The Squatting and Kicking strategies for self-provisioned, bandwidth resource sharing in multiclass networks

Master Thesis

Joan Antoni García Espin

Telematics Engineering Department (ENTEL), Technical University of Catalonia (UPC), Barcelona, Spain

Abstract—The bandwidth management problem in multiclass networks has become again a matter for study due to the increasing need to smartly share the network resources to achieve higher use efficiency. Moreover, traffic patterns in access networks have changed in the last years, powered by new generations of multimedia and mobile-based applications, which require a special planning of the bandwidth sharing strategies along the network and continuous monitoring of the quality of service. On its turn, telecom operators predict these tendencies to increase and consolidate, which may lead current provisioning techniques to obsoleteness if not revised and enhanced. Consequently, per-class quality of service assurance has leveraged operator’s interest in fine-grained control of the traffic aggregation techniques and bandwidth resource management at the edges of the transport networks. In general, this requires porting Differentiated Services schemes from the Internet access segment to its transport segment. Thus, traffic-engineered network technologies have to be made compatible with per-class bandwidth management rather than with traffic aggregates. The scalability of the network management systems and the adaptation to changing traffic loads in multiclass networks are also a partially resolved challenge. This article proposes a self-provisioned, Squatting and Kicking bandwidth resource sharing strategy for multiclass networks where differentiated services are not natively built. Moreover, this article provides a summary of the bandwidth constraints models and shows how the squatting and kicking strategies can be adapted to be the basis for a new bandwidth constraint model, which widens the range of techniques available to operators for bandwidth resource management in multiclass networks.

Index Terms—bandwidth management, class of service, differentiated services, squatting, kicking.

I. INTRODUCTION

In search of being able to offer better quality of service (QoS) in multiclass networks, a strategy and model that aim at allowing bandwidth resource sharing among different traffic classes, also named classes of service (CoS), according to their service needs and assigned priorities is proposed. This paper presents a bandwidth resource sharing strategy and explains the so-called Squatting and Kicking techniques, which provide a theoretical basis for a new Bandwidth Constraint (BC) model for Differentiated Services-aware (DiffServ-aware) multiclass networks. This model can be similarly applied to DiffServ-aware Multi-Protocol Label Switching (DS-MPLS) transport networks using their Traffic Engineering (TE) capabilities.

The Squatting and Kicking techniques are also considered the basis for a new bandwidth constraint model. Traditional BC models such as Maximum Allocation Model (MAM) [21], Maximum Allocation with Reservation (MAR) [22] and Russian Dolls Model (RDM) [23] are briefly described in section IV. It must be noted that the protocol specification for the Squatting and Kicking Model (SKM) and its performance comparison to MAM, MAR and RDM is out of scope of this article and thus, considered as future work.

In a technical plane, MPLS is considered one of the...
protocol suites that better performs the network service convergence of voice, video and data, required in most Universal Mobile Telecommunication System (UMTS) networks and High-Speed Packet Access (HSPA) networks. These networks represent straightforward examples of the usage of Differentiated Services from multiclass networks over traffic-engineered transport network technologies. However, the presented strategy is not only valid for these networks but also can be applied to other technologies, such as optical networks.

Stepping into MPLS-TE, it is well known that it fosters easy class of service tagging, traffic prioritization and bandwidth resource optimization. IP Differentiated Services [17] over the MPLS protocol stack have been proposed in the literature for guaranteeing a given QoS level per class and for raising network utilisation to the maximum. However, MPLS network functionalities have to be enhanced to fully support IP Differentiated Services together with automated, class-based, network service provisioning.

Basing on the latter scenario, this article and the work supporting it are focused in a novel technique for self-provisioned resource sharing, where idle resources from a given class of service can be squatted in by another class. Low priority classes of traffic can utilize resources reserved for higher priority ones when being unused. Similarly, a technique for letting high priority classes kick lower priority ones out of their currently allocated resources is described. In particular, the study has been carried out splitting the available bandwidth in a link among the pool of classes of traffic coming from IP-DiffServ network into the DiffServ-aware, TE-enabled network domain (i.e. multiclass network). Other works [12, 13] have studied some traffic management algorithms under DiffServ, but the scheme proposed here enhances the per-link total bandwidth utilization on a class of service basis.

This article is organised as follows: Section II provides an overview on the motivation context and related work. Section III shows a set of definitions and assumptions that contextualise the arguments in this article, as well as introduces the notation used. Section IV elaborates on the existing Bandwidth Constraints Models defined by IETF. Section V presents the squatting and kicking concepts. Relevant use cases for the techniques presented in this article are described in Section VI. Section VII introduces the concept of the Potential Usable Bandwidth (PUB), for modelling the maximum amount of bandwidth resources a class can allocate in the network, when using squatting and/or kicking techniques presented. A simple auto-provisioning model based on the novel techniques, where the potentially usable bandwidth resource is evaluated, is presented on Section VIII. An exemplification for a 3-class multiclass network is presented in section IX. Finally, conclusions are located together with future work actions in Section X.

II. MOTIVATION AND RELATED WORK

Since the standardisation of the Internet Protocol in RFC 791 in September 1981, several QoS management models have been broadly studied and described in the literature, from which Best Effort (BE) [16], Integrated Services (IntServ, IETF RFC 1633) and Differentiated Services (DiffServ) [17] were broadly analysed and implemented. These models are based on a specific use of the octet named Type of Service (ToS) in the IPv4 header (RFC 791), which was renamed to Traffic Class (TC) in the IPv6 protocol specification (RFC 2460). As a consequence, the Internet is, by specification, an interconnection of multiclass-enabled networks. Thus, network resource management techniques have to be sensible to the existence of multiplicity of traffic classes in the backbone. The evolution in the definition of the ToS/TC fields along the time is a clear indicator of this issue: almost each revision of the specification of the octet increased the number of bits dedicated to datagram classification. RFC 791 defined only 3 bits for this purpose named Precedence inside the ToS field that limited the maximum number of classes identifiable to eight. Three extra bits were added for characterizing the service, basing on delay, throughput and reliability (DTR). In 1992, RFC 1349 kept the same 3 bits for class identification but added a fourth bit to the service characterization for indicating monetary cost (DTRC). By 1998, with the standardisation of the DiffServ, RFC 2474 re-organised the ToS field in two unique fields: the Differentiated Services Code Point (DSCP) of 6 bits and an unused field of 2 bits. This time, DTRC was shrunken to 3 bits and merged with Precedence field for unified processing in the routers. Latest revisions of the ToS field have kept DSCP as it is, focusing only in the two remaining bits (allocated for Explicit Congestion Notification in RFC 3168, in 2001).

Best effort management model forces all IP datagrams to use the same service policies, that is to say, no distinction between packets is done. Packets are routed regardless of their ToS field. Logically, this model does not allow QoS-demanding applications to be run at the borders of the network, since all traffic patterns are treated the same way and no QoS is assured for any of them. This was one of the main
reasons for creating both IntServ and DiffServ models. An effective use of the ToS octet in IP datagram header was urging.

Integrated Services (IntServ) aimed at provisioning end-to-end, controlled QoS by means of applying strict resource reservations to classes. Traditionally, the target resource has been bandwidth. Three different classes of service are defined in IntServ: Guaranteed, Controlled Load and Best Effort. Guaranteed class (RFC 2212) was provided with assured bandwidth, limited delay and extremely low packet losses. RFC 2211 describes Controlled Load class, which only assures its QoS level if the network is not loaded. If the network starts being congested, the service policy for Controlled Load class is degraded, being comparable to best effort in hard congestion situations. Due to the strict QoS constraints for Guaranteed class type and the end-to-end definition of the service policies, IntServ service model does not scale well in multiclass networks where traffic tributaries come from multiplicity of sources.

Differentiated Services (DiffServ) aim at solving the limitations of IntServ by introducing three key operation primitives:

- Definition of local service policies at each router (the so called Per-Hop Behaviour or PHB),
- Utilisation of loose resource reservations for traffic classes, and
- Flexible traffic class identification mechanism based on three main classes plus class prioritisation.

