Achieving Bursty Traffic Guarantees by Integrating Traffic Engineering and Buffer Management Tools

Miriam Allalouf and Yuval Shavitt

School of Electrical Engineering,
Tel Aviv University
{miriama, shavitt}@eng.tau.ac.il

Abstract. Traffic engineering tools are applied to design a set of paths, e.g., using MPLS, in the network in order to achieve global network utilization. Usually, paths are guaranteed long-term traffic rates, while the short-term rates of bursty traffic are not guaranteed. The resource allocation scheme, suggested in this paper, handles bursts based on maximal *traffic volume allocation* (termed *TVAfB*) instead of a single maximal or sustained rate allocation. This translates to better SLAs to the network customers, namely SLAs with higher traffic peaks, that guarantees burst non-dropping. Given a set of paths and bandwidth allocation along them, the suggested algorithm finds a special collection of bottleneck links, which we term the *first cut*, as the optimal buffering location for bursts. In these locations, the buffers act as an additional resource to improve the network short-term behavior, allowing traffic to take advantage of the under-used resources at the links that precede and follow the bottleneck links. The algorithm was implemented in MATLAB. The resulted provisioning parameters were simulated using NS-2 to demonstrate the effectiveness of the proposed scheme.

1 Introduction

The latest Internet QoS (Quality of Service) design trends combine two approaches: DiffServ and MPLS. The first is based on reducing the computation complexity in core routers and on locating QoS entities such as policing and metering at the network edges. The DiffServ approach is based on per-hop QoS handling. In order to achieve global QoS guarantees or global profit gain, TE (Traffic engineering) tools are applied to design a connection-oriented network, e.g., using MPLS. In particular, QoS routing, where routes are assigned according to the service requirements, is an essential part to the end-to-end guarantees. Usually, the guarantees are applicable for long-term traffic rates, whereas the short-term rates of bursty traffic are not handled or guaranteed. This paper suggests a per-aggregate resource allocation algorithm that takes into account average traffic rates and also absorbs traffic bursts.

We consider as input a connection-oriented network where topology and directional link capacities are known. A typical rate demand of the network customer may represent aggregates of connections (e.g., TCP), such as client traffic (university campus, business client, client ISP), ATM VPs, or MPLS tunnels, and will be expressed by average or maximum required rate. The attitude of our resource allocation concept is to offer the network customers better SLAs with higher traffic peak rates that guarantees bursty traffic. It is a fast off-line algorithm that is performed during the network design phase.

F. Boavida et al. (Eds.): NETWORKING 2006, LNCS 3976, pp. 63–74, 2006.

Our resource allocation algorithm has two stages. In the first stage it seeks any QoS routing or bandwidth allocation algorithm that saturates the networks, such as maximum flow or max-min fair allocation [1]. Such algorithms use long-term average traffic demands as input, and allocate bandwidth using a signle rate parameter. In the second stage, we use buffers at specific locations for the short term traffic management, using the output of the long term TE algorithm. Note that we are not proposing to change the hardware whenever the demands are changed. All the routers will have their initial buffering resources, but our algorithms will use them optimally according to topology and demands analysis. These buffer analysis will determine the required flow regulation parameters at the edges of the network in order to enforce that traffic adheres to its designated maximal rate, while still isolating flows from each other. Specifically, we push the burst treatment to a point we term the *first cut*, which is an optimally selected set of bottleneck links. A burst is allowed to proceed unshaped until the destination, given the bottleneck link is not congested. In case of congestion the traffic is shaped at the *first cut* to the highest possible rate which guarantees the burst will not interfere with other flow traffic. Anyhow, the adjusted rate is never lower than the average rate determined by the long-term TE algorithm. Our algorithm determines provisioning parameters for the policy and regulation entities that are located at the edges of the network.

There are various methods for deterministic bandwidth allocation where the bandwidth is allocated using a single parameter, the maximal rate or the sustained rate parameter. The solutions of the different variants of the multi-commodity flow (MCF) problem for traffic engineering can be viewed as a long-term rate allocation method. Nichols *et al.* [2] describe two allocation methods for the DiffServ framework. The 'Premium service' is where the traffic is shaped at network edges. It provides the maximal permitted rate allocation contracts to its users, and it smoothes the jitter, provides certain delays, and guarantees peak rate flows. The 'Assured service' relies on statistical guarantees.

