Bandwidth profile for multi-timescale fairness

Szilveszter Nádas*, Balázs Varga*, Illés Horváth†, András Mészáros‡, Miklós Telek‡
*Traffic Analysis and Network Performance Laboratory, Ericsson Research, Hungary
{szilveszter.nadas,balazs.a.varga}@ericsson.com
†MTA-BME Information Systems Research Group, Hungary
horvath.illes.antal@gmail.com
‡Dept. of Networked Systems and Services, Budapest University of Technology and Economics, Hungary
{meszarosa, telek}@hit.bme.hu

***Abstract*—We propose a novel approach to provide fairness on multiple timescales in the framework of core-stateless resource sharing paradigm. We illustrate the unique advantages of multi-timescale fairness. We introduce a new marking scheme, called multi-timescale bandwidth profile. It assigns drop precedence to flows based on their traffic history on multiple timescales. Additionally, we provide dimensioning guidelines for the introduced profile in an access-aggregation network scenario and present its simulation-based performance analysis.**

***Index Terms*—fairness, multiple timescales, core stateless, resource sharing, bandwidth profile, QoS, fluid model**

I. Introduction

There are two major challenges for resource management in transport network with the advent of 5G. First, traffic is becoming more and more bursty due to the highly increased access bandwidth of 5G [1]. As a consequence the network bottleneck is no more limited to radio domain, but it may be inside transport networks as well, which requires more advanced resource sharing capabilities. For such bursty traffic, it is important to reach high rates temporarily, while the longer term traffic rate is much lower. Second, enhanced flexibility of 5G architecture is allowed by network slicing [2], [3], as it allows multiple logical networks to be created on top of a common shared physical infrastructure. Fairness between network slices and full control of resources consumed by network slices require new capabilities as well. Current techniques like resolving congestion for aggregate traffic at bottlenecks according to transport service definitions and related bandwidth profiles (BWPs) of the Metro Ethernet Forum (MEF) [4] have to be developed further in order to deal with the traffic of 5G Systems.

Since BWPs of MEF are insensitive to traffic history, we introduce a novel BWP that is temporarily able to assign high rate to traffic sources with good history measured on multiple timescales. As a results the new BWP provides fairness on multiple timescales by extending the currently used BWPs, e.g., [4]. For example, in an access-aggregation scenario, we establish timescale specific requirements on resource sharing and provide a dimensioning method for such a new BWP using a limited number of drop precedences.

Partially supported by the OTKA K123914 and the TUDFO/51757/2019-ITM grants.

To the authors' knowledge, there is no similar solution that formalizes and implements multi-timescale fairness and controls resource sharing accordingly: [5] 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 proposed multi-timescale BWP differs from existing token sharing solutions, e.g., [6], [7], where the buckets sharing tokens belong to different service classes, so they impact the fairness among sources belonging to *different* traffic classes (on a single timescale). In our solution, resource sharing controls the behavior of a single service class, so fairness is provided among sources *within* the same traffic class (on multiple timescales). Our solution can be used simultaneously with the above methods to achieve fairness at the same time *across and within* traffic classes.

In order to focus on the performance benefit of the proposed BWP, we simulate a fluid model that ignores the imperfection of packet level behavior and assumes an ideal congestion control.

II. Multi-Timescale Fairness

Fairness is usually interpreted as equal (or weighted) bandwidth experienced by entities [5] (e.g., traffic flows, aggregates, nodes, service endpoints). Throughput is a measure derived from total packet transmission during a time interval (the length of which is called timescale). For bursty traffic, throughput measured on multiple timescales (e.g., round trip time (RTT), 1 s, session duration) usually results in different values.

Current resource sharing control methods are usually based on throughput measured on a short timescale (e.g., RTT). From end-user perspective, network performance is better described by throughput during the active periods of a source. For different sources, the typical length of the active periods might be different, accordingly relevant throughput should be calculated over different timescales.

We propose an approach that provides fairness on multiple timescales, which means that sources with similar history shall receive the same throughput, while sources with "good history" (i.e., those which used less network bandwidth recently) shall receive higher throughput. Fig. 1 demonstrates the behavior of the proposed bandwidth sharing. If

