Stateless Resource Sharing in Networks with Multi-Layer Virtualization

Szilveszter Nádas*, Zoltán Turányi*, Gergő Gombos†, Sándor Laki‡
*Traffic Analysis and Network Performance Laboratory, Ericsson Research, Budapest, Hungary
{szilveszter.nadas, zoltan.turanyi}@ericsson.com
†3in Research Group, ELTE Eötvös Loránd University, Martonvásár, Hungary
‡Communication Networks Laboratory, ELTE Eötvös Loránd University, Budapest, Hungary
{ggombos, lakis}@inf.elte.hu

***Abstract*—Network QoS, fairness and resource sharing control are open challenges of network slicing and virtualization in 5G and future networks providing ultra-high speed Internet access. Traditional stateful solutions either employ the one-size-fits-all approach to provide services to end-users, regardless of the requirements of vertical services, or require real-time network monitoring and complex feedback loops for ensuring appropriate resource allocation at any time. In the past years, different core-stateless resource sharing solutions as scalable alternatives to traditional stateful approaches have recently emerged for closed networking domains like access, enterprise and data center networks. In this paper, we extend the core-stateless Per Packet Value (PPV) framework with the support of resource sharing policies across multiple layers of virtualization where policies defined by both physical and virtual network operators should be taken into account at the same time. To this end, we propose a re-marking mechanism that re-calculates packet markings during the transition from one virtualization layer to another, ensuring the desired resource sharing among end-users, network slices and physical networks without the need of real-time monitoring and complex feedback loops in the network core. To the best of our knowledge, this is the first core-stateless proposal that offers resource sharing for multiple layers of virtualization.**

***Index Terms*—Stateless, resource sharing, QoS, virtualization, network slicing, PPV**

I. Introduction

Network slicing and network virtualization are receiving significant attention in telecommunication as a means to provide network as a service for different use cases, enabling to build multiple virtual networks on a shared infrastructure. It has been proposed to address diversified service requirements, using isolated logical networks built from a set of virtual resources on the top of a physical infrastructure. Resource sharing among slice tenants across multi-layer of virtualization is, however, still a key issue in 5G and next generation networks [1], [2]: 1) the ultra-high speed end-user access makes overprovisioning of the core network difficult, costly and irrational; 2) the dynamic characteristics of network load requires fast and practical solutions for resource allocation at any congestion situations; and finally 3) highly scalable solutions are needed to control resource allocation among large number of end-users/flows.

Though Quality of Service (QoS) is one of the most studied area in networking with a vast array of valuable and practical knowledge, the issues of network QoS and resource management are still not fully solved problems. Traditional QoS and resource sharing solutions are stateful and either employ the one-size-fits-all approach to provide services to end-users, regardless of the requirements of vertical services or end-users, or require real-time network monitoring and complex feedback loops for ensuring appropriate resource allocation at any time.

However, different core-stateless solutions have also emerged to support different resource sharing policies by very simple mechanisms. Accordingly, the packets are marked by a color or a value at the network edges by applying the operator policy and then dropping and/or scheduling packets in the network core solely rely on packet markings. Since packet marking can be done in a highly distributed way, while packet dropping and scheduling are stateless and simple, such methods are far more scalable than traditional stateful solutions (especially if per user QoS policies should be ensured), providing alternatives in closed networking domains like high-speed access, enterprise and data center networks. The core-stateless idea originates from the early 2000s, but it has recently been revised by different proposals [3], [4], [5] for environments where weighted fairness needs to be ensured among a large number of end-users/flows and the overprovisioning of the core network is not feasible or irrationally costly.

In this paper, our goal is to propose a core-stateless QoS framework that support flexible resource sharing policies among entire virtual networks, while still supporting the policies for the flows within each individual virtual network over the same Resource Node (e.g. a router). The proposed solution does not require complex feedback loops and constant re-tuning of the network. The proposed method called Hierarchical Per Packet Values (HPPV) extends the single network marking concept called Per Packet Value (PPV) [4] to network/flow hierarchies, where in addition to providing services to end-users the operator itself can also be a user of another network operator and thus re-marking of the packets at the border of network domains are needed. As a result, the new packet value should express the operator policies of both domains (or the entire operator hierarchy if any).

The remaining part of the paper is organized as follows: Section II gives a brief overview of the related work. Then, in Section III, we introduce our system model and briefly overview the Per Packet Value (PPV) concept [4] the proposed approach is based on. In Section IV we introduce the proposed extension needed for the support of resource sharing policies across multiple layers of virtualization. Section V describes a practical algorithm for packet value re-marker nodes. This is followed by a simulation based evaluation in Section VI. Finally, we conclude the paper and outline future work in Section VII.