Other deterministic rate guarantees that consider the short-term rates [3, 4, 5, 6] were achieved by either the worst-case bounds on network internal buffer overflow or by end-to-end delays in the network. The rates of these traffic envelopes are not tight since they consider the worst-case bounds. A different line of research suggests statistical allocation guarantees. Christin *et al.* [7] examined the per-hop behavior of various real time streams having different constraints (such as delay or loss rate). Liebeherr [8] discusses different resource allocations and scheduling methods for the provision of delay sensitive video streams. Another approach is to allocate bandwidth according to an effective rate that takes into account statistical multiplexing between the burstiness of the flows [8, 9, 10]. Biton and Orda [11] provide QoS guarantees by coupling the scheduling mechanism and the routing schemes.

The resource allocation algorithm we propose in this paper reserves bandwidth according to the amount of traffic sent during a time interval (termed **TVAfB**, *maximal traffic volume allocation*) and not according to a single strict rate allocation (termed **MRA** in this work) used in previous suggestions.

The **TVAfB** cascading algorithm improves the state-of-the-art of service allocation and provisioning in a few ways. It allows bursty traffic to better exploit the existing network resources. It can also exploit the statistical multiplexing gain and still provides deterministic bandwidth and delay guarantees. For example, a burst that belongs to a flow

that has only one bottleneck link that finds no congestion at this link can be transmitted further without any delay. In case of a higher load, but still below capacity, it flows in a higher rate than its sustained rate with no loss danger. Only during periods of congestion the burst is shaped to its fair share. The novelty of this approach lies in our dealing with bursty traffic guarantees and the fact that it employs the buffer as an additional resource in traffic engineering design.

Further, our algorithm can lead to higher parameters assigned for policing and regulation without being restricted to any specific policy method. The mathematical derivations we present in this work concentrate on the case where traffic is policed at the edges using token buckets. However, the notation of first cut is important and can be used for other regulation scenarios, as well. Section 2 presents the problem. Section 3 outlines the two-stage algorithm where section 4 details the second algorithm. Section 5 describes the simulation results and evaluates this proposition.

2 Problem Presentation

The algorithm considers a connection-oriented network where topology and directional link capacities are known. The set of paths are set optimally using any bandwidth allocation criterion chosen by the network administrator. We model the network as a general directed graph where each arc label represents link capacity. The traffic flow is assumed to be bursty, though the peering networks cannot explicitly express the burstiness characteristics. It is regulated by token buckets at the edge nodes. The token bucket parameters we seek per customer demand are token rate and bucket size. The regulation using these parameters determines the committed rate, the peak rates and the maximum burst size per path (CIR, PIR, and CBS). Our goal is to set the SLA regulation parameters in order to maximize the burstiness each flow is allowed, while at the same time not dropping packets by optimally use buffers along the routes. We will show that it increases bandwidth utilization for this type of traffic compared to the maximal rate allocation (**MRA**) that is usually used for long-term guarantees. Our algorithm shows that for many scenarios, there are paths with only one bottleneck link per path. In these cases, if buffers are allocated in this set of bottleneck locations, higher rate traffic per-path can be allowed to enter the network.

To illustrates the problem, Figure 1 depicts a simple directed network with 4 unidirectional paths. There are 4 different clients each with a demand of 1Mbps as depicted. All link capacities are 4Mbps. Thus, the bandwidth reservation is 4Mbps on link $e7$, 2Mbps on links $e5$ and $e6$, and 1Mbps on links $e1$, $e2$, $e3$, and $e4$, respectively. It is maximally allocated because link $e7$ is saturated. If a burst with peak rate of 2Mbps is sent along path $r1$, the packets exceeding 1Mbps will be dropped, though links $e1$ and $e5$ are not fully used. The rational behind our approach is to