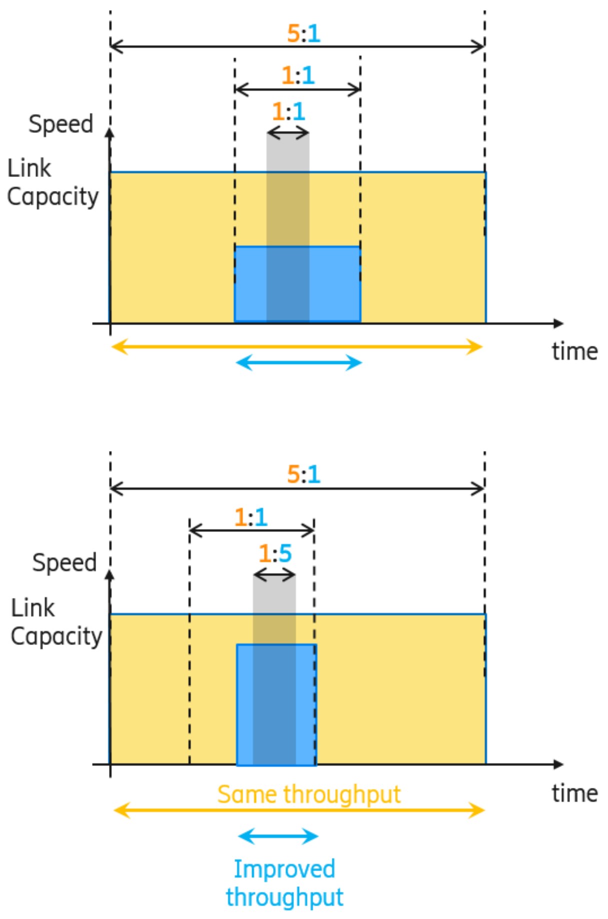

Fig. 1. Effect of providing multi-timescale fairness

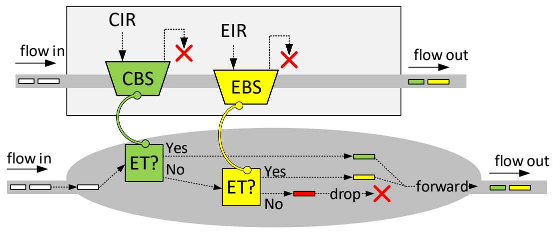

Fig. 2. trTCM bandwidth profile for a single priority

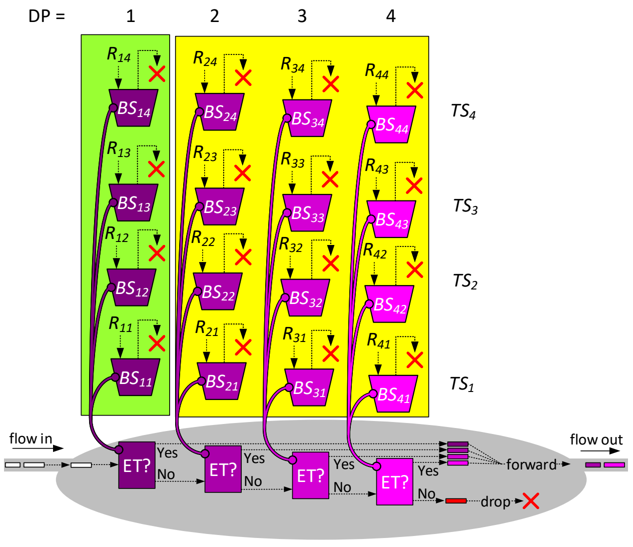

Fig. 3. A 4 × 4 Multi-Timescale Bandwidth Profile

the bandwidth sharing is independent of the history and proportional to the number of flows (which is the case in the upper diagram) the long/short (yellow/blue) flow receive equal bandwidth share (1:1) and is served during the period indicated by the long/short arrow below the time axes. If the bandwidth sharing on the short time scale favors the flow with “good history” and provides a higher share (1:5) to the short flow (c.f. lower diagram) the short flow gets served faster, while the service time of the long flow remains unchanged (indicated by the long/short arrow below the time axes). That is, the proposed method might increase the throughput of sources with “good history” without sources with “bad history” receiving lower throughput on longer timescales. Fig. 1 also displays bandwidth sharing and the ratio of throughput averages for the two sources over three timescales.

This bandwidth sharing is essentially different from current resource sharing control methods, e.g., [8], which do not take the history of the sources into account.

III. Multi-Timescale Bandwidth Profile

In this section, we extend the Two-Rate Three-Color Marker (trTCM) to provide balanced fairness on multiple timescales.

MEF defines multiple variants of trTCM for bandwidth profiling [4]. The simplest variant is depicted in Fig. 2. It has two rate parameters, the guaranteed Committed Information Rate (CIR) and the non-guaranteed Excess Information Rate (EIR). Both have an associated token bucket, whose sizes are typically set to Committed Burst Size, $CBS \approx CIR \times RTT$, and Excess Burst Size, $EBS \approx EIR \times RTT$. A packet is marked green (conform to CIR), yellow (conform to EIR) or dropped (red) based on the amount of tokens in the associated token buckets. A bucket must contain at least as many tokens as the packet size, marked as ET (Enough Tokens). Upon successful marking, the token bucket is decreased with the packet size.

We extend the trTCM by increasing the number of colors (i.e., drop precedences (DPs)) and by introducing multiple token buckets per drop precedence, representing different timescales (TSs), which we call Multi-Timescale Bandwidth Profile (MTS-BWP). Fig. 3 shows an example of MTS-BWP, where the number of DPs is $N_{DP} = 4$, and the number of TSs is $N_{TS} = 4$.

The darkest purple color in Fig. 3 (DP 1 – dropped last) is similar to green in Fig. 2 in the sense that we intend to guarantee transmission to packets marked dark purple. The lighter purple colors in Fig. 3 are similar to yellow in Fig. 2, though we intend to provide a more refined service than simple non-guaranteed delivery. The inbound token rate of the bucket associated with drop precedence

dp and timescale ts is $R_{dp,ts}$ and the bucket size of bucket $\mathrm{BU}_{dp,ts}$ is $BS_{dp,ts} \approx R_{dp,ts} \times TS_{ts}$ (for the precise calculation, see (10)). The definition of the time scales (TS_{ts}, $ts \in \{1, \ldots, N_{TS}\}$) is discussed in Section VI-B. Thus, $\boldsymbol{R} = \{R_{dp,ts}\}$ and $\boldsymbol{BS} = \{BS_{dp,ts}\}$ are matrices of size $N_{DP} \times N_{TS}$. A packet can be marked a given DP value dp if all buckets $\mathrm{BU}_{dp,ts}$, $\forall ts \in \{1, \ldots, N_{TS}\}$ (which we shorthand as $\forall ts$) contain enough tokens. Upon successful marking, all respective buckets are decreased with the packet size. This, in effect, profiles bandwidth on multiple timescales against predefined target bandwidths ($R_{dp,ts}$).