II. Related Work

Core-stateless resource sharing. Several QoS and resource sharing architectures have been proposed in the past decade that maintain no per-flow state inside the network. Such architectures typically perform certain packet marking at the edge of the network and require only simple processing in Resource Nodes (e.g. routers). The most widely known of such architectures is DiffServ [6] where markings identify a set of pre-defined policies, Per-Hop Behaviors (PHBs). One limitation of this approach is that Resource Nodes have to be re-programmed to support a new policy. In addition, they have to be tuned to the offered traffic to achieve desired QoS goals. Another such proposal is Core Stateless Fair Queuing [7]. CSFQ implements a single policy: proportional fair bandwidth sharing and marks packets of each flow with its estimated rate at the edge. ConEx [8] also aims to provide fairness at least for traffic that causes congestion using expected congestion indications as markings. pFabric [5] assigns a value to each packet and aims to maximize value transmitted, but its goal is not generic resource sharing, rather shortest-job-first data center networking instead. Rainbow Fair Queueing (RFQ) [9] introduced the notion of using more than a few drop precedence levels (colors). These levels are than used to select the packets to be dropped at bottlenecks. ABC [3] 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. Per Packet Value (PPV) [4] generalizes the concept of both ABC and RFQ, proposing scalar values assigned to the packets enabling arguments about value maximization and Throughput-Value Functions to express operator policies. PPV provides a practical, distributed approximate solution to the network utility maximization problem, requiring neither per flow processing inside the network nor feedback to the edges. PPV is applicable to networks, where routing is already catered for. Further improvements of the PPV concept are published in [10], [11] where the concept of value maximization is combined with AQM mechanisms for ensuring controlled delay. Building on PPV we extended it with smart re-marking for network virtualization. bb

Network slicing and virtualization. Network slicing exploits the concept and tools of network virtualization that enables flexible and dynamic network management to allow multiple heterogeneous and service-specific virtual networks to share a single physical infrastructure [1], [2]. Such a virtual network can be used by e.g. a mobile virtual network operator lacking network infrastructure or having limited capacity/coverage and leases resources from the operator of physical network, or vertical industries exploiting the physical infrastructure for services complementary to telecommunication industry. The efficient allocation of network resources to slices is a challenging problem that can be formulated as a integer linear program (ILP), similarly to virtual network embedding problems [12]. To solve it different network slicing approaches uses different methods: 1) for small-scale slices there exists exact algorithms; 2) for large-scale slices heuristic or meta-heuristic strategies are more efficient. The key challenge is that the algorithm should be practical enough and quickly adapt to changes in the network conditions. Most of the solutions use real-time monitoring and complex feedback loops to ensure the appropriate resource allocation at any time. A recent complex QoS proposal for private WANs is BwE [13] that relies on a centralized bandwidth broker to manage resource sharing in a global, virtualized packet network. In contrast, the PPV-based solution proposed in this paper uses an open-loop, distributed operation instead to fulfill similar requirements. In HPPV, each resource owner can decide and enforce its own policies via packet (re)marking, which simplifies trust issues and allows easier QoS interworking between different administrative domains and the multiple-layers of virtualization.

III. System Model

The proposed framwork is based on the PPV concept [4] that extends the idea of core stateless resource sharing methods like [9], [7] by marking each packet with a continuous value called Packet Value (PV). Informally, PV represents the reward given to the network operator when the packet is successfully delivered. The network aims at maximizing the total profit of the operator by maximizing the total aggregated PV delivered. The system model of PPV is split into two phases: 1) Packet marking at network edge; 2) Packet scheduling and dropping based on the Packet Value at resource nodes (e.g. routers) in the middle of the network.

First, packets are marked at the edge of the network by using the resource sharing policy of the operator. Note that marking may require flow-level, application-level or even user-specific information to determine the policy to be applied. Operator policies are described by Throughput-Value Functions (TVFs) (marked by $V(.)$), that basically defines the PV distribution of a flow for any sending rate. Note that the meaning of a flow may be different from one scenario to another: e.g. traffic generated by a specific application or application group of a given user, the total traffic of an end-user of a virtual or physical operator, the total traffic of a virtual operator, etc. Specifically, for any throughput value b, the traffic up to b shall receive a PV of $V(b)$ or higher. Accordingly, at high congestion only packets with high PVs are transmitted, more precisely packets with PV above a given Congestion Threshold Value (CTV). Note that the amount of high and low PV packets determines the resource share between various flows, resulting in that at high congestion, flows with larger share of high PV packets receive more throughput.

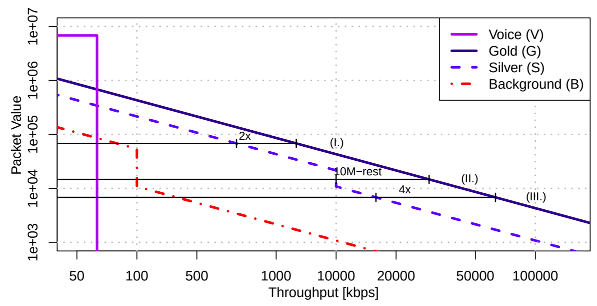

Fig. 1. TVF examples, log-log scale.

