Towards Core-Stateless Fairness on Multiple Timescales

Szilveszter Nádas
Ericsson
Budapest, Hungary
szilveszter.nadas@ericsson.com

Gergő Gombos, Ferenc Fejes and Sándor Laki
ELTE Eötvös Loránd University
Budapest, Hungary

ABSTRACT

Extending fairness to multiple timescales creates the right incentives for users and provides better QoE for short sessions, e.g. for web page download. In this paper, we show how to define and implement multi-timescale fairness among flows independently of actual traffic mixes and resource capacities. The proposed method is built on the top of the Multi-Timescale Bandwidth Profile concept and the core-stateless resource sharing framework called Per Packet Value (PPV). It adds two novel ideas: 1) Replacing the traditional weighted-fairness definition of PPV by extending Throughput-Value Functions to multiple timescales (MTS-TVF); 2) Providing an efficient packet marking algorithm using MTS-TVFs to assign values to packets. After marking the packets, the routers in the network core can work with any prior schedulers of PPV. Finally, our early results towards multi-timescale fairness are demonstrated by simulations.

CCS CONCEPTS

• **Networks → Packet scheduling**;

KEYWORDS

Fairness; Resource Sharing; Packet Marking; Rate Measurement; Timescales; PPV

ACM Reference Format:
Szilveszter Nádas and Gergő Gombos, Ferenc Fejes and Sándor Laki. 2019. Towards Core-Stateless Fairness on Multiple Timescales. In *ANRW '19: Applied Networking Research Workshop, July 22, 2019, Montreal, QC, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3340301.3341124

ANRW '19, July 22, 2019, Montreal, QC, Canada

ACM ISBN 978-1-4503-6848-3/19/07…$15.00
https://doi.org/10.1145/3340301.3341124

1 INTRODUCTION

Resource sharing among traffic flows has remained an area of interest in networking research. Fairness is usually interpreted as equal (or weighted) throughput [1] experienced by flows. In this paper we use the term *flow* for a traffic aggregate, which has an associated resource sharing policy, e.g. subscriber, node, service endpoint, traffic flow. By definition, throughput is a measure derived from total packet transmission during a time interval, the length of which is called *timescale*. Current resource sharing control methods are based on throughput measured only on a short timescale (e.g. round trip time (RTT)).

For bursty traffic, throughput measured on multiple timescales (e.g. RTT, 1 s, 10 s, session duration) usually results in different values. From the end-user perspective, network performance is better described by throughput during the active periods of a source as opposed to the general case when active and inactive periods are both considered. Taking the history of inactivity into account is advantageous for short transmissions like web downloads or initial buffering of adaptive video streaming. A comprehensive recent survey on fairness [1] states that "getting a scheme to instantly serve web flows for improved performance while maintaining fairness between other persistent traffic remains an open and significant design problem to be investigated." The authors of [8] introduce a Multi-Timescale Bandwidth Profile (MTS-BWP), which defines and implements multi-timescale fairness for a few sources in well-defined and well-dimensioned scenarios. MTS-BWP profiles against several token buckets per Drop Precedence representing increasing timescales of measurements. MTS-BWP does not scale as the number of drop precedences increases; therefore, it cannot provide fine-grained control.

In this paper, we extend the Per Packet Value (PPV) concept [6, 7] and the MTS-BWP [8] to provide fine-grained fairness on multiple timescales that is independent of traffic mixes and resource bandwidths. Compared to [6], the proposed solution only replaces the packet marker in the edge of the network and does not require any changes in the core of the network. To this end, we propose a practical

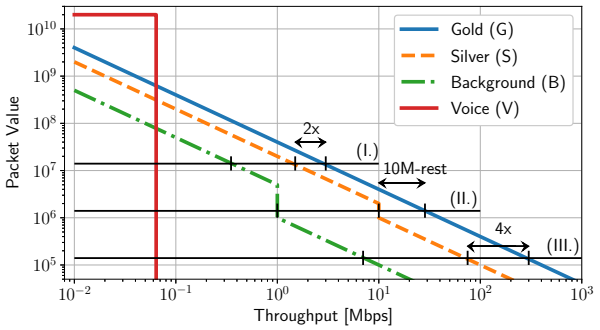


Figure 1: TVF examples, log-log scale.

multi-timescale packet marking method having moderate memory footprint and computational complexity.