If we want to enable more bursty traffic on shorter timescales, we have to offer smaller bandwidth on longer timescales. Thus the rows of $\boldsymbol{R}$ are decreasing, i.e.,

$$R_{dp,ts+1} \leq R_{dp,ts}, \ \forall dp, ts. \tag{1}$$

A 4×4 multi-timescale bandwidth profile provides significant improvement over trTCM (as shown in Section VII), and requires the maintenance of 16 token buckets (instead of 2 for trTCM). This level of complexity is feasible with standard devices. The core needs to support 4 color drop precedence markings instead of 2; a possible implementation is addressed in Section IV.

IV. Active Queue Management (AQM) algorithm

The proposed BWP builds on the core-stateless AQM algorithms presented in [8]–[10] and prototyped in [11], [12]. In the core-stateless resource sharing framework, packets are marked per traffic aggregate at the edge of the network and scheduling is performed in the core of the network based on the packet marking only, which scales well with the number of traffic aggregates. In the core of the network FIFO buffers with *drop the largest DP from head* AQM (DP 1 is dropped last) are implemented. More precisely, when the buffer is full, AQM determines the largest DP which has a packet present in the buffer and drops the first packet from the head which has this DP. This behavior can also be implemented on existing hardware by representing drop precedences with DSCP values in the IP header, configuring these DSCPs to the same queue, and configuring DP specific TailDrop thresholds to drop the different DPs at increasing queue lengths (the largest DP at the smallest queue length) [13].

For our simulation based performance evaluation, we describe the system by a fluid model, which assumes ideal AQM behavior.

V. Fluid model of the MTS-BWP

A. High-level model

We model a common bottleneck shared among several traffic sources (which we refer to as *nodes*) with identical BWP configurations. A node may have zero, one, or multiple *flows*. Nodes with zero flows are *inactive*. When no BWP is applied, nodes share the bottleneck bandwidth proportionally to the number of flows within the nodes (modeling the fairness achieved by TCP congestion control).

In contrast, for the proposed BWP, all traffic with DP lower than the (load and history dependent) *congestion DP* is transmitted and all traffic with higher DP is discarded. This defines the possible throughput range of each node and constrains the effect of TCP on fairness. The resulting node throughput sharing is as close to proportional (to the number of flows) as possible, subject to this constraint. It is typical that some nodes get either the minimum or the maximum throughput within this range.

The token level $TL^{n}_{dp,ts}(t)$ in bucket $\mathrm{BU}^{n}_{dp,ts}$ of node n is maintained and fluid flows out from each bucket of a given DP according to the transmission rate of the node on that DP $th_{dp,n}$.

Our system model assumes instantaneous adaptation and no bottleneck buffer: RTT is equal to 0, there is no packet loss, and packets are infinitesimal. These assumptions correspond to a fluid model where the throughput of each flow adapts instantly to varying conditions; in a packet level model, congestion results in packets lost and re-sent, while in this fluid model congestion leads to a lower throughput of the corresponding flow. Consequently, we set $BS_{dp,1} = 0$ in the fluid simulator (which results in maximum fluid rate of $R_{dp,1}$ on a given dp, see (2)).

B. Simulator settings

1) System parameters: The system is described by the following parameters: C: the capacity (total available bandwidth); N: the number of nodes; $\boldsymbol{R}, \boldsymbol{BS}$ (assumed to be the same for all nodes); $f^{\max}$ flow limit: the maximum number of concurrent flows at each node; further flows are discarded upon arrival.

2) Traffic Model: We use a compound Poisson point process with a discrete flow volume (referred to as file size in this document) distribution (given by possible file sizes and associated probabilities) as an input, and, based on that, we simulate the arrival time and the size of each arriving flow. Each node has a long-term fair share $S_n = C/N$, $n = 1, \ldots, N$ (corresponding to CIR). The *nominal load* of each node can then be calculated as 'nominal load' $=$ 'average file size' $\times$ 'arrival rate'$/S_n$. The *system load* is the average of the nominal loads for all nodes. The system is *underloaded* if its load is less than 1, and it is *overloaded* otherwise. A node has *low load* if its nominal load is less than the system load, and it has *high load* otherwise.

C. Bandwidth allocation model

In this subsection, we formalize the high-level model, i.e., the calculation of the congestion DP and the bandwidth allocation within the congestion DP.

At any given point in time, we collect the bandwidth bounds determined by the current bucket levels for drop precedence dp at node n into an $N_{DP} \times N$ matrix, denoted by $\boldsymbol{BD}$, thus

$$\boldsymbol{BD}_{dp,n} = \min_{ts=1 \ldots N_{TS}} \{R_{dp,ts} \,|\, TL^{n}_{dp,ts} = 0\}. \tag{2}$$

To present the bandwidth allocation we use the following notation: f_n is the number of flows in node n, th_n is the throughput of node n, initialized with 0, e_n shows whether node n is eligible for increase, initialized with True.

The iterative algorithm to calculate the throughput allocation is as follows.

Calculate the congestion DP as

$$dp_c = \min\left\{ i : \sum_{dp=1}^{i} \sum_{n=1}^{N} BD_{dp,n} \geq C \right\}. \qquad (3)$$

th_n is initialized for all n as $th_n = \sum_{dp=1}^{dp_c-1} BD_{dp,n}$. Then the procedure iterates the following 3 steps until $\sum_{n=1}^{N} th_n = C$:

1) Set nodes with $th_n = \sum_{dp=1}^{dp_c} BD_{dp,n}$ to non-eligible ($e_n =$ False).
2) Mark all eligible nodes for which the ratio th_n/f_n is minimal among all eligible nodes.
3) Increase th_n for all marked nodes by $f_n \cdot \delta$, where $\delta > 0$ is calculated as the maximal possible increase such that the following remain valid:
   - $th_n \leq \sum_{dp=1}^{dp_c} BD_{dp,n}$ for all n,
   - the ratio th_n/f_n among all marked nodes does not increase beyond the *second smallest ratio* th_n/f_n among all eligible nodes, and
   - $\sum_{n=1}^{N} th_n \leq C$.