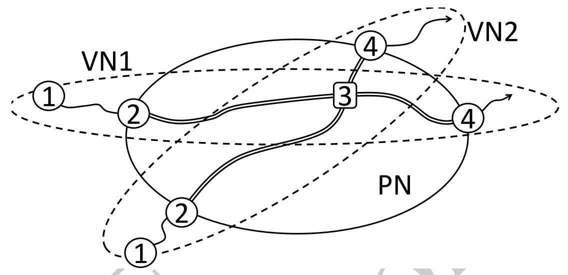

Fig. 2. Resource Sharing across Multi-Layer Virtualization.

In a stationary situation, the most value can be delivered over a shared bottleneck by transmitting all packets with PV above the Congestion Threshold Value and dropping the ones below. Assuming that CTV is determined accurately, transmitting any packets with PV below the threshold can only be done at the expanse of a packet with PV above, decreasing the total value delivered. As congestion increases the CTV also rises, while if the load is decreasing, the CTV is also reduced, enabling packets with smaller PV to be transmitted. Note that the CTV is generally not stationary, reflecting multiple factors from the available capacity and the amount of offered traffic to the observed PV distribution.

Fig. 1 shows different operator policies expressed as TVFs that describe conditional weighted resource sharing between three classes, Gold, Silver and Background while Voice has strict priority up to 64 Kbps. The intersection of the TVFs with horizontal lines representing different congestion levels (i.e. CTVs) defines the desired throughput of the classes at the given congestion level. Until Silver flows reach 10 Mbps, Gold flows get twice the throughput of Silver ones (I.). When the throughput of Silver flows is above 10 Mbps Gold flows get 4 times the throughput (III.). In between Silver flows get 10 Mbps and Gold flows get the rest (which will be between 20 Mbps and 40 Mbps) (II.). For this range of congestion levels the Silver policy pretends the behavior of a rate limiter at 10 Mbps. The figure also shows a background traffic class that has a small share of moderate PV to keep connectivity going, but receives larger bandwidth only if there is little or no congestion.

Second, Resource Nodes in the middle of the network treat packets without maintaining flow-states, solely relying on the carried PVs. Each such node aims at maximizing the total amount of value transmitted over the shared bottleneck. To this end, [4] proposes a simple scheduling algorithm that drops the packet with the smallest PV (even from the middle of the buffer) when the buffer length is too long, aiming at maximizing the total transmitted packet values.

IV. Resource Sharing across Multi-Layer Virtualization

Network virtualization happens, when a single physical network (PN) infrastructure is used to host several virtual networks (VNs), which share the capacity available in the physical network. Virtual networks themselves carry flows, which share the capacity allocated to the VN by the physical network operator. Note that virtual networks may extend beyond the infrastructure of the PN and use some of their own, dedicated physical infrastructure such as access or site infrastructures, continuing as a virtual network over a shared wide-area transport network.

The proposed HPPV can support such network virtualization scenarios, even with more than two layers of networks. A TVF defined by the PN operator is assigned to each virtual network to govern resource sharing between them. In addition, another set of TVFs defined by the given VN operator is also assigned to each flow within the VN, to govern how they share the resources currently available to their virtual slice. If virtual network A begins to receive less capacity, e.g. because another virtual network is ramping up its traffic and the policies give it higher priority, then flows in virtual network A will start seeing higher congestion even if their offered traffic does not change.

This scenario is depicted in Fig. 2 showing two VNs over a PN. The figure also depicts one example flow from each VN tunneled across the PN and sharing a common bottleneck. We assume that at the edge of each VN, packets are marked according to the TVF of their flow (#1 on the figure). When the packets of a VN arrive at the edge of the PN (which may be the same as the edge of the VN if the VN has no physical infrastructure of its own), the PN encapsulates the packet (as normal for virtualization to separate addressing spaces), saves the packet value in the inner header and re-marks the PV in the outer header (#2). Then the packet is transmitted through the PN which observes only the outer header (#3). At egress the packet is de-capsulated and continues to travel using its original PV (#4).

The difficult task of this scenario is the packet value re-marking. A good re-marking algorithm has to calculate the outer PV solely from the inner PV and the TVF of the VN (expressing the PN's policy for VN's traffic). Such algorithms must *1) keep value ordering*, that is, incoming packets with smaller PV shall be assigned a smaller outgoing PV; and *2) the value composition of outgoing packets shall match the VN's TVF* thus PVs after re-marking should follow the value distribution calculated from the VN's TVF (from throughput zero to the extent of offered traffic).