2 PPV OVERVIEW

The PPV concept [7] extends the idea of core stateless resource sharing solutions like [2, 9] by marking each packet with a continuous value called Packet Value (PV). The main goal in a PPV network is to maximize the total aggregate PV of delivered packets. 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.

First, packets are marked at the edge of the network by using the resource sharing policy of the operator. Operator policies are described by Throughput-Value Functions (TVFs) (marked by $V(.)$), that defines the PV distribution of a *flow* for any sending rate (R). Specifically, if one measures the throughput of the marked packets of a flow having PV of $V(b)$ or higher, it should be b (for any $b : 0 \leq b \leq R$). Accordingly, at high congestion only packets with high PVs are transmitted, more precisely packets with PV above a given Congestion Threshold Value (CTV) that reflects the actual congestion level. 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 shows examples for 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 four 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 implements 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, solving resource sharing in this way. To this end, PPV framework proposes simple PV-based scheduling algorithms [6, 7] including algorithms that drops the packet with the smallest PV (even from the middle of the buffer) when the buffer length is too long or using PI controllers to determine a PV threshold for packet dropping. Among the different proposals, we use a congestion control independent AQM algorithm called CSAQM [6] without any modification in this paper. CSAQM applies a simple PV threshold-based ECN-marking or packet dropping strategy in the middle of the network.

3 MULTI-TIMESCALE FAIRNESS

For bandwidth profiling, bitrate is typically measured on a short timescale in the order of RTT. It expresses the instantaneous resource usage and it can even capture short bursts. However, for ensuring long-term fairness (or network usage SLA) among flows with largely different profiles, bitrates on longer timescales are far more expressive.

In this paper, we use several timescales (TS) with different length: $TS_1 \approx RTT < TS_2 < \ldots < TS_n$. For a flow with an equally paced, stable traffic, after the time associated with the longest timescale has elapsed, we expect all those rate measurements to be the same ($R_i \approx R$, $\forall TS_i$). However, in transient situations, e.g. when transmission starts for a previously silent flow, we expect small rate measurements of long timescales, while R_1 (of $TS_1 \approx RTT$) may be high (i.e. $R_1 > R_2 > \ldots > R_n$). Rate measurements at shorter timescales react faster to the changes of network conditions, while at longer timescales temporal changes may remain invisible. Similar behavior with the opposite ordering can be seen for a case when a flow stops transmission (or its rate decreases) after a long active period. Fig. 4c depicts rate measurements for three TSs in a two-flow scenario where both transmission start and rate decrease can be observed.

Balancing rates on all timescales creates the right incentives for controlling a traffic flow, e.g. when it represents the aggregated traffic of a subscriber. After an inactive/silent period of a subscriber it gets advantage for its new session (in long term the network resource was underused), while if it is transmitting for a long time it does not (long term fairness does not allow further increase in the resource share).

Based on the detailed analysis of [5], we created two rate measurement algorithms (RMA): 1) a token bucket emulation-based one for the RTT timescale and 2) an efficient sliding window-based one for longer timescales.

Token bucket emulation RMA. This algorithm maintains a changing-rate token bucket and tries to set the rate such that the bucket never gets full or empty. It aims at modeling the fair throughput and buffer share of the flow at the bottleneck scheduler. It has the following parameters: TS representing the timescale, $f_{MBS} = 0.1$ a factor affecting the maximum bucket size (MBS) and $MBS_{min} = 5 \times 1500Bytes$ the smallest MBS (a few MTU).