For node n, $th_{dp,n}$ is calculated from th_n and $BD_{dp,n}$ by dividing th_n according to the bounds $BD_{dp,n}$ in the order $dp = 1, 2, \ldots, N_{DP}$. The resulting bandwidth allocation corresponds to the BWP described in Section V-A.

VI. Dimensioning guidelines

Proper dimensioning of the token rate matrix $\boldsymbol{R}$ and the token bucket size matrix $\boldsymbol{BS}$ is vital to obtain the desired properties of the bandwidth profile. We consider an access-aggregation scenario with N nodes with identical MTS-BWP configuration over a bottleneck link with capacity C. The required properties are the following:

1) Provide the (decreasing) bandwidth targets $BW_1, BW_2, \ldots, BW_{N_{fs}}$ for files of sizes $fs_1, fs_2, \ldots, fs_{N_{fs}}$ (increasing) for a node with good history, while all other nodes are either inactive or have bad history.
2) Provide the nominal bandwidth $S_n = C/N$ to each node in long-term average.
3) Provide the minimum guaranteed bandwidth $G_1, G_2, \ldots, G_{N_{TS}}$ (decreasing) in case of $TL^n_{11} = 0, TL^n_{12} = 0, \ldots, TL^n_{1,N_{TS}} = 0$ respectively.
4) Guarantee work conserving property (i.e., when there is traffic, the full link capacity shall be used).

BW_1 is also the peak bandwidth provided to a node after an inactive period.

In the following analysis, we focus on the $N_{DP} = 4$ and $N_{TS} = 4$ case, which allows for $N_{fs} = N_{TS} - 1 = 3$ file sizes with predefined bandwidth targets. Generalizing for $N_{TS} > 4$ is straightforward. We aim to minimize N_{DP} and will settle at $N_{DP} = 4$, providing insight into how the 4 DPs are used as well as what happens for fewer DPs.

A. Token rate matrix $\boldsymbol{R}$

We present a simple dimensioning method for a 4×4 $\boldsymbol{R}$ matrix based on requirements 1)–4) above.

We use the following intuitive guidelines for $\boldsymbol{R}$:

- DP 1 is used for the guaranteed bandwidths $G_1, G_2, \ldots$ and not intended to be the limiting DP;
- DP 1 and 2 are used to ensure that requirement 1) is satisfied for a low load node
- DP 3 or 4 is the congestion DP for high load nodes while low load nodes are inactive;
- DP 4 is used to guarantee the work conserving property.

In accordance with these guidelines, we propose the structure

$$\boldsymbol{R} = \begin{bmatrix} G_1 & G_2 & G_3 & G_4 \\ BW_1 - G_1 & BW_2 - G_2 & BW_3 - G_3 & \frac{C - BW_1}{N-1} - G_4 \\ C & C & S_n - \frac{C - BW_1}{N-1} & S_n - \frac{C - BW_1}{N-1} \\ C & C & C & C \end{bmatrix} \qquad (4)$$

The first row (DP 1) is straightforward and simply implements the guaranteed bandwidths. To avoid congestion on DP 1, $G_{ts} \leq S_n$ needs to hold for all ts.

$R_{2,1}$ is calculated so that $R_{1,1} + R_{2,1} = BW_1$ to ensure the predefined bandwidth target BW_1 on DP 1 and 2. Similarly, $R_{2,2} = BW_2 - R_{2,1}$ and $R_{2,3} = BW_3 - R_{3,1}$.

Element $R_{3,4}$ is defined so that

$$R_{1,4} + R_{2,4} + R_{3,4} = S_n \qquad (5)$$

holds. This important property will be called the *return rule*. If $R_{1,4} + R_{2,4} + R_{3,4} \leq S_n$, then any node n with nominal load larger than 1 will continue to deplete its token buckets and eventually end up with token levels

$$TL^n_{1,N_{TS}} = TL^n_{2,N_{TS}} = TL^n_{3,N_{TS}} = 0. \qquad (6)$$

Actually, (6) is exactly the way bad history is described within the system.

The return rule in (5) provides two important guarantees: in long-term average, only bandwidth S_n is guaranteed *on DP1–DP3* for any node n, but, since $S_n = C/N$, this also means that no node will be "suppressed" in long-term average by the other nodes. Also, if all other nodes are either inactive or have bad history (as in (6)), any node n with nominal load less than 1 will eventually have $TL^n_{3,4} > 0$, and thus potentially have access to a bandwidth larger than S_m (the node returns from "bad history" to "good history", hence the name of the rule). The general form of the return rule is that there exists a dp_r such that $\sum_{dp=1}^{dp_r} R_{dp,N_{TS}} = S_n$.

Next up is $R_{2,4}$, which is defined so that

$$(N-1)(R_{1,4} + R_{2,4}) + BW_1 = C. \qquad (7)$$

This will ensure that when a single node becomes active while all other nodes have bad histories, as in (6), the congestion DP will change to 2, with the single active node having throughput BW_1 and other nodes having throughput $(R_{1,4} + R_{2,4})$.

The last row guarantees the work conserving property: as long as at least one node is active, it has access to the entire capacity C.

