Weighted Fair Resource Sharing Without Queuing Delay

Benedek Kovács
Budapest University of Technology and Economics
H-1111, Egry József u. 1
Budapest, Hungary
kovacsbe@math.bme.hu

***Abstract*—This paper presents a new method that provides weighted fair sharing without queues and delay while outperforming traditional traffic throttling mechanisms. A mathematical description and model is presented to justify the findings and to provide better knowledge about traffic throttling characteristics. Simulation, numerical results and statistical discussion are also presented to underpin the findings especially where exact mathematical results are not or just partly available.**

***Keywords*-Network management; Service level agreement; Overload control; Traffic shaping; Fair sharing; Maximal throughput**

I. INTRODUCTION

There are many overload and congestion control and load sharing problems to be solved in telecommunication networks of various kinds e.g., in the Internet Multimedia Subsystem. Considering any type of network and signalling protocol a protocol operation flow consists of *messages*. The network nodes are entities receiving these messages and they process them using their resources such as CPU capacity or memory. If they lack the resource to process an offered message then the node is *overloaded*. Similarly, in packet switched networks, congestion control and fair resource (link bandwidth) sharing aims to keep the system utilizing its resources at its rated capacity while providing satisfactory service for the users and quality of service for service classes.

Many overload control mechanisms are developed to avoid overload and congestion situations. In some of them the target node (the critical resource) itself can deny to serve the request/forward a packet or sometimes reject/drop (send a negative reply message using minimal resources/ignore) it (see Figure 2). Another solution is that the sender (*source*) does not send out the message if it knows for some reason that the *target* will not be able to serve it. This information can be hard coded or can be gained from measurements by the source itself or gathered via special messages from peers (see Figure 1). In all cases i.e. in all overload control mechanisms there is a decision point and logic that decides to reject or admit/send out the request. This entity is called the *throttle* which is in the center of our interest. We suppose that the control information and the requirements on the behavior are available and up to date for the throttle. (Such information is to be gained with measurement or with some closed/open loop control mechanism. It has a large literature and is out of the scope of this paper to examine the characteristics of such mechanisms.) The throttle itself can be physically located in the target as Figure 2 shows, in the source or might also be distributed in multiple sources as Figure 1 shows.

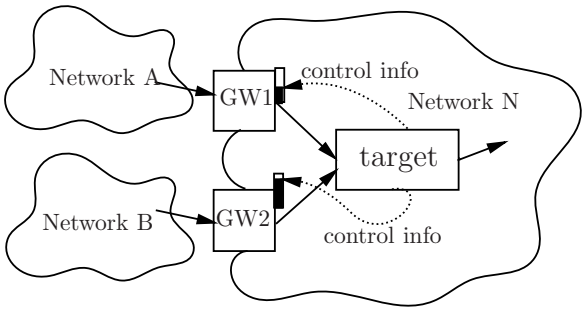

Fig. 1. A scenario where two operators A and B share a common core network N. The Gateways associated to each network implement distributed throttling to prevent the central nodes (target) to get overloaded. The central node—or any overload control agent— distributes control information to the Gateway Nodes to ensure fair service of requests from networks A and B while keeping maximum utilization of Network N.

The *throttle* entity discussed here is one very important and well defined part of overload control systems of any type as it has the role to reject (or drop) an *offer* or to let it go through i.e., admit it. When a *throttle* realizes a call gapping mechanism it makes the decision based only on previous offers thus no offers in the future are examined. (Call gapping means that calls are not admitted for a given period of time based on some measurements and collected information.) In our case the non-anticipative *throttle* is not allowed to delay an *offer* and only one *offer* arrives at a time i.e. the call gapping mechanism cannot buffer the *offer* and *admit* it later than it has arrived. This makes a fundamental difference from Weighted Fair Queuing and mechanisms like those in [4], [3]. Weighted fair resource sharing with no delay and maximum utilization is the main advantage of the method introduced and discussed in this paper.

For the sake of clarity, definition of terms and word usage is presented here:

 ISBN: 978-1-61208-133-5

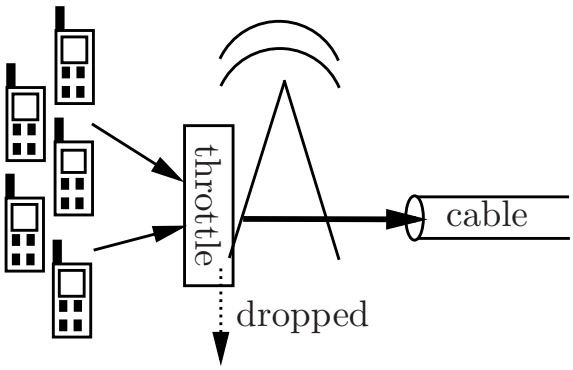

Fig. 2. The throttle is implemented in the Base Station and applied to avoid congestion on links when multiple users are sharing media in a small cell. Users not sharing media are also given a minimum share to be able to communicate. The throttle at the base station should select wisely which packet to be dropped and which to forward.

- The *throttle* decision function is a function mapping from the offer load point process to the set {*admission*, *rejection*}. (Each *throttle* is uniquely assigned a function γ that transforms the intensity process $\rho(t)$ of the income process to an intensity process of the admissions $\gamma(\rho(t)) = a(t)$. [9])
- An *offer* is the event for which the *throttle* has to decide on *admission* or *rejection*. If an offer is admitted it cannot be rejected (dropped) and vice versa, and there is no third possibility. An offer has properties: *arrival time, priority level and class* that can be measured.
- The *traffic class* and the *priority level* sets have finite elements.
- The *offered traffic* (or *offer load*) is the flow of *offers* modeled with a progressively measurable not necessarily stationary point process marked with the marks from the mark space that is the direct product of the set of priorities and classes. (This implies that the probability of two *offer* events occurring at the same time is zero.)
- The *admitted traffic* (or *throughput*) is the flow (i.e. the point process) of *admitted offers* (*offers* for which the *throttle* yields admission). The flow of admitted offers can be conditioned upon the whole history (past) of the offer load flow and upon the throttle parameters and undoubtedly on the decision strategy.

The above assumptions and definitions are natural and obvious and also necessary to make the discussion clear.