Initialization: Bucket rate $R = 0$, token level $L = 0$, last update $t_{last} = 0$.

Packet Arrival: When a packet of size ps arrives at time t we update $L = L + R \times (t - t_{last}) - ps$, and $t_{last} = t$.

Then *if the token level becomes negative,* ($L < 0$) we increase the rate according to the extra bits arrived, $R = R - L/TS$ and we set $L = 0$.

Otherwise, *if the token level is too large,* ($L > MBS$, $MBS = max(MBS_{min}, R \times TS \times f_{MBS})$) we decrease the token rate and adjust the token level, $R = R - (L - MBS)/TS$, $L = MBS$. We check the resulting R, and if it is smaller than the rate of the packet just arrived on TS, $R < ps/TS$, then we set $R = ps/TS$, $L = 0$.

The default is when *the token level is within the desired range.* In this case, we do not change R and L.

Sliding window-based RMA. We use a practical approximation of the Time-Dependent Rate Measurement algorithm with Time Window Moving Average (TDRM-TWMA) [5]. (TWMA calculates the rate as the "amount of bits arrived in the last TS time" divided by TS). Our approximation has the following parameters: TS represents the timescale, and N denotes how many disjoint time windows (of size TS/N) we use for the approximation (plus we use one more time window of changing size). The memory footprint of the algorithm is dominated by N that is set to 10 in our experiments shown later in this paper.

Initialization: Rate $R = 0$, total bits in all intervals $b = 0$, last interval start $t_{start} = 0$, bits arrived in all intervals $b[i] = 0$, $i = 0 \ldots N$. Accordingly, $b[i]$ ($i \geq 1$) represents bits arrived in the time interval $[t_{start} - i * TS/N,\ t_{start} - (i - 1) * TS/N]$, $b[0]$ represents the bits arrived since t_{start}. Note that $b = \sum_{i=0}^{N} b[i]$ is only used for optimization.

Packet Arrival: When a packet of size ps arrives at time t we first *maintain time windows.* Then we increase $b[0] = b[0] + ps$ (and $b = b + ps$). We calculate the rate as

$$R = \left(b - \frac{t - t_{start}}{TS/N} \cdot b[N] \right) / TS,$$

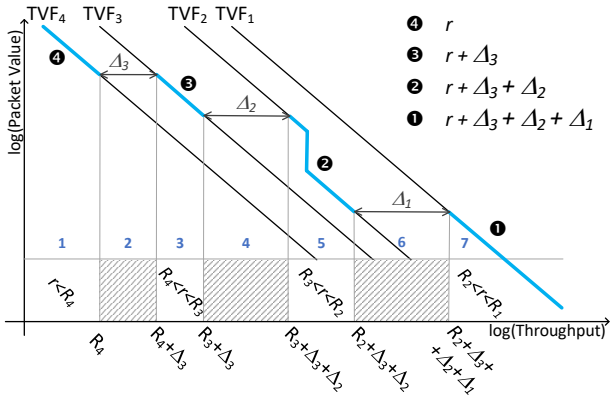


Figure 2: Determination of PV for a 4-timescale case

where we compensate that $t - t_{start}$ time has already elapsed in the current time window, $b[0]$, therefore a fraction according to that time window is subtracted from the oldest time window, $b[N]$.

Maintain time windows: In this part we make sure that the current time window, $b[0]$, is not longer than TS/N. When it becomes longer, we phase out old history and open a new time window[1]. While $t - t_{start} > TS/N$ we repeat the rest of the actions in this part. For all $b[i]$, $i = N \ldots 1$, we set $b[i] = b[i - 1]$, and we set $b[0] = 0$ and $b = \sum_{i=1}^{N} b[i]$. Then we set $t_{start} = t_{start} + TS/N$.