The first requirement ensures that whatever Congestion

Threshold Value emerges in the PN (in value space of the PN), it can be translated back to the value space of the VN and can serve as a CTV there (only dropping lower value packets). Thus the dropping pattern of the PN is also maximize the total transmitted packet value in the VN's value space and thus honoring the internal policies of the VN. The second requirement, on the other hand, ensures that resource sharing between VNs is according to their TVF. We note that it is also possible to encapsulate and re-mark the traffic of a PN to transport it over an additional underlay (e.g. when a part of the PN is also virtual). In this case, an additional set of TVFs can be assigned to the PN and its peers to govern resource sharing between them. As previously, any Resource Nodes need to consider only the outermost packet value to implement resource sharing targets for all layers.

The only practical limitation to the number of virtualization layers is the accuracy of the re-marking. Since the above two requirements can not fully be realized in a practical system with non-stationary incoming traffic, some imprecision is introduced by every re-marking step, resulting in divergence from the originally desired resource shares. However, at higher level of aggregation the traffic becomes more and more stable (e.g. short term fluctuations are not significant in a traffic mix of 100s of end-users), which on the other hand increases accuracy. In Section VI, we show that such divergence is not significant for two or three layers.

V. Implementation

As introduced in Section IV, packet re-marking at the border of virtualization layers is needed for the support of virtual networking. Assuming that incoming traffic is already marked with PV v and we need to apply TVF $V_{VN}(.)$ of the virtual network to calculate a new PV v_{new}. The new value is then added to the packet, e.g. by encapsulation, and used by the Resource Nodes for bandwidth sharing in the network of physical operator, as described in [4], [10], [11]. An optimal re-marking algorithm ensures all the requirements described in Section IV. In this section, we propose a practical re-marking algorithm that only requires a constant number of states to be stored.

The main steps of the re-marking algorithm are depicted on Fig. 3. One can observe that the key problem the re-marker needs to solve is to perform the translation between the packet value space of VN and PN. To this end, the proposed re-marking algorithm first determines the incoming *empirical TVF* $\tilde{V}(.)$ of the VN's packet value space as depicted on the left side of the figure and then applies the policy $V_{VN}(.)$, illustrated by the graph on the right. These two steps are detailed in the following paragraphs.

Construction of the *quantized empirical TVF*: Since the PV space can be large, to reduce the number of states to be stored the proposed algorithm first splits the PV space of the VN into K disjoint ranges: $(v_K, v_{K-1}]$, $(v_{K-1}, v_{K-2}]$, ..., $(v_1, v_0]$, where $v_i > v_{i+1}$. In our experiments, there are 655 ranges that follow a logarithmic sizing. The ranges may also be of equal sizes or other schemes are also possible (e.g. the PN may have prior knowledge on the expected eTVF). For each v_i ($1 \le i \le K$), the arrival rate (Th_i) of packets with PVs in range $(v_i, v_0]$ are measured, creating a quantized approximation of the incoming *eTVF* illustrated by the gray boxes in the left side of Fig. 3. We run K separate instances of a throughput estimator (e.g. based on exponentially weighted moving averaging (EWMA)) where the ith instance measures Th_i from the part of the incoming traffic with PV $v > v_i$.

Translation between packet value spaces: When a packet arrives with a value v that falls into range $(v_i, v_{i-1}]$, throughput value x is chosen uniformly at random[1] from the range $(Th_{i-1}, Th_i]$ and $v_{new} = V_{VN}(x)$ is marked on the packet as illustrated in Fig. 3.

One can observe that using quantization with random choice weakens the requirements described in Section IV for PVs falling to the same range. Basically, the ordering is only guaranteed between input values from different ranges. This can be remedied by more dense quantization of the packet value space and/or by knowledge about the incoming eTVF.Furthermore, the applied random method has the advantage of densely covering the outgoing value range, without the need of any assumption on the incoming values. In the remaining part of the paper, this approach is applied.

VI. Evaluation

We have implemented all the components of the HPPV framework in NS-3 [14], [15] simulator, including the packet marker and Resource Nodes as described in [4] as well as the re-marker node detailed in Section V. An overview of the topology used in our performance analysis is depicted in Fig. 4. It includes five source groups (S11, S12, S21, S22, S3) each containing one or more identical sources (representing, e.g. end-users) and four bottlenecks (BN-A1, BN-A2, BN-B, BN-C) controlled by four network operators (A1, A2, B, C). Each traffic source is connected to a separate packet marker placed at the boundary of the operator networks. Between networks of various operators, packet re-markers are placed implementing network virtualization. The bottlenecks rates are set to 1 Gbps (representing a high speed network virtualization), the buffer length is 10 ms in the Resource Nodes, and the TVFs in Fig. 1 are used. PV is logarithmically encoded into a 16 bit packet header field. Sources are either TCP or UDP as depicted, unless noted otherwise. Each TCP traffic flow consists of 5 parallel TCP connections, using the TCP CUBIC [16] implementation of Linux Kernel 4.1 through NS-3 Direct Code Execution [17]. UDP flows are greedy and non-congestion controlled, transmitting at a much higher rate than their desired share. There are no bottlenecks on the uplink and the round-trip propagation delay is 40 ms. In each scenario, the offered traffic load is modified at every 15 s to demonstrate both how the system provides the required resource sharing and how it behaves during transient periods. The ideal share of the flows is calculated from the TVFs by

[1] Instead of random selection other schemes for selecting throughput x are also possible (linear, logarithmic mapping, etc.).