The typical verbal definitions of the requirements we investigate the different throttles against are given here preliminary. They are not precise and many contradict and can have multiple exact (i.e. mathematical) definitions with different results. In this paper we give exact definitions of these requirements:

- **Requirement-A** *Maximal throughput with bound*: No *offer* should be rejected if there is enough available capacity in the system to serve it, but no *offer* should be admitted if there is not enough available capacity to serve it in the system.
- **Requirement-B** *Priority levels*: Each *offer* may be assigned a priority level and the *offer* with higher priority shall be admitted in favor of the one with the lower priority level.
- **Requirement-C** *Throughput share for traffic classes*: The offers can be classified and for the traffic class i the s_i portion of the capacity of the target shall be provided.

The three requirements are associated with three behavior aspects. We propose a so-called *rate based call gapping* mechanism and compare it with Token Bucket [7] against **Requirement-A** and **Requirement-B**.

In Section II, we introduce traffic throttling mechanisms, namely the Token Bucket and the rate based call gapping. Then in Section III, the new call gapping mechanism is proposed and it is shown how it meets the requirements. We also show how it can be extended with the original Token Bucket concept, achieving similar throughput characteristics to the original algorithm while keeping the good properties of the introduced one. In Section IV we present our simulations and some figures about the offer and admission traffic flows with the three mechanisms. Using statistics we show how each mechanism meets **Requirement-B**.

II. Classical requirements in a queue-less environment

A. Traffic throttling, priority handling and weighted fair sharing

Many call gapping mechanisms have been developed for different purposes with different characteristics. One of the most important call gapping algorithms is the Crawford algorithm [5] but one of the most widely implemented solutions is Token Bucket call gapping mechanism defined in the Overload Control Standard H.248.11 [13].

1) The Token Bucket with parameters (r, W): The Token Bucket call gapping mechanism is the following: there is a bucket of available tokens representing available resources (free capacity) of the system. Requests are offered to the system and each of them is assigned a number of tokens needed i.e., the amount of resources it requires to be served. Once there are enough tokens in the bucket the request is admitted or dropped. (Thus no queues are applied and no delay is present in the system because of the Token Bucket call gapping algorithm.)

By the definition of the original Token Bucket the tokens are generated into the bucket with exponential distribution and the offers arrive with a Poisson process in most of the models that means, the time interval between the consecutive arrivals is also exponentially distributed. We analyze and describe such a variant.

Firstly we mention that decision about serving a request are often implemented differently. The most important difference is in the interpretation: rather than consuming the tokens the bucket fill b is increased when a request arrives. The token generation is then realized with decreasing the bucket fill. The maximum fill is the watermark W that cannot be exceeded and

 ISBN: 978-1-61208-133-5

also the bucket fill can not be lower than 0. This concept is equivalent to the original algorithm.

Secondly we consider deterministic token generation instead of the exponential one that is used in most cases (e.g., [13]), because it is much easier to implement and sometimes to analyze, as well.

Then the Token Bucket mechanism we discuss works as follows: When a new request arrives at t_n than the needed bucket fill is calculated: $b(t_n)$ as if the request was served. This is done with calculating the expected number of tokens that would have been generated from the time when the former service was served (t_{n-1}) then multiply it with the throughput capacity of the bucket i.e., the Token Bucket rate at t_n: $r(t_n)$ and subtract it from the former bucket size at $b(t_{n-1})$. Then it tests it against the preset constant watermark: W.

Definition 1: Token Bucket call gapping strategy $\gamma_t(r, W)$.

$$b(t_n) = \max\{\chi(t), b(t_{n-1}) - r(t_{n-1})(t_n - t_{n-1}) + \chi(t)\}, \quad (1)$$

where $\chi(t) = 1$ iff there is an offer. Admit if $b(t_n) \leq W$. If the offer is admitted, the above definition is used for the next value of the bucket fill b. If the offer is rejected, then $b(t_n)$ is recalculated with $\chi(t) = 0$.

In many solutions the offers for the bucket can be of different types with different resource needs thus $S_i\chi_i(t)$ is used for update, where S_i is the so-called "splash amount" i.e., the expected number of tokens needed to serve the request of type i. From now on we suppose that $S_i = 1$, since the calculations would be much more difficult without any qualitatively different result with respect to the requirements we consider now.

2) Rate based call gapping: One obvious solution to regulate traffic is to limit the admitted offers in a given time frame. The Crawford algorithm worked as follows: for each time period T when the number of offers reached a certain number c, the capacity of the target, no other offers are admitted in the given period. This introduced bursts in traffic thus not preferred in most cases.

The idea and our proposal is to maintain an estimate of the traffic intensity $\hat{\rho}$ updated at each t_n an offer arrives. The provisional admission rate estimate $\hat{a}(t_n)$ is also calculated and then compared to some goal function $g(t_n)$, say the actual maximal throughput rate $c(t_n)$. Iff $\hat{a}(t_n) \leq g(t_n)$ then the offer is admitted.

The essence of such a mechanism lies in the way how the estimates are calculated and how $g(t_n)$ is determined. The first statistics that can be used and examined as an estimate for the intensity of a point process is the simplest first moment type estimate:

Definition 2 (Periodic intensity estimate):

$$\bar{\lambda}(t_n; T) := \begin{cases} \dfrac{N(t_n) - N(t_n - T)}{T} & \text{if } t_n \geq T \\ \dfrac{N(t_n)}{t_n} & \text{if } t_n < T \\ \bar{\lambda}(0; T) := 0 & \end{cases} \quad (2)$$

To calculate the value of the periodic intensity estimate one has to maintain all $t < t_i \leq t - T$ event times and it is often too resource consuming thus not feasible in real time systems. An adaptive estimate is often preferred and introduced by patent [14] to control network traffic. The definition is the following:

Definition 3 (Dynamic intensity estimate):

$$\begin{aligned} \hat{\lambda}(t_n; T) &:= \max\{\tfrac{1}{T}, \tfrac{T\hat{\lambda}(t_{n-1};T) - (t_n - t_{n-1})\hat{\lambda}(t_{n-1};T) + 1}{T}\}, \\ \hat{\lambda}(t_0; T) &:= 0. \end{aligned} \quad (3)$$

We do not want to go into details of intensity estimation of inhomogeneous point processes in this paper.

