PPV framework consists of two key elements: 1) a packet marker which assigns values to each packet of a flow (or in more general a traffic aggregate) according to a predefined resource sharing policy; and 2) a scheduler that solely uses the values carried by packets to make a decision on which packet to drop or mark with ECN Congestion Encountered (CE) flag if the buffer is full or its size exceeds a predefined threshold. Accordingly, each flow (or traffic aggregate) has its own marker instance that labels each packet entering the PPV networking domain with a drop precedence called Packet Value (PV). Though policy-based marking is aware of flow states, each flow has its own marker operating independently, and thus can be implemented and deployed in a distributed way.

The scheduler is implemented by all the nodes (e.g., routers, end-hosts) in the PPV domain as each of them could be a potential bottleneck. When the buffer is congested, these nodes always drop (or mark with ECN CE) one or more packets with the smallest PV, instead of dropping the last packet or uniformly at random. Several works, such as [5], [6], [7], and [3], have shown that this dropping strategy

can efficiently be implemented using simple FIFO queues (see Sec. II).

Note that in the remaining part of the paper, we use the term *flow* to express the traffic whose packets belong together. Accordingly, a flow can represent the packets of a TCP connection, the traffic of an application, the total traffic of a given user, or any combinations of them.

A. POLICY ENCODING

As mentioned, the core essence of PPV is how packets of a flow are tagged with PVs. To this end, we use a Throughput-Value Function (TVF) that is defined as the derivative of a utility function: $V_a(x) = U'_a(x)$. Note that for each flow a the utility function $U_a(x)$ expresses the application gain (or the value realized by the network operator) if throughput x is assigned to the flow from the shared network-resources. The derivative of the utility function represents the extra value (e.g., the increase in profit for the operator) that can be generated if extra throughput is given to flow a.

In the PPV method, TVFs are used to label packets where the packet value expresses the gain that is only realized if the packet is delivered (marginal utility in other words). The TVF $V(\cdot)$ defines the PV distribution of a flow for any sending rate R. Specifically, the throughput contribution of packets with PV at least $V(x)$ in the flow should be x (for any $x : 0 \leq x \leq R$). A practical packet marker [7] of flow a continuously measures the flow's sending rate R, chooses a rate x from range $[0, R]$ uniformly at random at packet arrival and assigns PV $p = V_a(x)$ to the given packet.

Fig. 1 illustrates how the TVFs and PVs can be used to share the bottleneck capacity between various flows. In the first case, the bottleneck capacity is 10 Mbps shared between three flows. The red, blue and green curves on the right side represent the TVFs of Flow1, Flow2 and Flow3, resp. The gray dotted line illustrates the cutoff value that results in a resource allocation 0, 6.25 and 3.75Mbps for Flow1, Flow2 and Flow3, resp. This allocation is ensured by only transmitting packets with PV above the cutoff level. One can observe that Flow1 has no packet with PV above this threshold and thus it cannot even transmit a single packet. In the case of a 45Mbps bottleneck, the cutoff value is much smaller and thus all three flows have non-zero assigned throughput. The purple dotted line represents the cutoff value, leading to 10, 17.5 and 17.5Mbps throughput allocation for Flow1, Flow2 and Flow3, resp. In this case, packets below the threshold marked by the purple line are only dropped (or marked with ECN CE). One can observe that the inverse function of a TVF is basically a bandwidth function introduced in [13] to express resource sharing policies, and thus bandwidth function policies can easily be mapped to TVFs in the PPV system, and vice versa.

In general, at high congestion only packets with high PVs are transmitted, more precisely packets with PV above a certain PV threshold that we call Congestion Threshold Value (CTV). The CTV at a bottleneck represents the minimal PV that can successfully be transmitted via the congested link.

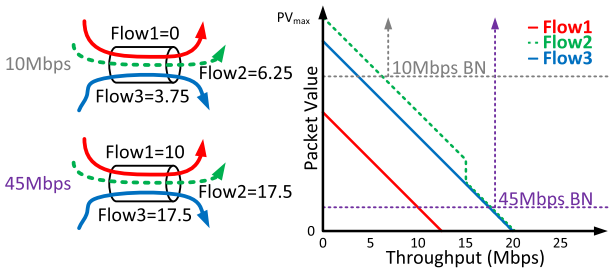


FIGURE 1. Resource sharing with the PPV framework.

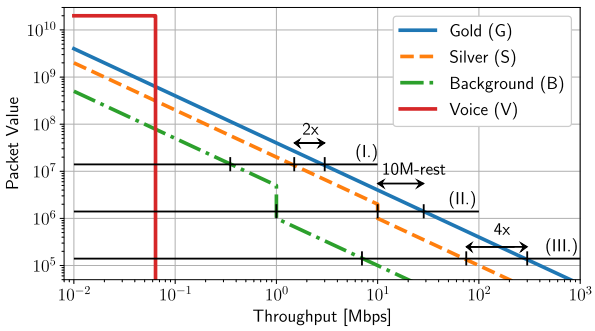


FIGURE 2. Policy examples as throughput-value functions.

The observed CTV reflects the actual congestion level, since at a congested link the total throughput of packets having PV at least the current CTV is exactly the bottleneck capacity. CTV is not a parameter of the proposed system, but an emergent property of the applied drop minimum-PV first AQM strategy. The amount of high and low PV packets determines the resource share between various flows, resulting in that at high congestion, aggregates with larger share of high PV packets get more throughput.

Fig. 2 depicts few example policies expressed as TVFs. Voice traffic is rate limited and has a guaranteed throughput need. It is defined by the red curve, assigning the maximum PV up to the rate limit, and above that the smallest value is assigned to the packets, emulating a rate limiter policy. The blue, orange and green TVFs express weighted fairness between the different flow groups, where the weights rely on the congestion level (CTV). For example, at high congestion (I.) a Gold flow can get twice the throughput of a Silver flow; at medium congestion (II.) Silver flows get 10 Mbps throughput and the rest can be used by Gold flows; at low congestion (III.) 1:4 resource share is defined between Silver and Gold flows. For the evaluations in this paper we configured the Gold and Silver policies to always achieve 2:1 resource sharing.

B. POTENTIAL APPLICATIONS

Good Quality of Service in novel applications such as AR/VR, cloud rendered gaming, HD or holographic video conferencing and remote presence requires high bandwidth, low latency or both. End users connect to the Internet with different subscriptions and access properties. As gigabit-speed access links became widespread, the possibility of temporal

and even permanent overloads in the Access Aggregation Network has increased. These periods can be handled by over-provisioning, but it has a high price: high infrastructure cost and underutilization in most of the time. Weighted Fair Queueing (e.g., DRR scheduling) is a widely adopted solution to ensure complex resource sharing policies in access networks, where resource sharing is controlled among traffic aggregates (TAs), e.g., between virtual operators, network slices, users, or subflows of users. Nowadays, QoS is typically enforced in the border gateway of the access network, since all traffic going towards or coming from the Internet flow through this node. Note that this solution puts high computational load on the gateway node and has further limitations. PPV could serve as a lightweight alternative for such network environments.

Another area where PPV-like resource sharing could have potential benefits is cloud networking where it can ensure complex resource sharing policies between tenants and/or its subflows. The access to the shared resources at tenant level is described in the Service Level Agreement, reflecting the per-tenant payment granularity applied by today's data centers.

Moreover, PPV could also have potential benefits in any other physical network infrastructures (e.g., private WAN, 5G/6G RAN, etc.) that are shared among virtual networks (e.g., slices, virtual operators). It can solve not only the isolation of virtual networks but the differentiation between traffic groups inside virtual slices.

Though PPV requires new packet tagging and scheduling mechanisms that make its deployment difficult, programmable switches could enable its real-world and incremental deployment in the future.

IV. MODEL AND ANALYSIS OF STEADY STATE

In this section, we introduce a mathematical model to analyse the network-wide properties of the PPV method in steady state, and show that it can solve the generalization of max-min fair allocation problem over arbitrary topology.

Let $N = (V, E)$ be a directed graph (the network) with vertex set V, edge set E and link capacities $g \in \mathbb{R}^E_{>0}$. We consider a k-commodity flow problem with fixed paths: P_i is a path from $s_i \in V$ to $t_i \in V$, $d_i \in \mathbb{R}_+$ is the maximum demand of the i-th flow, and $v_i : [0, d_i] \to \mathbb{R}_+$ is a strictly monotone decreasing, continuous throughput-value function (TVF) with $v_i(d_i) = 0$ $(i \in [1, k])$. Note that the strictly monotone decreasing and continuous properties of TVFs (e.g., see Fig. 2) can be ensured by replacing horizontal or vertical line segments with segments having a small or large negative slopes, resp. A *throughput allocation* is a k-tuple $x = (x_1, \dots, x_k)$ where $0 \le x_i \le d_i$ $(i \in [1, k])$. We use the notation

$$x(e) = \sum_{i:e\in P_i} x_i \quad (e \in E).$$

An allocation x is *feasible* if

$$x(e) \le g_e \quad \text{for every } e \in E.$$

For a throughput allocation x and a link $e \in E$, we define the *cutoff value* (CTV as defined in Sec. III) $\alpha_e(x)$ as

$$\alpha_e(x) = \inf\left\{\alpha \ge 0 : \sum_{i:e\in P_i} \min\{x_i, v_i^{-1}(\alpha)\} < g_e\right\},$$

where $v_i^{-1}(\alpha)$ is defined as 0 if $\alpha > v_i(0)$.

For an allocation x, the *truncated allocation* $\bar{x} = (\bar{x}_1, \dots, \bar{x}_k)$ is defined as

$$\bar{x}_i = \min\left\{x_i, \min_{e\in P_i} v_i^{-1}(\alpha_e(x))\right\} \quad (i \in [1, k]).$$

Since TVFs v_i are continuous, an allocation x is feasible if and only if $x = \bar{x}$. A feasible allocation is called an *equilibrium allocation* if

$$x_i = \bar{x}_i = \min_{e\in P_i} v_i^{-1}(\alpha_e(x)) \quad (i \in [1, k]).$$

Note that $v_i^{-1}(\alpha_e(x))$ is between 0 and d_i for any x.

A. EXPLANATION

This model corresponds to a static interpretation of the PPV framework. The value x_i corresponds to the sending rate of flow i, and g_e is the maximum capacity of link e. The TVF v_i is the same as the function V introduced in Sec. III, which is the derivative of the utility function.

Each packet of flow i is marked with a random value $v_i(z)$ where $z \in [0, x_i]$ is chosen uniformly at random. Link e discards every packet with value less than $\alpha_e(x)$, representing that the PPV method drops packets with smallest packet values when the buffer is full. To compute the effective rate of flow i, we should consider the rate of non-discarded packets; this can be computed as

$$\min\left\{x_i, \min_{e\in P_i}\{v_i^{-1}(\alpha_e(x))\}\right\} = \bar{x}_i.$$

Thus, $\bar{x}_i$ is the effective rate of flow i. The throughput allocation x is feasible if no packet is discarded; it is in equilibrium if it is feasible and an increase in the sending rate of any flows results in packet loss.