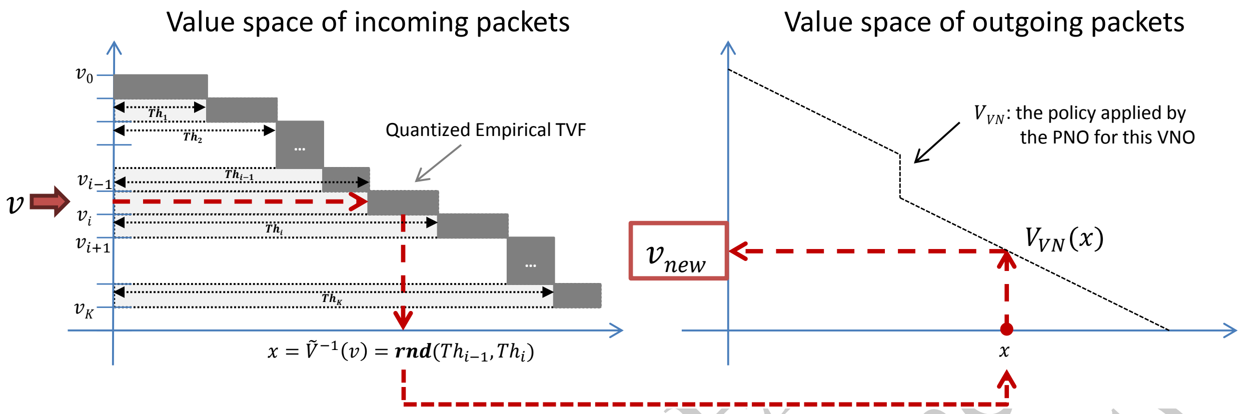
Value space of incoming packets
Value space of outgoing packets
v_0
Th_1
Th_2
Quantized Empirical TVF
v_{i-1}
Th_{i-1}
v
v_i
Th_i
v_{i+1}
Th_K
v_K
$x = \tilde{V}^{-1}(v) = \boldsymbol{rnd}(Th_{i-1}, Th_i)$
V_{VN}: the policy applied by the PNO for this VNO
v_{new}
$V_{VN}(x)$
x

Fig. 3. Packet value re-marking concept.

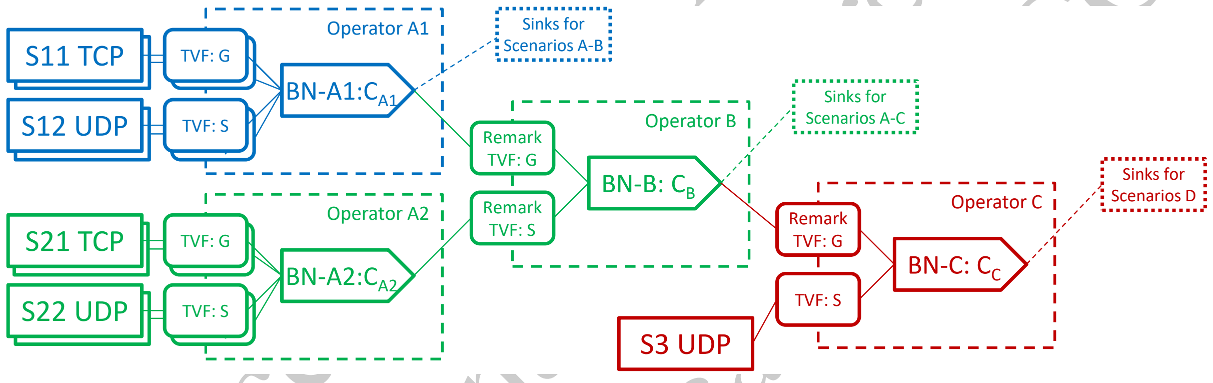

Fig. 4. Evaluation topology with four network domains: A1, A2, B and C

network calculus as described in [4], denoted by gray lines ("Desired") on the throughput figures.

In the first few experiments, we have two virtual networks (A1 and A2) transmitting over a physical network (B), see the blue and green area on Fig 4. Operators A1 and A2 are assigned a Gold and Silver policies by network B, respectively, and they share the 1 Gbps capacity at BN-B (as defined by the TVFs, they get 800 and 200 Mbps, respectively). Resource sharing among internal flows of Operator A1 and A2 remains to be governed by their internal policies (marked just after the sources as in previous simulations). Note again that this is achieved without Operator B (neither its bottlenecks nor re-markers) being aware of those internal policies.