Each traffic class defined by DiffServ receives a given PHB service policy at each node along the path. Different traffic classes are applied different PHBs policies. Although ToS field allows up to 64 codes, only 21 of them are standard PHB codes, all of them ending in a bit set to zero. The DSCP bits determine the traffic class, grouped in three main blocks: Class Selector or CS (RFC 2474), Assured Forwarding or AF (RFC 2597) and Expedited Forwarding or EF (RFC 3246). CS class is similar to best effort, AF is comparable to Controlled Load from IntServ (although lacking of service policy degradation) and EF class is the one with better QoS assurance. Further details on DiffServ classes specification are out of scope of this article.

However, DiffServ model is unable to ensure end-to-end QoS levels by its own, since no traffic management is supported. At this point, MPLS-TE attracted much attention [4, 18, 19]. The ability to provide end-to-end QoS is limited in classical MPLS-TE due to the fact that it works with aggregated traffic flows towards the same routing destinations. This is achieved by aggregating classes inside the same Label Switched Path (LSP). An LSP is basically a sequence of routers in which every forwarding decision is made basing on the labels located at the same level in the MPLS label stack. To address this limitation, the Internet Engineering Task Force (IETF) proposed the DiffServ-aware Traffic Engineering (DS-TE), which is able to perform TE on a class basis, regardless of having the same topological destination [20]. With this new approach, traffic classes heading to the same destination can use different LSPs to be transported and, thus, be applied different service policies along the path. It has to be mentioned that these service policies can be influenced by bandwidth constraints derived from traffic engineering, which indeed provides a substantial advantage given the flexibility and variety of the existing Bandwidth Constraint Models.

Going in deep with TE capabilities, another interesting feature is the clear separation TE specifies between performance objectives in the network. As described in RFC 2702 [18], the key performance objectives associated with traffic engineering can be classified as being either traffic-oriented or resource-oriented. On the one hand, traffic oriented performance objective includes the aspects that enhance the QoS of traffic streams. In a single class scenario, the key traffic oriented objectives consider the best-known QoS parameters, that is: minimization of packet loss, delay and jitter, and maximization of throughput. Under a single class service model, minimization of packet loss is one of the most important traffic oriented performance objectives. Statistically bounded traffic oriented performance objectives (such as peak to peak packet delay variation, loss ratio, and maximum packet transfer delay) might become useful in the DiffServ-aware networks.

On the other hand, resource oriented performance objectives include the aspects regarding to the optimization of resource utilization. Efficient management of network resources, such as bandwidth, is the way to achieve resource oriented performance objectives. In general, it should be assured that some network resources do not become over utilized and congested while other subsets along alternate paths remain underutilised. Bandwidth is a crucial resource in contemporary networks. Therefore, advanced techniques for bandwidth resource allocation and
management are required. This article presents mechanisms and their applicability to bandwidth constraints models basing on resource-oriented objectives in TE.

In any case, to assure efficient bandwidth resource sharing and QoS guarantees (especially the higher priority ones), it is necessary a strategy that combines (i) lower class of traffic pre-emption when contention for system resources occur; and (ii) easy access to spare and/or idle bandwidth resource for any class [11]. In the former, class pre-emption is based in the same principle as LSP pre-emption in MPLS: allocated low priority classes may be misplaced in order to allocate a higher priority one.

Some studies have assessed DS-TE solutions basing on different bandwidth constraints models, by parameterising QoS and bandwidth management procedures in multiclass networks. A proposed methodology consists in the definition of the so called Key Performance Indicators (KPI) [7], which can be used to objectively quantify the QoS level for each considered service given a traffic load and a specific network topology altogether. The optimality of the method is represented by means of a cost function that jointly handles expected QoS fulfilment and network resource usage.

Other works [2, 4, 6] consider the use of matrices for simultaneously handling bandwidth and traffic classes with traffic prioritization. Not only this facilitates the use of traffic matrix theory for modelling TE behaviour but also allows a flexible inter-class bandwidth management. As an example of the latter, we can consider the “user it or lend it” strategy [6], which is used to guarantee minimum bandwidth resources for each class of traffic. The squatting scheme presented in this article shares basic functional principles with the mentioned strategy, as will be seen in the coming sections.

III. DEFINITIONS, ASSUMPTIONS AND NOTATION

This section has a three-fold purpose: on the one hand, it introduces the terminology that will be used along the document, part of which is based on RFC 3246; on the other hand, it presents the assumptions that create the context for the work presented; and, finally, it shows and describes the notation used along the use cases description, analytical model and evaluation sections.

A. Definitions

This sub-section is divided in two parts: the first one defines a set of terms that characterise the classes of traffic depending on their behaviour in the network, whereas the second one defines the terms used to specify the utilisation state of the bandwidth resource.

The following definitions do not supersede the ones in the above mentioned requests for comments. In any case, they must be considered as a simplification of the RFCs.

Finally, the key words “must (not)”, “required”, “shall (not)”, “should (not)”, “recommended”, “may” and “optional” in this document are to be interpreted as described in RFC 2119 [14].

1) Traffic classes.

A traffic class (also class or class of service or CoS) is a logical group of data packets that meet a given constraint, such as equal value in a specific header field (e.g. source-destination).

Traffic classes populate the so called multiclass networks. A multiclass network is used to transmit multiple classes of service at the same time. Therefore, the multiclass network implements the necessary mechanisms to allow specific traffic management per class.

In this article, we define two main strategies to handle bandwidth among classes: the Squatting and the Kicking:

- **Squatting**: act or action of occupying resources allocated to other classes when their holders are not using them. It must be noted that squatting can be applied over resources allocated to either higher priority classes (default behaviour) or lower priority ones. This concept is further elaborated in the following sections.
- **Kicking**: act or action of expelling a lower priority class from its allocated resources, either partially or totally.

Any class can adopt either a squatting or a kicking behaviour. Moreover, any class can have a subject or a target role in a squatting or kicking process, depending on whether it is executing the process (subject role) or it is receiving the action (target role). Consequently, classes are characterised as follows:

- **Squatter class**: class of traffic that is performing squatting (subject role).
- **Squatted class**: class of traffic that is affected by a squatting action (target role).
- **Kicker class**: the class of traffic that is performing kicking (subject role).
- **Kicked class**: the class of traffic that is affected by a kicking action (target role).
2) The bandwidth resource.

The key concepts for the different characteristic states the bandwidth can transit from and to, are presented below. In all definitions, a maximum reservable bandwidth resource (Max-RBR) exists. Max-RBR corresponds to the total bandwidth resource capacity in a link in the multiclass network (cf. link capacity).

The bandwidth resource states considered in this article are:

- **Reserved bandwidth resource** \((A_i)\): it is the pre-assigned bandwidth resource for a given class. It corresponds to each of the logical divisions in which the Max-RBR (link capacity) can be partitioned. In some literature, it is also known as “pre-allocated bandwidth”.

- **Allocated bandwidth resource** \((S_i)\): it is the bandwidth resource that a given class is using from the link. In general, if a bandwidth resource is allocated for a given class, the class is currently transmitting data.

- **Cross-allocated bandwidth resource** \((S_{ij})\): it is the allocated bandwidth resource in this case where a class \((C_i)\) has been allocated in a reserved bandwidth resource that is owned by another class \((C_j)\).

- **Demanded bandwidth resource** \((G_i)\): it is the bandwidth resource required by a given class at a given point in time, just before being allocated.

- **Unused bandwidth resource of class** \(C_i\) \((US_i)\): it is the bandwidth resource pre-allocated for class \(C_i\) that is not being used by its owner class \((C_i)\).

- **Idle bandwidth resource of class** \(C_i\) \((IS_i)\): it is the bandwidth resource pre-allocated for class \(C_i\) that is not used by any class.