B. EQUILIBRIUM: EXISTENCE AND UNIQUENESS

We first show that the equilibrium allocation exists, is unique and results in a fair allocation. The mathematical proof of this theorem is described in Appendix A.

Theorem 1: For a throughput allocation x, let $v^{\downarrow}(x)$ denote the vector obtained by ordering $v_1(x_1), \dots, v_k(x_k)$ in decreasing order. There is a unique equilibrium allocation, and it is the unique feasible allocation for which $v^{\downarrow}(x)$ is lexicographically minimal among all feasible throughput allocations.

As a corollary of this theorem, we can also claim the following statements: 1) since $v^{\downarrow}(x)$ (representing the vector of cut-off values) is lexicographically minimal and $v(\cdot)$ is strictly monotone decreasing, the equilibrium allocation is the feasible allocation that maximizes the delivered packet

values. 2) If we assume that all flows apply the same TVF $v(\cdot)$ (same resource sharing policy), the equilibrium allocation $x_1, \ldots, x_k$ is a max-min fair allocation. 3) If each TVF $v_i(\cdot)$ is the derivative of an appropriate utility function $U_i(\cdot)$, the equilibrium allocation is also the feasible throughput allocation that maximizes the minimum of flow utilities, generalizing the definition of max-min fair allocation.

V. CONGESTION CONTROLLED FLOW MODEL

In the previous section, we have investigated the properties of PPV framework in steady state. Flows on the Internet have no prior knowledge about the network state. Some flows use constant sending rates while others are controlled by congestion control mechanisms. In this section, we examine the dynamic behaviour of flows with respect to various update protocols which are simplified models of congestion control algorithms. In this simplified model, flow rates are updated in discrete time steps (phases). Each phase is considered as a steady state where the definitions of the previous section apply, and the sending rate x_i is updated for the next phase based on the difference between x_i and $\bar{x}_i$. Note that $\bar{x}_i$ expresses effective rate of non-discarded packets (i.e., the arrival rate at the destination of the flow). We say that a flow x_i *has loss* in a given phase if $\bar{x}_i < x_i$; otherwise, it is *lossless*. We will show that the following types of update protocols are well-behaved.

A. TIGHT CUT UPDATE PROTOCOLS

Definition 2 (Tight Cut Update): If $x_i > \bar{x}_i$, then the sending rate is $\bar{x}_i$ in the next phase. Otherwise the sending rate is at least $\min\{d_i, x_i + \Delta\}$ for some fixed Δ.

One specific instance of the tight cut update strategy is what we call the slowly growing protocol. It emulates the behavior of scalable congestion controls. Though scalable congestion control like DCTCP [20] was originally introduced for data center networks to remedy the performance issues of traditional TCP, recent works propose the application of similar solutions (e.g., TCP Prague [21], Google's BBRv2 [22]) over the Internet or at least in closed networking domains (e.g., access aggregation networks, private WANs).

Definition 3 (Slowly Growing): For the next phase, the sending rate of terminal i is $\bar{x}_i$ if x_i had loss in the last phase, otherwise the sending rate is $\min\{d_i, x_i + \Delta\}$, where Δ is fixed in advance.

One can observe that tight cut strategies always probe the throughput carefully and reduces the rate proportionally to the congestion level in case of packet loss. It is not surprising that this strategy shows convergence to equilibrium allocation:

Theorem 4: If each flow follows some tight cut update protocol, then the throughput allocation converges to the equilibrium of the network.

Proof of the theorem and additional technical lemmas are detailed in Appendix B.

B. AIMD, A NON TIGHT CUT UPDATE PROTOCOL

We also examine a (non tight cut) update protocol that mimics the classic AIMD-based congestion control (e.g., TCP Reno, NewReno).

Definition 5 (AIMD): For the next phase, the sending rate of terminal i is $\frac{x_i}{2}$ if $x_i > \bar{x}_i$, otherwise it increases to $\min\{d_i, x_i + \Delta\}$, where Δ is fixed in advance.

AIMD halves the sending rate in case of congestion, otherwise it increases the rate linearly by adding a constant factor, resulting in the well-known sawtooth-like oscillations.

It does not have such nice behavior as tight cut protocols, but the use of the PPV framework can guarantee a lower bound for the assigned capacity even for this case. Our first result recovers the empirically known fact that if the AIMD protocol is used, sending rates might fluctuate.

Claim 6: If some flows use the AIMD update protocol, then the flow throughput values might not converge to the equilibrium.

Proof: We show an example where the network is a single edge with two flows. Let $d_1, d_2 = 2$, $v_1(x) = v_2(x) = 2 - x$. Let $g_e = 3$. In this case the equilibrium is $(x_1, x_2) = (\frac{3}{2}, \frac{3}{2})$. This is clearly a feasible allocation, and for $\alpha_e = \frac{1}{2}$, $x_1 = \bar{x}_1 = x_2 = \bar{x}_2 = \frac{3}{2}$.

To show a process that does not converge to this equilibrium, let us start with flow throughput values $(x_1, x_2) = (\frac{11}{8}, \frac{7}{8})$. It is easy to check that in this case the consecutive throughput values will be $(\frac{12}{8}, \frac{8}{8})$, $(\frac{13}{8}, \frac{9}{8})$, $(\frac{14}{8}, \frac{10}{8})$, $(\frac{14}{8}, \frac{11}{8})$, $(\frac{7}{8}, \frac{11}{8})$, $(\frac{8}{8}, \frac{12}{8})$, $(\frac{9}{8}, \frac{13}{8})$, $(\frac{10}{8}, \frac{14}{8})$, $(\frac{11}{8}, \frac{14}{8})$, $(\frac{11}{8}, \frac{7}{8})$ and from this point, the protocol will cycle. □

As another negative result, we show the following:

Claim 7: A flow using AIMD-like update protocol might receive smaller channel capacity in average than a flow using a tight cut protocol, even if their TVFs agree.

Proof: We give an example with two AIMD and a slowly growing protocol, where the AIMD protocols cycle, and on average they get smaller capacity than the slowly growing protocol.

Let our network consist of an edge. We will have three flows x_1, x_2 and x_3. x_1 and x_2 will follow the AIMD protocol, while x_3 the slowly growing protocol. Let $d_1, d_2, d_3 = 2$, $v_1(x) = v_2(x) = v_3(x) = 2 - x$ and $g_e = \frac{9}{2}$.

Let the protocol start with $(x_1, x_2, x_3) = (\frac{11}{8}, \frac{7}{8}, \frac{13}{8})$. Then the throughput values of the following phases will be: $(\frac{12}{8}, \frac{8}{8}, \frac{14}{8})$, $(\frac{13}{8}, \frac{9}{8}, \frac{15}{8})$, $(\frac{14}{8}, \frac{10}{8}, \frac{14}{8})$, $(\frac{7}{8}, \frac{11}{8}, \frac{13}{8})$, $(\frac{8}{8}, \frac{12}{8}, \frac{14}{8})$, $(\frac{9}{8}, \frac{13}{8}, \frac{15}{8})$, $(\frac{10}{8}, \frac{14}{8}, \frac{14}{8})$, $(\frac{11}{8}, \frac{7}{8}, \frac{13}{8})$, from where the protocol will cycle.

For x_1 and x_2, the average capacity is $\frac{21}{16}$. While for x_3, the average capacity is $\frac{28}{16}$. As the d_i and the v_i are symmetric for the three flows, the equilibrium is where each flow gets one third of the capacity, that is, $\frac{3}{2} = \frac{24}{16}$. □

However, we can show that using the PPV method for dropping packets, there is a guaranteed lower bound for the capacity assigned to flows using the AIMD protocol.

Theorem 8: If a flow x_i follows the AIMD protocol, then after some time,

$$x_i \geq \frac{1}{2}\min\{d_i, \min_{e\in P_i} v_i^{-1}(\beta_e)\}$$

always holds.

Accordingly, each flow gets at least half of their capacity in the equilibrium allocation. The proof of the theorem can be found in Appendix C. Note that this lower bound cannot hold if packets are dropped randomly, instead of using the PPV method.

VI. POLICING UNFRIENDLY FLOWS

In general network typologies, per link control of resource sharing cannot solely ensure that the ideal resource allocation is reached. The current Internet relies on end-to-end congestion control that reduces the sending rates of flows when they detect network congestion.

It has been shown that the efficiency of this approach depends on two assumptions: 1) all flows are cooperative and 2) they are homogeneous or at least react to congestion similarly. However, these assumptions can easily be violated by unfriendly flows that do not implement congestion control at all, are drop-tolerant or increase their transmission rates too aggressively and thus they are not TCP-friendly. In the presence of unfriendly flows, links could be busy transmitting packets that will only be dropped later along the network path, occupying scarce bandwidth from other well-behaving flows on links before the congested link. It also means that some packets may be dropped because of other packets called dead packets [8] that end up being dropped on a later link, causing unnecessary packet drops. This problem is mitigated by end-to-end congestion control as that keeps packet loss small. Flow policers (e.g., [23]) can similarly mitigate this problem by enforcing good flow behavior even for unresponsive flows, but they require additional per-flow states in network nodes. It would be desirable to apply a lightweight flow policer which is easy to deploy and implement without increasing the complexity of network nodes.

In this section, we introduce the concept of Core-Stateless Policing (CSP), which extends our PPV framework with flow policing, enabling to handle the problem of dead packets efficiently. It only keeps per-flow states in the flow's packet marker instance thus does not require any new flow states in the core of the network. It requires an extra bit called notification flag in packet headers and infrequent signaling messages from the bottleneck scheduler to the packet marker. Typically no (or very rare) signaling is needed for congestion controlled flows. However, CSP regulates unresponsive flows within the core-stateless network domain. Its control loop is clocked by the round trip delay between the bottleneck nodes and the given packet marker instance instead of the end-to-end RTT of the flow.

To control unresponsive flows, CSP requires the addition of four basic mechanisms to the packet marker component and the scheduler: In addition to labeling packets with PVs as previously, a packet marker instance also maintains a cut-off value called marker CTV (CTV_m) and a PV range defined by PV_{min} and PV_{max}, expressing packet values If a packet arrives with a PV less than CTV_m, the marker will drop it as the policing action. However, if the PV is in range $[PV_{min}, PV_{max}]$ and a packet has not been tagged with the notification flag for $t_{tagging} = 5$ ms it sets the notification flag in the outgoing packet. Whenever a bottleneck scheduler drops a packet, if its notification flag is set, it also sends a control message to the packet marker containing the PV of the dropped packet. When a marker receives a control message, where the carried value PV is greater than CTV_m, it sets $CTV_m = PV$.

