

Per Packet Value: A Practical Concept for Network Resource Sharing

Szilveszter Nádas, Zoltán Richárd Turányi and Sándor Rácz Ericsson Research, Traffic Analysis and Network Performance Laboratory
H-1097, Könyves Kálmán krt. 11. B, Budapest, Hungary
szilveszter.nadas@ericsson.com

***Abstract*—Governing how network resources are shared among various traffic flows is not a completely solved problem. The ideal system can implement a wide variety of detailed and flexible policies; enforces those policies under all possible offered traffic combinations and scales well with the number of flows. In this paper we introduce a framework that approximates the ideal system well. It consists of flow-aware *Packet Value* marking at the edges and flow-unaware operation inside the network. We show that inside the network relatively simple scheduling and AQM algorithms operating solely on Packet Value are suitable to implement the policies set at the network edge. We also discuss, how the framework can be used to support resource sharing over radio networks (where the cost of transmission varies) and how it enables load balancing, resource balancing and optimal network-wide load distribution.**

I. Introduction

Quality of Service is one of the most researched areas in networking literature. As usual for such a mature field, several solutions exist for most conceivable networking scenarios. However, the issues of QoS and resource management are still not fully solved problems. For example, "QoS framework" is considered a key issue in 5G standardization [1]. Usually resource sharing policies are specified at the network edge and executed inside the network. We aim for a system where the 1) set of possible policies are very wide and 2) the execution is lightweight. First, policies shall not be limited to priority or weighted fair queuing, but an arbitrary combination of these, for example: flow X shall have absolute priority over flow Y, but only up to 1Mbit/sec above which resources shall be shared evenly. Existing resource sharing schemes do not provide a language rich enough to describe such policies. Second, execution of the resource sharing shall not require flow awareness in the nodes inside the network, as this would eventually limit the number of flows the network can admit. Instead, network devices shall operate with a small, fixed cost per packet irrespective of the number of flows or the type of policies specified.

In this paper we discuss a QoS framework based on packet marking, where each packet carries its own marginal utility. Our aim was to address as many aspects of QoS and resource sharing as possible using a single, coherent and scalable concept. Our focus is on generic communication networks of today, where the network operator wants to have control over who gets how much resources. We do not target networks which require no resource management due to heavy over-provisioning.

As with most packet marking schemes, packets are marked at the network edge – at this location operators can express their resource sharing and QoS policies. Here flow-level (or even application or user-specific) information may be used to determine the policies. In contrast, the inside of the network operates solely based on the packet markings with no flow awareness. Throughout the paper we will use the term *Resource Node* to denote internal nodes of the network which own and manage actual transmission resources; and we use the term *flow* to refer to a piece of traffic to which we apply resource sharing policies.

Policies are expressed by assigning each packet a *Packet Value*, which represents the gain of the operator when this packet is delivered (marginal utility in other words). The network will strive to maximize the total aggregate Packet Value delivered using the available resources. In addition to the Packet Value, packets are also assigned a *Delay Class*, which indicate the delay requirements of the flow.

We argue that the information represented by these two values in each packet enables a rich enough interface to carry a very wide range of resource management and QoS policies from the edge of the network to the Resource Nodes. We also argue that these two values represent a universal coordinate space to precisely quantify important aspects of resource management. They can be used to compare the importance of any two packets or describe the level of congestion in a way that is meaningful if the network carries flows of different importance. In addition, since the Packet Value is scalar, arithmetic is possible on it enabling to trade two packets for a single one, if the sum of the Packet Values of the two packets is still smaller than that of the single one. Furthermore it becomes easy to handle situations where the resource cost of transmitting bits is different for different packets, such as in the case of radio transmission.

The Per Packet Value (PPV) framework complements and works well together with end-to-end congestion control. Today congestion control is used to ensure 1) low, controlled loss; and 2) proper resource sharing (typically fairness). The latter is enforced by PPV, which simplifies the task of Congestion Control to the former. Thus even previously incompatible congestion control algorithms can safely coexist in networks implementing PPV.

In Section II we compare our proposal to selected resource management frameworks. In Section III we describe the proposed concept in detail. In Section IV we propose practical

marker and Resource Node implementations. In Section V we demonstrate that the algorithms work well with TCP end-to-end congestion control, enforce the desired resource sharing policies and protect well behaving flows from misbehaving ones. Finally, in Section VI we outline a number of more advanced uses of the concept besides basic resource sharing.