It must be noted that idle and unused bandwidth concepts are not equal. Unused bandwidth for class \(i\) monitors the resources that have not been allocated for class \(C_i\), regarding to the initial resource reservation for this class \((A_i)\). Formally, \(US_i\) is defined as follows:

\[
US_i = A_i - S_i
\]

On the other hand, idle bandwidth for class \(i\) keeps track of the resources that have not been allocated for class \(C_i\) or any other class that has been authorised to make use of them. Formally, idle bandwidth is defined as in the following equation:

\[
IS_i = A_i - S_i - \sum_{j \neq i} S_{ji}
\]

Hence, there is a trivial, direct relationship between the used and idle bandwidth resources:

\[
US_i = IS_i + \sum_{j \neq i} S_{ji}
\]

**B. Assumptions**

In this article, the maximum number of classes is \(N\). Moreover, traffic classes are prioritised at the borders of the network domain. Classes are assigned a priority number, ranging from \(1\) (maximum) to \(N\) (minimum). For simplicity, in the rest of the article, a higher or highest priority class will be named HiPri; and similarly a lower or lowest priority class will be named LoPri. It is out of scope of this article to discuss what the borders of the network domain are and how the traffic classes are prioritised there.

Since this article is focused on a problem identified in multiclass networks, the term resource refers to bandwidth. Nevertheless, it must be noted that the use cases and model presented in this article can be abstracted and, thus, generalised for being applied to several kind of resources.

Finally, as expressed in the introduction, the goal of the auto-provisioning, squat-/kick-based model presented in this article is to achieve more efficient use of the resources; motivated by the observation of the usage of the link bandwidth in multiclass networks from a per-class resource usage perspective.

**IV. THE BANDWIDTH CONSTRAINTS MODELS**

IETF’s literature on Bandwidth Constraints (BC) Models describes three main techniques for bandwidth management in DiffServ-aware networks using TE: the Maximum Allocation Model (MAM), the Maximum Allocation with Reservation Model (MAR), and the Russian Dolls Model (RDM). The three of them are based on the requirements for support DiffServ-aware MPLS traffic engineering, as described in RFC 3564 [20].

For the sake of keeping compatibility with RFCs 4125 to 4128 [21, 22, 23, 24], and according to traffic engineering terms, the Bandwidth Constraint \(BC_i\) for class is defined as \(BC_i = A_i\) (soft constraint). Thus, the \(BC\) for a given class \(i\) corresponds to the initially reserved bandwidth resource for this class. It must be noted that, as commented in [21], the shares for each class are not isolated. Consequently, the existence of
the cross-allocated bandwidth resource cannot be obviated.

Regarding the first of the models, MAM is described in RFC 4125 [21]. It presents a simple model that allows each class of service having a reserved bandwidth and a full share of the overall resources, as far as:

- The sum of reserved bandwidths for all classes (considering a fixed maximum number of classes of eight) is less or equal to the Max-RBR. In general, Max-RBR may not be the same as the Link Capacity (LC).
- The sum of BCs for all classes is less or equal to the Max-RBR.
- The sum of the BCs for all classes is higher or equal to the sum of reserved bandwidths for all classes.

MAM is attractive in some DiffServ-aware TE environments for its simplicity and intuitiveness, easy bandwidth control policy definition, easy CoS isolation, and high bandwidth efficiency.

MAR is described in RFC 4126 [22]. It is an enhanced MAM system that implements a Call Admission Control (CAC) function based on the current reserved bandwidth for a given class and its associated BC, considering the total unreserved bandwidth in the link and a parameter called the "reservation bandwidth threshold" (RBTh) to govern the CAC function. MAR operates as follows:

- Given a class \( i \), if its allocated bandwidth is less or equal to \( BC_i \), a new connection for class \( i \) is admitted if the demanded bandwidth is less or equal to the unreserved bandwidth in the link.

\[
S_i \leq BC_i \Rightarrow \text{Admit if } G_i \leq LC - \sum_{v \in j} S_{ij}
\]

- If its allocated bandwidth is higher than \( BC_i \) (soft constraint), a new connection for class \( i \) is admitted if the demanded bandwidth is less or equal to the unreserved bandwidth in the link minus the reservation bandwidth threshold.

\[
S_i > BC_i \Rightarrow \text{Admit if } G_i \leq LC - \sum_{v \in j} S_{ij} - RBTh
\]

The strong point of MAR is that under normal, non-congested network conditions, all classes share all the available bandwidth resources but, in case of congestion, CAC function prevents greedy classes from seizing all the allocated bandwidth for each single class.

RDM is described in RFC 4127 [23]. It presents a more sophisticated technique for bandwidth resource sharing among classes than MAM or MAR. RDM mechanism defines a CAC that blocks any new class allocation if violating a simple rule:

\[
\sum_{i=1}^{N} S_i \leq BC_j
\]

Where \( N \) indicates the maximum number of classes considered; \( i \) is the iteration index and \( j \) is a given class number.

It can be easily derived from the previous equation that the allocated bandwidth for each class is recursively nested in the contiguous class resources (for \( N = 8 \)):

- \( S_2 \leq BC_2 \)
- \( S_6 + S_7 \leq BC_6 \)
- \( S_5 + S_6 + S_7 \leq BC_5 \)
- \( \ldots \)

V. SQUATTING AND KICKING CONCEPTS

Prior to the definition of the squatting and kicking concepts, a step further on the characterisation of the used/unused and idle bandwidth must be done. In a traditional DS-TE scenario, traffic classes make use of the bandwidth resources in two different ways: reservation and allocation. Reserved bandwidth resources are only signalled because the control plane of the network plans to dedicate them to a given class of traffic. Thus, the binding between DiffServ and TE is performed, that is, a traffic class is assigned a given pool of bandwidth resources. On the contrary, allocated bandwidth resources are currently used by a given traffic class, which is being transmitted along the data plane. Multiclass networks considered in this article always perform a reservation of the bandwidth resources for a given class of service on beforehand to the allocation. By smartly combining the reservation and allocation states of the bandwidth resources and being those attached to classes of service, we can build the framework for squatting and kicking definition.

The squatting and kicking terms were defined in the previous section. However, a further detailed vision of the concepts is developed and contextualised in the present section. Consequently, the so-called hard and soft squatting strategies, as well as the kicking strategy are presented. The rest of this section is divided in two parts: the first one elaborates on the squatting strategy
whereas the second one does so about the kicking strategy.

A. Soft and Hard Squatting

The squatting strategy is based on the idea that a given class of traffic can make use of the resources—bandwidth resources—that were originally reserved for a class of different priority, in case of excessive bandwidth resource demand.

As defined before, squatting is a pacific bandwidth resource sharing strategy based on the “use it or lend it” idea for the idle bandwidth resources per class. However, this strategy can be implemented in two different ways, depending on what the target classes of the squatting action are, in terms of priority. These implementations are named:

- Soft Squatting (SS)
- Hard Squatting (HS)

Soft Squatting implementation initiates squatting process against higher priority classes, starting from the present one and upwards. The way idle bandwidth resources are requested is sequential. That is, given class $C_i$ the initiator of the squatting process, it will first occupy idle resources from class $C_{i-1}$, then $C_{i-2}$, after that $C_{i-3}$, and so on, up to the highest priority class, $C_1$.

On the other hand, Hard Squatting implementation initiates squatting in the opposite direction, that is, taking idle bandwidth resources from the adjacent lower priority class ($C_{i+1}$), which has the most similar QoS parameters to $C_i$. After that, a lower priority one ($C_{i+2}$) will be considered, and so will iterate up to the lowest priority one, $C_N$.

In either case, the squatting strategy follows two primary rules:
- A given class will never expel another class from its allocated resources, in any case.
- If the target class of service does not have (enough) idle resources to satisfy the demand of the squatting initiator, the next adjacent class will be selected instead. This process can be iterated if needed.

Other rules have been considered at the time of writing this article, but have been reserved for future studies in the field. An example of these other rules is the definition of the “squatting thresholds”, that is, the parameters that will define the maximum amount of idle resources that can be allocated for another class of service for a given priority context.