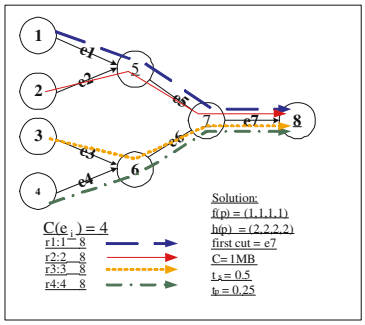


Fig. 1. Example 1

exploit links $e1$ or/and $e5$ capacity limits and still guarantee the traffic at the bottleneck, which in this case is link $e7$. By using another resource we can define extended allocation using more parameters, increase the usage of the under used links, and assign more flexible contracts. A 1Mbit buffer at the output port of node 7 to link $e7$ enables an agreement of 2Mbps peak rate, 1Mbps sustained rate and maximum burst time of 0.25 second for each path. The burst size for each path can grow as high as 2Mbit for a period of 0.25 seconds providing it is followed by a silence period of 0.25 seconds. Now consider an underload situation where only one client transmits bursty traffic of 2Mbps peak rate. This stream will be transmitted without any buffering delay all the way. Otherwise, if all the sources transmit using their peak rate, the buffer at node 7 will shape (using any GPS-compliant scheduler) the traffic per path to the sustained rate.

3 Algorithmic Solution

Below is an outline of the algorithm that achieves deterministic guarantees for bursty traffic. The algorithm is based on a few algorithms activated in cascade.

3.1 Solution Outline

1. 1^{st} **stage - Routing and Average Rate Allocation**: Find, using LP (Linear Program) formulation and solver, the QoS routing that identifies maximum flow (or other criterion) allocation of the bandwidth. The output is the set of paths and the net flow that is assigned per path. This stage is described in Section 3.2.
2. 2^{nd} **stage - TVAfB cascading algorithm - Traffic Volume Allocation for Bursts**:
   (a) Find a special set of bottleneck links, termed the *first cut* (Section 4.1).
   (b) Indicate which buffers at the *first cut* enable us to increase the rate at the edges.
   (c) Calculate the permitted peak rate over each path taking into account all the arcs not included in the *first cut* for each path. Again, we use LP solver over the residual graph 'before' and 'after' the first cut (Details in Section 4.2).
   (d) Based on the previous calculations, decide for each path whether it can gain additional burstiness using buffering. If yes:
      – Analyze buffer behavior at the bottleneck link, in case of congestion (4.3).
      – Set a contract (SLA) per-path (Section 4.4).

3.2 1^{st} stage: Long-Term Routing and Bandwidth Allocation

This stage specifies a set of paths in the network, and allocates them bandwidth. TE tools are used to choose paths between a given set of ingress-egress pairs. Any resource allocation criterion can be used, in order to saturate the network.

In this paper we are particularly considering the Maximum Multi-commodity Flow (MCF) problem. The input to this problem is the network topology, the directional links capacities, and a list of ingress-egress pair (clients). It finds the maximum of the total net flows over all commodities (e.g, paths), the routing to be used between each pair, and the net flow per each path. This problem can be solved using LP solver in a polynomial number of steps. We specifically consider this problem since it achieves network saturation and leaves minimal excess capacities. Other routing algorithms that

allocate bandwidth and saturate the network can also fit this framework. In [1] we suggested bandwidth allocation method according to the max-min fair criteria that can be used for the *TVAfB* algorithm.

4 2^{nd} Stage: TVAfB Cascading Algorithm - Traffic Volume Allocation for Bursts

4.1 The Bottleneck Links for Buffering Analysis

The 1^{st} stage solution found the set of paths between (s_i, t_i)-pairs and a per path net flow $f(P)$ in the graph $G(V, A)$. Based upon the routing found previously this subsection will find the strategic location for the buffers, which is defined below as *first cut*. First, we will define a few terms.