II. Related Work

Several comprehensive resource management frameworks have been proposed in the past. In the ATM [2] and Integrated Services [3] model, signaling messages are used to transmit the policies regarding individual flows to Resource Nodes, where each flow is scheduled individually. This resulted in poor scalability of the schedulers. In a class of more lightweight signaled resource sharing frameworks packet scheduling is performed based on aggregate traffic classes and signaling is used just to reserve resources and to establish the forwarding path. Such per-flow signaling was also found to be of limited scalability and has lead to the development of QoS architectures without per-flow signaling.

The bearer model of 3GPP [5] is another form of signaled QoS model, geared to support radio operations. Traffic of a terminal with the same QoS requirements is grouped into the same *bearer*. The QoS provided by a bearer is described by its QoS Class Identifier (QCI), that is used as a reference to a specific packet forwarding behavior. Bearers are the unit of radio scheduling and may be subject to admission control. It is difficult to differentiate flows or packets within the same bearer or to change resource sharing policies among bearers when the traffic changes.

Similarly to our framework DiffServ [4] uses packet marking at the edge to convey the intended resource sharing policy to the Resource Nodes. In contrast to our scheme, DiffServ markings do not represent Packet Values, but identify a set of pre-defined policies, Per-Hop Behaviors (PHBs). Contrary to our concept, to introduce a new policy, Resource Nodes have to be changed to support it. Another difference is that in our concept packets of a flow are typically marked with a wide range of packet markings, whereas in DiffServ it is one or a few values only.

The Network Utility Maximization (NUM) framework [6] provided much inspiration for our work. This framework introduces the notion of utility, but instead of assigning a single Packet Value to each packet it assigns complete throughput-utility functions to whole flows. The maximized utility might be implemented in different ways, e.g., by informing Resource Nodes of the utility functions or by end-to-end congestion control. In our concept markings on the packets of flow together describe the utility function and the encoding is such that Resource Nodes do not need flow awareness to maximize utility.

The report by Congestion Control Research Group of the IRTF [8] summarizes the open issues in congestion control. Although it is from a few years back, its authors still refer to it as a relevant summary of open issues in this field. The solution in this paper directly addresses “Challenge 1: Network Support”, which is about the balance of functions between the endpoints and network equipments.

The concept in pFabric [12] shows a number of similarities with our concept, such as assigning a wide range of priority levels to packets and scheduling on highest priority. On the other hand pFabric is designed to minimize flow completion times (or other policies) in Data Centre environments with co-operating endpoints as opposed to controlling resource sharing in general networking environments, which is our goal.

Rainbow Fair Queueing (RFQ) is a packet coloring and buffer management scheme that emulates the fair sharing of WFQ but avoids packet classification and per-flow state operations at the Resource Node [7]. Our concept extends RFQ in a number of ways. First, instead of marking colors on the packets we mark scalar Packet Values, which can be used for arithmetic operations. Second, we add a Delay Class. Third, we introduce the throughput-Packet Value functions as a way to express rich resource sharing policies. Finally, we list a number of advanced uses not discussed by [7].

III. Concept

The PPV concept described in this paper uses two simple values assigned at the network edge to convey a rich set of resource sharing policies to the flow unaware Resource Nodes inside the network.

A. Packet Value and Delay Class

Packet Value represents the gain of the operator when the packet is delivered. The aim of the network, and every Resource Node within is to maximize the total Packet Value of the packets transmitted. Packet Values are expressed as value per bit, to make direct comparison among packets of different sizes possible. The total value of a longer packet is higher than a shorter one with the same Packet Value marking, but normally it also takes proportionally more resources to transmit. We talk about value to the network operator, but naturally this encompasses the value to the user of the network, as well. In addition, it also allows the network operator to incorporate its own preferences, by giving higher value to more important users or traffic, such as emergency services.

Packet Values express the relative importance of one packet to another. Resource Nodes will transmit packets with high PV at the expense of packets with low PV, which are delayed or dropped. Resource Nodes may also take care to preserve packet ordering within a flow, when needed.

In an ideal system at any given moment at every Resource Node there is a *Congestion Threshold Value*, which separates the packets with Packet Values that get transmitted from the ones that get dropped. The Congestion Threshold Value is a momentary emergent value resulting from the combination of available capacity, amount of offered traffic and the Packet Value composition of the offered traffic. As congestion increases, this threshold value increases as only more and more important packets make it through the available capacity. As congestion is easing, the threshold value decreases, as packets with lower PVs can get transmitted. This threshold do not have to be calculated in the resource node to maximize the total transmitted PV, rather it is the result of that maximization.