3.1 Multi-Timescale TVF (MTS-TVF)

Instead of having a single TVF per flow type, we define one TVF per TS (per flow type). Fig. 2 depicts an example with four TVFs: $TVF_4 \ldots TVF_1$ (disregard the other markings for the time being). A high-level description is that a given TVF_i shall dominate the resource share of the flow when the rate on that timescale (R_i) is dominant. E.g. if a flow has dominant R_4 and another one has dominant R_1, their resource share shall be according to TVF_4 and TVF_1, as if those would be single-timescale TVFs. At the same time, we aim at smooth transmission as the relations between the rates (R_i) changes. As we want to achieve higher throughput for small bursts, $TVF_i(x) \geq TVF_{i+1}(x)$ for all i,x.

3.2 MTS Rate Measurement-based Marker

For a single-timescale TVF we can mark packets by measuring the bitrate R, determining a uniform random throughput $r \in [0,\ R]$, and setting the PV to TVF(r). We aim to generalize this simple packet marking algorithm for the MTS-TVF. We create and explain an algorithm, which chooses the right

[1] In the simulator implementation we use a circular buffer structure to avoid copying $b[i]$s. We describe the algorithm this way because of clarity.

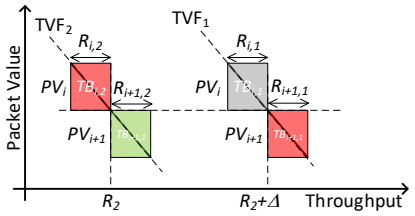


Figure 3: A quantized MTS-TVF example

TVF_i and the right throughput value, based on the measured bitrates and a random throughput.

The TVFs in Fig. 2 could be quantized to a Token Bucket matrix like in [8]. That would mean several thousand token buckets (i.e. the quantized PV range × the number of TSs), which is not practical to implement. This model however helps us to design an efficient packet marker implementation.

Fig. 3 depicts a quantized two-timescale MTS-TVF example. Similarly to the MTS-BWP [8] a packet can be marked to PV_i if Token Buckets $TB_{i,2}$ and $TB_{i,1}$ (with rates $R_{i,2}$ and $R_{i,1}$) both contain enough tokens. $R_{i,1} > R_{i,2}$, and $R_{i+1,1} > R_{i+1,2}$, because the TVFs are parallel on logarithmic scale. At the same time, maximum token levels for the token buckets ($BS_{i,j} = R_{i,j} * TS_j$) normally have different order ($BS_{i,1} < BS_{i,2}$ and $BS_{i+1,1} < BS_{i+1,2}$), because the multipliers between TS_is are usually larger than the ones between TVF_is.

Specifically in case of PV_i for a burst, $TB_{i,1}$ will first be emptied (before $TB_{i,2}$). When $TB_{i,2}$ is also emptied that means that on the TS_2 the token buckets representing TVF_2 measure $R_2 = \sum_{j=1}^{i} R_{j,2}$. (It is easy to see that buckets $TB_{j,2}$, $j < i$ are emptied before $TB_{i,2}$). Assuming that $TB_{i+1,2}$ is not yet emptied, then until R_2 the TVF_2 will be used when marking packets. Above R_2, TVF_1 will be used. However, the region between R_2 and $R_2 + \Delta$ cannot be used (of any TVF) for packet marking. Δ is the distance between the two TVFs at $PV = TVF_2(R_2)$.

This suggests that by measuring the rate on both timescales we can simplify the packet marking for this two-TVF case. We determine a uniform random number, r between 0 and R_1. If $r \leq R_2$ we mark the packet to $TVF_2(r)$, otherwise we mark it to $TVF_1(r + \Delta)$.

We generalize this concept to several timescales and call the resulting packet marker MTS Rate Measurement-based Marker (MTS-RMM). In Fig. 2 we demonstrate the method using a 4 TS example. When updating the rate measurements for different timescales (R_i) we also update the distances of TVF_i from TVF_{i+1} (called Δ_i) at the relevant rates. Only regions 1, 3, 5, 7 are used for determining PV; the blue TVF is the one used in each region. Depending on the relation between the random throughput $r \in [0, R_1]$ and R_is, the region and Δ_is to add are chosen (indicated by the numbers in black circles next to the blue regions). E.g. if $R_3 < r < R_2$, the PV is set to $TVF_2(r + \Delta_3 + \Delta_2)$.