Definition 1. *A link a is saturated, denoted: $sat(a) = 1$ if it is assigned bandwidth equal to its capacity. Otherwise it is not saturated which is denoted $sat(a) = 0$.*

Definition 2. *a_i is the fbn link of a path $p = (a_1, a_2, \ldots), a_j \in A, if i = \min\{j | sat(a_j)\}$*

Definition 3. *A* first cut *is the set of the first bottleneck links ($fbns$).*

Definition 4. *Given a graph $G(V, A)$ and a set $(s_i, t_i), \forall i = 1..K$ of source-terminal pairs, a cut M of the graph is a subset $M \subset A$ such that the subgraph $G' = (V, A \setminus M)$ has no $s_i \rightarrow t_i$ path, $\forall i = 1..K$.*

Using the above definitions we can state the main construction of this subsection.
The *first cut* properties

1. Each path has exactly one fbn link. The number of fbn links $\leq$ the paths.
2. For each path, the links that are prior to its first bottleneck link are under-used.
3. Each *first cut* link can be saturated by flows that this link is their fbn link and by other flows that already met their fbn link before (discussed in 4.2).
4. The *first cut* is a cut of the graph. If we delete the arcs of the first cut no traffic will flow (The proof can be found in [12]). Thus, we can use it as the location for absorbing the peak rates of the bursts.

4.2 Peak Rates Calculations

The *Traffic Volume* allocation assigns peak rate $h(p)$ per path p on top of the sustained rate, $f(p)$, which was found in the 1^{st} stage. The lower bound for each $h(p)$ is $f(p)$. The goal of this work is to enable flow transmission over a predefined path using its peak rate when the buffer is used only in case of congestion. Therefore, the peak rates calculation is derived out of the excess bandwidth of the links, which are not saturated, and is divided among all the paths flowing through them.

This subsection calculates the possible peak rates per path in each first bottleneck link (fbn) subject to capacities constraints of all the preceding and following arcs over this path. For this purpose we use the same TE algorithm used in the first stage over the

residual graph arcs that reside 'before' and 'after the first cut. The specific TE algorithm (maximum flow, max-min fair, etc.) also determines how the excess bandwidth will be divided among the paths.

The construction of the 'before' and 'after' residual graph is as follows. According to property 2 of the *first cut*, all the links of path p prior to its fbn are under used and can accommodate higher rates than the sustained rate $f(P)$. However, property 3 is more complicated. Consider a link fbn_1 (belonging to the *first cut*) and a set of paths that are traversing it. Note that fbn_1 may not be the first bottleneck link for some of the paths that traverse it. Assume a path p_i which passes through the saturated links fbn_2 and fbn_1 in this order. By definition only fbn_2 is p_i's first bottleneck link. However, peak rates calculation, residual graph construction and buffer management vary if p_i has more than one bottleneck link. Essentially, this variation arises due to the need to allocate these peak rates along the arcs that lay between the bottleneck links (fbn_2 and fbn_1).

We developed two algorithms. The first, algorithm A, saves buffering resources by allowing burstiness (some peak rate) only for paths that traverse a single saturated link. The second, algorithm B, enables burstiness also for paths that traverse multiple saturated links, but requires more buffering resources. In both algorithms, shaping of the peak rate to the sustained rate is performed only when congestion occurs, otherwise, the flow's peak rate is allowed.

Peak Rate Calculation Algorithm A: Enabling Burst Flow only for Single-fbn Paths. The first algorithm benefits paths that traverse a single fbn link whose other links (not in the **first cut**) are under used. The excess bandwidth in the under-used links is divided among these paths, which permits a possible peak rate per path. Not every topology and demand flow can benefit from this algorithm, though the algorithm can check its usefulness. Section 5 discusses briefly the topologies that are likely to be beneficial by the algorithm. The traffic flow is controlled at the ingress, using the peak rate. Other traffic flows are controlled using the sustained rate. In case of congestion, buffers at the first cut will be used to shape the peak rate to a lower rate (but not lower than the sustained rate).