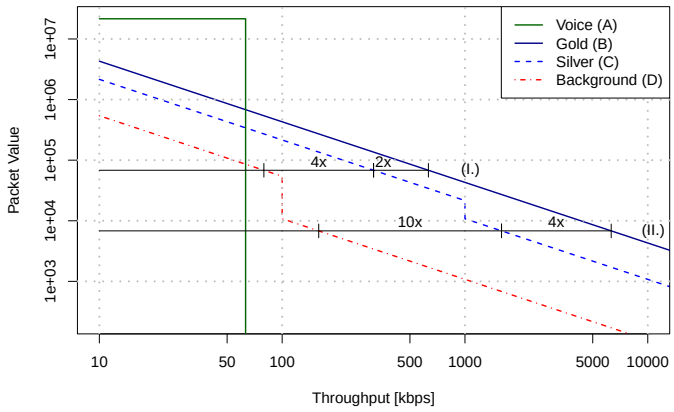

Fig. 1. Example throughput-Packet Value functions. Due to the logarithmic x axis, parallel curves represent weighted resource sharing.

The Packet Value is a scalar, with preferably many bits to represent it. The larger the value space the more precision resource sharing can be expressed and enforced with.

The other value attached to packets at the edges is the *Delay Class*. Unlike the PV that is scalar, we allow only a few distinct Delay Classes. Resource nodes shall forward packets according to the operator-defined maximum delay for the Delay Class of the packet. Since packets in different Delay Classes can naturally get re-ordered, all packets of a flow (within which in-order delivery is preferred) shall be marked with the same Delay Class. Both Packet Value and Delay Class can be carried in newly created or existing packet header fields, such as in DSCP or in an MPLS label.

Packet Values and Delay Classes are orthogonal. Marking a packet low delay and low PV makes sense for, e.g., for enhancement layer of conversational video content. Such packets will get delivered only at light or no congestion - but then at low delay.

Note that the above two parameters augmented with some routing information (label or address) and flow hash (to signal packet ordering requirements, e.g., an entropy label) are sufficient information for L2 and L3 protocols to implement most wide-area networking requirements.

B. Flow Policies

An easy way to visualize the policy assigned to a flow is via the *throughput-Packet Value* function $V(b)$, which is the derivative of the utility function $(U_i(x_i))$ in [6]. This function describes the quota of packets marked with certain range of Packet Values and it is the direct input of packet marking algorithms. Specifically for every throughput value of b, b worth of packets shall receive Packet Value $V(b)$ or higher. This function also tells us how much throughput the flow will get in a certain congestion situation described by the Congestion Threshold Value. This throughput will be the value b, where $V(b)$ equals the Congestion Threshold value. Figure 1 shows an example of four different function examples demonstrating both priority and weighted fair resource sharing.

- Function A: Voice. Up to 64 kbps (an example rate for a voice call) it enjoys a PV higher than any other traffic, but nothing above that. This will result absolute priority as long as its rate is limited to 64 kbps.
- Function B, C: Gold/Silver Internet access. These demonstrate proportional resource sharing. Until 1 Mbps is reached Gold gets twice the throughput of Silver. Above that it gets 4 times the throughput. (Note the log scale.)
- Function D: Background download. This traffic receives 1/4th of Silver up tp 100 kbps, above that it receives 1/10th of Silver.

The intersection of the throughput-Packet Value function and a horizontal line at the Congestion Threshold Value shows the desired throughput in that congestion situation. For example on Fig. 1 the horizontal line (I.), which corresponds to the Congestion Threshold Value of 68000, intersects functions A, B, C and D at throughput of 64, 80, 320 and 640 kbps respectively.

The power of the concept lies in the fact that through the markings we encode instructions how the network shall treat this flow under any congestion level. Therefore no additional information is needed inside the network nor feedback to network edges to execute the desired resource sharing policy even if congestion conditions vary. This results in open loop control of resource sharing.

It is up to the edge to decide which specific packets to mark with high and low PV, within the quota allowed by $V(b)$. For example, in case of a media stream the quota of high and low PVs can be distributed among packets belonging to enhancement layers and the base layer such that the base layer gets the higher PV markings. This results in that (under ideal node behavior) packets from enhancement layers get transmitted only if all the base layer packets of all flows are transmitted, as well. Thus, the marking policy do not have to consider throughput (alone), it can respect other information.

The policy applied at the edge can be easily changed over time. If a user has exhausted its quota, started a new, premium application or the application has switched to another mode of operation the marking can change to reflect the new policy. Note that no signaling is needed from the edges to the inside of the network to enact the change - the (updated) PVs are sufficient.