B. Kicking

Similarly to the squatting scheme, the kicking strategy is based on the idea that a given class of traffic can make use of the resources—bandwidth resources—that were originally reserved for a class of different priority, in case of excessive bandwidth resource demand.

In this case, kicking is an aggressive bandwidth resource sharing strategy based on the idea of expelling lower priority classes from their allocated bandwidth resources in case of high demand rates in higher priority classes. This allows the situation where a class or service steals used/busy bandwidth resource to another class, under certain constraints.

The kicking strategy is restricted to a unique, directed priority scheme: high-to-low priority classes. That is, a given class $C_i$ can only kick resources from classes $C_j$ if and only if $j > i$. Although constraints like the “kicking thresholds” have also been considered, as in the squatting case, they are relegated to future work.

C. Considerations on idle and unused resources

As described at the beginning of this section, idle and unused bandwidth resources are computed differently and do not identify the same magnitude in the context of this article.

On a first instance, unused resources, $US_i$, only monitor the difference between the resources allocated by a given class in its original link share and the amount of resource reserved for the share (see eq. (2)).

Due to the fact that squatting (both SS and HS) and kicking techniques allow a given class of service using more bandwidth resources than initially reserved, we can find a situation where the allocated resources exceed the reserved resources. This behaviour is modelled as follows:

- If $S_i \leq A_i$ then $C_i$ is partly or totally using its initially reserved resource.
- If $S_i > A_i$ then $C_i$ is performing SS and/or HS and/or kicking.

Considering the previous situations and substituting in equation (2), we discover that the unused bandwidth resources for a given class can be negative:

$$\text{If } S_i \leq A_i \Rightarrow US_i \geq 0$$

$$\text{and}$$

$$\text{If } S_i > A_i \Rightarrow US_i < 0$$

The relation in (5.b) shows how unused bandwidth resources from class $i$ can be lower than zero, due to
the fact that this class can potentially use resources from other classes and thus consume more resources than initially reserved.

On a second instance, the idle bandwidth resources \((IS_i)\) suffer a similar effect. However, idle bandwidth resources take into account the contribution cross-allocations from other classes produce in the unused resources. That is, as shown in eq. (3), whereas \(US_i\) only monitors the difference between \(S_i\) and \(A_i\); \(IS_i\) monitors \(A_i\) minus the allocation \(S_i\) minus the cross-allocations other classes perform in \(C_i\)’s bandwidth resource share \(A_j\).

The cause of the cross-allocations can be found in any of the three strategies discussed in this article. In general, we assume that:

- If \(S_i < A_i\), then the term \(\sum_{j \neq i} S_{ij}\) can be influenced by either squatting or kicking actions, or both of them at the same time.
- If \(S_i \geq A_i\), then the term \(\sum_{j \neq i} S_{ij}\) can only be caused by kicking actions, since not enough bandwidth resource is available for the pacific squatting actions. It must be noted that \(S_i\) is not only considering self-owned bandwidth for class \(C_i\) but also squatted or kicked resources to higher/lower or only lower priority classes, respectively.

D. Prioritised use of the squatting/kicking strategies

As it has been commented before, Soft Squatting, Hard Squatting and Kicking have a considerable different operating mood. Whereas Soft and Hard squatting perform non-aggressive processes when gathering and using resources from other classes, Kicking forces third parties to quit their current allocation, regardless of their needs, for the simple fact of being of a lower priority than the kicker class.

Consequently, the way a class behaves when forced to initiate one of the previous processes is:

1. Perform Soft Squatting
2. Perform Hard Squatting
3. Perform Kicking

The previous steps are always executed sequentially by all classes of service. If one of the steps is not feasible for any reason, it will be skipped and the next step will be considered. For instance, class 1 (highest priority) cannot perform the 1st step, since no higher priority classes exist. Analogously, class \(N\) (lowest priority) cannot perform the 2nd step in any case, since any lower priority classes exist. Moreover, class \(N\) can never perform the 3rd step, for

the same reason commented before.

VI. Sample Use Cases and Behaviour

This section contains a set of fundamental situations or use cases where the squatting and kicking techniques gain importance. These situations constitute the building blocks of the squatting and kicking mechanisms theory in the upcoming sections.

Seven scenarios have been considered. Three of them only consider two actors: an already allocated class using its own resources and an over-demanding class that will start an occupation process against the allocated one, which are named Single Class processes. Other three of them consider multiple classes interacting with each other where the occupation processes may involve more the resources assigned to more than one class, which are named Multiple Class processes. Finally, the last scenario considers a mixture of all the previous ones, but it is left out of scope of this article.

A. Use cases

1) Single Class Soft Squatting (SCSS).

Case 1.1 (adjacent classes):

\(C_i \ (i \in \{2...N\})\) squats in idle resources of \(C_{i-1} \ (i \in \{2...N\})\).

Note that \(C_{i-1}\) is the immediate HiPri class for \(C_i\).

Case 1.2 (non-adjacent classes):

\(C_i \ (i \in \{2...N\})\) squats in idle resources of \(C_j \ (j < i)\).

Note that \(C_j\) is one of the HiPri classes of \(C_i\). Therefore, the latter is not restricted to squat in its immediate HiPri class, because the immediate upper class may not have enough resources available to meet the demand. The squatter class \(C_i\) will select the next HiPri class that has available resources to be squatted \(C_j\), looking up from \(C_{i-2}\) and increasing priority up to \(C_1\).

Remark: In both cases 1a and 1b, \(C_i\) cannot squat in resources, since no more HiPri classes are defined.

What if \(C_i\) tries to squat in resources of \(C_j \ (j > i)\) that have already been squatted by \(C_k \ (k > i)\)?

- Approach 1: resources that are already squatted are considered as blocked, so that \(C_i\) cannot squat in \(C_j\), although \(C_k\) (the squatter class already allocated in the resources of \(C_i\)) has a lower priority than \(C_i\). In this situation, case 1b
should be applied.

- Approach 2: a contention mechanism between $C_i$ and $C_k$ should be defined in order to decide which one of them squats in resources from $C_j$. For example, a simple priority contention protocol can be applied, in order to allow $C_i$ expelling $C_k$ from resources of $C_j$ and later on squatting them. Consequently, $C_k$ will have to initiate a new squatting process over the resources of $C_{j-1}$. This approach is considered only for future study basis.

2) Single Class Hard Squatting (SCHS)

**Case 2.1 (adjacent classes):**

$C_i$ ($i\in[1..N-1]$) squats in idle resources of $C_{i+1}$ ($i\in[1..N-1]$). Note that $C_{i+1}$ is the immediate LoPri class for $C_i$.

**Case 2.2 (non-adjacent classes):**

$C_i$ ($i\in[1..N-1]$) squats in idle resources of $C_j$ ($j > i$). Note that $C_j$ is one of the LoPri classes of $C_i$. Therefore, the latter is not restricted to squat in its immediate LoPri class. The squatter class ($C_i$) will select the first LoPri class that has available resources to be squatted (in this case, $C_j$), looking up from $C_{i+2}$ and decreasing priority up to $C_N$.

It must be noted that in both cases, $C_N$ cannot squat in resources, since no LoPri classes are defined.

**Special case:**

$C_i$ tries to squat in resources of $C_j$ ($j < i$) that have already been squatted by $C_k$ ($j < k < i$). Two approaches can be analysed for this special case:

- **Approach 1:** resources that are already squatted are considered as blocked, so that $C_i$ cannot squat in $C_j$, although $C_k$ (the squatter class already allocated in the resources of $C_i$) has a lower priority than $C_i$. In this situation, case 2b should be applied.

- **Approach 2:** a contention mechanism between $C_i$ and $C_k$ should be defined in order to decide which one of them squats in resources from $C_j$. For example, a simple priority contention protocol can be applied, in order to allow $C_i$ expelling $C_k$ from resources of $C_j$ and later on squatting them. Consequently, $C_k$ will have to initiate a new squatting process over the resources of $C_{j+1}$.

3) Single Class Kicking (SCK)

In this case, only one sub-case can be considered, since by definition, a given class cannot occupy busy resources from higher priority classes. This restriction is needed in order to be compatible with pre-emption and holding priorities in TE.