We detail the generic algorithm below (it works for any order of R_is, not only for $R_1 > R_2 > \ldots > R_n$ as in the example). When a packet arrives into the packet marker, after updating the rate measurements (R_i), we update Δ_is as follows:

Initialization: k is the number of TSs (e.g. 4), $R'_k = R_k$, $i = k - 1$.

Calculation of Δ_is: $R'_i = max(R'_{i+1}, R_i)$ is the largest measured bitrate on all TS_j, $j \geq i$.

$$\Delta_i = TVF_i^{-1}\left(TVF_{i+1}\left(R'_{i+1} + \sum_{j=i+1}^{k-1} \Delta_j\right)\right) - \left(R'_{i+1} + \sum_{j=i+1}^{k-1} \Delta_j\right).$$

Then set $i = i - 1$ and repeat the calculation until $i < 1$.

If the longest timescale bitrate that changed is R_j, then all Δ_is, $i \geq j$ remain unchanged. By only updating R_ks when they would significantly change (instead of for every packet arrival), the computation demand can be further optimized.

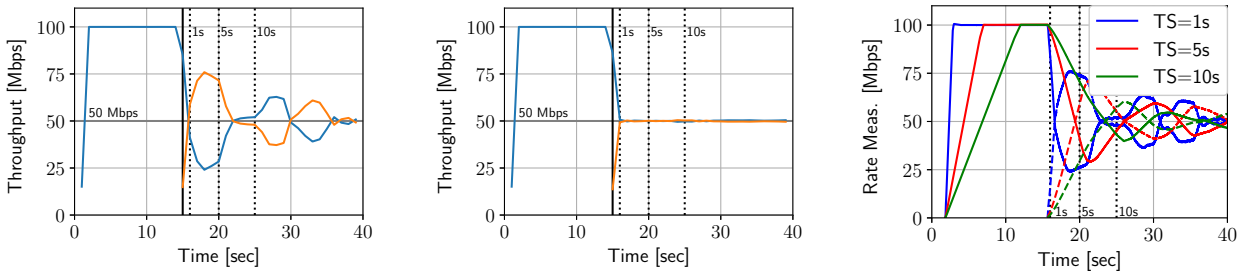
After updating R_is and Δ_is, we determine a uniform random throughput $r \in [0, R_1]$. We search for the maximum i, where $r \leq R_i$ then we mark the packet according to $PV = TVF_i\left(r + \sum_{j=i}^{k-1} \Delta_j\right)$. This finds the right region in Fig. 2, chooses the Δ_js to add, and the TVF_i to use.

4 EVALUATION

MTS-RMM described in Section 3 has been implemented and evaluated in the NS-3 simulator [4]. The bottleneck node applies the CSAQM scheme described in [6] with 10 ms queueing delay target. In all the examined simulation scenarios, we consider a single downlink bottleneck where the same bottleneck buffer is shared by several flows. We also assume that there is no uplink bottleneck. The term flow refers to a piece of traffic to which an operator policy expressed by a specific MTS-TVF is applied.