Note that applying the right policies provides significant protection and isolation among flows. No matter how much traffic arrives on a flow, only a controlled amount can get high PV. This prevents misbehaving flows to take resources from well-behaving ones. This will be demonstrated in Section V.

C. Flow Calculus

Assuming a flow model it is possible to determine the amount of resources flows get at a bottleneck from their throughput-PV function. We introduce a function composition operator $\oplus$ as $f \oplus g := (f^{-1} + g^{-1})^{-1}$. This operator can be used to compute the *Combined Throughput-PV Function* of two or more of flows. Assuming we have a set of flows with (strictly monotonically decreasing) $V_i(b)$ functions going through a bottleneck of capacity C, the Congestion Threshold Value will be $V_{CongTh} = \oplus_i V_i(C)$. Substituting this back to the individual throughput-PV functions, we can get the desired throughput of individual flows as $b_i = V_i^{-1}(V_{CongTh})$.

D. Variable-cost Transmission

A conspicuous property of cellular radio transmission is that depending on the location of individual mobile terminals, the amount of resources to transmit the same amount of data may vary significantly. For terminals at bad locations a less aggressive modulation scheme, more channel coding overhead and transmission power are needed to transmit reliably. As a result some terminals can send a lot more data than others using the same amount of radio resources. Consequently, in order to maximize the total PV transmitted over such links, the PVs of the packets cannot be directly compared, but must be normalized by the cost of their transmission (r). We define the *Effective Packet Value* of a packet as its PV divided by its transmission cost (V/r) and *Effective Congestion Threshold Value* as the Effective Packet Value threshold separating transmitted and dropped packets in an ideal case.

Traditionally fairness and radio resource efficiency are contradicting requirements when the cost of transmission is different between different mobile terminals. Scheduling policies vary from bitrate fair through Round Robin and Proportional Fair to maxCQI[1]. The total throughput is the smallest in case of bitrate fair scheduling, because a high amount of resources has to be allocated to the mobile terminal with the worst channel resulting small channel efficiency. At the other extreme the maxCQI scheduler may lead to starvation of mobile terminals with poor radio conditions, thus it is rarely used.

Fig. 2 depicts a $V(b)$ function designed to take into account both throughput targets and resource efficiency. The Effective Packet Value function is shown for ideal channel[2] (r=1) and for a channel where the cost of transmission is twice that of the ideal (r=2). The curve has different slopes in different throughput regions, and the steepness determines different relative radio resource sharing and throughput ratio in the different regions. We indicated the throughput ratio of the two channels at three different Effective Congestion Threshold Values. Up to 100 kbps a twofold decrease in transmission cost results in 1.4 times increase in throughput, which means that a terminal with worse channel gets more radio resources allocated, but its throughput is still smaller than that of the terminal with the ideal channel. The rationale behind this policy is that in this low-throughput regime throughput fairness is preferred as everyone should be receiving a tolerable minimum service, regardless of how poor radio conditions are. Between 100 kbps and 1 Mbps a twofold cost decrease results in twice the throughput, which stands for equal radio resource allocation. Above 1 Mbps the same twofold cost decrease results in 4 times more throughput, as the mobile terminal with the better channel gets twice the radio resources in this range. In this last regime, all terminals already receive good enough throughput and efficient resource utilization is more important, thus users with momentarily good radio conditions are favored disproportionally.

[1]Higher Channel Quality Indicator (CQI) represent better radio channel. The maxCQI scheduling policy always schedules the mobile terminal with the best channel only in order to maximize the total throughput without regard for fairness.

[2]We assume that the (normalized) transmission cost of the ideal channel is 1.

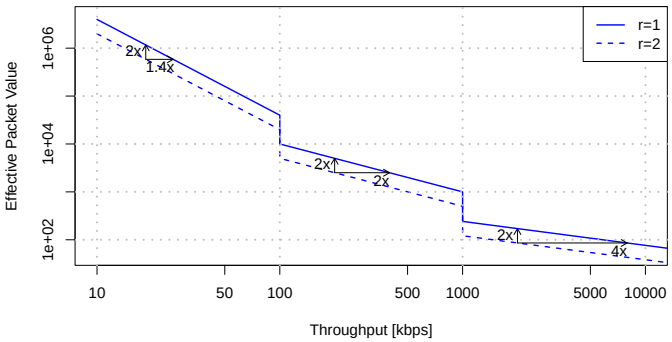

Fig. 2. Effective Packet Value example.

In a mobile network r is not constant and might be changing fast (making closed loop control from a potentially distant network edge impractical). The task of the Resource Node remains to maximize the realized PV. The Resource Node has the most up to date information about r and can also utilize prediction. The above curve design still represents the throughput dependent cost aware resource allocation in this case.