The impact of the elements $R_{3,1}, R_{3,2}$, and $R_{3,3}$ is relatively minor in most scenarios; their setting in (4) is selected to avoid extra limitations compared to $R_{3,4}$. The resulting $\boldsymbol{R}$ should conform with (1). If that is not the case for a given set of parameters $\boldsymbol{R}$ has to be adjusted accordingly.

The BWP can be extended (simplified) to have more (or fewer) timescales to accommodate more (or fewer) predefined file sizes and bandwidth targets. In case of fewer DPs, we have to take into account that:

- omitting the first row of $\boldsymbol{R}$ results in no strictly guaranteed bandwidths;
- omitting the second row removes the predefined bandwidth targets, resulting in a system very similar to trTCM, with nearly no memory;
- omitting the third row violates the return rule;
- omitting the last row results in a non-work-conserving system, where the whole system capacity may not be used at all time (e.g., when there is only one active node).

Example 1: For the parameters $N = 5$, $C = 10$ (Gbps), guaranteed bandwidths $G_1 = G_2 = G_3 = 2, G_4 = 0.75$ (Gbps), file sizes are $fs_1 = 0.1, fs_2 = 1, fs_3 = 11.25$ (GByte). The bandwidth targets $BW_1 = 6, BW_2 = 4, BW_3 = 3$ (Gbps), the following 4×4 matrix is suitable:

$$\boldsymbol{R} = \begin{bmatrix} 2 & 2 & 2 & 0.75 \\ 4 & 2 & 1 & 0.25 \\ 10 & 10 & 1 & 1 \\ 10 & 10 & 10 & 10 \end{bmatrix} \tag{8}$$

B. Bucket size matrix $\boldsymbol{BS}$

The sizes of the buckets are calculated from the rates in R and the list of timescales TS, which we define as

$$TS = [0, fs_1/BW_1, fs_2/BW_2, fs_3/BW_3], \tag{9}$$

so, e.g., in Example 1, $TS = [0,\ 0.133,\ 2,\ 30]$ (in sec).

$TS_1 = 0$ represents the ideal behavior of the fluid model. The remaining timescales correspond to transmission times of the predefined file sizes. We use the last timescale (TS_4) to define how long a node must send with at least S_n throughput to be considered to have bad history. In Example 1, we set $TS_4 = 30$ sec (to allow a 30 second active period, before a node is considered to have bad history) and calculate fs_3 accordingly.

We set the bucket sizes according to the formula

$$BS_{dp,ts} = \tag{10}$$
$$\begin{cases} 0 & \text{for } ts = 1 \\ TS_2(R_{dp,1} - R_{dp,2}) & \text{for } ts = 2 \\ \sum_{k=2}^{ts}(TS_k - TS_{k-1})(R_{dp,k-1} - R_{dp,ts}) & \text{for } ts > 2 \end{cases}$$

which will result in a previously inactive node emptying bucket $\mathrm{BU}_{dp,ts}$ after time TS_{ts} (assuming the rate at DP dp is limited only by the node's own history and not by other nodes), taking into account that it has different throughput on different timescales. Buckets with $BS_{dp,ts} = 0$ act as rate limiters in the fluid model, due to (2).

Setup	N_{low}	N_{high}	low load	system load
A	1	4	0.5	0.6, 0.8, 1.0, 2.0
B	2	3	0.5	0.6, 0.8, 1.0, 2.0
C	3	2	0.5	0.6, 0.8, 1.0, 2.0
D	4	1	0.5	0.6, 0.8, 1.0, 2.0
E	1	4	0.2, 0.4, 0.6, 0.8	1.1
F	2	3	0.2, 0.4, 0.6, 0.8	1.1
G	3	2	0.2, 0.4, 0.6, 0.8	1.1
H	4	1	0.2, 0.4, 0.6, 0.8	1.1

TABLE I
SIMULATION SETUPS AND PARAMETERS

When using the above $\boldsymbol{R}$ and $\boldsymbol{BS}$ dimensioning method, the flow throughput of a single flow of size fs_2 can reach as high as

$$BW_2' = \frac{TS_2 \cdot BW_1 + (TS_3 - TS_2) \cdot BW_2}{TS_3} \geq BW_2. \tag{11}$$

If one wants to replace the current *maintainable throughput requirement* to a *flow throughput requirement* for fs_2, BW_2 in (4) should be replaced by the (slightly smaller) solution of (11) for BW_2 when setting the left-hand side equal to the *flow bandwidth requirement*. For Example 1, and for the anticipated meaningful input values, the difference between BW_2 and BW_2' is very small; specifically, $BW_2' = 4.1333$ (Gbps).

The above calculations are for the fluid model. For actual packet-based networks, bucket sizes $BS^*_{dp,ts}$ must have a minimum: at least MTU (maximum transmission unit) to be able to pass packets, furthermore they must also allow bursts on the RTT timescale. In summary, $BS^*_{dp,ts} = \max(BS_{dp,ts},\ MTU,\ R_{dp,ts} \cdot RTT)$.

VII. Simulation

Simulations are carried out by a dedicated discrete event simulator implemented in Julia [14]. In all simulations, the MTS-BWP rates ($\boldsymbol{R}$), bucket sizes ($\boldsymbol{BS}$), and the system parameters are set according to Example 1. In the input process, we use file sizes fs_1 and fs_2 from Example 1 with identical 50% probability. The flow limit is $f^{\max} = 20$ for each node.

We have two groups of nodes with identical nominal loads within a group. We specify the nominal load for low load nodes (*low load*) and the *system load*, and calculate the nominal load for high load nodes using the equations in Section V-B2. The simulation setups are summarized in Table I, with the number and load of each node type varying for a total of 8×4 actual setups, denoted A–H.