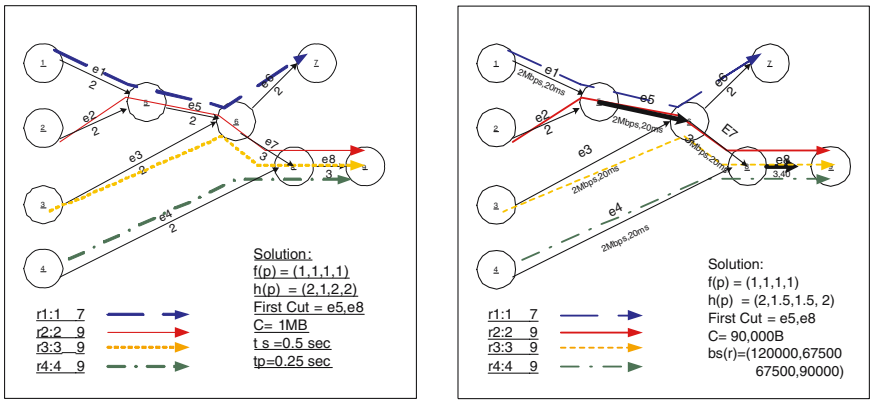
The input for this algorithm is the graph $G(V, A)$; its arc capacities; set of paths over G and the assigned net flows over them and the *first cut* arcs. The algorithm finds $h(P)$, the permitted peak rate per path in two steps. The first step constructs a sub-graph $G^-(V, A^-)$ (see in Figure 2). The second step applies the TE algorithm used in the 1^{st} stage over A^- and identifies the highest possible rates over the paths subject to A^- capacity constraints.

Consider the example in Figure 3(a), where the arc capacities of links $e1 - e6$ is 2Mbps and of links $e7 - e8$ is 3Mbps. The optimal bandwidth assignment per-path, calculated by the *first-stage TE* algorithm is 1Mbps. We consider this rate to be the sustained rate. The *first cut* consists of the links $e5$ and $e8$. Paths $r2, r3$, and $r4$ are traversing arc $e8$. Note that $fbn(r3) = fbn(r4) = e8$ but $fbn(r2) = fbn(r1) = e5$. Paths $r1, r3$, and $r4$ have only one *fbn* link, thus, their rate can be increased. Path $r2$, however, is excluded from the set of the beneficial paths because it has two bottleneck links and can not have burstiness. A^- contains links $e1$ and $e6$ (that precedes and follows $e5$ respectively), $e3, e4$, and $e7$ (that are prior to $e8$). The residual capacity of $e1, e3$ and $e4$ in A^- is 1 (originally was 2) and the capacity of $e7$ is 1 (originally 3). Buffer located

Constructing set of links A^-

1. Set $FPATHS$ to be the set of all input paths (from TE stage), $NEWFPATHS = FPATHS$
2. **for** each bottleneck link a in 'first cut':**do**
3. Set $FP(a)$ to be all the paths passing through a
4. **for** $f_i \in FP(a)$ **do** /* Consider only paths with single fbn */
5. **if** $a = fbn(f_i)$ and $\forall a_f \in f_i, a_f \neq a, a_f \notin firstcut - a$ **then**
6. **for each** $a_f \in f_i, a \neq a_f$ **do** $A^- = A^- \bigcup a_f$
7. **else** $NEWFPATHS = NEWFPATHS - f_i$
8. /* Get the residual graph : for the excess rates calculation */
9. **for each** $f_i \in FPATHS$ **do, for each** $a_f \in f_i$ **do** $c(a_f) = c(a_f) - f(f_i)$

Fig. 2. Algorithm A G^- construction: selecting links for the peak rate



(a) **(b)**

Fig. 3. Results of algorithms A and B for a network with various arc capacities

at the *first cut* links $e5$ and $e8$ absorbs the sum of the peak rates of the traversing paths (which is (2,1,2,2) for paths 1,2,3 and 4). The derivation of the maximum peak period per path that is allowed subject to the buffer size and the calculated peak rate is described in subsection 4.3. In this algorithm, each flow peak rate is only considered once in the buffers calculation, at its first bottleneck link. This means that our usage of the buffering resources is minimal and is not sensitive to whether the first cut is the minimum cut or what is the number of the links of the first cut. The maximal peak rate, R_p, that can be handled at each one of the first cut links is **not the sum** of the peak rates of the paths that traverses it, but is given by $\forall a \in A^-, R_p^a = \sum_{\text{already shaped paths}} f(p) + \sum_{a \text{ is their fbn}} h(p)$.