This case is as follows: $C_i$ ($i\in[1..N-1]$) kicks $C_{i+1}$ and occupies its resources, either partially or totally.

It must be noted that a given class, by the definition of kicking, cannot initiate a kicking process against HiPri classes, due to a lack of privileges and for the sake of compatibility with traffic engineering, as commented above.

4) Multiple Class Soft Squatting (MCSS)

This use case is a generalisation of case 1 for an environment where multiple classes coexist.

$C_i$ ($i\in[2..N]$) squats in, either partially or totally, as many HiPri classes as required in order to satisfy its demanded resources; starting from $C_{i-1}$ until $C_1$.

Therefore, class $C_i$ can squat in several HiPri classes at the same time. The total amount of squatted resources is the sum of the amounts gathered from each squatted class.

**Special case:**

$C_k$ ($k < i,j$) was squatted by $C_j$ and now $C_i$ ($j \neq i$) initiates a squatting process including resources from $C_k$. In this situation, we must dissect the problem into two different sub-cases:

- **Sub-case 1:** $j > i$
- **Sub-case 2:** $j < i$

However, the study of this special case is left for future work in the field.

5) Multiple Class Hard Squatting (MCHS)

This use case is a generalisation of case 2 for an environment where multiple classes coexist.

$C_i$ ($i\in[1..N-1]$) squats, either partially or totally, as many LoPri classes as required in order to satisfy its demanded resources; starting from $C_{i+1}$ until $C_N$.

Therefore, class $C_i$ can squat in several LoPri classes at the same time. The total amount of squatted resources is the sum of the amounts gathered from each squatted class.

**Special case:**

$C_k$ ($k > i,j$) was squatted by $C_j$ and now $C_i$ ($j \neq i$), initiates a squatting process including resources
from $C_k$. Again, this situation requires two different sub-cases to be studied:
- Sub-case 1: $j > i$
- Sub-case 2: $j < i$

However, the study of this special case is left for future work in the field, as the previous one.

6) **Multiple Class Kicking (MCK)**

This use case is a generalisation of case 3 for an environment where multiple classes coexist.

$C_i \ (i \in \{1..N-1\})$ kicks as many LoPri classes as required, either partially or totally, in order to satisfy its demanded resources; starting from $C_{i+1}$ until $C_N$.

Therefore, class $C_i$ can kick several LoPri classes from their allocation at the same time. The total amount of occupied resources is the sum of the amounts expelled to each kicked class.

**What if** $C_k$ was kicked by $C_j$ and now $C_i \ (j \neq i)$, initiates a kicking process against $C_k$?
  - Case 1: $j > i$
  - Case 2: $j < i$

7) **Combined situation**

This use case aims to define complex situations combining use cases from the previous ones, where multiple classes coexist.

According to the general approach presented before, a given class can gather resources from other classes using a combined strategy based on the following cases:
  - Perform case 4 (HiPri squatting)
  - Perform case 5 (LoPri squatting)
  - Perform case 6 (kicking)

**B. Considerations**

Some considerations must be taken into account for a good understanding of the previous use cases, which can be found in this subsection, as follows.

1) **Technique aggressiveness**

Squatting model, in any of its two variations, is a less aggressive technique than kicking. Therefore, in case of a class requiring extra resource allocation, squatting technique is generally preferred over kicking.

On the other hand, in order to avoid classes being fully expelled from their original resource assignment, thresholds for limiting the amount of resources squatted and kicked must be defined. It must be noted that this issue is not addressed in the present article, as it is considered as future work by the author.

2) **Hypothetical situations**

Basic cases keep class priority ordering, whereas “what if” situations may introduce alterations on them, which are specified. “What if” cases should be designed to determine what problems appear when considering specific situations that can be derived from the basic cases and, moreover, specify a protocol or procedure devoted to fix any situations where the general rules are not enough.

3) **Single vs. Multiple class situations**

In all cases, the situations for multiple class processes presented before extend the single processes. Therefore:
  - Use case 1 is included in case 4
  - Use case 2 is included in case 5
  - Use case 3 is included in case 6

The general approach for case 7 is derived from combining cases 1 to 6. If a given class requires more resources than allocated, it will follow a simple protocol to get them, as follows:
  - Firstly, the class considered will squat in resources from HiPri classes (non aggressive and better service policy).
  - Secondly, if more resources are required, the class will start squatting resources from LoPri classes (non aggressive and poorer service policy but still admitted).
  - Thirdly, if not enough resources can be gathered in the previous steps, the class will start kicking (aggressive and poorer service policy but still admitted).

4) **Restoration and Network Re-planning**

If a connection is closed, the associated class frees all the resources it was using. Therefore, pre-allocated, squatted and kicked resources are freed. Consequently, all remaining classes have to rearrange their allocated resources in order to keep them as close as possible to their native service policy.

At the time of writing this article, the restoration process is considered only to be triggered by a traffic class stopping its activity and being de-allocated from the resources it was using. However, other events provoking a restoration process are under study by the author, such as excessive resource use based on global resource usage indicators.
C. Example: “The hungry person”

A hungry person starts eating his/her own food, but it is scarce and rapidly runs out of it. If the person keeps on being hungry, it is always prepared to eat better quality foods, if granted. Therefore, a soft squatting is happening. However, the hungriness is considerable, and the person runs out of food again. In this case, he/she considers the possibility of eating lower quality food, in order to be satisfied. At this stage, a hard squatting situation is produced. Finally, the person is stroke by famine, being the previous feeding efforts completely insufficient. In this situation, the person may even feel like stealing low quality food, due to a less restrictive security policy for this good. Unfortunately, the person has incurred in a kicking process.

The example of the hungry person adequately illustrates the situations where soft/hard squatting and kicking may occur, reinforcing the idea of the generalisation of the model presented and its portability to many kinds of resources.

VII. THE POTENTIAL USABLE BANDWIDTH

This section defines the Potential Usable Bandwidth (PUB) and its analytical formulation, depending on the strategy considered (Squatting and/or Kicking). Firstly, the effects of the Squatting and Kicking strategies over the PUB are studied separately. Secondly, the combined effect is described in order to formulate a general expression for the PUB.

The Potential Usable Bandwidth quantifies the maximum amount of bandwidth resource a class of service can allocate in the network under certain circumstances, when squatting and/or kicking strategies are applied. Consequently, the PUB is an indicator of the receptiveness a multiclass network has towards a specific class of traffic.

Another characteristic feature of the PUB is that it is proportional to the bandwidth resource amount the class can consume from the network. The higher the PUB for a specific class is, the more resources the class will be able to allocate, if needed.

A. Effects of the squatting strategy over the PUB

When studying the effects of the squatting strategy over the PUB, both soft and hard squatting are taken into account. In this situation, the bandwidth contributions to the PUB are the following:

- Initially assigned bandwidth resources for the class \(A_i\).
- Soft-squatted bandwidth resources \((BW_{SS})\).
- Hard-squatted bandwidth resources \((BW_{HS})\).

Analytically, the PUB for a class \(i\) is the sum of the previously mentioned bandwidth contributions, as follows:

\[
PUB_i = A_i + BW_{SS} + BW_{HS}
\]

Whereas the first contribution, \(A_i\), is defined constant due to its nature, both \(BW_{SS}\) and \(BW_{HS}\) obey to cumulative bandwidth resource expressions. The soft-squatted bandwidth resource agglutinates all available bandwidth resources in higher priority classes, from 1 (highest priority class) to \(i-1\) (upper immediate adjacent class to squatter class). This is obtained by summing the difference between the reserved bandwidth for the object class \((A_i)\) and the minimum between the allocated and the reserved bandwidths for the same class, at each iteration step:

\[
BW_{SS} = \sum_{i=1}^{i-1}(A_i - \min(S_i, A_i))
\]

The minimum is forced using a bandwidth constraint in order to obtain a stable system, where the resources available for squatting are always positive or zero, but never negative.