B. Priority handling with Token Bucket and Rate-based call gapping

At first we would like to clarify that once the offered traffic is modeled with a point process and the throttle meets **Requirement-A** we cannot provide priority between the offers. Why? Suppose that we have an offer in the system and we have to decide if we should admit it or not. **Requirement-A** tells us to admit the offer if we have the capacity to serve it. Suppose that this is the case and see that if the throttle would not admit the current offer to reserve this capacity for offers of higher priority then it might happen that there will be no higher priority offer in the future and the throttle would suffer a loss of workload.

1) Priority handling with Token Bucket: However, giving up the maximal throughput requirement with turning it into an efficient-enough requirement, some priority handling naturally can be done. In the Token Bucket concept different watermarks are assigned to each priority level. The offers of lower priority are checked with a lower watermark. This means reserving a set of tokens (system resources) to the higher priority traffic. This method violates **Requirement-A** whenever $b(t)$ declines to 0 before rejecting an offer. Whenever this event has a low probability, using different watermarks for different priority levels is a good solution to meet **Requirement-B** with a Token Bucket throttle.

This is a strict priority handling in the sense that when possible, resources are always reserved for higher priority offers. The probability of rejecting an offer on a given priority level is given by a formula which is, together with other regulation properties, discussed in [10].

2) Priority handling with rate based call gapping only: For rate based call gapping mechanisms the above solution cannot be applied but less strict priority handling can be achieved using the dynamic intensity estimate 3 with different settings of parameter T.

The dynamic estimate is asymptotically unbiased with low variance proving its usability and also, the dynamic one often performs better than the most commonly used periodic estimate 2. Such discussion is pure mathematics thus out of the scope of this article.

It can be also proved that the expected bias of the dynamic estimate 3 is lower than the real intensity and this property is

 ISBN: 978-1-61208-133-5

monotonous in T. More exactly, at each t_n an offer arrives if $T_1 < T_2 < \infty$ then $\mathbf{E}[\hat{\lambda}(t_n;T_1)] < \mathbf{E}[\hat{\lambda}(t_n;T_2)] < \mathbf{E}[\hat{\lambda}(t_n;\infty)] = \lambda(t_n)$. The load is thus slightly underestimated, but the more likely the load is underestimated the more likely the offer is admitted i.e., higher priority offers should have lower T parameter and **Requirement-A** is met.

C. Weighted fair sharing

Fair sharing is about the differentiation between offers on the same priority level. (Note that traffic classes are not the same as priority levels.) Suppose a scenario like Figure 1 where the resources of Network N should be equally split between the two contractors while emergency (priority) calls still have to be served. In our other example minimal share for different data types should be achieved on limited bandwidth while it is allowed that there are some priority e.g., time critical packets such as voice, video, media, etc.

The solution in an environment without queues proposed in this paper is rather straightforward: one should measure the incoming rates for each class and check each offer against online calculated goal levels. The goal throughput rates can be adjusted dynamically according to the weighted fair sharing agreements and also according to the maximal throughput requirements.

The problem with the solution using Token Bucket only is that it does not measure the incoming traffic rates. (Using the bucket fill instead of measurement gives false result.) However, using any kind of rate based call gapping method the problem is solved. One can also measure the incoming rates for each class with the estimates presented above and set the bucket fill-up rate r according to the desired goal levels but this is already a combination of the methods. A solution for the network level case is proposed in the GOCAP standard [11].

III. Weighted fair sharing with no delay

In this Section our new method, the proposed fair sharing method with no delay is presented. At first we introduce the complete proposed procedure clearly. Then we discuss and prove how it satisfies all the requirements and what possible extensions, modifications or other solutions might result in a similar good algorithm. At the end of the discussion we present the relationship between the new method and the original Token Bucket algorithm.

A. The new call gapping algorithm $\gamma_g(c,T,g,s)$

Suppose that the consecutive offers arrive to the *throttle* at $... < t_{n-1} < t_n < t_{n+1} < ...$ time instants respectively. Each *offer* has a well defined *priority level* $j, j \in 1..J$ and *traffic class* $i \in 1..I$. Each *priority level* j has a priority parameter T_j assigned ($T_j \geq T_k$) if the *offer* with priority k has the higher priority and each *traffic class* has a pre-configured weight $i \mapsto s_i \in (0,1) \subset R$, (where $\sum s_i = 1$). For each i the algorithm maintains an estimation of the incoming *offer rate* $\hat{\rho}_i(t)$, a *provisional admission* rate $\hat{\alpha}_i(t)$ from which it calculates a *bounding rate* $g_i(t)$ and then according to the decision it estimates an *admission rate* $\hat{a}_i(t)$.

We suppose that the rate of the throttle varies with the following function: $c(t)$. (This value is determined and given for the algorithm and represents the capacity of the *throttle* and might be different from $r(t)$).

Definition 4: Define the proposed throttle decision strategy γ_g in the following way. Let us suppose that at t_n an *offer* arrives and the system is in state $\{t_{n-1}, \hat{\rho}_i(t_{n-1}), \hat{a}_i(t_{n-1})\}$ and $c(t_n)$:

1) Determine priority constants i.e., calculate T_j;
2) Update the incoming rates estimated for all i classes: $\hat{\rho}(t_n)$ with $\chi_k(t_n) = 1$ iff $i = k$, 0 otherwise;
3) Calculate a provisional admission rate for all i: $\hat{a}(t_n)$ with $\chi_k(t_n) = 1$ iff $i = k$, 0 otherwise;
4) Calculate the bounding rate for class i only: $g_i(t_n)$;
5) If $\hat{\alpha}_i \leq g_i$ then *admit* the offer and $a(t_n) := \alpha(t_n)$ else *reject* the *offer* and update $\hat{\alpha}(t_n)$ with $\chi_k(t) = 0, \forall k(!)$;
6) (Continue with 1. for the next event).