IV. Implementation

In this section we introduce example marking, scheduling and Active Queue Management (AQM) algorithms implementing the proposed PPV framework.

A. Packet Marker

The task of the packet marker is to translate the throughput-PV function to actual PVs marked on each and every packet.

A simple implementation of this function could apply a series of token buckets (one for each region of the throughput-PV function), with associated packet values (the PV around the region of the token bucket). For each incoming packet we select the highest PV token bucket with available tokens and assign its value to the packet. This ensures that the amount of traffic marked with a specific value is as proscribed by the throughput-PV function.

More in detail, we quantize the throughput region logarithmically, e.g., $b_i = 10^{1+1/30 \cdot i}$ kbps, $i = 0 \ldots 149$ covering the throughput range $[10$ kbps, 1 Gbps$]$. For each such throughput (b_i) we assign Packet Value $V_i = V(b_i)$. This quantization is illustrated and throughput (b_i) - Packet Value (V_i) pairs of the quantized throughput-Packet Value function for Background traffic are shown for $i = 28, 29, 30, 31$ on Fig. 3.

The maximum size of each bucket is $l_i^{max} = b_i \cdot d$, where d is the averaging delay. These buckets are initialized as full ($l_i = l_i^{max}$) and are continuously filled with speed b_i. Whenever a packet with size s arrives it gets the Packet Value V_k, where k is the index of the first bucket having at least s amount of tokens ($k = \text{argmin}_j(l_j \geq s)$), then all buckets with index larger than or equal to k are decreased with s.

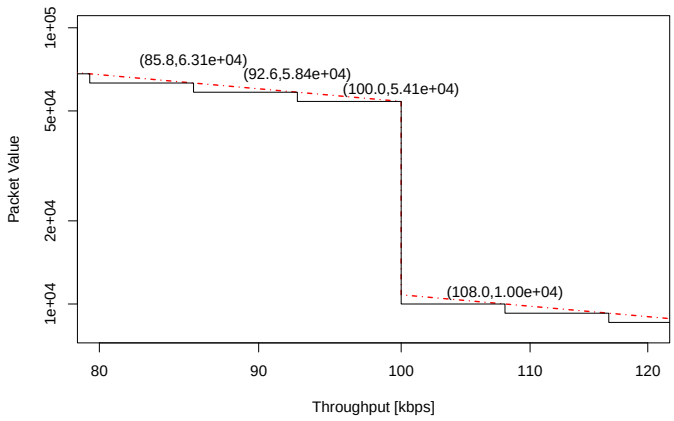

Fig. 3. Quantized throughput - Packet Value function of Background traffic, magnified part.

This token bucket structure is hierarchical, each bin represents the throughput interval $[0, b_i]$, as opposed to non-hierarchical structure where bucket i represents interval $[b_{i-1}, b_i]$. The reason for this is that it results larger bucket sizes ($b_i \cdot d$ instead of $(b_i - b_{i-1}) \cdot d$) and thus a full sized IP packet fits into the maximum bucket size sooner. This becomes especially important if we increase the granularity of quantization to better approximate the throughput-Packet Value function. The consequence of this is that all buckets with index larger than or equal to k has to be decreased with s and that this might result in negative bucket size temporarily.

B. Resource Node

The task of the Scheduler and AQM algorithms at the Resource Node is to maximize the total PV of the transmitted packets. There might be other constraints on the algorithm, e.g., in-order delivery, packet delay requirement and node implementation constraints (such as packet dropping is only possible upon arrival). Another factor to take into account is whether the cost of all bits is the same (see Section III-D).

The simplest algorithm is to always serve packets with highest PV first. In most practical cases though it is not sufficient, because it does not provide in-order delivery and delays can be arbitrarily high.

A simple, but already practically working, algorithm is as follows. We keep a buffer of constant length (e.g., determined based on a common delay requirement) and whenever an incoming packet does not fit into this buffer we *drop* the packet(s) with the *smallest Packet Values* (potentially from the middle of the queue) until it fits. This algorithm can be adapted for radio transmission by dropping the packet with the smallest Effective Packet Value instead.

In the case when packet dropping is only possible upon packet arrival it is possible to define a packet drop probability function with its inputs being the queue length and the PV of the incoming packet. More generally, it is possible to adapt, e.g., the PIE [10] algorithm to also take into account the PV of the incoming packets when determining the packet drop probability.

Another way to reuse existing AQM algorithms is to quantize Packet Values to a few levels (e.g., DSCPs) and configure Weighted RED [13] profiles for each level. This can approximate the behavior of the algorithms above on existing network equipment.