Similarly, the hard-squatted bandwidth resource is obtained. In this case, \(BW_{HS}\) performs the same sum operation but altering the class range, which is now defined from class \(i+1\) (lower immediate adjacent class to squatter class) up to \(N\) (lowest priority class).

\[
BW_{HS} = \sum_{i=1}^{N}(A_i - \min(S_i, A_i))
\]

As a result, we can obtain an expression for the PUB substituting 6 and 7 in 5:

\[
PUB_i = A_i + \sum_{l=1}^{i-1}(A_l - \min(S_l, A_l)) + \sum_{l=i+1}^{N}(A_l - \min(S_l, A_l))
\]

The sums of the previous equation can be simplified as follows:

\[
PUB_i = A_i + \sum_{l \in (1,N), l \neq i}(A_l - \min(S_l, A_l))
\]
B. Effects of the kicking strategy over the PUB

In case only the kicking strategy is applied, the bandwidth contributions to the PUB are the following:
- Initially assigned bandwidth resources for the class ($A_i$).
- Kicked bandwidth resources ($BW_K$).

Since the kicking strategy consists of expelling lower priority classes from its assigned bandwidth (cf. section IV), the kicked bandwidth resources are computed as the sum of the reserved bandwidth for the class plus all the reserved bandwidths corresponding to lower priority classes:

\[ PUB_t = A_i + \sum_{l=i+1}^{N} A_l \] (9)

C. Effects of the squatting and kicking strategies over the PUB (generalised case)

In a generalised case, we consider four different bandwidth resource contributions to the PUB:
- Initially assigned bandwidth resources for the class ($A_i$).
- Soft-squatted bandwidth resources ($BW_{SS}$).
- Hard-squatted bandwidth resources ($BW_{HS}$).
- Kicked bandwidth resources ($BW_K$).

However, it must be observed that the bandwidth resources a class can get from a hard squatting action are equally gettable by applying the kicking strategy. This happens because $S_i$ is computed as the allocated bandwidth for a class, given its original reservation $A_i$. Thus, in order to avoid duplicity in the analytical model $BW_{HS}$ is not reflected in the formulation, although it may exist during normal operation conditions in the network.

Therefore, the generalised analytical model for the PUB is formulated as follows:

\[ PUB_t = A_i + BW_{SS} + BW_K = A_i + \sum_{l=1}^{N} (A_l - \min(S_l, A_l)) + \sum_{l=i+1}^{N} A_l \] (10)

It must be noted that a finer-grained formula for PUB can be calculated from how classes transfer the allocated bandwidth resource among them, as a result of the soft-/hard-squatting and kicking techniques. This can be done by analysing the cross-allocation parameter $S_{ij}$, since it provides a more accurate trace of bandwidth transfers among classes. Nevertheless, that analysis is not in scope of this article.

VIII. SIMPLE AUTO-PROVISIONING MODEL BASED ON PER-CLASS SQUATTING AND KICKING

This section presents a simple auto-provisioning model where a given class from a DS-TE pool implements the squatting and kicking strategies. Although it does not model all situations presented in the previous sections, it is compatible with all the use cases previously introduced.

A. Assumptions

On the one hand, for simplicity in this article, we assume the initial bandwidth resource assignment for all classes to be constant and equal for all classes:

\[ A_i = 1/N \quad \forall i \] (11)

This assumption equally distributes the link capacity (Max-RBR) among all classes of service, which facilitates the graphical representation and evaluation of the behaviour per class. We leave for future work the evaluation of a general case.

On the other hand, besides the reserved bandwidth, this subsection elaborates on the demanded bandwidth resource per class, $G_i$. In general, $G_i$ is a function $f$ characterised by the traffic pattern of the class, being it an external input to the multiclass network. This article only evaluates situations where traffic demand patterns follow a linear approach, for simplicity again. Let’s assume that class $i$ increases $G_i$ linearly with a slope $s$, being $g_i$ the independent demand variable for class $i$:

\[ G_i = f(g_i) = s \cdot g_i \] (12)

The demand function is considered unbound, so that any class can demand as much bandwidth resource as desired.

On the contrary, the BC for the $i$-th class ($BC_i$) is the upper bound for the allocable bandwidth ($S_i$). Typically, multiclass networks fix $BC_i$ to the reserved bandwidth $A_i$. In this section, I will name $BC_i$ as $S_i^{\text{max}}$. It must be noted that if the index $i$ refers to the subject class (squatter or kicker), then $S_i^{\text{max}}$ is non-existent. Oppositely, if the index $i$ refers to the target class (squatted or kicked), then $S_i^{\text{max}}$ is bounded to $A_i$.

Hence:

\[ S_i^{\text{max}} = A_i \quad (\forall i \neq \text{subject class}) \] (13)
B. Squatting strategy applied

As presented before, the analysis of the squatting strategy is performed considering both soft- and hard-squatting.

1) Soft squatting

In this scenario, we consider \( C_N \) to be the subject class, that is, the squatter class. This allows the \( N \)-th class to increase its allocated over the initial reserved, if needed.

In this case, given the assumption in eq. (13):

\[
S_i = \begin{cases} 
  s \cdot g_i & \text{if } g_i < A_i/s \\
  A_i & \text{if } g_i \geq A_i/s 
\end{cases} \quad \forall i \neq N
\]

This constrained allocation function considers a hard bandwidth resource demand limit \((S_i^{\text{max}})\) and applies to all classes of service except for the \( N \)-th class (minimum priority), which is allowed to increase its bandwidth resource demand without limitation.

As a consequence, the Potentially Usable Bandwidth for all classes but the \( N \)-th can be defined as:

\[
PUB_i = A_i \quad \forall i \neq N
\]

This means that all classes have a limited amount of bandwidth resources to be used, always under its initially reserved bandwidth amount, since the limitation \( S_i^{\text{max}} \) is hard.

At a given point in time, \( G_N \) increases over the initially reserved value \((A_N)\), which means that \( G_N \) needs extra resources in order to raise \( S_N \) up to the demanded rate \( g_N \).

From the expression in eq. (14) we can easily find that the aggregated, normalized bandwidth available for squatting by class \( N \) is:

\[
\sum_{i \neq N} (A_i - S_i)
\]

If we consider all classes but the \( N \)-th have the same homogeneous bandwidth limit and all behave identically (slope \( s \) is constant and all classes increase demand synchronously, \( g_i = g \) \( \forall i \neq N \)), equation (16) can be re-written as follows:

\[
(N - 1) \cdot \left( \frac{1}{N} - s \cdot g \right)
\]

Therefore, the total PUB for class \( N \) corresponds to the self-owned one plus the squatted one. Basing on equation (11), the \( PUB_N \) can be written as follows:

\[
PUB_N = \frac{1}{N} + (N - 1) \cdot \left( \frac{1}{N} - s \cdot g \right)
\]

This trivially leads to the simplified expression:

\[
PUB_N = 1 - (N - 1) \cdot s \cdot g
\]

Under normal operation conditions, the allocated bandwidth for the \( N \)-th class corresponds to its linear demanded bandwidth. In case of excessive demand, the allocated bandwidth could escalate as high has the \( PUB_N \). And consequently we can specify the total allocated bandwidth resource for class \( N \), if squatting available, as:

\[
\begin{cases} 
  S_N = s \cdot g_N & \text{for } g_N \leq \frac{1}{s} - g \cdot (N - 1) \\
  S_N \leq 1 - (N - 1) \cdot s \cdot g & \text{for } g_N > \frac{1}{s} - g \cdot (N - 1)
\end{cases}
\]

When comparing \( S_i \) \( \forall i \neq N \) in eq. (15) and \( S_N \) in eq. (20), we can appreciate the effect of the squatting technique on the bandwidth resource allocation function for class \( N \), which escalates much higher than all other classes thanks to a much higher bound for \( PUB_N \). Figure 1 shows this effect.