A. TCP Flows Only

In the first scenario, Operators A1 and A2 have only Gold TCP flows (UDP sources S12 and S22 are inactive). We increase the number of flows (5 TCP conn. per flow) from 10 to 20 and then 40 in source groups S11 and S21, every 15 s. In Fig. 5 one can see that the flows get throughput close to their desired share after a 1-2 s transient. This transient is caused by the newly arrived TCP flows ramping up to their desired shares. One can also observe that as the number of flows increases the deviation form the ideal throughput curves decreases. This is due to TCP's congestion control behavior: at lower speeds the TCP congestion control can recover its sending rate faster after a single packet drop. Similarly, if we check the bandwidth share of the two operators, the total traffic of the two operators fits well to their ideal share (4:1 ratio) and as the flow number is increasing, the difference basically becomes negligible.

B. TCP and UDP Flows

In this scenario, aggressive (and unresponsive) Silver UDP sources (S12 and S22) are added to both Operator A1 and A2. All four sources start transmitting 5 flows and then increase it to 10 and 20 each, every 15 s. The access speed of S11 flows is limited to 100 Mbps, representing a high residential access rate. Fig. 6 shows the desired and the achieved throughput of these flows. It can be observed that in the first 15 s that Op.1 UDP flows (S12) get slightly higher throughput at the expense of Op.1 TCP flows (S11). As the number of flows increases this deviation decreases. This is due to the effect of the 100 Mbps bottleneck of S11 flows and the previosly noted congestion control behavior. The expected (4:1) share among operators is kept as depicted in the bottom figure.

C. Varying Bottleneck Rate

In this scenario, the number of flows is fixed to 10 for each of the four source groups and the capacity of the internal bottlenecks (BN-A1 and BN-A2) are varied. At the beginning they are large enough such that BN-B is the only bottleneck.