We propose to update $\hat{\rho}_i, \hat{\alpha}_i, \hat{a}_i$ according to the following equation:

$$\hat{\lambda}(t_n) := \frac{\chi(t_n)}{T_j} + \max\{0, \frac{T_j\hat{\lambda}(t_{n-1}) - (t_n - t_{n-1})\hat{\lambda}(t_{n-1})}{T_j}\}, \tag{4}$$

where $\hat{\lambda}$ is an estimator asymptotically unbiased for the $\lambda(t)$ real intensity of a point process thus to be replaced by $\hat{\rho}_i, \hat{\alpha}_i, \hat{a}_i$ and indicator $\chi_i(t_{n-1}) = 1$ iff the *offer* is of type i and 0 otherwise (or further specified like in step 5). Note that the time parameter T_j changes in time as well according to the priority level and that the former always has to be remembered.

To calculate the bound rate we introduce $u(t)$ the provisional used capacity according to **Requirement-B**:

$$\begin{aligned} u(t) &:= \sum_{\forall i} \min\{s_i c(t), \hat{\rho}_i(t)\} \\ &= \sum_{\hat{\rho}_i(t) \leq s_i c(t)} \hat{\rho}_i(t) + \sum_{s_i c(t) < \hat{\rho}_i(t)} s_i c(t) \end{aligned} \tag{5}$$

Thus the remaining (unused) capacity in the system is $c(t) - u(t)$. This has to be split between traffic classes with higher incoming rate then the agreed share $\hat{\rho}_i(t) > s_i c(t)$. Then

$$g_i(t) := \min\{\hat{\rho}_i(t), s_i c(t) + (\hat{\rho}_i(t) - s_i c(t))\frac{c(t) - u(t)}{\rho - u(t)}\}. \tag{6}$$

It is important to see that our method is capable to handle other class-wise throughput criteria than fair sharing and maximal throughput. Giving upper or lower bounds for g one can implement fairly complex throttle mechanisms.

As one can see the new method is more complex than the original token bucket mechanism. However, the processing cost of updating the few variables introduced is significantly smaller than processing the offers, thus it does not count even in case of overload.

B. γ_g meets all the requirements

Now that the strategy is introduced we prove that it meets all the requirements. We define each requirement mathematically

 ISBN: 978-1-61208-133-5

then we show how they are satisfied. We introduce some notation to make the discussion clear.

- $c(t)$ represents the true capacity of the system expressed in rate, deterministic and coming from an external input source.
- $\rho(t)$ is the real intensity of the offered traffic and $\hat{\rho}(t)$ is its estimate with (4).
- $a(t)$ is the real intensity of the admitted traffic and $\hat{a}(t)$ is the estimation of the intensity with (4).
- $\alpha(t_n)$ is the preliminary admitted traffic intensity for which the following stands: $\alpha(t) = a(t), \forall t < t_n$, and $\alpha(t_n)$ is the intensity $a(t_n)$ would have if the offer was admitted at time t_n, and its estimate is $\hat{\alpha}(t)$ accordingly.

1) Requirement-A : This requirement consists of two parts. Firstly, it says that there exists an upper bound for the system that should not be exceeded i.e., it limits the admission rate to avoid overload. Secondly, it tells us that once the limit is not exceeded then all the offers should be admitted to maximize the utilization. However, in theory the words capacity and bound can have many different definitions depending on the model we use for the target node.

The target node is often modeled with an inverse Token Bucket, i.e., a server with deterministic serving rate s and a queue of maximal length Q. It is very easy to see that the Token Bucket throttle $\gamma_t(s, Q)$ can perfectly meet the requirement in this case. (Note that this is only true supposing that there is no delay in the system between the throttle and the protected entity while $s(t) = r(t)$ is satisfied.)

Another approach is to assume that the target can handle requests on a maximal call rate c that is used as the bound at the throttle.

Both models have benefits and drawbacks while a mixture of them is used in practice. Speaking about the capacity of a node in Next Generation Networks engineers often refer to the call rate value in industrial contracts and Service Level Agreements. It is very important to note that the feedback driven overload control mechanisms work with call rate information too (see [13]). On the other hand a server with queue is a common model in the academic literature for the CPU capacity and Token Bucket (or versions of it) is proposed in many standards (e.g., [13] again) and implemented into nodes.

As a consequence we say that although it is rather difficult to give an exact definition for **Requirement-A** we can give a certain definition grabbing a few properties depending on the method we use.

Definition 5: Call rate bound. **Requirement-A** is met if $\sum E[a_i(t_n)] \leq c(t_n)$ (the throughput rate is bounded in expected value).

Theorem 1: The throttle with strategy γ_g meets the **call rate bound** requirement.

Proof: The proof relies on the fact that the estimator is asymptotically unbiased i.e., $\lim_{T\to+\infty} E[\hat{a}; T_i] = E[a_i]$ with negative bias if $T \geq 1/a_i$ (thus $E[\hat{a}_i] < E[a_i]$). The proposed strategy γ_g limits a_i in a way that $a_i \leq g_i$, thus showing $g(t) := \sum g_i(t) = c(t)$ competes the proof.

Define $u_1(t) := \sum_{i:\hat{\rho}_i(t)<s_ic(t)} \hat{\rho}_i(t)$ and $u_2(t) := \sum_{i:s_ic(t)<\hat{\rho}_i(t)} s_ic(t)$ thus $u = u_1 + u_2$ and then $g_i = \min\{\hat{\rho}_i, s_ic + (\hat{\rho}_i(t) - s_ic(t))\frac{c-u}{\rho-u}\}$. Although the system is non-stationary it is homogenous in time so $f(t) = \text{const.}$ for all functions. Now calculate $g(t)$:

$$\begin{aligned} g &= \sum g_i = \sum \min\{\hat{\rho}_i, s_ic + (\hat{\rho}_i - s_ic)\frac{c-u}{\rho-u}\} = \\ &= \sum_{i:\hat{\rho}_i<s_ic} \hat{\rho}_i + \sum_{i:s_ic<\hat{\rho}_i} s_ic + (\hat{\rho}_i - s_ic)\frac{c-u}{\rho-u} = \\ g &= u_1 + u_2 + (\rho - u_1 - u_2)\frac{c-u_1-u_2}{\rho-u_1-u_2} = c. \end{aligned} \quad (7)$$

■

Corollary 1: The following calculation of g can also be used:

$$g_i(t) = \min\{\hat{\rho}_i, s_ic(t) + (\hat{\rho}_i(t) - s_ic(t))\frac{c(t)-u(t)}{\rho-u(t)}\}, \quad (8)$$

where $u(t) = \sum_{i:\hat{\rho}_i(t)<s_ic(t)} \hat{\rho}_i(t) = \alpha(t)$. Then (7) becomes:

$$g' = u_1 + u_2 + (\rho - u_1 - u_2)\frac{c-u_1-u_2}{\rho-u_1-u_2} = c. \quad (9)$$

The difference between the two strategies is that in case of g the remaining capacity is split between the classes with higher offer rates proportionally to their weights while using g' the remaining capacity is split proportionally to the remaining offer rates. Both satisfies **Requirement-A** and **Requirement-C**as we will show. From now on g means either g or g' and the results will be obviously the same.