C. Delay Aware Resource Node

A simple algorithm concept for delay aware Resource Nodes is described in this section. In the Resource Node create a queue per Delay Class. Schedule these queues with strict priority in increasing order of their delay requirements (D_i). For each such queue periodically estimate the delay of the last packet (d_i) based on the time-stamp of the first packet in that queue ($t_{i,1}$); the total amount of packets waiting in that queue and the queues above ($s_{j,k}$ is the size of the k^{th} packet in the j^{th} queue); and the available bottleneck capacity (C):

$$d_i = t_{i,1} + \Big(\sum_{j=0}^{i} \sum_{k=0}^{n_j} s_{j,k} \Big) / C. \quad (1)$$

While $d_i > D_i$ drop the packet with the smallest PV (V_l) in queues $0 \dots i$. In addition to that for a period D_i drop all packets with smaller PV ($V \leq V_l$) arriving to queues above queue i. This ensures that low PV low delay packets do not starve high PV packets with more relaxed delay requirement.

V. Evaluation

In this section we present measurement results of the concept. We implemented the concept on a Ubuntu 14.04.3 LTS PC using Linux kernel 3.13.0. Both the packet marker and the Resource Node has been implemented in C using [9]. The packet marker is implemented according to IV-A, the Resource Node is implemented according to the *"drop smallest Packet Values"* algorithm in IV-B. The packet marker sets the ToS field of the IP packet, the Packet Values are encoded using the formula $30 \cdot \log_{10}(V(b))$. This is to encode a wide range of Packet Values efficiently. We use a single Delay Class.

We set up a small emulated network for the measurements with a single bottleneck. Available capacity at the bottleneck was 10 Mbps with a 40 ms long buffer and the end-to-end propagation delay is also 40 ms. In all cases the same buffer is shared among all the flows. We implemented a simple TCP client to generate greedy TCP flows and use iPerf [11] to generate and measure UDP flows. We configured the packet marker according to Fig. 1 and used the same line styles to plot the throughput of the flows from different classes (measured in 1 s intervals). We also plot the desired share of flows in gray, calculated according to III-C. Apart from the PVs marked on every packet there was no additional communication between the packet markers and the Resource Node.

Figures 4 and 5 depict measurements with Gold and Silver TCP flows. The number of TCP connections per flow is 1 on Fig. 4; and 5 on Fig. 5. The number of Gold and Silver flows starts at 1-1 then increases to 2-2, 4-4 and 8-8 every 30 s. It can be seen that the desired resource sharing is honored: until the Silver flows can have 1 Mbps throughput the Gold flows achieve 4 times more throughput than Silver, otherwise twice the throughput. There are higher oscillations in case of 1 TCP connection per flow, because a single packet loss in this case

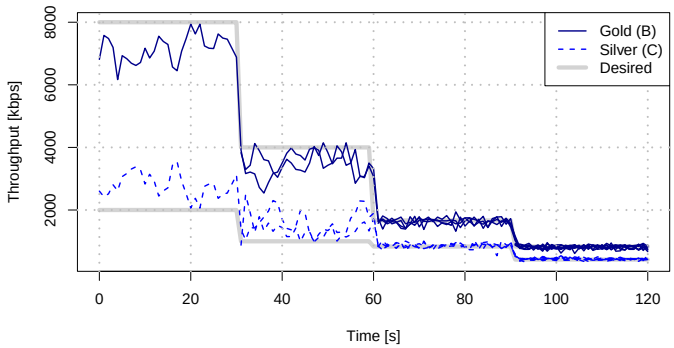

Fig. 4. Gold and Silver TCP flows. 1 TCP connection per flow. 1-1, 2-2, 4-4, 8-8 flows.

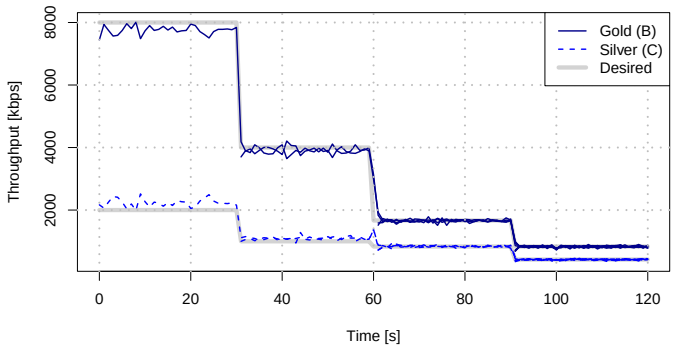

Fig. 5. Gold and Silver TCP flows. 5 TCP connections per flow. 1-1, 2-2, 4-4, 8-8 flows.