A. Time-series simulation example

Fig. 4 depicts the evolution of the bandwidth allocation in a time interval for setup A with a system load of 1.0. Colors correspond to nodes and shades within a color correspond to DPs. Node 1 (red) is the low load node. Some events are also marked (a)–(g).

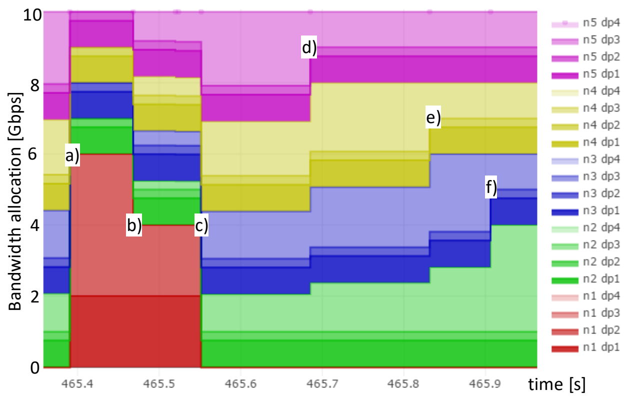

Fig. 4. Example of bandwidth allocation over time

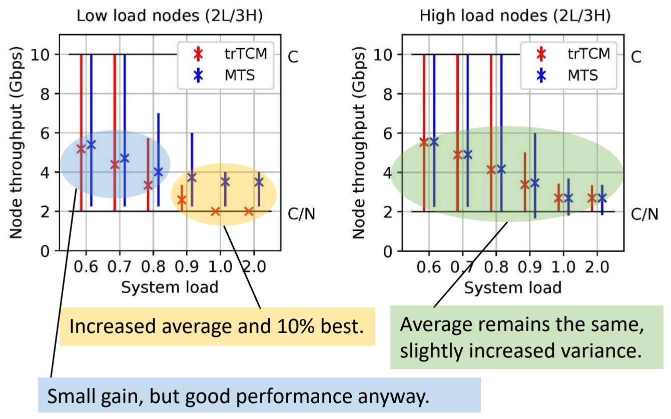

Fig. 5. Gain for MTS over trTCM

Node 1 is inactive in the beginning, and the congestion DP is 3. Then a flow starts in node 1 (a) and the congestion DP changes to 2. Node 1 starts using 2 Gbps ($R_{1,1}$) + 4 Gbps ($R_{2,1}$) of the available capacity on DP 1 and 2 respectively, while nodes 2–5 start using 0.25 Gbps ($R_{1,4}$) + 0.75 Gbps ($R_{2,4}$) respectively. Eq. (7) ensures that all traffic on DP 2 can be transmitted for this case.

As time progresses, the buckets $\mathrm{BU}^1_{2,2}$ and $\mathrm{BU}^1_{1,2}$ become empty (b), and the bandwidth share of node 1 drops accordingly to 2 Gbps ($R_{1,2}$) + 2 Gbps ($R_{2,2}$). The congestion DP switches back to 3, but DP 3 is dominated by nodes 2–5, because those nodes have high numbers of flows, while node 1 has a single flow only. That single flow can still achieve 4 Gbps (BW_2) as dimensioned.

Once node 1 finishes its flow (c), the available bandwidth is reallocated to nodes 2–5 on DP 3. Buckets which were filled previously (specifically $\mathrm{BU}^2_{3,4}, \ldots, \mathrm{BU}^5_{3,4}$) empty one by one, and their bandwidth shares on DP 3 drop accordingly: first for node 5 (d), then node 4 (e), and finally node 3 (f). The exact order depends on the bucket levels of $\mathrm{BU}^2_{3,4}, \ldots, \mathrm{BU}^5_{3,4}$, which depend on their earlier history, not visible in the depicted time interval.

B. Analysis of different traffic scenarios

Based on the simulator output, we calculate the following two statistics: the node throughput for active periods (periods when there is no traffic at the respective node are excluded); and the flow throughput for the different flow sizes, which is the flow size divided by the transmission time. The figures depict the average with a $\times$ symbol and the 10% best – 10% worst interval with bars.

We compare the suggested MTS-BWP versus the trTCM profile of CIR$= S_n = 2$ Gbps and EIR$= C - S_n = 8$ Gbps as a baseline for various setups.

Fig. 5 is a takeout where the advantages of the proposed MTS-BWP are visualized assuming two low load nodes (with 0.5 nominal load) and three high load nodes (whose load is determined by the indicated system load). The left plot depicts the throughput of the low load nodes, which is similar for both profiles for system load less than 0.8, and significantly improved for higher system loads. The right plot depicts the effect of the BWPs on the high load nodes and shows that the increased throughput of the low load nodes does not cause significant throughput degradation for MTS-BWP compared to the trTCM profile. This is because traffic from low load nodes is indeed served faster, but the total amount of traffic served from low load nodes is the same.

Fig. 6 (setups A–D, with fixed low load 0.5 and varying system load) and 7 (setups E–H, with fixed system load 1.1 and varying low load) compare the node throughput of low load nodes for trTCM and MTS. MTS consistently outperforms trTCM, i.e., it allocates more bandwidth to low load nodes. Additionally, Fig. 8 and 9 show that the associated effect on the high load node throughput is minimal. These figures provide the same conclusion as Fig. 5 in many different scenarios.

The difference in the performance of low load nodes between MTS and trTCM BWPs increases with the total number of low load nodes in the scenario. This is because typically one low load node competes with all high load nodes, and having fewer high load nodes results in higher throughput for the low load node.