2) Requirement-B: As pointed out before, the priority requirement for call gapping is the most complex one in a way, since in the gapping algorithms it is supposing that we make decisions using measures on the past and the present offer. No future events can be used, thus **Requirement-B** is always satisfied. There is always one offer in the system and the throttle can admit or reject it according to **Requirement-A** and **Requirement-B**.

In the case of the Token Bucket call gapping, different watermarks W_j are introduced for each priority level j. One interpretation is that the bucket allows larger peaks for traffics with higher priority, thus $W_j < W_k$ whenever k represents the higher priority level. Doing this, the bucket implicitly reduces the throughput for lower priority traffics (the extra peak in the bucket has to be refilled with tokens i.e., $b(t)$ has to decline below the low watermarks to admit low priority traffic). Note that the different watermark levels have no effect if the offer rate is low with small peaks thus the rejection probability is small i.e., if there is no overload. Supposed that the true bound is $W = \max\{W_j\}$, this system preserves capacity for high priority traffic.

We give a similar solution for the problem through the timer parameter of the estimators: T. As defined above, we introduce a function of $T : j \mapsto T_j$ where $T_k \leq T_j$ if k represents the higher priority. (Note that it is the other way round for W_js.) The interpretation is that the estimator forgets the high offer

rates faster for the traffic of the higher priority. Let $T_m = \min\{T_j\}$ the true bound on the throttle using different T_js. This means that for low priority traffic it remembers the high peaks for a longer period, thus reserves capacity for the higher priorities similarly to the Token Bucket.

The two methods have different characteristics, but one thing is common. Both reserve capacity for higher priority traffic. Now we say that to meet **Requirement-B** the system has to have this ability and define it in the following way.

Definition 6: **Requirement-B**. Suppose that the throttle has rejected an offer at time t_{n-1}. Let $t_{n;j}$ be the closest time the throttle is able to admit an offer of priority level j. **Requirement-B** is met iff $\forall k, l(t_{n;k} \leq t_{n;l}) \Leftrightarrow (k \geq l)$ (k represents a higher priority).

The exact proof of this statement is not ready yet. The simulation results show that the proposed strategy satisfies **Requirement-B**. We discuss the statement in the Numerical Results Section.

3) Requirement-C: This is referred to as the throughput share requirement and tells us that there should be at least an s_i portion of the capacity dedicated to traffic class i.

Definition 7: **Requirement-C**. The **Minimum share requirement** is met if $\forall i : (\rho_i(t_n) \leq s_i c(t)) \Rightarrow E[a_i(t_n)] = \rho_i(t_n)$ i.e., if the offer rate of a traffic class is less than the agreed share, it should be fully admitted.

Theorem 2: The throttle with strategy γ_g meets **Requirement-C** in expected value.

Proof: We have the asymptotical unbiasedness for our estimators, thus $\lim_{T\to\ +\infty} E[a;\hat{T}_i] = E[a_i]$, meaning that the proof is true for the expected value of a_i.

Statement $\hat{a}_i(t_n) = \hat{\rho}_i(t_n)$ whenever $\forall i \hat{\rho}_i(t_n) \leq s_i c(t)$ is equivalent to the statement ($g_i(t_n) \geq \hat{\alpha}_i(t_n)$ thus) $g_i(t_n) \geq \hat{a}_i(t_n)$ whenever $\hat{\rho}_i(t_n) \leq s_i c(t)$. According to strategy γ_g: $g_i(t_n) = \hat{\rho}_i(t_n)$ whenever $\hat{\rho}_i(t_n) \leq s_i c(t)$ and since $\hat{\alpha}_i(t_n) \leq \hat{\rho}_i(t_n)$ because $\hat{a}_i(t_{n-1}) \leq \hat{\rho}_i(t_{n-1})$, it is true that $\hat{\alpha}_i(t_n) \leq g_i(t_n)$ thus the offer is admitted (and also $\hat{a}_i(t_n) \leq g_i(t_n)$). ■

C. Rate model for Token Bucket and a joint algorithm merging the methods

In this section we introduce a model for Token Bucket that is equivalent to the definition in Section II but makes calculations easier.

Definition 8: Token Bucket Rate Model Strategy: $\tilde{\gamma}_t(r, W)$ Let us define $T(t) = W/r(t)$ and use the following equation for updating the bucket rate variable:

$$\tilde{a}(t_n) = \frac{\chi(t_n)}{T} + \max\{0, \frac{T\tilde{a}(t_{n-1}) - (t_n - t_{n-1})r(t_n)+}{T}\}$$

where $\chi(t) = 1$ iff there is an offer at time t. Admit the offer iff $\tilde{a}(t_n) \leq r(t_n)$. If the offer is admitted then the above definition is the used for the next value of the bucket rate variable $\tilde{a}(t)$. If the offer is rejected then $\tilde{a}(t_n)$ is recalculated with $\chi(t) = 0$.

Theorem 3: The Token Bucket and the Token Bucket Rate Model Strategy are the same: $\gamma_t = \tilde{\gamma}_t$.