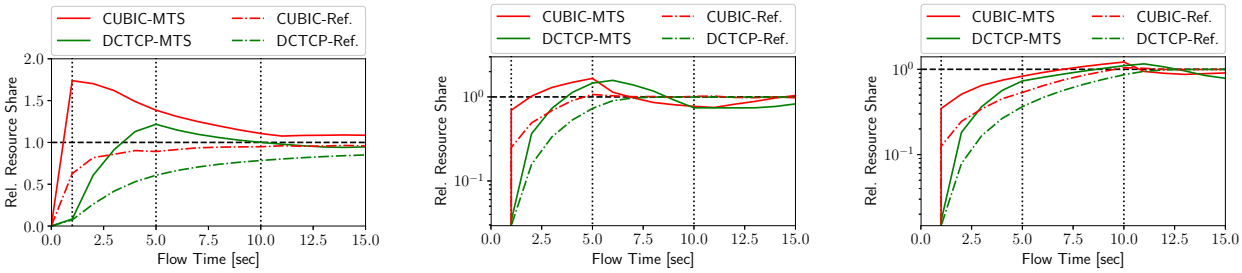
In our simulations, each flow consists of 1 or 4 TCP connections using DCTCP or CUBIC [3] as the congestion control algorithm, respectively. The default round-trip propagation delay is moderate (10 ms). In the simulations, MTS-RMM is used with 4 timescales: 10ms, 1s, 5s, and 10s. TVF_4 is configured to be Gold or Silver, according to Fig. 1. TVFs for shorter timescales are configured to have weights 2, 4 and 8 compared to TVF_4, i.e. $TVF_3(x) = TVF_4(x/2)$, $TVF_2(x) = TVF_4(x/4)$, $TVF_1(x) = TVF_4(x/8)$. The single-timescale marker of the original PPV framework is used as reference in the presented scenarios. It is configured with a single TVF: $TVF(x) = TVF_4(x)$.

Greedy flows of the same traffic class. We configure permanent background Gold TCP flows (1 or 10) in the network, and a new TCP flow from the same class (Gold) arrives at



(a) DCTCP flows with MTS Marker (b) DCTCP flows with Ref. Marker (c) Rate measurements of MTS Marker

Figure 4: Resource sharing for a new flow with good history, same traffic class, C=100 Mbps.



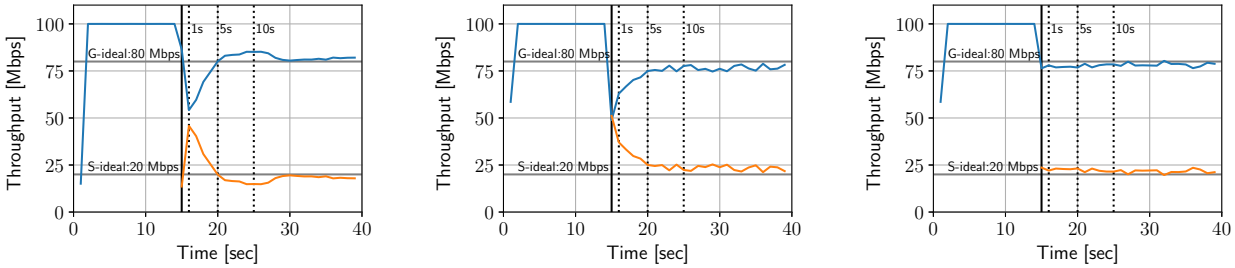
(a) Flow-Time Average (b) 5s Time-Window Average (c) 10s Time-Window Average

Figure 5: Relative resource share for the new flow, same traffic class, 10 background flows, C=1 Gbps

the bottleneck. Every new flow belongs to a separate traffic aggregate, thus upon start-up all its rate measurements are 0. The bottleneck capacity (C) is 100 Mbps or 1Gbps. A time-series plot of flow throughput with 1 background flow and $C = 100$ Mbps is depicted in Fig. 4a for DCTCP flows. For reference, we depict the throughput for DCTCP flows with (single-timescale) PPV in Fig. 4b. For the MTS-RMM, the new flow increases its throughput much faster in the first seconds than for the reference. It is visible that the high initial boost for the new flow will be compensated later in favor of the old flow and after some oscillation, the flow throughput will be equal. Fig. 4c is the time-series plot of the rate measurements in the MTS-RMM used in Fig. 4a for the background flow (solid lines) and the new flow (dashed lines).