In addition to setting CTV_m, the marker decreases CTV_m periodically: if no control message was received with $PV > CTV_m$ within an update period $t_{update} = 30$ ms, the coded marker CTV is decreased with a constant $CTV_{step} = 64$ (it stands for $\sim$ 3% increase in rate). CTV is not decreased below 0.

Whenever the marker's rate measurement (R) or CTV (CTV_m) is updated the PV range is updated as follows: $r_{max} = min((1-f_1)\cdot R, TVF^{-1}(CTV_m))$, where $TVF^{-1}(CTV_m)$ is the policing rate determined by CTV_m. $PV_{min} = TVF(r_{max})$ and $PV_{max} = TVF(f_2 \cdot r_{max})$. f_1 (0.1 by default) represents the allowed loss rate of the flow within the domain, before policing is activated. Packets representing the rate range $[0.9{\cdot}R,\ R]$ are not generating control messages, which means that until 10% (f_1) of the flow's packets are lost, the policing is not activated. It was designed so that congestion controlled flows will not activate policing in normal cases. Furthermore, policing itself limits the loss rate thus the control itself limits the necessary signaling. f_2 (0.8 by default) is introduced in order to have more meaningful tagged packets. It limits how high the PV of a packet tagged with the notification flag could be, and dropping a packet with smaller PV has higher probability.

The signaling message from the scheduler to the packet marker can be constructed using only packet header information and scheduler state, no per-flow state or policy knowledge is required in the core-stateless scheduler. The tagged packets may include the address of the marker instance in an extension header. Alternatively, a scheduler may send the signaling directly to the source address of the packet, when it is ensured in the domain that these packets can be terminated in the relevant packet marker. This design is lightweight enough for implementing the signaling in P4 programmable switches [1].

One can observe that CSP artificially extend non-responsive flows with congestion control following the previously introduced tight-cut update protocol, and thus Theorem 4 also holds:

Claim 9: If each flow is controlled by CSP (or follows another tight cut update protocol), then the throughput allocation converges to the equilibrium of the network.

Note that the algorithm parameters could be tuned based on the allowed loss rate of the flow within the domain, the signaling load, and how conservative the policing shall be.

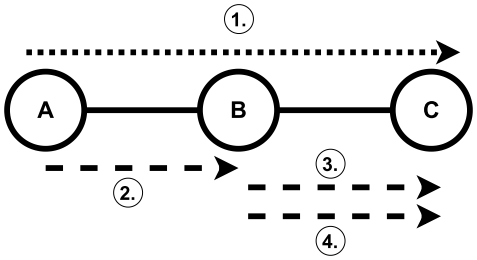


FIGURE 3. Simple scenario to demonstrate the effect of Core-Stateless policing.

VII. SIMULATION-BASED EVALUATION

Though the theoretical results described in the previous section hold on an arbitrary topology, they rely on simplified models. To examine the performance of the PPV framework under more realistic conditions, we have carried out a large number of packet-level simulations with NS-3.31 network simulator. Both packet marker and scheduler components as described in Sec. III have been implemented in the simulator, including the Core-Stateless Policing (CSP) implementation. CSP was disabled by default, when enabled it is explicitly stated. For packet marking, we use two resource sharing policies: Gold and Silver. The corresponding TVFs are depicted in Fig. 2. The packet marker encodes the PV into a 16 bit header field (the identification field of IPv4 header). The scheduler component can work in both drop and ECN mode. In drop mode, it simply implements the drop minimum PV first AQM strategy when the buffer is full. Note that it can drop either the newly arrived packet or one or more packets from the middle of the buffer. In ECN mode, the scheduler buffer is infinite, the unmarked packets having the smallest PVs are marked with ECN CE until the amount of unmarked packets is below a predefined marking threshold. We used NewReno or DCTCP for TCP congestion control. A single flow consisted of 5 TCP connections.

A. CORE-STATELESS POLICING

To show the benefits of Core-Stateless Policing we created a simple parking lot scenario with three nodes, A, B and C, as depicted in Fig. 3. We load this topology with 4 flows. Flow 1 transmits between nodes A and C and thus loads both the AB and BC bottlenecks, it has an end-to-end RTT of 14 ms. Flow 2 is between AB, and Flow 3 and Flow 4 are between BC, each has an end-to-end RTT of 9 ms. The capacity of both links is 100 Mbps and the buffer is 125 KB long.

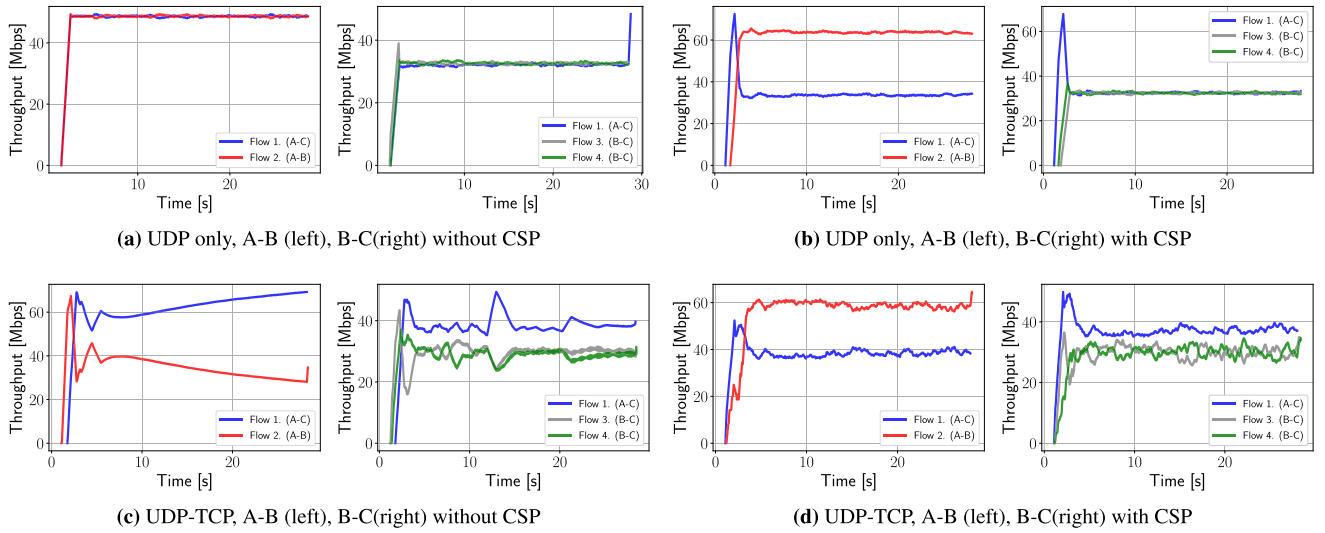
Fig. 4a shows the resulting baseline throughput on the two links when all flows are using unresponsive UDP transmitting at 100 Mbps. Though Flow 1 only gets 33.3 Mbps on link BC, it still takes its share of 50 Mbps on link AB, resulting in 16.6 Mbps worth of dead packets. Fig. 4b shows the same scenario with CSP switched on. While the throughput on link BC do not change, as a consequence of the core-stateless policing, Flow 1 gets much smaller throughput on AB and consequently Flow 2 can use the remaining capacity of the AB link getting close to the ideal 66.6 Mbps. To demonstrate

Algorithm 1 CTV-Search Algorithm

Throughput allocation, $x = \{x_1, \ldots, x_k\}$
Initial allocation, $\forall j \in [1, k] : x_j \leftarrow d_j$
Initial cutoff values, $\forall e \in E : \alpha_e \leftarrow 0$
Set of ready edges, $\hat{E} \leftarrow \emptyset$
for $it = 1$ to $|E|$ **do**
    Max. cutoff value, $\alpha_{max} \leftarrow -\infty$
    Bottleneck link w. max. cutoff, $e_{max} \leftarrow$ None
    for each $e \in E \backslash \hat{E}$ **do**
        $\alpha_e \leftarrow \text{findCTV}(e, x)$
        if $\alpha_e > \alpha_{max}$ **then**
            $\alpha_{max} \leftarrow \alpha_e$
            $e_{max} \leftarrow e$
        end if
    end for
    for each $x_j \in x \mid e_{max} \in P_j$ **do**
        $x_j \leftarrow \min(x_j, v_j^{-1}(\alpha_{max}))$
    end for
    $\hat{E} \leftarrow \hat{E} \cup \{e_{max}\}$
end for

the operation of our CSP algorithm in this case, we depict the scheduler CTVs of the two links and the marker CTV used in Flow 1 and Flow 2 in Fig. 5. The link CTVs are only depicted when a packet is lost on a given link in the last 100 ms. It is visible how the marker CTV used by the policer for both flows are oscillating around the CTVs of respective bottleneck links. Note that link AB is the bottleneck for Flow 2, while link BC is the bottleneck for Flow 1. The marker CTV is always somewhat smaller than the link CTV (as intended, to allow some packet loss for the flows). It is regularly set to a higher value closer to the link CTV when the algorithm decreased the marker CTV too much.

To evaluate the effect of CSP on TCP traffic we investigated mixed scenarios where Flow 1 is still an unresponsive UDP, but the rest of the flows (2, 3, 4) are using NewReno TCP congestion control. Fig. 4c is the baseline without CSP. In addition to the degradation seen in the UDP only case (Fig. 4a), the Reno flows cannot even completely utilize their fair shares on any of the links due to the aggressiveness of the UDP flow. Fig. 4d shows that when CSP is applied, the traffic of the UDP flow (Flow 1) will be less overwhelming for TCP flows: Flow 3 and Flow 4. Also, the UDP flow is limited on link AB close to its fair share on the other link BC, resulting in higher TCP throughput for Flow 2. The resulting sharing is less perfect than in the previous UDP-only case, but still there is significant improvement not only by solving the dead packet problem, but also by controlling the aggressiveness of UDP traffic. Fig. 6 depicts the link and marker CTVs for this case. It is visible that for Flow 2 (TCP) the marker CTV is only rarely set, and it is restored fast by the algorithm, while for Flow 1 (UDP) it is continuously kept close to the CTV of link BC. The CTV of link BC is somewhat less stable than in the UDP-only case, due to the presence of the adaptive TCP traffic of Flow 3 and Flow 4.



(a) UDP only, A-B (left), B-C(right) without CSP

(b) UDP only, A-B (left), B-C(right) with CSP

(c) UDP-TCP, A-B (left), B-C(right) without CSP

(d) UDP-TCP, A-B (left), B-C(right) with CSP

FIGURE 4. The effect of core-stateless policing on unresponsive flows.