Proof: It is easy to show that $b(t_{n-1}) = \tilde{a}(t_{n-1})T \Rightarrow b(t_n) = \tilde{a}(t_n)T$ and the decision is $b = T\tilde{a}(t) \leq Tr(t) = W$ also trivial. ■

If one extends the Token Bucket for traffic class handling with some role like in the proposed mechanism it will not provide traffic class fairness. The reason is hidden in the fact that unlike $\tilde{\rho}, \tilde{\alpha}, \tilde{a}, \tilde{\beta}$ and all such estimators is not asymptotically unbiased i.e., $E[\tilde{\lambda}] = \lambda$ as $t \to \ +\infty$ is not true for the estimators defined with:

$$\tilde{\lambda}(t_n) = \frac{\chi(t_n)}{T_j} + \max\{0, \frac{T\tilde{\lambda}(t_{n-1}) - (t_n - t_{n-1})r(t_n)}{T}\}. \quad (10)$$

The bucket fill does not represent at all the used capacity in the system it only measures the peakedness of the traffic but these peaks can happen on low offer rates too.

On the other hand, the proposed method does not allow such big transient peaks in the traffic. Now we aim to make the proposed new call gapping to behave like Token Bucket. We define the following strategy that is a mixed architecture.

Definition 9: Rate Based Call Gapping with Bucket-type Aggregate Characteristics: γ_x Take all the definition from the new call gapping mechanism γ_g for $\hat{\rho}, \hat{\alpha}, \hat{a}, u, g_i$ and define $T_j(t) = W_j/r(t)$. Take W_j and the bucket fill change definition b from the original token bucket γ_t. Perform all the steps like in γ_g but decide using the following constraint equation: $\frac{b(t_n)}{W_j}\hat{a}_i(t_n) \leq g_i(t_n)$.

We will show numerically that the mixed algorithm behaves like Token Bucket on aggregate level and meets all the requirements. The source of the idea comes from the fact that $\hat{a}(t)$ places a strict bound on the rate thus $\hat{a}(t) \leq r(t)$ is always true as required. However we decrease the value of $\hat{a}$ thus allow peaks in the traffic like Token Bucket does. (See that Token Bucket γ_t allows temporary bounding violation rate-wise unlike γ_g but like γ_x. The bucket size related to the whole bucket is a kind of measure of this violation.)

1) $\gamma_{t'}$ and γ_x and Requirement-A : Here we discuss how the different algorithms meet the maximal throughput requirement. It is obvious that Token Bucket cannot meet **Requirement-A** in the way it was defined before since that definition assumed that the target has an infinite queue.

We do not aim to give an exact definition to **Requirement-A** but we derive relations between the bucket and the estimator based throughput characteristics. The number of admitted offers i.e., the probability of admission is in the center of our interest.

The probability of admission for token bucket depends on the offer rate with the following formula: $1 - \mathrm{Erlang}[\rho, r]$. Thus the probability of losing calls is only defined at given values of ρ.

For rate based call gapping, since the estimator always over-estimates the rate ($\lambda < \hat{\lambda}$) and cuts the traffic strictly with c the admission rate is always below the target. But for the same reason it is possible that the offer is rejected although it could have been accepted according to the bound. The probability of this is the probability of estimating higher rate than c while the

true offer rate is lower: $P[\hat{\alpha} > c|\alpha < c] = 1 - \frac{P[\alpha<c-B[T]]}{P[\alpha<c]}$, where $B[T] = 1/(T(1-F[T]) + E[\Delta t|t < T]) - \alpha$ is the bias. (Knowing the exact bias if constant intensity is supposed for the offer rate, the bound can be modified to have maximal throughput and strict bound at the same time.)

The two methods can only be compared at a given value of the intensity. For all those values when the value of the intensity is not between $c - B[T]$ and c the γ_g strategy works perfectly. The Token Bucket drops a call with positive probability for any value of the offer rate and also might admit when the intensity is higher than allowed. This means that we cannot tell which method is better or has the higher throughput since it depends very much on the offer rate.

Theorem 4: The mixed strategy γ_x meets **Requirement-A** with appropriate watermark settings.

Proof: It is shown in Theorem 1 that $\sum g_i(t) = g(t) = c(t)$ and since the definition of g was not changed we should only examine what means to compare g_i to $\frac{b}{W_j}\hat{a}$ rather than to $\hat{a}$.

When we admit a request then $1 \leq b(t_n) \leq W_j \leq W_{\max}$ thus $\frac{1}{W_{\max}}\hat{a}_i \leq \frac{1}{W_j}\hat{a}_i \leq \frac{b(t_n)}{W_j}\hat{a}_i \leq \hat{a}_i$. This tells us that γ_x lets through more messages than γ_g since $E[\frac{b(t_n)}{W_j}\hat{a}_i] \leq E[\hat{a}_i]$. Fortunately the maximal watermark limits this overflow error $\frac{1}{W_{\max}}E[\hat{a}_i] \leq E[\frac{b(t_n)}{W_j}\hat{a}_i]$. It tells us that there is a setting of watermarks that guarantees bounding. (It is obvious that if $W_{\max} \to \ +\infty$ then $\frac{b}{W_{\max}}\hat{a}$ becomes very small and we always admit the request thus the theorem cannot be proved for any watermark settings.) ■

2) γ_x and Requirement-B: Some simple theorems are proved to show that the mixed strategy meets the priority and the throughput share requirements.

Theorem 5: Token Bucket strategy γ_t meets **Requirement-B**.

Proof: Obviously, the time to accept the next offer of priority level j is the time when the bucket level declines sufficiently to $b(t) \leq W_j$. For all levels $k > j$, $W_k > W_j$ i.e., $b(t)$ declines under the lower threshold later in time and the requirement is met. ■

Again it is rather hard to show that the mixed strategy γ_x meets **Requirement-B**. However, it seems to be trivial that γ_x satisfies **Requirement-B** more drastically than γ_t does. We have interesting simulation results presented about this property. We can see numerical results about this in Section IV.

3) γ_x and Requirement-C:

Theorem 6: The mixed strategy γ_x meets **Requirement-C**.

Proof: As pointed out γ_x admits at least all the offers γ_g does since $\forall i, \frac{b}{W}\hat{\alpha}_i \leq \hat{\alpha}_i$ is compared to $s_i c$ while a comparison of $\hat{\alpha}$ would be enough. This means that the mixed strategy provides minimum throughput share and fulfills **Requirement-C**. ■

IV. Numerical results and analysis

Although we have nice proofs on the good behavior of the proposed rate based call gapping mechanism the complete mathematical discussion about the differences and similarities with Token Bucket is not ready yet. It is also true that the requirements can be interpreted with definitions slightly different from those we gave. Therefore, we would like to present some simulation results and show that the findings are valid.