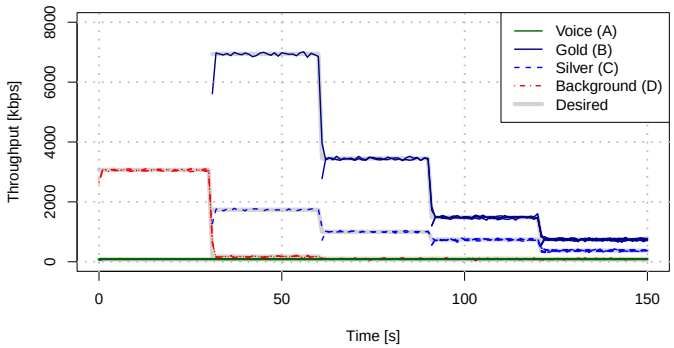

Fig. 6. 10 Voice, 3 Background, Gold and Silver flows. All flows use UDP. Gold and Silver class has 1-1, 2-2, 4-4, 8-8 flows.

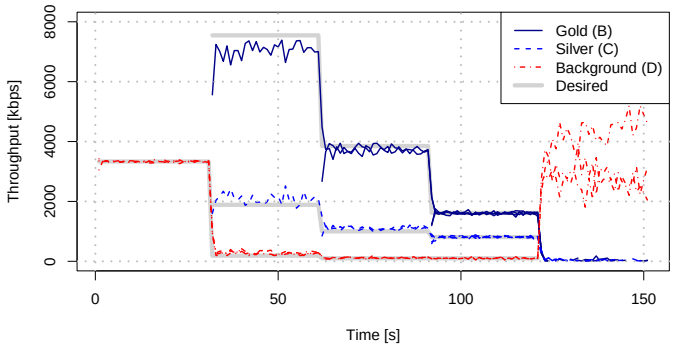

Fig. 7. Gold and Silver TCP flows. 3 Background UDP flows. 5 TCP connections per flow. Gold and Silver class has 1-1, 2-2, 4-4 flows.

results in a more significant change in the total TCP congestion window, especially for the single Gold TCP connection in the first 30 s.

Fig. 6 depicts measurements of all 4 classes using UDP. There are 10 voice flows of 64 kbps and 3 background flows of 5 Mbps during the whole measurement. Gold and Silver flows are of speed 10 Mbps, they are started at 30 s and their number is increased similarly to the TCP examples. It can be seen from the iPerf results that voice flows experienced no packet loss during the whole measurement, even though the packet loss for background flows sometimes reached 99% (as desired) and all flows shared the same buffer. It is also visible how the 1 to 4 and 1 to 2 weighted resource sharing between Gold and Silver flows is met. Background and Voice flows utilize all of the capacity when Gold and Silver flows are not present (0 to 30 s). Background flows receive much less as soon as there is more important traffic (as defined by Fig 1).

Fig. 7 depicts the effect of aggressive Background UDP traffic on TCP flows. There are 3 background UDP flows of 10 Mbps during the whole measurement. Those share the available capacity equally in the beginning resulting in 3.33 Mbps throughput each. At 30 s Gold and Silver TCP flows are added and it can be seen how these flows get their desired share even though the aggressive flows remain. The number of Gold and Silver flows is then increased to 2-2 and 4-4 at 60 and 90 s. At 120 s the packet markers are configured to mark all packets with PV 1, which results in a simple head drop AQM behavior. Aggressive Background UDP flows overwhelm the TCP flows immediately, which were well protected before.

VI. Advanced Use Cases

A. Feedback

The Congestion Threshold Value represents the congestion situation well, as it considers not only the volume of offered traffic in relation to the available capacity, but also the importance of the offered traffic. It is a different congestion situation if a lot of low PV, background traffic is lost compared to the case of dropping the same amount of traffic from emergency video supporting, e.g., first responder action. The Congestion Threshold Value is therefore quite suitable to provide feedback to the network edge nodes to implement, for example, admission control.

To perform admission control, at the arrival of a new flow (probably an event signaled to the network edge) the network edge can compare the current Congestion Threshold Value (towards the destination of the new flow) with the profile of the new flow. If it deems that the network has not enough resources (congestion is too high) to serve the new flow, it can block it (perhaps in a probabilistic way). It is also possible to preempt ongoing flows in a similar manner.