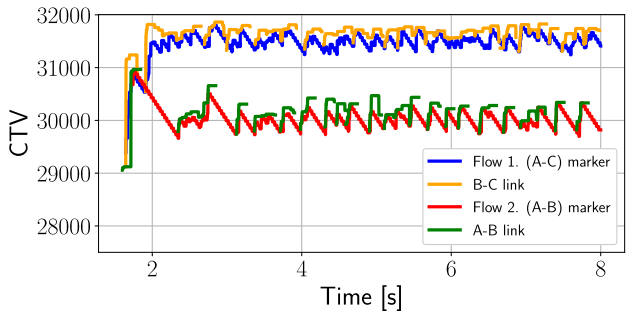


FIGURE 5. UDP-only scenario, marker (CSP) and link CTVs.

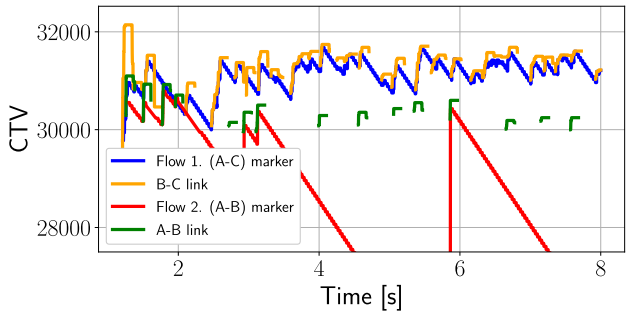


FIGURE 6. Mixed scenario with one UDP and three TCP flows, marker (CSP) and link CTVs.

B. IDEAL THROUGHPUT ALLOCATION

To calculate the ideal (equilibrium) throughput allocation introduced in the previous sections, we designed an algorithm called CTV-Search (Alg. 1). CTV-Search was inspired by the well-known Waterfilling algorithm [24] used to calculate the max-min fair allocation.

The algorithm calculates the cutoff values (CTVs) applied by each edge in the network and the equilibrium allocation according to the definition of Sec. IV. In each iteration (line 5) the algorithm finds the largest cutoff value among the non-ready edges and updates the flow throughput values x_j accordingly. In line 9, findCTV(e, x) finds the cutoff value α_e so that $\sum_{x_j \in x | e \in P_j} \min(x_j, v_j^{-1}(\alpha_e)) = g_e$. This method can be approximated by applying a binary search in the packet value range until $|\sum_{x_j \in x | e \in P_j} \min(x_j, v_j^{-1}(\alpha_e)) - g_e| < \epsilon$ does not hold. One can observe that the loop in line 8 iterates over all the non-ready edges and finds the possible maximal cutoff value among the non-ready edges. The edge where this cutoff value is applied is marked as ready. Note that the largest cutoff value leads to the most significant throughput limitation as the TVFs are strictly monotone decreasing. One can observe that the algorithm determines the cutoff values in decreasing order.

In the following sections, we use this algorithm to calculate the ideal throughput allocation of flows to which we compare the empirical throughput values of packet-level simulations.

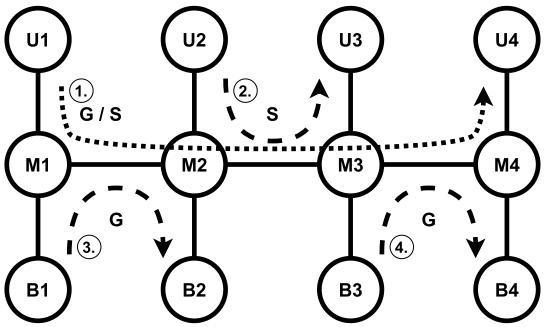


FIGURE 7. The parking lot topology with various traffic patterns. Silver and Gold flows are marked with S and G, resp.

C. STATIC FLOWS IN A PARKING LOT TOPOLOGY

In this scenario, we use a simple parking lot topology depicted in Fig. 7 with different traffic intensities. The traffic is generated along four different routes in the topology: (1) A designated flow between source U1 and sink U4, using either Gold

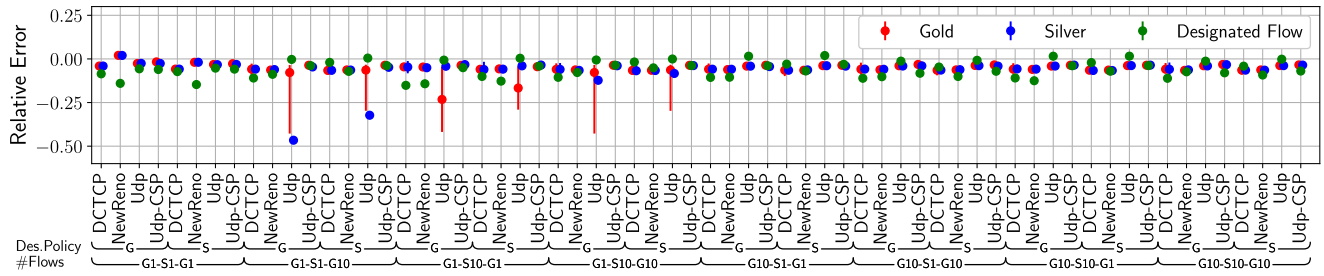


FIGURE 8. Simulation results on the parking lot topology with different traffic intensities and congestion controls. The green dot represents a designated flow (either Gold or Silver policy).

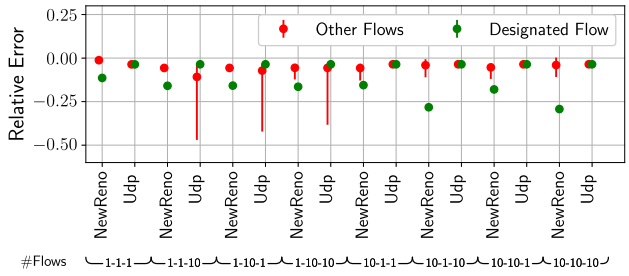


FIGURE 9. Simulation results of the DRR method on the parking lot topology with different traffic intensities and congestion controls. The green dot represents a designated flow.

or Silver policy (see Fig. 2); (2) flows with Silver policy use the route between source U2 and sink U3; (3) We generate traffic with Gold policy from B1 to B2 and (4) from B3 to B4. We have a single designated flow, while the number of flows along other routes is varied (1 or 10 flows). Each link in the topology has 100 Mbps capacity and 2.5 ms link delay. The flows are TCP connections with either DCTCP (scalable) or NewReno (AIMD-like) congestion control, or non-responsive UDP traffic without or with Core-Stateless Policing (CSP). The buffer size used in the scheduler is 190 KB for NewReno and 50KB for UDP experiments. In the DCTCP scenario, the scheduler operates in ECN mode and the ECN marking threshold is set to 190 KB. The TCP sources are greedy with infinite data, while the UDP sources have a constant 100 Mbps sending rate. Fig. 8 depicts the simulation results for various settings and traffic intensities. The y-axis represents the relative error of the observed throughput values for Gold and Silver flows and the single designated flow. The relative error of a flow i is defined as $(x_i - \bar{x}_i)/\bar{x}_i$, where x_i is the observed average throughput of flow i, while $\bar{x}_i$ is the ideal throughput according to the equilibrium allocation. The x-axis represents the various scenarios with flow types (DCTCP, NewReno, UDP, UDP-CSP), the policy of the designated flow (G as Gold, S as Silver) and the number of flows with the applied policies along routes (3), (2) and (4). For example, G1-S1-G10 represents 1 Gold flow along (3), 1 Silver flow along (2) and 10 Gold flows along (4).

As it can be observed the UDP scenarios deviate the most from the ideal share. These flows do not respond to congestion signals (packet drops) which is assumed by the definition of equilibrium allocation. Note that a UDP source in our simulation sends unresponsively at a constant rate, thus unnecessary drops due to dead packets can happen. For example, in the G1-S1-G10 case, the Gold designated UDP flow shares the link M1-M2 with another Gold UDP flow, leading to 50 Mbps-50 Mbps. On the second link (M2-M3) the Silver flow gets also 50 Mbps and the designated flow does not experience any drops. Finally, the last link (M3-M4) is shared between the designated flow and 10 other Gold UDP flows, limiting the designated flow to 9.1 Mbps (same as the other 10 Gold flows in (4)). In the equilibrium allocation the Silver flow gets $100 - 9.1 = 90.9$Mbps that cannot be guaranteed in presence of unresponsive flows. It is visible that enabling CSP solves this issue, and all allocations are close to ideal.

The second highest deviation is for DCTCP flows, though that is in the acceptable range. The deviation is even smaller for Reno flows, while for UDP-CSP it is consistently close to the ideal.

As a comparison, we also executed the same scenarios with DRR scheduling. Fig. 9 depicts simulation results for TCP NewReno and UDP flows. In DRR, the flows are equally weighted, expecting fairness between them. One can observe that the results are very similar to what we get with the PPV method. In NewReno cases, the single designated flow get slightly lower throughput than the equal share that is caused by its conservative congestion control. In case of UDP flows, the designated flow causes dead packets in several scenarios (e.g., 1-1-10, 1-10-1, 1-10-10), leading to significant deviations from the fair share. This is similar to the UDP cases without CSP in Fig. 8.

D. DYNAMIC TRAFFIC BEHAVIOR

In the second scenario, we also use the same parking lot topology with the same link settings (100 Mbps link capacity and 2.5 ms link delay). The traffic is varied over time: 1) the single designated Gold flow is present from the beginning of the simulation to its end; 2) 5 Silver flows from U2 to U3 starts at 10 s and ends at 60 s; 3) 5 Gold flows enters the system along route B1 to B2 at 20 s and they are closed at 40 s; 4) 5 Gold flows from B3 to B4 are present between 30 and 50 s.

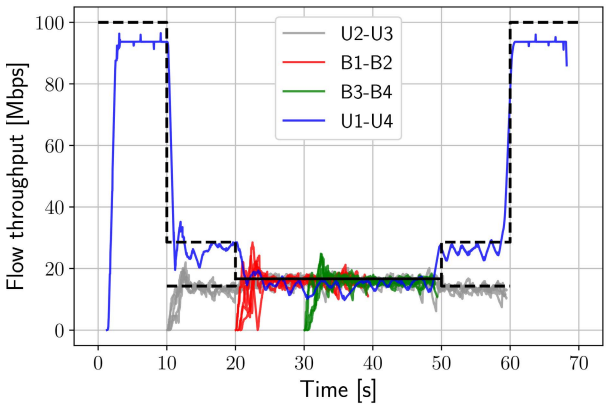


FIGURE 10. The traffic intensities along the different routes are varied. The flows use TCP NewReno congestion control.