As the load of the low load node approaches 1, the difference between trTCM and MTS gradually disappears, because the difference between the low and high load nodes vanishes.

Next we examine the prioritization of small flows (fs_1) compared to large flows (fs_2) provided by MTS-BWP compared to trTCM BWP. Fig. 10 and 11 show flow throughput statistics for small flows in low load nodes. MTS outperforms trTCM, i.e., it allocates more bandwidth in these cases for every setup, but particularly for overloaded systems, where the difference is huge, both for average and also for best 10% values. For MTS BWP, the best 10% values for small flows reach BW_1 for all scenarios. Again, as the low load is approaching 1, the difference between trTCM and MTS decreases (similar to the node throughput).

Fig. 12 and 13 display the same statistics for large flows (1 GB) at low load nodes. Again, MTS outperforms trTCM significantly. The best 10% throughput matches BW'_2 for most scenarios, and in many scenarios, even the average

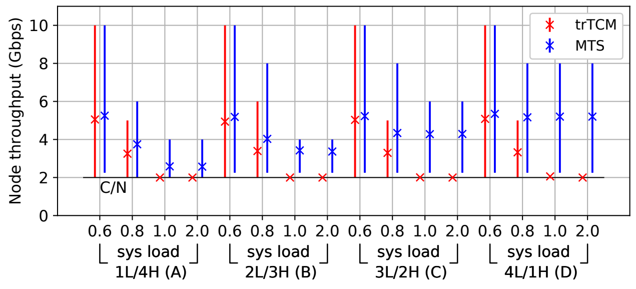

Fig. 6. Node throughput for low load nodes; low load = 0.5

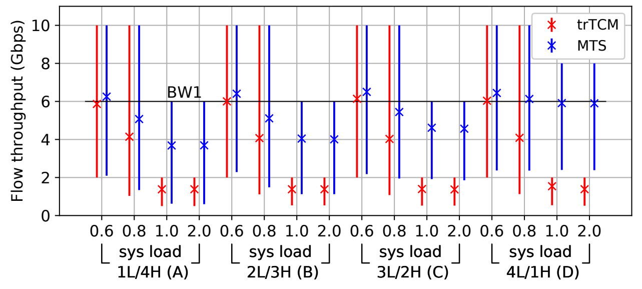

Fig. 10. Performance of small flows at low load nodes; low load = 0.5

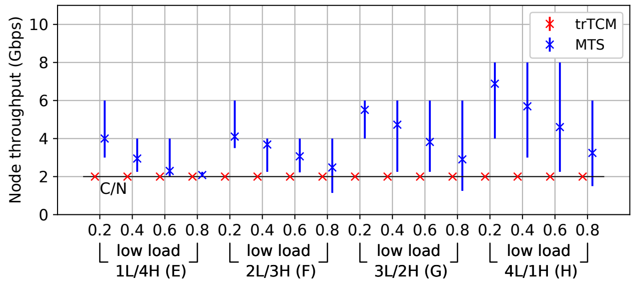

Fig. 7. Node throughput for low load nodes; system load = 1.1

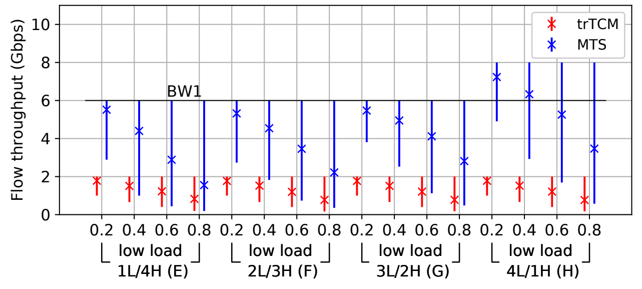

Fig. 11. Performance of small flows at low load nodes; system load = 1.1

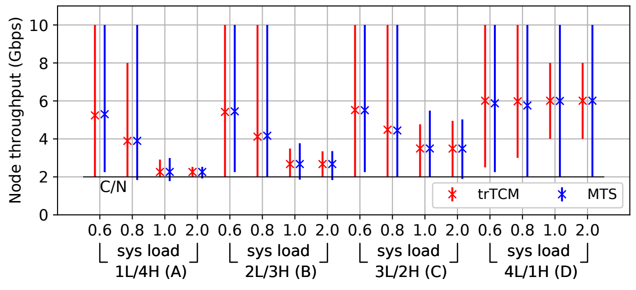

Fig. 8. Node throughput for high load nodes; low load = 0.5

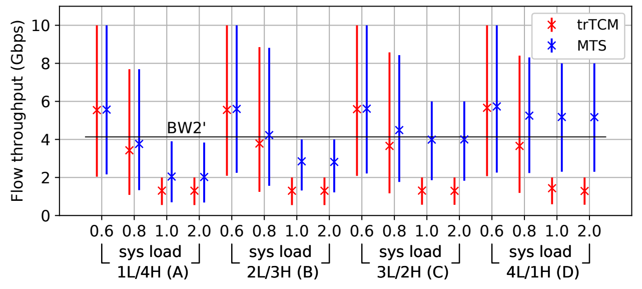

Fig. 12. Performance of large flows at low load nodes; low load = 0.5

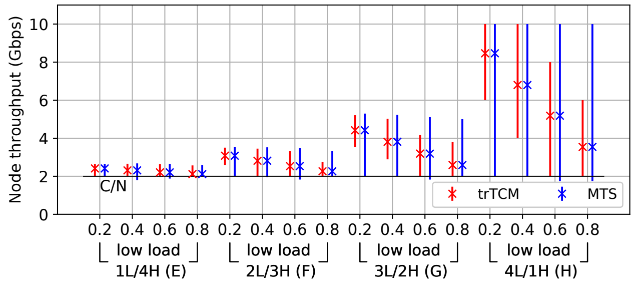

Fig. 9. Node throughput for high load nodes; system load = 1.1

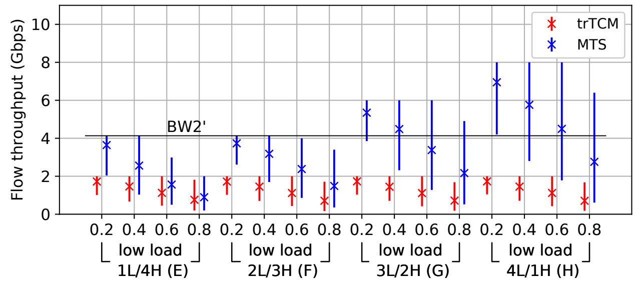

Fig. 13. Performance of large flows at low load nodes; system load = 1.1

throughput reaches BW_2', which is close to the dimensioned BW_2 (see Section VI-A).

One more possible approach for a comparison of MTS and trTCM BWPs is the introduction of the *experienced system load*, that is, the system load as perceived by low load nodes. The idea behind the concept is that low load nodes in the MTS BWP environment perceive the system underloaded even if the entire system is overloaded. We define the experienced system load as follows: for a low load node in an overloaded system with MTS policy, we select the closest comparison from among *underloaded* systems with trTCM policy based on the average node throughput of low load nodes. The result of the comparison is displayed in Fig. 14; for example, the rightmost blue dot in Fig. 14 can be interpreted as follows: a low load node in an overloaded system (system load 2.0) with four low load nodes (load 0.5) and one high load node with MTS policy experiences approximately the same average throughput as a low load node in a system with system load 0.6 (the blue dot) with four low load nodes (load 0.5) and one high load node with trTCM policy.