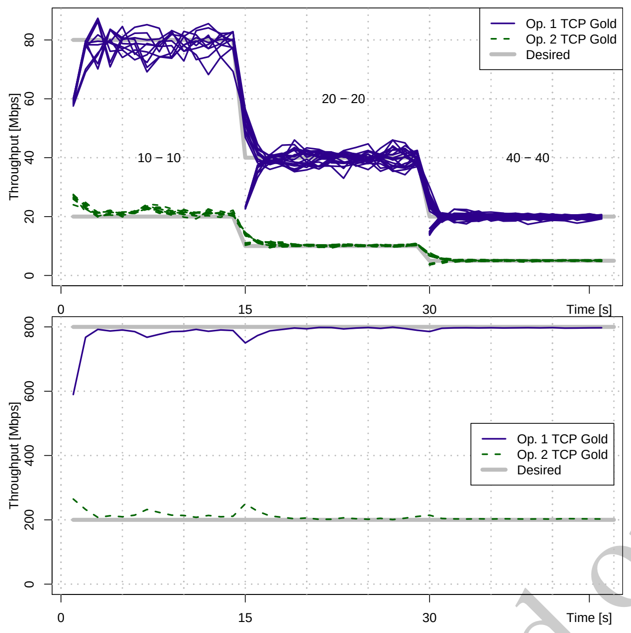

Fig. 5. Two virtual networks with TCP traffic only

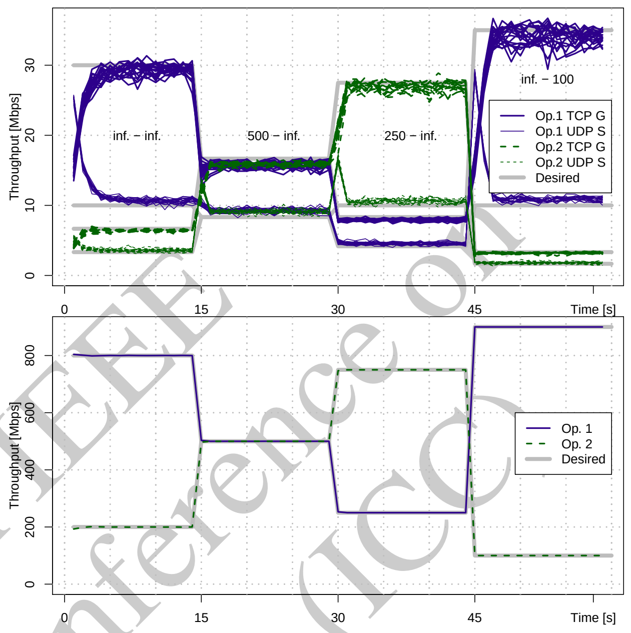

Fig. 7. Varying bottleneck rates in two virtual networks

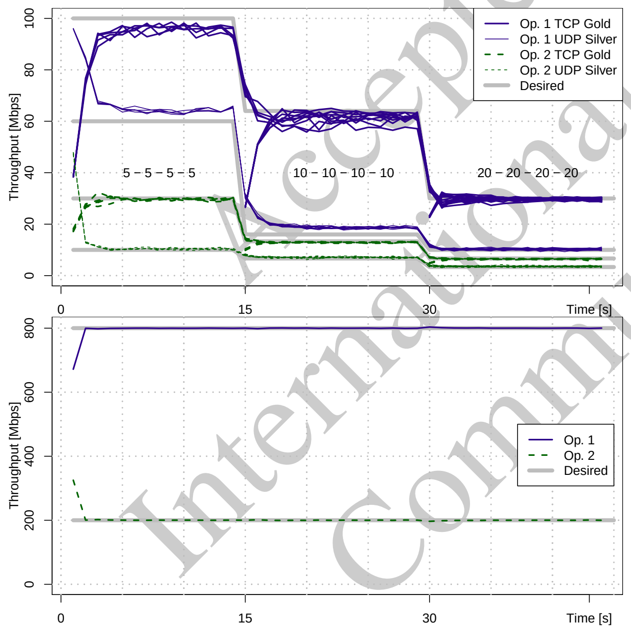

Fig. 6. Two virtual networks with Gold TCP & Silver UDP traffic

At 15 s we introduce a 500 Mbps bottleneck for BN-A1. Thus between 15-30 s sources S21 and S22 together utilize half of the 1 Gbps BN-B bottleneck, since sources S11 and S12 are limited by BN-A1 (see desired vs. achieved shares on the bottom chart of Fig. 7). During this interval resource sharing between S11 and S12 flows is governed by the scheduler at BN-A1. At 30 s, we further decrease the BN-A1 to 250 Mbps thus 750 Mbps remains for Operator A2. At 45 s, we lift BN-A1 and introduce a 100 Mbps bottleneck in BN-A2. The top chart of Fig. 7 shows the ideal vs. achieved throughput of individual flows. This figure demonstrates how the system can reach the ideal resource share under highly varying conditions. Note again that no feedback is applied (apart from TCP Congestion Control) and that the none of the resource sharing policies or markers are changed during the simulation, only the bottleneck capacities. Similarly to the previous sections, it takes a few seconds for TCPs to ramp up to the new ideal share after a transient.

D. Three Layers of Virtualization

In this final scenario, we introduce another layer of virtualization below Operator B. Operator C carries both the traffic of Operator B (after re-marking as Gold) and source S3, marked as Silver. BN-C is set to 1 Gbps, thus Operator B and S3 shall receive around 800 and 200 Mbps, respectively (4:1 ratio). Sources S12 and S22 are turned off, so Operator A1 and A2 only send TCP traffic. Again, we start the simulation by having S11 and S21 emitting 10-10 flows, then successively increasing them to 20-20 and 40-40, every 15 s. S3 has a single UDP flow during the entire simulation.

In Fig. 8 the throughput achieved by the flows is shown. One can observe that S3 kept its roughly 200 Mbps share during the entire experiment. The other two operators share the remaining 800 Mbps capacity in 1:4 as defined by their TVFs. The operator shares depicted follow the desired throughput curves with slightly higher deviation than in the previous examples, especially in the initial transient caused by the ramping up behavior of TCP slow start. One can also observe that the larger the number of flows are, the better fairness among the flows in the given classes can be experienced. Similarly to the previous cases, the large deviations seen in the first 15 s (with only 10-10 TCP flows) are also caused by the too conservative rate reduction and increase of TCP congestion control. Since the UDP source is aggressive and unresponsive, as soon some

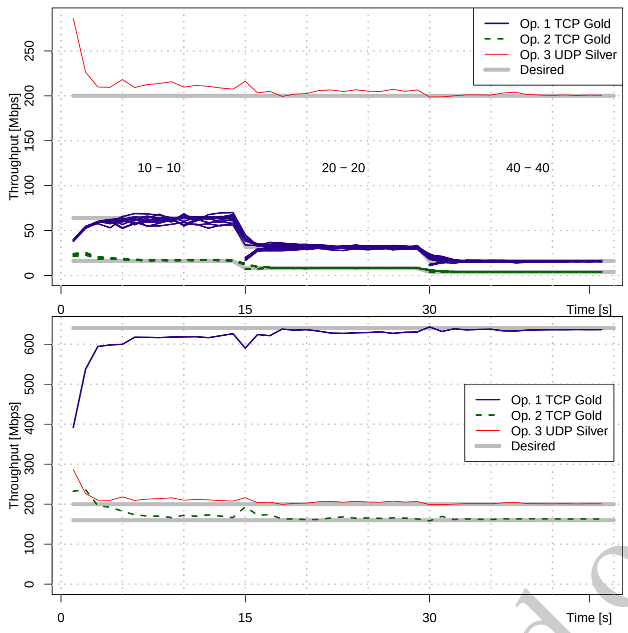

Fig. 8. Three layers of virtualization

capacity is available at the bottleneck, it is obtained by the UDP source, resulting in a slightly higher average throughput than its ideal. This phenomenon is more visible when the number of TCP sources is limited (e.g. in the first 15 s).