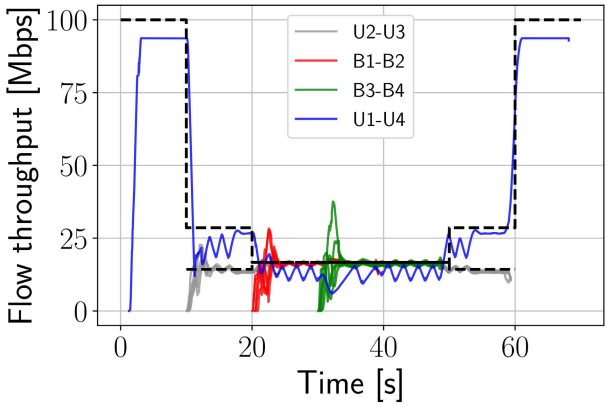
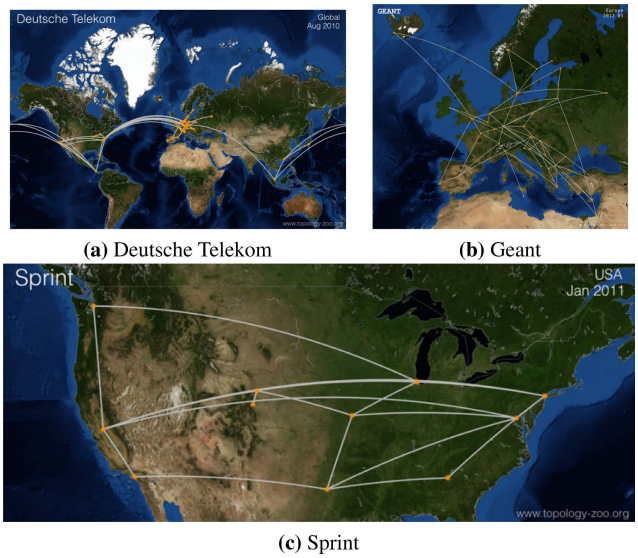


FIGURE 11. The traffic intensities along the different routes are varied. The flows use TCP DCTCP congestion control.



(a) Deutsche Telekom (b) Geant

(c) Sprint

FIGURE 12. We selected three topologies from the database of TopologyZoo [9].

Fig. 10 depicts the throughput of TCP flows with NewReno congestion control. The flow throughput values along the different routes are marked with solid lines while black dashed curves denote the ideal throughput levels. The simulation splits into 7 regions: 1) (0-10 s) the designated Gold flow solely uses the network, resulting in slightly less throughput than 100 Mbps; 2) (10-20 s) the designated flow shares the link M2-M3 with 5 Silver flows. According to the applied TVFs, the desired resource share between Gold and Silver flows is 2:1 that is also reflected by the observed throughput values (both Silver and Gold). 3) (20-30 s) In addition to the Silver flows, the designated flow also shares the link M1-M2 with 5 Gold flows. One can observe that all flows get the same throughput. This is caused by the M1-M2 link where the designated flow is limited to $100/6 = 16.7$ Mbps, and thus Silver flows on link M2-M3 immediately occupy the unused capacity. 4) (30-40 s) Another 5 Gold flows enter the system, crossing link M3-M4. One can observe that this extra traffic does not affect the throughput of other flows. 5) (40-50 s) The 5 Gold flows crossing M1-M2 link are closed, but the designated flow remains limited by another bottleneck link M3-M4, sharing the resource with 5 other Gold flows. 6) (50-60 s) The 5 Gold flows crossing M3-M4 leave the system. We get the same resource share as in range 10-20 s. 7) (60-70 s) The designated flow occupies the available resources. Fig. 11 shows the same experiment with DCTCP congestion control, the results are very similar to NewReno.

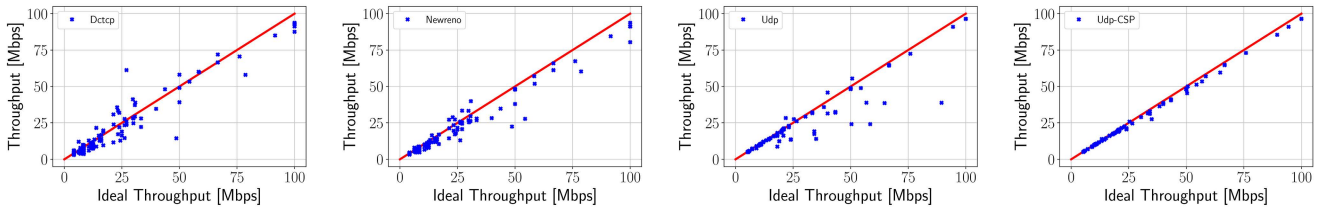
E. REALISTIC TRAFFIC

In addition to static flows, we also evaluated the PPV method in the parking lot topology (Fig. 7) with more realistic dynamic traffic patterns. We assume Poisson arrival with arrival rate 2 flows/s on each path. NewReno is used as TCP congestion control. For each arrival, the file sizes (flow sizes) to be downloaded are chosen from four discrete values (10 kB, 100 kB, 1 MB and 10 MB) with equal probability. The maximum number of active downloads per path is limited to 50, each flow has a separate Gold or Silver packet marker. We created evaluation scenarios to compare the resource sharing accuracy with and without the PPV method. Each evaluation scenario lasts 600 s.

Fig. 19 depicts the download times (flow completion times) of the 10 kB flows for all 4 paths with both simply tail drop (no resource sharing control) (*Ref*) and the PPV method (*G/S*). The box plot shows the minimum value, the 25th (bottom of the box), 50th (black line), 75th (top of the box), and 99th percentiles, while the mean value is illustrated as a triangle. Fig 20 depicts the same for flows of 10 MB size.

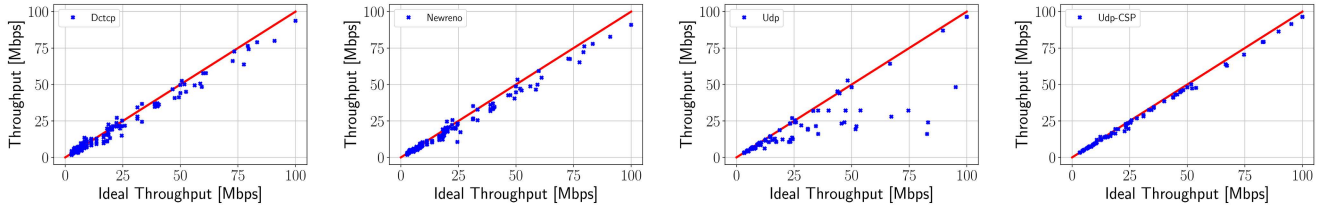
One can observe that in case of tail drop flows on path U1-U4 experience the longest download times. On this path, only 571 flows were launched out of the 1129 flow arrivals according to the applied arrival rate in the Poisson model and because of the applied concurrent download limit 50. This path competed with all 3 other paths and thus resulted in increased RTTs and packet losses.

When we applied the PPV method the download times of path U1-U4 became comparable to that of other paths. The download times of other paths increased not only because of the PPV-based resource sharing, but also because of serving



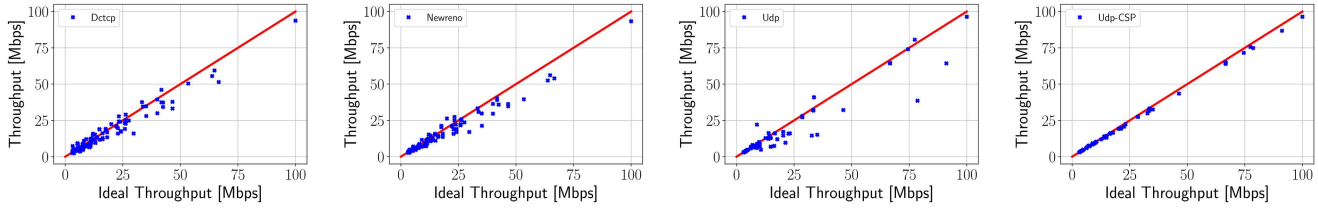
(a) DCTCP (b) NewReno (c) UDP (d) UDP-CSP

FIGURE 13. Deutsche Telekom; Ideal vs. Experimental throughput.



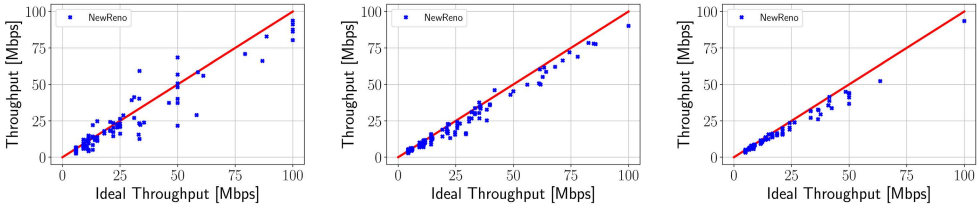
(a) DCTCP (b) NewReno (c) UDP (d) UDP-CSP

FIGURE 14. Geant; Ideal vs. Experimental throughput.



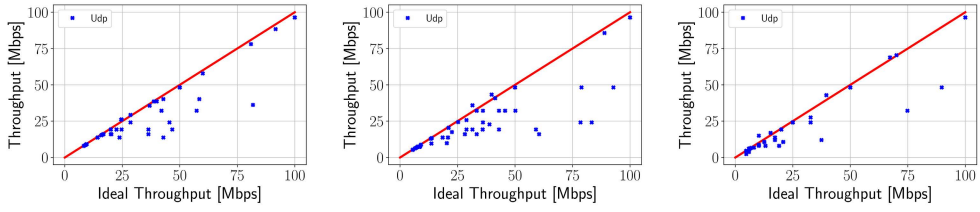
(a) DCTCP (b) NewReno (c) UDP (d) UDP-CSP

FIGURE 15. Sprint; Ideal vs. Experimental throughput.



(a) Deutsche Telekom (b) Geant (c) Sprint

FIGURE 16. DRR NewReno Ideal vs. Experimental throughput.

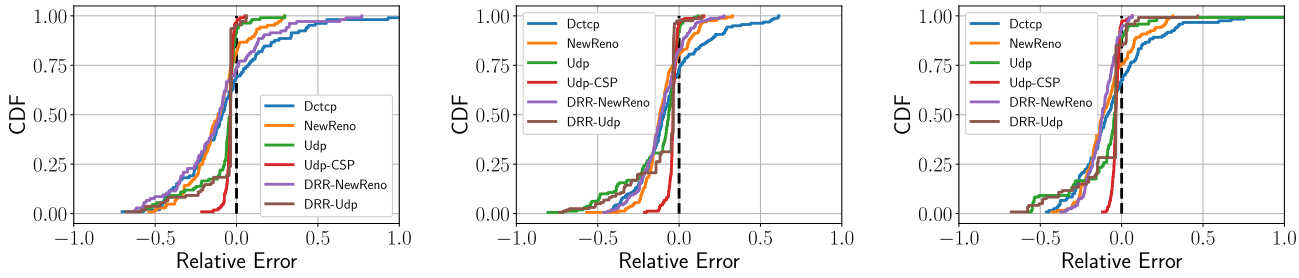


(a) Deutsche Telekom (b) Geant (c) Sprint

FIGURE 17. DRR Udp Ideal vs. Experimental throughput.

more flows on the long path. With PPV, all the flow arrivals were served on the long path, therefore the overall load of the system increased One can also see that the download times in case of 99% of the 10 kB flows also decreased for all paths, as PPV-based per-flow resource sharing improves consistency.