B. Load Balancing

If multiple parallel links or paths are available towards the same destination, splitting the traffic among them can result

in higher total aggregate Packet Value realized towards that destination. The capacity of the alternatives may vary and they may carry other traffic, as well, which results in varying congestion situations. The node which splits the traffic needs to balance the load among the alternatives to maximize total realized PV. This is achieved if the Congestion Threshold Value on the alternatives is the same. If not, then moving flows from a more congested alternative to a less congested one will result in the transmission of extra PV.

C. Resource Balancing

In case of resource balancing it is not the traffic that is balanced across two given resources, but the resources are balanced under two traffic flows. For example, different terminals may be using different radio technology operating in the same spectrum. In this case the system cannot influence how much traffic is offered using each technology, but it can (re-)allocate the resources (e.g., spectrum, power or processing) among the radio technologies. This re-allocation can be performed in a similar manner to the load balancing example before: The amount of resources are increased to the access(es) where the Effective Congestion Threshold Value is higher form the technologies where this value is lower until this threshold value is equalized.

D. Network Value Maximization

The above solutions can be generalized to the network level. The task in this case is to find the optimal resource allocation and routing in a way that the total PV realized by the whole network is maximized. This problem is already studied in the Network Utility Maximization (NUM) [6] framework in a quite detailed way. The additional problem to solve is not to find the theoretical maximum, but to create simple algorithms, which can approximate that in a practical, highly dynamic case.

VII. Summary

In this paper we extended the RFQ concept by marking a scalar Packet Value on each packet, which can govern resource sharing even when devices need different amount of resources to transmit the same amount of data. We also introduced the throughput-Packet Value function to describe and visualize a rich set of resource sharing polices. We have outlined a packet marking algorithm for these functions, which is used by Network edges to assign a Packet Value to each packet.

The key advantage of the concept is that Resource Nodes inside the network only need to maximize the total aggregate Packet Value transmitted, they do not have to be aware of flows. This enables low implementation complexity in these nodes and scales well with increased number of flows.

Simulations show that the policies remain executed even under changing network load without any changes at the edges or tuning inside the network. PPV protects congestion controlled TCP traffic from mis-behaving or unresponsive flows even under serious overload from the latter.

We have introduced the notion of Congestion Threshold Value as a congestion measure that considers not only the volume of offered traffic in relation to the available capacity, but also its importance. It is a different congestion situation if a lot of low PV, background traffic is lost compared to the case of dropping the same amount of traffic from emergency voice.

We have addressed how delay requirements can be considered via marking packets with a Delay Class, which is orthogonal to Packet Value. A packet marked with low delay and low Packet Value will get delivered only at light or no congestion - but then at low delay.

Finally, we have listed a number of advanced use cases of PPV to implement load balancing, resource balancing and optimal network-wide load distribution.

It remains for future work to evaluate our delay aware resource sharing proposal and to develop algorithms for the advanced use cases.

References

[1] 3GPP TR 23.799, “Study on Architecture for Next Generation System”, V0.2.0 (2016-02)
[2] J. Martin et al., “Asynchronous Transfer Mode : ATM Architecture and Implementation”, Prentice Hall, November 1996
[3] RFC 1633: “Integrated Services in the Internet Architecture: an Overview”, June 1994
[4] RFC 2475: “An Architecture for Differentiated Services”, December 1998
[5] 3GPP TS 23.203, “Policy and charging control architecture”, V13.6.0 (2015-12)
[6] Yung Yi and Mung Chiang. “Stochastic network utility maximisation - A tribute to Kellys paper published in this journal a decade ago.” Eur. Trans. Telecomm., 19(4):421-442, 2008
[7] Zhiruo Cao et al., “Rainbow fair queueing: theory and applications”, Elsevier Computer Networks 47 (2005) pp 367-392
[8] Internet Research Task Force, “Open Research Issues in Internet Congestion Control”, RFC6077, February 2011
[9] The netfilter.org “libnetfilter_queue” project, Checked 2016-03-17 http://www.netfilter.org/projects/libnetfilter_queue/
[10] R. Pan et al., “PIE: A lightweight control scheme to address the bufferbloat problem”, High Performance Switching and Routing (HPSR), 2013 IEEE 14th International Conference on, Taipei, 2013, pp. 148-155
[11] “iPerf - The network bandwidth measurement tool”, Checked 2016-03-25, https://iperf.fr/
[12] M. Alizadeh et al., “Deconstructing Datacenter Packet Transport”, Proc. ACM HotNets XI, pp. 133-138, October 2012
[13] “Cisco IOS Quality of Service Solutions Configuration Guide”, Release 12.2, Congestion Avoidance Overview, Checked 2016-03-31