Fig. 5 depicts a scenario with 10 greedy background flows and $C = 1$ Gbps. The new flow arrives 15 s after the background flows. We compare the performance of MTS-RMM and the single-timescale PPV marked "Ref.". The relative resource share is the calculated throughput divided by the long term fair share throughput of the flow. We show the Flow-Time Average (5a) that for any time instance $t > 0$ is calculated as the total number of bits of the flow transmitted till t divided by t, where $t = 0$ is the start time of the new flow. For TCP flows the Flow-Time Average grows faster than in the reference case and it grows above the long-time fair share in the first 10 s as desired. By 10 s it gets close to the long-term fair share. In the Cubic case, we have 4 TCP connections per flow (vs. 1 for DCTCP), which explains the faster start. Figs. 5b,5c illustrate the 5s and 10s time window average of the observed throughput. For any time t, the time window average is calculated as the average throughput of the flow in interval $[t - W, t)$, where W denotes the window size (5s and 10s). These figures show that a flow approaches its long-time fair share much faster with MTS-RMM. E.g. the 5s average throughput of the Cubic flow on Fig. 5b reaches it long-time fair share after being in the system for 2 s.

Greedy flows of different traffic classes. In these cases, the new flow is from the Silver class, which means that its long-time fair share is smaller than that of the flows already in the system. Still short term it is to be boosted when it has no previous transmission. Fig. 6 depicts time-series plots for DCTCP (6a) and Cubic (6b) TCP flows. Fig. 6c depicts a reference (PPV) simulation. The observed behavior is similar to the previous scenario, MTS-RMM marking gives more resource to the newcomer flow in the defined timescale ranges



(a) DCTCP flows with MTS-RMM (b) Cubic flows with MTS-RMM (c) DCTCP flows with Ref. Marker

Figure 6: Resource sharing for a new flow with good history, different traffic class, C=100 Mbps.

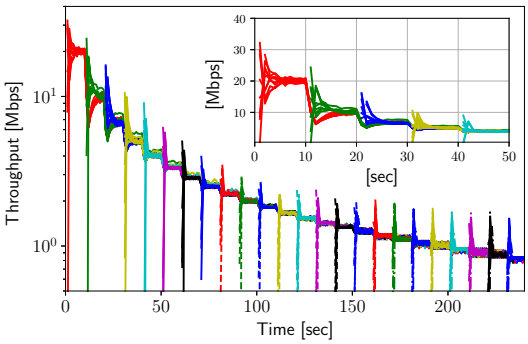


Figure 7: Continuous arrival, 10 flows every 10 s, same traffic class, C=200 Mbps.

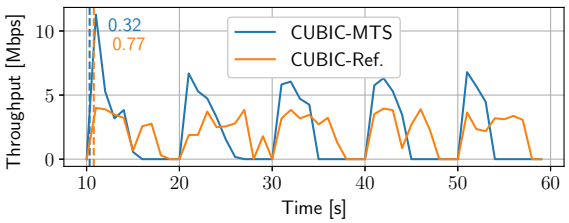


Figure 8: Adaptive streaming comparison

and converges to the long-time fair share after 10 s. Cubic flows still get a higher initial share, again due to the 4 connections resulting in a more aggressive slow start.

Considering Flow-Time Average, the 5s and 10s Time-Window Avarage relative to the long-term fair share, the achieved relative acceleration is even higher than in the previous scenario shown in Fig. 5, especially for the Cubic case, as the fair share of Silver flows is smaller while the TCP start-up speed is the same.

Continuous flow arrival. In this case, 10 new Gold TCP flows arrive to a $C = 200$ Mbps bottleneck every 10 s (240 flows in the end). Fig. 7 depicts this case; the different batches are denoted by different colors. It can be observed that the new flows are temporarily boosted, and that flows in the system for a longer period have equal shares.