This scenario is based on complex bottleneck structure [25] where congestion control itself cannot handle RTT unfairness



(a) Deutche Telekom **(b)** Geant **(c)** Sprint

FIGURE 18. Cumulative distribution function of relative errors.

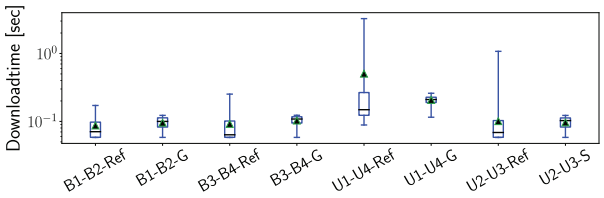


FIGURE 19. Download times for 10 kB file sizes.

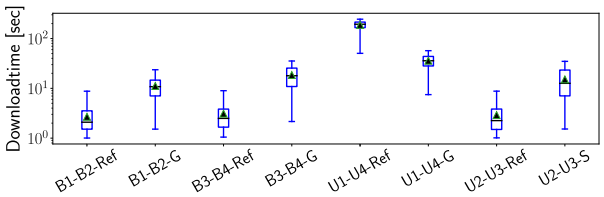


FIGURE 20. Download times for 10 MB file sizes.

and thus flows on the long path gets much less throughput share than their fair share. However, PPV-based resource sharing can also ensure good fairness among flows with very different RTTs.

F. EVALUATION IN REALISTIC TOPOLOGIES

In the last scenario, we evaluate the performance in three real-world network topologies chosen from the Topology-Zoo [9] collection: 1) Deutsche Telekom, 2) Geant and 3) Sprint. The topologies are illustrated in Fig. 12. Since these networks are different in size, we set the number of simulated flows to be proportional to the number of nodes N in the network: $N^2/10$ flows for Deutsche Telekom and Geant, and N^2 for Sprint. Note that Sprint topology only consists of a small number of nodes. The source and the destination of each flow have randomly been selected. Half of flows use Gold while the other half Silver policies. The sending rate of each the UDP flows is 100 Mbps. The packet size is 800 bytes for all sources. Every link on the topology has the same capacity of 100 Mbps while the link delay is calculated according to the geographic distance of its two end-points. To this end, we have applied the formula described in [26] for converting geographic distances to link delays. The flows are either TCP connections with DCTCP or NewReno congestion control, unresponsive UDP, or UDP controlled by CSP.

TABLE 1. Root-mean-square error (RMSE in Mbps) of the simulation on the different topolgies.

	DCTCP	NewReno	UDP	UDP-CSP
Deutsche Telekom	7.18	6.22	8.26	1.77
Geant	3.43	3.60	11.11	1.54
Sprint	3.84	4.03	6.15	1.18

TABLE 2. Root-mean-square error (RMSE in Mbps) of the simulation on the different topolgies with DRR.

	Deutsche Telekom	Geant	Sprint
NewReno	8.06	3.75	3.23
UDP	7.71	10.27	6.69

Figs. 13, 14 and 15 show scatter plots comparing the observed throughput of flows to their ideal throughput allocation for Deutsche Telekom, Geant and Sprint topologies, resp. One can observe that using NewReno congestion control there is only a small deviance from the ideal throughput for all topologies. DCTCP shows similar behavior with a somewhat wider deviance around the ideal throughput. The Deutsche Telekom case shows some significant outliers (e.g., 25 Mbps assigned throughput instead of the 50 Mbps ideal). These measurement points belong to flows crossing intercontinental links where the delay is large and thus the buffers used in the simulation are undersized, prohibiting these TCP flows to fully utilize the available capacity. Though unresponsive UDP flows show large deviations from the ideal throughput, in most cases flows get smaller effective throughput than their ideal and there are no cases when the occupied throughput is significantly larger than the equilibrium allocation. For accurate resource sharing with the PPV framework, the flows need to be congestion controlled. Still the PPV framework ensures that unresponsive sources cannot make extra benefit from their greedy behavior, thus the smaller throughput values. When Core-Stateless policing is enabled for the unresponsive UDP flows (UPD-CSP) the allocation is almost perfect. This shows that the PPV framework together with CSP not only avoids the dead packet problem, but it can also converge well to the ideal allocation.

Table 1 shows the root-mean-square error of the throughput values for the different investigated scenarios.

Figs. 16 and 17 show the same evaluation scenarios with DRR scheduling for NewReno and UDP flows, resp. The results are similar to what we get with the PPV method. Sprint seems slightly better.

Fig. 18 shows the distribution of relative errors experienced in the different topologies. Most relative error values are close to 0 and the deviance is more significant in the negative direction where the flows get smaller throughput than their ideal. One can also observe that the CDF of the DCTCP case is less steep than the one of other flow types. Its reason is that DCTCP has been developed for smaller RTTs where it can react to congestion faster. UDP-CSP results in an allocation that is close to the ideal. These figures also show that DRR results in very similar relative errors as the PPV method. The purple and orange, and the brown and green curves run very similarly. Using NewReno, PPV results in positive relative errors with larger probability than DRR, leading to the conclusion that DRR slightly underutilize the network resources. In case of UDP, there are not significant differences between the two resource sharing methods.

Table 2 shows the root-mean-square error of the throughput values for DRR scenarios.

VIII. DISCUSSION OF THE RESULTS

Our theoretical analysis and simulation-based evaluation show that PPV method can maximize the minimum utility of flows in an arbitrary topology if the sending rates are controlled by either a congestion control or a policer like CSP. In this section, we discuss the results with respect to state-of-the-art methods.

PPV shares the core-stateless idea of Core-Stateless Fair Queueing (CSFQ) [15]. CSFQ labels packets with flow rate estimates and calculates the fair rates at each potential bottlenecks. In case of congestion, the ratio of the two rate estimates determines the drop probability to be applied on the incoming packet. As a result, routers drop the throughput of flows above the fair rate. One can observe that the behavior of CSFQ can be emulated by the PPV method. If each flow uses the same TVF (e.g., $v(x) = 1/x$, $x > 0$), the drop minimum packet value first strategy leads to a cutoff value (CTV) that represents the fair share at a given bottleneck. The PV-based dropping will cut the flow throughput to the fair share. As a consequence, our theoretical results hold for CSFQ as well: it ensures max-min fair allocation with both classic and scalable congestion controls. Max-min fair allocation is a well-known property of CSFQ. Though CSFQ can ensure simple weighted fairness, it is not able to handle bandwidth-function like policies as PPV.

Among the stateful alternatives, DRR [10] scheduling can also be used to ensure max-min fair allocation. However, it needs to maintain a large number of queues at each potential bottlenecks in the network, resulting in scalability issues. If queues are shared with other flows, the resulted fairness is getting worst. DRR can also be extended to support flows with different weights but weights need to be maintained at any points of the network, making the management and configuration of such an approach much more complicated than CSFQ and PPV. The dead packet problem caused by unresponsive flows also holds for DRR.

Another stateful approach, BWE [13] was proposed by Google to implement bandwidth function-based resource sharing in its private WAN. BWE uses the principle of Software Defined Networking (SDN) and solves resource sharing in a centralized way. Similarly to PPV, BWE also provides a provable solution for the generalized max-min fair allocation problem. PPV follows a distributed approach while BWE is a centralized method that requires continuous feedback between the logically centralized controller and the network nodes.

IX. CONCLUSION

In this paper, we have analyzed the network-wide behavior of a recent core-stateless resource sharing proposal called Per Packet Value (PPV). Our theoretical results show that the PPV framework provides a distributed solution for the generalized max-min fair allocation problem in an arbitrary topology. We have shown the existence and uniqueness of the equilibrium allocation which is a feasible throughput allocation maximizing the transmitted packet values in the network. We have extended the mathematical model to emulate the behavior of both scalable (DCTCP-like) and classic (AIMD-like) congestion controls. The results have also been validated with thorough packet-level simulations, using various topologies. The simulation results are in accordance with our theoretical analysis, showing that the PPV method can effectively solve resource sharing between congestion controlled flows and it also ensures that non-responsive UDP flows cannot gain advantages over well-behaving flows. Finally, we have also proposed core-stateless policing that can regulate unfriendly flows that do not react to packet losses. It applies a lightweight mechanism that does not affect the behavior of well-behaving flows.

APPENDIX A
THE EXISTENCE AND UNIQUENESS OF EQUILIBRIUM IN STEADY STATE

Proof of Theorem 1: We first prove that every feasible allocation x for which $v^{\downarrow}(x)$ is lexicographically minimal is an equilibrium. Suppose that x is not an equilibrium allocation; then there is an index i such that $x_i < d_i$ and $x_i < v_i^{-1}(\alpha_e(x))$ for every $e \in P_i$. Let $F = \{e \in P_i : x(e) < g_e\}$. If $e \in P_i \setminus F$, then there is at least one index j such that $x_j = v_j^{-1}(\alpha_e(x))$ by the definition of $\alpha_e(x)$. Let

$$J = \bigcup_{e \in P_i \setminus F} \{j \in [1,k] : x_j = v_j^{-1}(\alpha_e(x))\}.$$

Note that $v_i(x_i) > v_j(x_j)$ for every $j \in J$. We can choose an $\epsilon > 0$ such that $\epsilon < g_e - x(e)$ for every $e \in F$ and $v_i(x_i+\epsilon) > v_j(x_j - \epsilon)$ for every $j \in J$. Define

$$x'_j = \begin{cases} x_j + \epsilon & \text{if } j = i, \\ x_j - \epsilon & \text{if } j \in J, \\ x_j & \text{otherwise.} \end{cases}$$

Then x' is a feasible allocation such that $v^{\downarrow}(x')$ is lexicographically smaller than $v^{\downarrow}(x)$, a contradiction.

For the second part of the proof, let x be an equilibrium allocation. Let $t_1 > t_2 > \cdots > t_s$ be the values in $\{v_1(x_1), v_2(x_2), \ldots, v_k(x_k)\}$ in decreasing order, without multiplicity. For any allocations x', let $I_q(x') = \{i \in [1,k] : v_i(x_i') = t_q\}$. We prove the following property, which implies the theorem:

Claim 10: If x' is a feasible throughput allocation for which $v^{\downarrow}(x')$ is lexicographically minimal, then $I_q(x') = I_q(x)$ for every $q \in [1,s]$.