![Figure 1. Per-class normalised, allocable bandwidth resource using the Soft Squatting strategy \((S_i)\), compared to non-squatting classes \((S_i, i\neq N)\).](image-url)
2) **Hard Squatting**

Analogously to soft-squatting presented before, the hard-squatting implements the squatting technique but considering that the higher priority class suffers an increase on its demand and starts occupying idle resources from lower priority classes.

Assuming that all the classes are incrementing the bandwidth in the same way, linearly up, under the same context conditions than in the soft-squatting scenario, we can conclude that the potentially usable bandwidth in the same way, class increased its demand over the reserved resources from lower priority classes.

Since class 1 can occupy all the resources from all other classes due to the fact that it is the highest priority class.

As a result of the kicking strategy, the allocable bandwidth resource for the 1st class is:

\[
\begin{align*}
S_1 &= s \cdot g_1 & \text{for } g_1 \leq \frac{1}{s} - g \cdot (N - 1) \\
S_1 &\leq 1 - (N - 1) \cdot s \cdot g & \text{for } g_1 > \frac{1}{s} - g \cdot (N - 1)
\end{align*}
\]

And it has an identical graphical representation to equation (20), considering the new indexes. Figure 2 shows this hard-squatting scenario, where class 1 squats in idle resources from any other class.

![Figure 2. Per-class normalised, allocable bandwidth resource using the Hard Squatting strategy (\(S_i\)), compared to non-squatting classes (\(S_i, i \neq 1\)).](image)

**C. Kicking strategy applied**

As commented in previous sections, the kicking strategy is an aggressive mechanism that permits classes of service expelling lower priority ones from their allocated bandwidth resources in case of a resource demand that exceeds the class’s reservation plus any other extra resource available (for instance, extra resources gained as a result of a squatting process). Considering this, we define a scenario where the contour conditions are similar to the Hard Squatting case before, in which the highest priority class increased its demand over the reserved resources.

In this context, the allocable resources to any class but the 1st class are kept as in equation (15). The potentially usable bandwidth resource for class 1 is defined as:

\[
PUB_1 = 1
\]

Since class 1 can occupy all the resources from all other classes due to the fact that it is the highest priority class.

As a result of the kicking strategy, the allocable bandwidth resource for the 1st class is:

\[
\begin{align*}
S_1 &= s \cdot g_1 & \text{for } S_1 \leq A_1 \\
S_1 &\leq A_1 + \sum_{i \neq 1} k_{1i} \cdot s \cdot g_i & \text{for } S_1 > A_1
\end{align*}
\]

Where \(k_{1i}\) is the so called *kicking parameter*. In eq. (23), the kicking parameter is the unitary portion of the bandwidth resources that were previously allocated to class \(i (i \neq 1)\) and are now occupied by class 1 (kicked resources). In general, the kicking parameter \(k_{ij}\) provides the notion of which proportion of the resources allocated to class \(j\) have been kicked out, and thus occupied, by class \(i\).

It cannot be obviated the effect the kicking parameter has over equation (15), which has to be re-defined for the kicking scenario as follows:

\[
S_i = \begin{cases} 
(1 - k_{ji}) g_i & \text{for } g_i \leq A_i/s \\
A_i \cdot (1 - k_{ji}) & \text{for } g_i > A_i/s
\end{cases}
\]

If and only if: \(i \neq 1\) and \(j \neq i\).

Figure 3 shows a graphical representation of the total allocated bandwidth resource per class, where class 1 is progressively performing kicking against all other classes.

The values marked as \(x_i\) on the horizontal axis represent the bandwidth resource amounts at which a given class \(i\) will be completely expelled from its allocated resources. They can be recursively calculated as follows:

- \(x_2 = A_i/s + \frac{A_i}{k_{12}} = A_i \cdot \left(\frac{1}{s} + \frac{1}{k_{12}}\right)\)
- \(x_3 = x_2 + \frac{A_i}{k_{13}} = A_i \cdot \left(\frac{1}{s} + \frac{1}{k_{12}} + \frac{1}{k_{13}}\right)\)
- \(x_4 = x_3 + \frac{A_i}{k_{14}} = A_i \cdot \left(\frac{1}{s} + \frac{1}{k_{12}} + \frac{1}{k_{13}} + \frac{1}{k_{14}}\right)\)
- \(\ldots\)

We can easily extract a general term for generating
these values by isolating the common factors, given that $A_i$ is constant and equal for all classes in this example:

$$x_i = A_i \cdot \left(\frac{1}{s} + \sum_{i=2}^{N} \frac{1}{k_{i1}}\right)$$

(25)

![Figure 3. Per-class normalised potentially usable bandwidth (PUB) using the Kicking strategy in $S_1$, compared to PUB from kicked classes ($S_i, i\neq 1$).](image)

As a final remark, this kicking scenario described has potential class starvation consequences, that is, lower priority classes may suffer of scarce or null access to their reserved resources in case of highly demanding higher priority classes. This problem is easily fixable by introducing the “kicking threshold”, which defines an upper-bound value for the kicking parameter (similarly to the squatting thresholds commented in previous sections). In any case, the definition and study of the kicking threshold are not under scope in this article and is considered as future work.

IX. SIMPLE CASE EVALUATION AND RESULTS

In this section, we consider a multicloud network where three classes of service coexist: $C_1, C_2$ and $C_3$ $(N = 3)$, enumerated from highest to lowest priority.

The Potentially Usable Bandwidth is evaluated for the three of them as a result of implementing the squatting and kicking strategies.

By default, all initial reservations ($A_i$) and demand functions ($G_i$) are defined so that:

$$A_i = \frac{1}{3} \quad \forall i$$

(26)

And:

$$G_i = s \cdot g_i \quad \forall i$$

(27)

Since the demand function should never overpass the initial reservation when all classes are under-allocating resources, we consider that:

$$(28.a) \quad G_i \leq A_i \quad \forall i$$

So that:

$$(28.b) \quad s \cdot g_i^{max} = \frac{1}{3} \Rightarrow g_i^{max} = \frac{1}{3s} \quad \forall i$$

Consequently, the bandwidth resource allocation function and PUB for all three classes can be characterised as follows:

$$(29.a) \quad S_i = \begin{cases} s \cdot g_i & \text{if } g_i \leq 1/3s \\ 1/3 & \text{if } g_i > 1/3s \end{cases} \quad \forall i$$

$$(29.b) \quad PUB_i = 1/3 \quad \forall i$$

As we can observe, the bandwidth resource shares for each class are hardly delimited, and each class of service can only use a bandwidth amount less or equal to its initial reservation. Since this approach is highly inefficient in terms of total link bandwidth usage and blocking probability, we study how the strategies proposed previously in this article improve the usage rates of all classes in the following sub-sections.

A. Soft-squatting strategy applied

Firstly, we evaluate the PUB per class when a soft-squatting strategy is implemented.

Class 1 is not able to perform soft-squatting, since no higher priority classes can be found. In this case, its PUB is equal to its initial reservation:

$$(30) \quad PUB_1 = A_1 = 1/3$$

Class 2 is able to gather resources from class 1, in case of need. Thus, its PUB is formulated as follows:

$$(31.a) \quad PUB_2 = A_2 + (A_1 - S_1) = 2/3 - S_1$$

In the best case ($S_1 = 0$), we obtain:

$$(31.b) \quad PUB_2 = 2/3$$

As a final note, the bandwidth resource allocation function and PUB for all three classes can be characterised as follows:

$$(29.a) \quad S_i = \begin{cases} s \cdot g_i & \text{if } g_i \leq 1/3s \\ 1/3 & \text{if } g_i > 1/3s \end{cases} \quad \forall i$$

$$(29.b) \quad PUB_i = 1/3 \quad \forall i$$

As we can observe, the bandwidth resource shares for each class are hardly delimited, and each class of service can only use a bandwidth amount less or equal to its initial reservation. Since this approach is highly inefficient in terms of total link bandwidth usage and blocking probability, we study how the strategies proposed previously in this article improve the usage rates of all classes in the following sub-sections.
And finally, class 3 is able to squat in resources from both classes 1 and 2. Therefore, its PUB is formulated as:

\[ (32.a) \quad PUB_3 = A_3 + (A_1 - S_1) + (A_2 - S_2) = 1 - S_1 - S_2 \]

In the best case \((S_1 = S_2 = 0)\), we obtain:

\[ (32.b) \quad PUB_3 = 1 \]

### B. Hard-squatting strategy applied

Secondly, we evaluate the PUB per class when a hard-squatting strategy is implemented.