The blue (load 0.5) and especially the green dots (load 0.2) are all considerably below 1.0, meaning that the experience of the low load nodes under MTS policy is indeed comparable to their experience in an underloaded system with trTCM policy.

VIII. Conclusion

We proposed a novel Bandwidth Profile to provide fairness on multiple timescales. We illustrate its unique advantages and its motivation in 5G transport network. We created

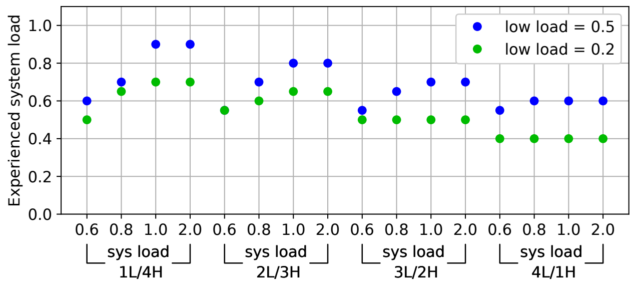

Fig. 14. Experienced system load for low load nodes

a dimensioning method for the BWP and investigated its properties using a fluid level simulator. The results confirmed that the dimensioning targets are met: the throughput of traffic sources with good history can approach the total access bandwidth, while the throughput of traffic sources with bad history does not decrease significantly compared to bandwidth sharing according to the widely-used trTCM BWP.

References

[1] Ericsson AB, “5G Radio Access-Research and Vision,” *Ericsson White Paper 284 23-3204 Uen, Rev C*, April 2016.

[2] 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.

[3] N. Alliance, “Description of network slicing concept,” Tech. Rep., January 2016.

[4] “EVC ethernet services definitions phase 3,” Aug. 2014, Metro Ethernet Forum 6.2.

[5] G. Abbas, Z. Halim, and Z. H. Abbas, “Fairness-driven queue management: A survey and taxonomy,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 324–367, 2016.

[6] “Technical specification mef 41 - generic token bucket algorithm,” 2013. [Online]. Available: https://www.mef.net/Assets/Technical_Specifications/PDF/MEF_41.pdf

[7] J. L. Valenzuela, A. Monleon, I. San Esteban, M. Portoles, and O. Sallent, “A hierarchical token bucket algorithm to enhance QoS in IEEE 802.11: proposal, implementation and evaluation,” in *IEEE Vehicular Technology Conference*, vol. 4, 2004, pp. 2659–2662.

[8] S. Nadas, Z. R. Turanyi, and S. Racz, “Per packet value: A practical concept for network resource sharing,” in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec 2016, pp. 1–7.

[9] S. Nádas, G. Gombos, P. Hudoba, and S. Laki, “Towards a Congestion Control-Independent Core-Stateless AQM,” in *ANRW ’18*, 2018, pp. 84–90.

[10] 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.

[11] S. Laki, G. Gombos, P. Hudoba, S. Nádas, Z. Kiss, G. Pongrácz, and C. Keszei, “Scalable Per Subscriber QoS with Core-Stateless Scheduling,” in *ACM SIGCOMM Industrial Demos*, 2018.

[12] F. Fejes, S. Laki, G. Gombos, S. Nádas, and Z. Kiss, “Decoupling delay and resource sharing targets with efficient core-stateless aqm,” in *Proceedings of the ACM SIGCOMM 2019 Conference Posters and Demos*. ACM, 2019, pp. 128–130.

[13] F. Baker and R. Pan, “On Queuing, Marking, and Dropping,” RFC 7806, Apr. 2016. [Online]. Available: https://rfc-editor.org/rfc/rfc7806.txt

[14] “The Julia Programming Language,” https://julialang.org/.