Proof: We prove the claim in increasing order of q; suppose it is true for $1, \ldots, q-1$. Notice that it is enough to prove $I_q(x') \supseteq I_q(x)$, because if $I_q(x') \supsetneq I_q(x)$, then $v^{\downarrow}(x')$ cannot be lexicographically minimal. Let $i \in I_q(x)$. Since x is an equilibrium allocation, we have

$$x_i = \bar{x}_i = \min_{e \in P_i} v_i^{-1}(\alpha_e(x)).$$

If $x_i = d_i$, then obviously $x_i' \le d_i = x_i$, but it cannot be strictly smaller since $v^{\downarrow}(x')$ is lexicographically minimal.

Suppose that $x_i < d_i$; then there is an edge $e \in P_i$ such that $x(e) = g_e$ and $x_i = v_i^{-1}(\alpha_e(x))$. Let $J = \{j : e \in P_j\}$. Since $x_i = v_i^{-1}(\alpha_e(x))$, $i \in \operatorname{argmin}_{j \in J} v_j(x_j)$, and therefore $J \subseteq I_1(x) \cup \cdots \cup I_q(x)$. Since $v^{\downarrow}(x')$ is lexicographically minimal and $I_\ell(x') = I_\ell(x)$ if $\ell < q$, we must have $x_j' \ge x_j$ for every $j \in J$. But $\sum_{j \in J} x_j' \le g_e = \sum_{j \in J} x_j$, so $x_j' = x_j$ for every $j \in J$, in particular $x_i' = x_i$. □

If $I_q(x') = I_q(x)$ for every $q \in [1,s]$, then $x = x'$, so the claim implies the theorem. □

APPENDIX B
CONVERGENCE TO THE EQUILIBRIUM ALLOCATION WITH TIGHT CUT UPDATE PROTOCOLS

Let us start with a technical lemma before proving Theorem 4.

Lemma 11: If $\alpha_e(x) > 0$ for a given edge $e \in E$, then $\alpha_e(x) = \max\left\{\alpha \ge 0 : \sum_{i:e \in P_i} \min\{x_i, v_i^{-1}(\alpha)\} \ge g_e\right\} = \max\left\{\alpha \ge 0 : \sum_{i:e \in P_i} \min\{x_i, v_i^{-1}(\alpha)\} = g_e\right\}$.

Proof: If $\alpha_e(x) > 0$, then there exist α such that $\sum_{i:e \in P_i} \min\{x_i, v_i^{-1}(\alpha)\} \ge g_e$. Since the functions v_i^{-1} are monotone decreasing and continuous, also the function $\alpha \mapsto \sum_{i:e \in P_i} \min\{x_i, v_i^{-1}(\alpha)\}$ is monotone decreasing and continuous. Hence, the set $\{\alpha : \sum_{i:e \in P_i} \min\{x_i, v_i^{-1}(\alpha) \ge g_e\}$ is a closed interval. It follows that

$$\inf\left\{\alpha \ge 0 : \sum_{i:e \in P_i} \min\{x_i, v_i^{-1}(\alpha)\} < g_e\right\} = \max\left\{\alpha \ge 0 : \sum_{i:e \in P_i} \min\{x_i, v_i^{-1}(\alpha)\} \ge g_e\right\},$$

and by continuity this point has $\sum_{i:e \in P_i} \min\{x_i, v_i^{-1}(\alpha_e(x))\} = g_e$. □

For the proof of Theorem 4, we also need a couple of definitions and additional lemmas.

Definition 12 (Equilibrium Bound of an Edge): Let e be an edge in the network N, and suppose that $x_{i_1}, \ldots, x_{i_j}$ are the flows that use edge e. Take the network N_e that consists of the single edge e, and throughput allocation $x_{i_1}, \ldots, x_{i_j}$ with their original throughput-value functions. We denote by β_e the cutoff value of the equilibrium allocation on N_e.

Lemma 13: The value of β_e is

$$\max_{x \text{ alloc. for } N_e}\left\{\alpha \ge 0 : \sum_{k=1}^{j} \min\{x_{i_k}, v_{i_k}^{-1}(\alpha)\} = g_e\right\}.$$

Proof: By Lemma 11, it is enough to prove

$$\beta_e = \max\{\alpha_e(x) : x \text{ is a throughput allocation for } N_e\}.$$

As the equilibrium allocation of N_e is a throughput allocation for N_e, it is enough to prove that $\alpha_e(x) \le \beta_e$ for each allocation of N_e. We can assume without loss of generality that flows using edge e are $1, \ldots, j$.

Let y be the equilibrium allocation of N_e. By definition, $\beta_e = \alpha_e(y)$.

If $\beta_e = 0$, then $\sum_{i=1}^{j} d_i \le g_e$ and $y_i = d_i$ for each $i = 1, \ldots, j$. Then for each throughput allocation x, we have $\sum_{i=1}^{j} x_i \le \sum_{i=1}^{j} d_i \le g_e$, hence $\alpha_e(x) = 0$.

If $\beta_e > 0$, then $\sum_{i=1}^{j} v_i^{-1}(\beta_e) = g_e$. Suppose that for an allocation x, $\alpha_e(x) \ge \beta_e$. Then $v_i^{-1}(\alpha_e(x)) \le v_i^{-1}(\beta_e)$ by the monotonicity of the functions v_i. Since $\alpha_e(x) \ge \beta_e > 0$, we have $\alpha_e(x) > 0$. Hence by Lemma 11, we have $\sum_{i=1}^{j} \min\{x_i, v_i^{-1}(\alpha_e(x))\} = g_e$. Thus, $\sum_{i=1}^{j} \min\{x_i, v_i^{-1}(\alpha_e(x))\} = \sum_{i=1}^{j} v_i^{-1}(\beta_e) = g_e$. As all the summands in the second sum are greater or equal to the corresponding summand of the first sum, this is only possible if $\min\{x_i, v_i^{-1}(\alpha_e(x))\} = v_i^{-1}(\beta_e)$ for each $i = 1, \ldots, j$. By the strict monotonicity of the TVFs, this implies $\alpha_e(x) = \beta_e$. □

Lemma 14: For any edge e and throughput allocation x of network N, $\alpha_e(x) \le \beta_e$.

Proof: This follows directly from Lemma 13, since the restriction of x to coordinates used by e gives an allocation for N_e, and $\alpha_e(x)$ only depends on these coordinates. □

Lemma 15: Let our network consist of a single edge e, and suppose that all flows use some tight cut protocol. Then the value of α_e increases monotonically in each phase.

Proof: If $\alpha_e = 0$ in a phase, then this is obviously true, so assume $\alpha_e > 0$. A flow x_i has loss if $x_i > v_i^{-1}(\alpha_e(x))$, and in this case in the next phase, it is going to be $\bar{x}_i = v_i^{-1}(\alpha_e(x))$. A flow x_i is lossless if $x_i \le v_i^{-1}(\alpha_e(x))$, and in this case in the next phase, its sending rate is going to be larger than $x_i = \bar{x}_i$.

Hence in both cases, for throughput allocation x' of the next phase, $\min\{x_i', v_i^{-1}(\alpha(x))\} \ge \min\{x_i, v_i^{-1}(\alpha(x))\}$. Adding up, $\sum_{i=1}^{k} \min\{x_i', v_i^{-1}(\alpha(x))\} \ge \sum_{i=1}^{k} \min\{x_i, v_i^{-1}(\alpha(x))\} \ge g_e$, thus, $\alpha_e(x') \ge \alpha_e(x)$ by Lemma 11. □

Lemma 16: Suppose that in a network, the edge e is used by flows $x_1, \ldots x_j$, and $x_i \ge v_i^{-1}(\beta_e)$ for each $1 \le i \le j$. Then $\alpha_e(x) = \beta_e$.

Proof: Let y be the equilibrium allocation of N_e on $x_1, \ldots x_j$ and defined arbitrarily for the rest of the coordinates.

If $\beta_e = 0$, then $y_i = d_i$ and $\sum_{i=1}^{j} y_i = \sum_{i=1}^{j} d_i = \sum_{i=1}^{j} v_i^{-1}(\beta_e) \leq g_e$. If $\beta_e > 0$, then by Lemma 11, $\sum_{i=1}^{j} y_i = \sum_{i=1}^{j} v_i^{-1}(\beta_e) = g_e$. Since $\alpha_e(x)$ only depends on $x_1, \dots x_j$, for the determination of $\alpha_e(x)$ we can imagine that the network only consists of the edge e and apply Lemma 14, to deduce that $\alpha_e(x) \leq \beta_e$. Hence for each i, $v_i^{-1}(\alpha_e(x)) \geq v_i^{-1}(\beta_e)$. Thus, $\sum_{r=1}^{j} \min\left\{x_i, v_i^{-1}(\alpha_e(x))\right\} \geq \sum_{i=1}^{j} v_i^{-1}(\beta_e) \geq g_e$. From the definition of $\alpha_e(x)$, it follows that $\sum_{r=1}^{j} \min\left\{x_{i_r}, v_{i_r}^{-1}(\alpha_e(x))\right\} \leq g_e$. Hence $\min\left\{x_i, v_i^{-1}(\alpha_e(x))\right\} = v_i^{-1}(\beta_e) = y_i$ for each $i = 1, \dots j$. From the strict monotonicity of the TVFs, we conclude that $\alpha_e(x) = \alpha_e(y) = \beta_e$. □

Proof of Theorem 4: Let $N = (V, E)$ be our network. Partition the edges $E = E_1 \cup \dots \cup E_s$ such that $\beta_e = \beta_{e'}$ if $e, e' \in E_i$ for some $1 \leq i \leq s$, and $\beta_e > \beta_{e'}$ if $e \in E_i$, $e' \in E_j$ and $1 \leq i < j \leq s$.

Claim 17: If $e \in E_1$ and $x_1, \dots x_j$ are the flows such that $e \in P_i$, then eventually, α_e gets fixed to β_e, and for $i = 1, \dots, j$, x_i gets fixed to $v_i^{-1}(\beta_e)$ regardless of the rest of the network.

Proof: Suppose that in a given phase, a flow x_i has loss because of an edge e', that is, $\bar{x}_i = v_i^{-1}(\alpha_{e'}(x))$. By Lemma 14, $\alpha_{e'}(x) \leq \beta_{e'}$. Also, since $e \in E_1$, $\beta_e \geq \beta_{e'}$. Hence $\bar{x}_i = v_i^{-1}(\alpha_{e'}(x)) \geq v_i^{-1}(\beta_{e'}) \geq v_i^{-1}(\beta_e)$ using the monotonicity of v_i.