Class 1 is now able to perform hard-squatting. In this case, its PUB can be expressed as follows:

\[ (33.a) \quad PUB_1 = A_1 + (A_2 - S_2) + (A_3 - S_3) = 1 - S_2 - S_3 \]

In the best case \((S_2 = S_3 = 0)\), we obtain:

\[ (33.b) \quad PUB_1 = 1 \]

Class 2 is able to gather resources from class 3. Thus, its PUB is formulated as follows:

\[ (34.a) \quad PUB_2 = A_2 + (A_3 - S_3) = 2/3 - S_3 \]

In the best case \((S_3 = 0)\), we obtain:

\[ (34.b) \quad PUB_2 = 2/3 \]

And finally, class 3 is not able to squat in any bandwidth resources, since no lower priority classes exist. Therefore, its PUB is formulated as:

\[ (35) \quad PUB_3 = A_3 = 1/3 \]

### C. Kicking strategy applied

Thirdly, we evaluate the PUB per class when a kicking strategy is implemented.

Class 1 is now able to perform kicking over all lower priority classes. In this case, its PUB can be expressed as follows:

\[ (36) \quad PUB_1 = A_1 + A_2 + A_3 = 1 \]

Class 2 is able to kick class 3 from its allocated resources. Thus, its PUB is formulated as follows:

\[ (37) \quad PUB_2 = A_2 + A_3 = 2/3 \]

And finally, class 3 is not able to expel any lower priority class from its reserved bandwidth resources, since no lower priority classes exist. Therefore, its PUB is formulated as:

\[ (38) \quad PUB_3 = A_3 = 1/3 \]

### D. Evaluation summary

In table 1 we show the potentially usable bandwidth resources, in percentage over the total of the link, for all three classes in the different scenarios described before: non-sharing, soft-squatting, hard-squatting and kicking.

What refers to class 1 (HiPri), it dramatically increases its PUB when using either hard-squatting or kicking. By contrast, no improvement over the non-sharing strategy is shown using soft-squatting.

Regarding class 2, it is equally benefited from all strategies. Class 2 is able to double its PUB with respect to the non-sharing scenario, independently of the strategy chosen.

Finally, class 3 is only able to boost its PUB using the soft-squatting strategy. No improvement over the non-sharing scenario is observed when neither hard-squatting nor kicking strategies applied.

<table>
<thead>
<tr>
<th>Class 1</th>
<th>Non-sharing</th>
<th>Soft-squatting</th>
<th>Hard-squatting</th>
<th>Kicking</th>
</tr>
</thead>
<tbody>
<tr>
<td>33 %</td>
<td>33 %</td>
<td>100 %</td>
<td>100 %</td>
<td></td>
</tr>
<tr>
<td>Class 2</td>
<td>33 %</td>
<td>67 %</td>
<td>67 %</td>
<td>67 %</td>
</tr>
<tr>
<td>Class 3</td>
<td>33 %</td>
<td>100 %</td>
<td>33 %</td>
<td>33 %</td>
</tr>
</tbody>
</table>

**Table 1.** Potentially Usable Bandwidth resource (PUB) in percentage, per class, in four different scenarios: not sharing strategy, soft-squatting applied, hard-squatting applied and kicking applied, in columns, from left to right.

Basing on the table above, we can evaluate the improvement of the proposed techniques by considering the link has eight possible states, depending on which class or classes are being transmitted at different times:

- Link is empty one out of eight possibilities,
- One class is being transmitted in three out of eight possibilities,
- Two classes are being transmitted in three out of eight possibilities, and
- Three classes are being transmitted in one out of eight possibilities.

The squatting and kicking techniques do not have any effect in the first situation, since no traffic is transmitted. In all the other situations, the effects are as follows:

- Class 1 is able to triplicate its allocated bandwidth in the 50% of the cases,
- Class 2 is able to duplicate its allocated bandwidth in the 25% of the cases and triplicate it in another 25% of the cases, and
- Class 3 is able to duplicate its allocated bandwidth in the 25% of the cases and triplicate it in the 12.5% of them.

Compared to a non-sharing scenario, the squatting and kicking techniques improve the link utilisation in the 87.5% of the cases.

It must be noted that, in the best effort Internet, the lowest priority class is always transmitted with no guarantees. In the squatting and kicking scenarios, the best effort class maps directly to the lowest priority class using a soft-squatting mechanism.

X. CONCLUSION AND FUTURE WORK

In this article, the Squatting and Kicking strategies and their special variations, Soft and Hard squatting, have been presented as means for efficient bandwidth resource utilisation in multiclass networks.

Squatting is a new strategy devoted to pacifically make a more efficient network utilization by allowing classes of service to use idle resources from others classes, following a set of rules based on class prioritisation. On the other hand, Kicking is an aggressive alternative to get resources from lower priority classes, which is based on expelling lower priority classes from their allocated resources in order to satisfy current class’s demand.

Additionally, a set of both simple and complex scenarios, also named use cases in the article, have been presented. A scheme to use the Squatting and Kicking mechanisms has been formally formulated. Additionally, a simple analysis has been presented, showing the incremented bandwidth resource allocation for the squatter/kicker class in comparison to the squatted/kicked classes.

The squatting and kicking techniques presented in this article can be generalised for being able to operate not only for bandwidth resource allocation, but also for generic resource management. The author envisions that this algorithm can be adapted to any context where resources require reservation plus allocation phases, among different entities. For example, businesses, power line usage patterns, food prices and many others can apply an adapted version of the generalised squatting and kicking strategies, to be defined in future work.

What specifically refers to the increased allocable bandwidth resources for squatter/kicker classes, we can conclude that the low priority classes can use the idle bandwidth from other classes by means of initiating a squatting process. Therefore, the potentially usable bandwidth for these classes is greater than the originally reserved. Alternatively, when using the kicking mechanism an analogue conclusion can be derived, but the benefits are now for the higher priority classes. In any case, the need for defining squatting and kicking thresholds has been identified, in order to prevent some classes dominating resources by means of uncontrollably applying either or the other techniques.

For the future work, the aforementioned thresholds in squatting and kicking should be defined for avoiding resources beat down for any class of service and for guaranteeing a minimum always-allocable bandwidth from the initial reserved one. Moreover, consecutive squatting using several classes should be studied, and convergence and stabilization time must be analyzed. Additionally, the strategy when the owner of a squatted or kicked resource requires the bandwidth back to its possession should be studied. Crank-backing mechanisms are foreseen to be complex, but a more accurate view is only possible after studying the behaviour of a system implementing both techniques at the same time.

Laterally, what refers to the bandwidth constraints models, MAM, MAR and RDM do offer good bandwidth resource efficiency thanks to allowing the classes of service sharing the whole link capacity. However, none of these three models permit optimising resource usage on a per-class basis, but rather on an aggregate basis, such as native traffic engineering does.

Finally, some other issues to be studied in future work are: analysis of the convergence and stability derived from the combined used of the squatting and kicking techniques; definition of squatting and kicking thresholds to keep a minimum QoS per-class service and definition of the performance indications and evaluators for a comprehensive scalability and convergence characterisation.
ACKNOWLEDGEMENT

The author thanks Xavier Hesselbach for his valuable contribution to the definition of the concepts and the analysis of the strategy proposed. Additionally, the author thanks Sergi Figuerola and the i2CAT Foundation for having given him the chance to participate in the first Enigma project. This work has been indirectly and partly supported by the Enigma, Enigma Enhanced and E3MS projects consortia and the Spanish Ministry for Industry, Tourism and Commerce (MITyC), 2006-2008.

REFERENCES