Simple adaptive streaming model. A Gold adaptive streaming flow shares a $C = 35$ Mbps bottleneck with 9 greedy Gold TCP flows. All flows have 1 TCP connection. We use a simplified adaptive video streaming model, starting at $t = 10$ s, the streaming flow downloads 2,5 MByte data initiated every 10 s. Fig. 8 compares the throughput of the adaptive streaming flow in the MTS-RMM and single TS (Ref.) case. The dotted vertical lines show when 0.25 Mbyte has been downloaded, which models the time to play the video, i.e. having enough initial buffers. The silent periods stand for having enough playout buffer, while the restart of the session models the case when the buffer needs to be refilled. MTS-RMM has clear advantages, the time to play is much shorter, and it always fills the playout buffer faster, which means shorter transmission periods, which results in smaller battery use.

5 DISCUSSION

Though the initial results presented in this paper demonstrate the benefits of ensuring fairness on multiple timescales, several questions are left unsolved. On one side, we have presented that a significant performance gain can be obtained with the proposed concept by accelerating the starting phase of new flows, and providing better long-term fairness for flows with on-off behavior (e.g. DASH-like traffic) and better network access (higher throughput) for short flows. However, on the other side several details of the main concept require further research, including open questions like 1) What is the practical number of timescales to be used? 2) How shall the timescales be dimensioned? 3) How to design multi-timescale TVFs? Does it make sense to use a different kind of policy at various timescales? 4) What further policies that have practical relevance can be described in this model?

ACKNOWLEDGEMENT

The authors thank the support of Ericsson Hungary Ltd. G. Gombos also thanks the support of the European Union, co-financed by the European Social Fund (EFOP-3.6.2-16-2017-00013). The research of S. Laki was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences. The authors thank the anonymous reviewers for their suggestions on how to improve clarity of the paper.

REFERENCES

[1] Ghulam Abbas, Zahid Halim, and Ziaul Haq Abbas. 2016. Fairness-driven queue management: A survey and taxonomy. *IEEE Communications Surveys & Tutorials* 18, 1 (2016), 324–367.

[2] Zhiruo Cao, Ellen Zegura, and Zheng Wang. 2005. Rainbow fair queueing: theory and applications. *Computer Networks* 47, 3 (2005), 367 – 392. https://doi.org/10.1016/j.comnet.2004.07.018

[3] Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: A New TCP-friendly High-speed TCP Variant. *SIGOPS Oper. Syst. Rev.* 42, 5 (July 2008), 64–74. https://doi.org/10.1145/1400097.1400105

[4] Thomas R Henderson, Mathieu Lacage, George F Riley, C Dowell, and J Kopena. 2008. Network simulations with the ns-3 simulator. In *Sigcomm (Demo)*, Vol. 14.

[5] Michael Menth and Frederik Hauser. 2017. On moving averages, histograms and time-dependentrates for online measurement. In *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering*. ACM, 103–114.

[6] Szilveszter Nádas, Gergő Gombos, Péter Hudoba, and Sándor Laki. 2018. Towards a Congestion Control-Independent Core-Stateless AQM. In *ANRW '18*. 84–90. https://doi.org/10.1145/3232755.3232777

[7] Szilveszter Nádas, Zoltán Richárd Turányi, and Sándor Rácz. 2016. Per Packet Value: A Practical Concept for Network Resource Sharing. In *IEEE Globecom 2016*.

[8] Szilveszter Nádas, Balázs Varga, Illés Horváth, András Mészáros, and Miklós Telek. 2019. Multi timescale bandwidth profile and its application for burst-aware fairness. *arXiv preprint arXiv:1903.08075* (2019).

[9] Ion Stoica, Scott Shenker, and Hui Zhang. 2003. Core-stateless Fair Queueing: A Scalable Architecture to Approximate Fair Bandwidth Allocations in High-speed Networks. *IEEE/ACM Trans. Netw.* 11, 1 (Feb. 2003), 33–46. https://doi.org/10.1109/TNET.2002.808414