VII. Conclusion

One of the key benefits of HPPV introduced in this paper is that flow and policy knowledge is only required at the border of various network domains including edge nodes and transition points between different virtualization layers where packet marking or re-marking is performed. Within the network, the marked traffic is then handled by Resource Nodes (e.g. routers) that aim to maximize the total amount of transmitted Packet Values and with it implement the resource sharing rules that govern the marking. To the best of our knowledge, HPPV is the first core stateless QoS solution that also supports resource sharing over multiple layers of virtualization without the need for re-tuning any element if traffic or bottleneck capacity changes. To this end, a re-marking algorithm has been introduced that describes how packet values needs to be recalculated at the edge of a shared infrastructure to ensure that the policies of both virtual and physical network operators are simultaneously taken into account. In Section V, we have provided a proof-of-concept algorithm and the performance of the proposed solution has been analyzed in a range of simulation scenarios. With the multi-layer virtualization support, HPPV provides a stateless, loop-free and scalable alternative to traditional resource sharing solutions for future networks like 5G access networks or other closed networking environments where QoS insurance is needed among a large number of end-users, flows or network slices.

Acknowledgment

The research of G. Gombos has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.2-16-2017-00013, Thematic Fundamental Research Collaborations Grounding Innovation in Informatics and Infocommunications). S. Laki thanks the support of the János Bolyai Research Scholarship of the Hungarian Academy of Sciences. The authors also thank the support of Ericsson Hungary Ltd.

References

[1] X. Li, M. Samaka, H. A. Chan, D. Bhamare, L. Gupta, C. Guo, and R. Jain, "Network slicing for 5g: challenges and opportunities," *IEEE Internet Computing*, vol. 21, no. 5, pp. 20–27, 2017.
[2] N. Alliance, "Description of network slicing concept," Tech. Rep., January 2016.
[3] M. Menth and N. Zeitler, "Fair resource sharing for stateless-core packet-switched networks with prioritization," *IEEE Access*, vol. 6, pp. 42 702–42 720, 2018.
[4] S. Nádas, Z. R. Turányi, and S. Rácz, "Per packet value: A practical concept for network resource sharing," in *IEEE Globecom 2016*, 2016.
[5] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "pfabric: Minimal near-optimal datacenter transport," in *Proceedings of ACM Sigcomm*, ser. SIGCOMM '13. New York, NY, USA: ACM, 2013, pp. 435–446. [Online]. Available: http://doi.acm.org/10.1145/2486001.2486031
[6] M. Carlson, W. Weiss, S. Blake, Z. Wang, D. Black, and E. Davies, "An architecture for differentiated services," RFC 2475, December, Tech. Rep., 1998.
[7] 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. [Online]. Available: http://dx.doi.org/10.1109/TNET.2002.808414
[8] M. Mathis and B. Briscoe, "Congestion exposure (conex) concepts, abstract mechanism, and requirements," RFC 7713, Tech. Rep., 2015.
[9] Z. Cao, E. Zegura, and Z. Wang, "Rainbow fair queueing: theory and applications," *Computer Networks*, vol. 47, no. 3, pp. 367 – 392, 2005. [Online]. Available: //www.sciencedirect.com/science/article/pii/S1389128604002361
[10] S. Laki, G. Gombos, S. Nádas, and Z. Turányi, "Take your own share of the pie," in *Proceedings of the Applied Networking Research Workshop*. ACM, 2017, pp. 27–32.
[11] S. Nádas, G. Gombos, P. Hudoba, and S. Laki, "Towards a congestion control-independent core-stateless aqm," in *Proceedings of the Applied Networking Research Workshop*. ACM, 2018, pp. 84–90.
[12] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.
[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 *Proceedings of ACM Sigcomm*, ser. SIGCOMM '15. New York, NY, USA: ACM, 2015, pp. 1–14. [Online]. Available: http://doi.acm.org/10.1145/2785956.2787478
[14] "The ns3 simulator," Available: http://www.nsnam.org/, 2016.
[15] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, "Network simulations with the ns-3 simulator," in *In Sigcomm (Demo*, vol. 14, 2008.
[16] S. Ha, I. Rhee, and L. Xu, "Cubic: A new tcp-friendly high-speed tcp variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, Jul. 2008. [Online]. Available: http://doi.acm.org/10.1145/1400097.1400105
[17] H. Tazaki, F. Uarbani, E. Mancini, M. Lacage, D. Camara, T. Turletti, and W. Dabbous, "Direct code execution: Revisiting library os architecture for reproducible network experiments," in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. ACM, 2013, pp. 217–228.