The simulation is written in *Mathematica* [15] and a *notebook* is available at *http://www.math.bme.hu/ kovacsbe/rbcg/BENEDEK-KOVACS-rate-based-call-gapping-PRELIMINARY-VERSION.nb* as an electronic appendix.

A. *Requirement-A*

The figure shows that all the mechanisms limit the admitted offer rate while trying to keep the highest throughput. In this scenario we examine the traffic on aggregate level i.e., there is only one traffic class for which the capacity of the throttle should be maximized and limited. The capacity is 1 offer/sec for the simple simulation case while the average number of offers per sec increases from 0.8 to 2 meaning that there is a 200% load on the node.

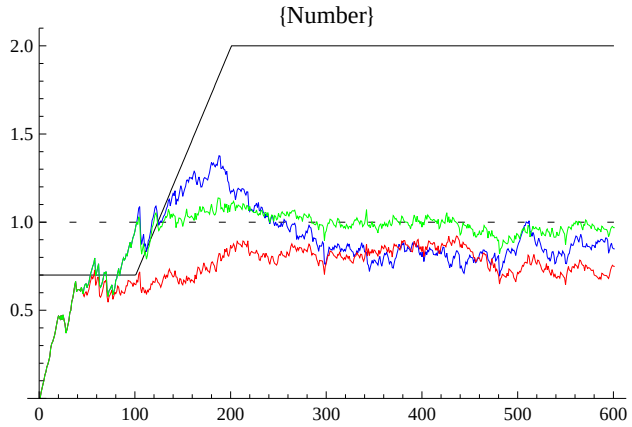

Fig. 3. The new algorithm (γ_g) on aggregate level. (Black: nominal offer rate, red: the token bucket's, blue: γ_g's, green: γ_x's throughput.)

As it can be seen in Figure 3 all three mechanisms limit the admitted traffic although Token Bucket allows considerable peak at the beginning. (The size of the peak depends on the parameters we set. Here the 1 offer/sec capacity is very small compared to the watermark which is set to 10.) On the other hand, rate based call gapping seems to under-utilize the system while the joint mechanism seems to have the smoothest and also maximal throughput.

After a total 600 offers from each traffic with the same exact trajectory the results show that $\gamma_t, \gamma_g, \gamma_x$ has admitted 415, 386, 404 number of calls respectively.

The problem with the mathematical discussion of maximal throughput is that the results depend very much severely on the value of the offer rate and capacity. It is only possible to compare the mechanisms at given rates, which is useless for real applications.

 ISBN: 978-1-61208-133-5

90	100	$W_H = 10$ $W_L = 15$	{0.,1.} [.002,.002]	{0.38,0.62} [.015,.015]	{0.01,0.99} [.006,.006]
150	100	$W_H = 10$ $W_L = 15$	{0.2,0.98} [.003,.003]	{0.4,0.6} [.007,.007]	{0.05,0.95} [.005,.005]
10	10	$W_H = 10$ $W_L = 20$	{0.,1.} [.000,.000]	{0.31,0.69} [.014,.014]	{0.,1.} [.000,.000]
10	10	$W_H = 10$ $W_L = 10$	{0.5,0.5} [.008,.008]	{0.5,0.5} [.009,.009]	{0.5,0.5} [.007,.007]

TABLE I
IN EACH ROW THE FOLLOWING QUANTITIES ARE PRESENTED RESPECTIVELY: TOTAL OFFER RATE: ρ, MAXIMAL THROUGHPUT: c, WATERMARK SETTINGS: $W_{\text{HIGH}}, W_{\text{LOW}}$ WHILE $T_j := W_j/c$. THEN PORTION IN REJECTED MESSAGES FOR TOKEN BUCKET, RATE BASED CALL GAPPING AND THE MIXED MECHANISM RESPECTIVELY.

B. *Requirement-B*

To discuss **Requirement-B** we provide the reader with some statistical results. The sample is generated with our simulation program. Generally there are two priority levels: normal and emergency calls. Each call is one of the two types with 1/2 probability. The means and the standard deviations are presented of 100 samples with 10 000 offers handled in each sample. The further setups for the simulation can be seen in Table IV-B.

It can be seen that all three methods reject less offer from those of higher priority but Token Bucket (γ_t) and the mixed mechanisms (γ_x) enforce a more strict priority handling than the simple proposal. Note that in case of sustained overload (row 2) almost all dropped offers are the lower priority ones.

C. *Requirement-C*

The results tell explicitly that unlike the new rate based call gapping proposal the original Token Bucket algorithm does not meet **Requirement-C**. We consider a scenario when there are two traffic classes Class A and Class B. The agreed share for Class A is the 20% of the total capacity of the node while the share for Class B is the remaining 80%. The offer rates set for the simulator are exactly the inverse of this for the two types of traffic.

The aggregate offer rate increases from 0.7 offers/sec to 2 offers/sec and reaches the scenario of 100% overload (the capacity of the node is 1 offer/sec while the offered rate is a 2 offers/sec on average). The offer rate of traffic Class B is 0.4 i.e., it is still under its provided share, thus all such calls are admitted. On the other hand, the whole remaining capacity should be granted to traffic Class A and it should be admitted on a higher level than the agreed share and only those exceeding the capacity limit are to be rejected.

Figure 4 shows the behavior of the rate based call gapping mechanism. With the proposed mechanism the minimum share is guaranteed for traffic Class B (the admission line is around the offered), while the requirement fails for Token Bucket. With the proposed method there is no rejected message of Class B, since it never offers on a higher rate than the agreed share. The throughput of the throttle is limited but also maximized, since Class A is granted all remaining capacity.