If flow i is lossless, then x_i increases by at least Δ or attains d_i in the next phase. Hence eventually each flow rate x_i, $i = 1, \dots j$ will have value at least $v_i^{-1}(\beta_e)$. Hence by Lemma 16 in the next phase, $\alpha_e(x) = \beta_e$, and $x_i = v_i^{-1}(\beta_e)$. From this time on, e prevents flow rates $x_1, \dots x_j$ from increasing, while by our previous argument, another edge e' cannot cut these flows below the actual values, hence flow rates $x_1, \dots x_j$ get fixed on the actual values, and so does α_e on β_e. □

Stop the process after $\alpha_e(x)$ gets fixed to β_e for each $e \in E_1$. Let us call $\beta_1 = \beta_e$ for any edge $e \in E_1$. We can assume that for some j, $x_1, \dots x_j$ are the flows that use any of these edges. Then from this point, such a flow x_i will always have value $v_i^{-1}(\beta_1)$. Hence we get an equivalent problem if we contract each $e \in E_1$ to a point, forget the flows $x_1, \dots, x_j$, and for each edge e' and each $1 \leq i \leq j$, if $e' \in P_i$, then we subtract $v_i^{-1}(\beta_1)$ from $g_{e'}$.

This way we have a problem with a smaller number of edges and flows. Eventually we get a situation where each flow rate is fixed, hence the system is in equilibrium. □

APPENDIX C
LOWER BOUND FOR THE AIMD-LIKE UPDATE PROTOCOL

In this section, we provide the mathematical proof for the guaranteed lower bound for the capacity assigned to flows by the AIMD protocol.

Proof of Theorem 8: Let $B = \min\{d_i, \min_{e \in P_i} v_i^{-1}(\beta_e)\}$. We show that after x_i first has loss, it always has value at least $B/2$. While x_i is lossless, its value always increases by at least Δ, hence eventually x_i will have loss or it will attain d_i.

If x_i has loss in a phase, then $\bar{x}_i = v_i^{-1}(\alpha_e(x))$ for some $e \in P_i$. Moreover, by Lemma 14 and by the monotonicity of v_i, $v_i^{-1}(\alpha_e(x)) \geq v_i^{-1}(\beta_e)$, so $\bar{x}_i \geq B$. As x_i follows the AIMD protocol, in the next phase, $x_i' = \frac{x_i}{2} \geq \frac{\bar{x}_i}{2} \geq \frac{B}{2}$. In subsequent phases, if x_i has loss then we can repeat the same argument to show that it is at least $B/2$ in the next phase. Thus, x_i remains above $B/2$ in all subsequent phases. □

REFERENCES

[1] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, 2014.

[2] Z. Yu, J. Wu, and B. Vladimir, "Twenty years after: Hierarchical core-stateless fair queueing," in *Proc. 18th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, Apr. 2021, pp. 29–45. [Online]. Available: https://www.usenix.org/conference/nsdi21/presentation/yu

[3] F. Fejes, S. Nadas, G. Gombos, and S. Laki, "A core-stateless L4S scheduler for P4-enabled hardware switches with emulated HQoS," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, May 2021, pp. 1–2.

[4] S. Nadas, Z. R. Turanyi, and S. Racz, "Per packet value: A practical concept for network resource sharing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–7.

[5] S. Laki, G. Gombos, S. Nádas, and Z. Turányi, "Take your own share of the PIE," in *Proc. Appl. Netw. Res. Workshop*, Jul. 2017, pp. 27–32.

[6] S. Nádas, G. Gombos, P. Hudoba, and S. Laki, "Towards a congestion control-independent core-stateless AQM," in *Proc. Appl. Netw. Res. Workshop*, Jul. 2018, pp. 84–90.

[7] S. Laki, S. Nadas, G. Gombos, F. Fejes, P. Hudoba, Z. Turanyi, Z. Kiss, and C. Keszei, "Core-stateless forwarding with QoS revisited: Decoupling delay and bandwidth requirements," *IEEE/ACM Trans. Netw.*, vol. 29, no. 2, pp. 503–516, Apr. 2020.

[8] T. Kelly, S. Floyd, and S. Shenker, "Patterns of congestion collapse," Int. Comput. Sci. Inst., Univ. Cambridge, Cambridge, MA, USA, Tech. Rep., 2003.

[9] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, Sep. 2011.

[10] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round Robin," in *Proc. Conf. Appl., Technol., Architectures, Protocols Comput. Commun. (SIGCOMM)*, New York, NY, USA, 1995, p. 231, doi: 10.1145/217382.217453.

[11] K. Kogan, D. Menikkumbura, G. Petri, Y. Noh, S. I. Nikolenko, A. Sirotkin, and P. Eugster, "Towards software-defined buffer management," *IEEE/ACM Trans. Netw.*, vol. 28, no. 5, pp. 2337–2349, Oct. 2020.

[12] K. Nagaraj, D. Bharadia, H. Mao, S. Chinchali, M. Alizadeh, and S. Katti, "NUMFabric: Fast and flexible bandwidth allocation in datacenters," in *Proc. ACM SIGCOMM Conf.*, New York, NY, USA, Aug. 2016, pp. 188–201.

[13] A. Kumar, S. Jain, U. Naik, A. Raghuraman, N. Kasinadhuni, E. C. Zermeno, C. S. Gunn, J. Ai, B. Carlin, M. Amarandei-Stavila, M. Robin, A. Siganporia, S. Stuart, and A. Vahdat, "BwE: Flexible, hierarchical bandwidth allocation for wan distributed computing," in *Proc. ACM Sigcomm*, New York, NY, USA, 2015, pp. 1–14.

[14] M. Carlson, W. Weiss, S. Blake, Z. Wang, D. Black, and E. Davies. (Dec. 1998). *An Architecture for Differentiated Services*. Internet Requests for Comments, RFC Editor, RFC 2475. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2475.txt

[15] I. Stoica, S. Shenker, and H. Zhang, "Core-stateless fair queueing: A scalable architecture to approximate fair bandwidth allocations in high-speed networks," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 33–46, Feb. 2003.

[16] Z. Cao, E. Zegura, and Z. Wang, "Rainbow fair queueing: Theory and applications," *Comput. Netw.*, vol. 47, no. 3, pp. 367–392, Feb. 2005.

[17] F. Kelly, "Charging and rate control for elastic traffic," *Eur. Trans. Telecommun.*, vol. 8, no. 1, pp. 33–37, Jan./Feb. 1997.

[18] S. Nádas, G. Gombos, F. Fejes, and S. Laki, "A congestion control independent L4S scheduler," in *Proc. Appl. Netw. Res. Workshop ZZZ*, Jul. 2020, pp. 45–51.

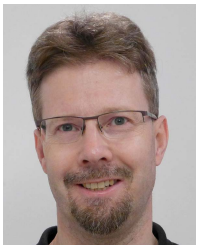
[19] M. Menth and N. Zeitler, "Fair resource sharing for stateless-core packet-switched networks with prioritization," *IEEE Access*, vol. 6, pp. 42702–42720, 2018.

[20] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center TCP (DCTCP)," *ACM SIGCOMM Comput. Commun. Rev.*, New York, NY, USA, vol. 40, no. 4, pp. 63–74, Aug. 2010.
[21] B. Briscoe, K. De Schepper, O. Tilmans, M. Kühlewind, J. Misund, O. Albisser, and A. S. Ahmed, "Implementing the 'prague requirements' for low latency low loss scalable throughput (L4S)," in *Proc. NetDev 0x13*, Prague, Czech Republic, Mar. 2019, pp. 1–11.
[22] N. Cardwell, Y. Cheng, S. H. Yeganeh, I. Swett, V. Vasiliev, P. Jha, Y. Seung, M. Mathis, and V. Jacobson, "Bbrv2: A model-based congestion control," presentation in ICCRG at IETF 104th Meeting, Mar. 2019.
[23] B. Briscoe, R. Woundy, and A. Cooper. (Dec. 2012). *Congestion Exposure (ConEx) Concepts and Use Cases*. RFC. [Online]. Available: https://www.rfc-editor.org/info/rfc6789
[24] W. Yu, W. Rhee, S. Boyd, and J. M. Cioffi, "Iterative water-filling for Gaussian vector multiple-access channels," *IEEE Trans. Inf. Theory*, vol. 50, no. 1, pp. 145–152, Jan. 2004.
[25] J. Ros-Giralt, A. Bohara, S. Yellamraju, M. H. Langston, R. Lethin, Y. Jiang, L. Tassiulas, J. Li, Y. Tan, and M. Veeraraghavan, "On the bottleneck structure of congestion-controlled networks," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 3, no. 3, pp. 1–31, Dec. 2019, doi: 10.1145/3366707.
[26] S. Laki, P. Matray, P. Haga, T. Sebok, I. Csabai, and G. Vattay, "Spotter: A model based active geolocation service," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 3173–3181.

GERGŐ GOMBOS received the M.Sc. degree in computer science from Eötvös Loránd University (ELTE), in 2012, and the Ph.D. degree, in 2018. The topic of his thesis is the semantic web and the distributed computing in Hadoop environment. Since 2018, he has been working as an Assistant Professor at the Department of Information Systems, ELTE. His research interests include computer networks, Hadoop and Spark environments, big data architectures, and NoSQL databases.

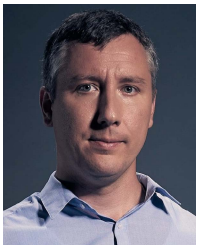
DÁVID KIS received the M.Sc. degree in computer science from Eötvös Loránd University, where he is currently pursuing the Ph.D. degree. His current research interests include computer networks, the IoT, programmable data planes, and their applications.

LILLA TÓTHMÉRÉSZ received the M.Sc. and Ph.D. degrees in mathematics from Eötvös Loránd University. Her research interests include combinatorial optimization, algorithms, and combinatorial geometry.



TAMÁS KIRÁLY received the M.Sc. and Ph.D. degrees in mathematics from Eötvös Loránd University, Budapest, in 1999 and 2004, respectively. He is currently an Associate Professor at the Department of Operations Research, Eötvös Loránd University, and also a member of the MTA-ELTE Egerváry Research Group on Combinatorial Optimization. His research was presented at conferences like FOCS, SODA, and IPCO, and he published more than 30 articles in journals, including *Mathematical Programming*, *Journal of Combinatorial Theory*, *Mathematics of Operations Research*, and *Combinatorica*. His main research interests include optimization, polyhedral combinatorics, approximation algorithms, and game theory.

SZILVESZTER NÁDAS received the M.Sc. degree in electrical engineering from the Budapest University of Technology and Economics, in 2000. Since then, he has been with Ericsson Research, Hungary. He has been working with traffic management for 20 years, and his main interest is controlling resource sharing. He is also interested in the interaction of different mechanisms of traffic management (e.g., AQM and congestion control) and he believes that more coordination among the mechanisms is necessary.



SÁNDOR LAKI (Member, IEEE) received the M.Sc. and Ph.D. degrees in computer science from Eötvös Loránd University, in 2007 and 2015, respectively. He is currently an Assistant Professor with the Department of Information Systems, Eötvös Loránd University. He has authored over 40 peer-reviewed papers and demo papers, including publications at the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, INFOCOM, ICC, and SIGCOMM. His research interests include active and passive network measurement, traffic analytics, programmable data planes, and their application for new networking solutions.