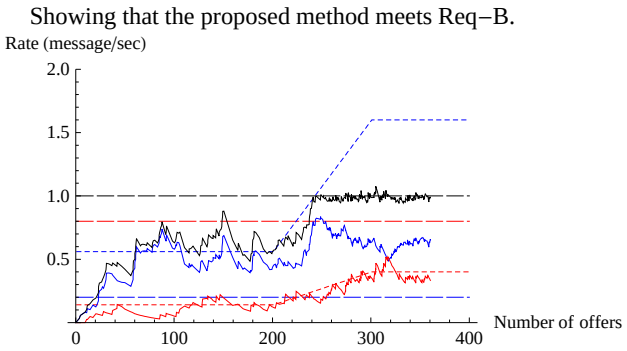

Fig. 4. The new algorithm (γ_g) with two traffic classes. Black: aggregate throughput rate, Red and Blue: the throughput rate of each traffic class, Solid: throughput, Dashed (large): nominal capacity, required minimal throughput, Dashed (small): nominal offer rates

V. CONCLUSION

We have presented a weighted fair sharing mechanism with no delay and its extension with the original Token Bucket to provide good network characteristics as well. These unique mechanisms meet the *maximal throughput with bound* requirement, handle priority messages and offers, and give minimum share for different traffic classes without using message buffers or queues. New ways of measuring traffic intensities are also proposed.

Examining the properties of the mechanisms we gave mathematical definitions of the three requirements and accompanied the mathematical model with several theorems. Still, the proof of priority handling is missing for the new methods, we presented statistical analysis instead. We used simulation that we implemented to underpin our proposal and findings.

Our rate based call gapping strategy can use different traffic intensity estimators. It is still remains an open task to find the optimal estimator and the optimal parameter setting of the estimators considering Poissonian input traffic with variable intensity or even a non-Poissonian (e.g., general renewal or Hawkes type) input process. The mathematical background on to prove the properties of the estimates of the intensity of a point process is to be published in the near future.

 ISBN: 978-1-61208-133-5

VI. APPENDIX

Notations:

$a, a(t)$	Real admission rate
$\hat{a}, \hat{\alpha}(t)$	Estimated admission rate
$b, b(t)$	Actual bucket fill
$c(t)$	Maximal capacity of the target (rate)
$g_i, g_i(t)$	Goal rate for traffic class i
$g, g(t)$	Sum of goal rates of all traffic classes
$r, r(t)$	Token Bucket's token generation rate
T	Parameter of the estimator
T_j	Parameter of the estimator for priority level j
$u, u(t)$	Used capacity according to **Requirement-B**
W	Watermark for Token Bucket
W_j	Watermark for offers of priority level j
$\hat{\alpha}, \hat{a}(t)$	Estimated preliminary admission rate
$\beta, \beta(t)$	Preliminary bucket size
γ_t	The token bucket throttle function
γ_g	The rate based call gapping throttle function
$\gamma_{g'}$	The variant of the rate based call gapping throttle function
γ_x	The rate based call gapping throttle function with Token Bucket extension
$\lambda, \lambda(t)$	Intensity (rate) of a Poisson process
$\hat{\lambda}, \hat{\lambda}(t)$	Estimated rate (intensity)
$\rho, \rho(t)$	Real offer rate
$\hat{\rho}, \hat{\rho}(t)$	Estimated offer rate

ACKNOWLEDGEMENTS

The research was motivated by Ericsson Research Hungary (Ericsson Telecommunications Hungary Ltd.) and the High Speed Network Laboratory, and was partially funded by the Hungarian Ministry of Culture and Education with reference number NK 63066 and the National Office for Research and Technology with reference number TS 49835. Special thanks to János Tóth (Budapest Univ. of Tech. and Eco., Dept. of Math. Analysis) for the careful reviews and support.

REFERENCES

[1] A. Demers, S. Keshav, and S. Shenker: Analysis and simulation of a weighted fair queuing algorithm, Journal of Interworking Research and Experience, pp. 3–26, 1990

[2] A. K. Erlang: "Solution of some problems in the theory of probabilities of significance in automatic telephone exchanges", Post Office Electrical Engineer's Journal 10 (1917–18), pp. 189–197.

[3] A. K. Parekh and R. G. Gallager: "A generalized Processor Sharing Approach to Flow Control in Integrated Services Netwokrs: The Single-Node Case", IEEE/ACM Transactions On Networking, Vol. 1, No. 3, June 1993, pp. 344–357.

[4] J. Rexford, F. Bonomi, A. Greenberg, and A. Wong: Scalable Architectures of Integrated Traffic Shaping and Link Scheduling in High-Speed ATM Switches, IEEE Journal On Selected Areas in Communications, vol 15, issue 5, pp. 938–950, 1997.

[5] K. E. Crawford: "Method of controlling call traffic in a communication system", U.S. patent, no. 4224479, 1980.

[6] L. Takács: "Introduction to the theory of queues", Oxford University Press, New York, 1962, pp. 186–188.

[7] M. A. Gilfix: Method for Distributed Hierarchical Admission Control Across a Cluster, United States Patent, no. 0008090, 2006.

[8] P. P. Tang, Tsung-Yuan C. Tai: Network traffic characterization using token bucket model, INFOCOM'99 conference proceedings, vol 1. pp. 51–62

[9] Y. Ogata: "The asymptotic behavior of maximum likelihood estimators for stationary point processes", Annals of Institute of Satatistics and Mathematics, 30 (1978), Part A, pp. 243–261

[10] B. Kovács: "Mathematical remarks on Token Bucket", IEEE Conference on Software Telecommunications and Computer Networks, 24-26 Sept. 2009, pp. 151-155.

[11] M. Whitehead: "GOCAP – one standardised overload control for next generation networks", BT Technology Journal, Volume 23, Issue 1 (January 2005), pp. 147–153

[12] H.248 v2 protocol specification: *http://tools.ietf.org/html/draft-ietf-megaco-h248v2-04*, Last Access: 19 Mar. 2011.

[13] H.248.11 extension specification: ITU-T recommendation H.248.11

[14] "verload control in a quality-of-service aware telecommunications network", European Patent Office, The Hague, no. PCT/EP2008/059693, 2008, WO/2010/009764, App. no.: PCT/EP2008/059693, Publication: 28.01.2010, filing date: 24.07.2008

[15] *Mathematica*, Wolfram Research Inc. *http://www.wolfram.com*, Last Access: 19 Mar. 2011

 ISBN: 978-1-61208-133-5