Peak Rate Calculation - Algorithm B: Enabling Bursts Flow for all the Paths, with more buffers. This algorithm enables peak rates assignment also to paths with more than one *fbn* link though this requires more buffering resources. As in algorithm A, we build a new sub graph $G^-(V, A^-)$ and apply the same TE algorithm on G^- to find $h(P)$, the per-path permitted peak rate. A^- consists of all the links except the *first cut* links. In this algorithm, assuming there is no congestion in the network, a flow of a path

that traverses more than one bottleneck link can reach the second bottleneck link with a higher rate than its sustained rate. Portion of the buffer in this *fbn* has to be assigned to guarantee the higher rates. Consequently, more buffering resources should be added at each first cut link to accommodate the peak rates.

Figure 3(b) shows algorithm B execution on the same graph used in Figure 3(a). The rate of path $r2$ can be increased even though it has two *fbn* links, and its peak rate is calculated using arcs $e2$ and $e7$. There will be 2 buffers: one located at node 5 towards $e5$ to treat bursts from routes $r1$ and $r2$ and the other is located at node 8 towards $e8$ to treat the bursts of routes 2,3 and 4. Assuming locating buffers of size 90,000 bytes at the output ports of nodes 5 and 8 towards links $e5$ and $e8$. The sustained rates are (1Mbps, 1Mbps, 1Mbps, 1Mbps), peak rates are (2Mbps, 1.5Mbps, 1.5Mbps, 2Mbps), and the sizes of the token buckets are (120,000, 67,500, 67,500, 90,000) bytes for routes (1, 2, 3, 4), respectively. The details of this calculations can be found in subsections 4.3 and 4.4. Note that the *fbn* link $e5$ allows a burst size of $90,000$ for path $r2$ but this burst size was decreased by the fbn $e8$ upper bound. As in the previous algorithm, in case where a path cannot gain a peak rate that is higher than its sustained rate, it will be policed to its sustained rate at the ingress. Otherwise, the peak rate will be used.

4.3 Buffer Management Analysis at the *First Cut*

The buffers, located at the **first cut**, are used for holding the bursts that may arrive with a maximal rate of $h(p)$ for any path p. The buffer sizes are determined by the peak rates calculated in 4.2. Given the shaping capabilities at the first cut, we can calculate the possible traffic envelopes at the first cut. The way we handle the traffic at the first cut affects the control parameters of the traffic at the ingress nodes. Many previous papers estimated the bounds on the size of traffic envelopes at the core based on the traffic pattern at the source nodes. Since our calculations are derived from the TE routing stage, we are able to set regulation rules at the ingress. Specifically, we assume the incoming flows are regulated per path using token buckets at their source node. We derive the per-path token bucket parameters (i.e., peak rate, sustained rate, and burst size) from the first cut buffer analysis. Figure 4 describes the node's functionalities with buffer capacity C, link output rate, R_{out}, peak rate of arriving traffic, $R_{peak,in}$, and a peak interval, t_p. The transmission rate of the outgoing traffic is bounded by the link output rate, R_{out}. If the rate of the offered traffic is $R_{in} \leq R_{out}$, a queue will not build up. In case of bursty traffic the buffer is used for storing the incoming packets which are smoothed by the transmission rate. The most extreme case is an On-Off streams in an interval t_s, which are composed of peak rate $R_{peak,in}$ for the burst duration t_p followed by a silence period of length $t_s - t_p$. The longest period of time t_p that a burst can be sent, given, $R_{peak,in}, R_{out}$ and C is expressed by:

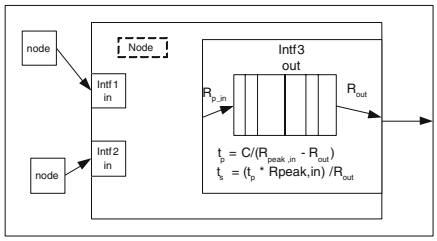


Fig. 4. Buffer management at the output port

$$t_p = C/(R_{peak,in} - R_{out}) \tag{1}$$

The minimal length of the interval t_s can be derived by equating the amount of incoming and outgoing data:

$$t_s = R_{in} \cdot t_p / R_{out} \tag{2}$$

Alternatively, we require that the generated amount of data v in the interval t_s: $v \leq R_{out} \cdot t_s$. The maximum delay at a node is given by the emptying time of a full buffer C/R_{out}. A general definition of v will be to integrate the arrival rate, given $g(t) \stackrel{\text{def}}{=} R_{in}(t)$: $\int_{t-t_s}^{t} g(t)d(t) \leq (R_{out} \cdot t_s)$ where t_s is calculated from using Eq. 1 and Eq. 2. We have shown that if the above parameters on the arriving traffic are kept, the traffic is guaranteed to be conforming. Next we will prove the correctness of traffic envelope bounds. Consider streams $i = 1, 2, \ldots$ with peak rates $h(p_i)$, sustained rates $f(p_i)$, and $\int_{t-t_s}^{t} g_i(t)d(t) \leq f(p_i) \cdot t_s$. The following Lemma states the conditions for conformance.

Lemma 1. ***Assuming*** *outgoing link rate R_{out}, permitted peak rate $R_{peak,in}$, buffer capacity C, time t_s and m input traffic streams.* ***If*** *(1) $\sum_{i=1}^{m} h(p_i) \leq R_{peak,in}$, (2) $\sum_{i=1}^{m} f(p_i) \leq R_{out}$ and (3) $\forall i = 1, \ldots, m \int_{t-t_s}^{t} g_i(t)d(t) \leq f(p_i) \cdot t_s = h(p_i) \cdot t_p$* ***holds, then*** *the total volume $v \leq R_{out} \cdot t_s$.*

The proof can be found in [12]. The sum of burst sizes of the input streams equals to the maximal permitted g(t) so there will be no data loss.

4.4 Setting Per-Path Token Bucket Parameters

The following subsection describes the algorithm that assigns each path with its token bucket parameters: the token fill rate and the bucket size. The token fill rate governs the per path sustained rate and the bucket size is calculated by the maximal burst time interval t_p multiplied by the peak rate. We derive these parameters by traversing each first cut arc. We assume all first cut links have the same buffer size C. By applying these parameters to the token bucket at the ingress of this path, the traffic is assured to be conforming.

– **Perform** for each $a^k \in A^-$ with outgoing rate R^k_{out}

1. For each incoming path p_i: $h(p_i) = \begin{cases} f(p_i) \text{ /*cannot increase its rate*/} \\ h(p_i) \text{ /*otherwise */} \end{cases}$
2. Set $R^k_{peak,in}$ to be the incoming peak rate of a^k, $R^k_{peak,in} = \sum_{path\ i \in a^k} h(p_i)$.
3. set t^k_p to be the maximal burst interval for arc a^k using Eq. 1, C, R^k_{out}, and $R^k_{peak,in}$.

Table 1. provisioning parameters can be systems wide (the only one here is buffer size), per path, or per node interface

Parameters	**Per-fbn**	**Per-Path**
Buffer size, Same for all $fbns$	C	
R_{out}	The fbn interface link rate	
$R_{peak,in}$	The sum of peak rates per-path ($\sum h(p_i)$)	calculated per fbn in subsection 4.2
t_p	Calculated using C, R_{out} and $R_{peak,in}$ (Eq. 1)	The minimum over all *first cut* links it traverse
Burst size	$R_{peak,in} \cdot t_p$	$h(p_i) \cdot t_p$

4. Apply to all the paths of a^k (that a^k is their fbn) the values $f(p_i), h(p_i)$ and t_p^k. Set the token bucket contract to be: token rate $= f(p_i)$ and $bs = h(p_i) \cdot t_p^k$

Table 1 summarizes the parameters this system needs for provisioning and the order of their derivation. All the stages of the algorithms were implemented using MATLAB.

5 Simulation Results and Evaluation

Simulations. In order to evaluate the gain from our algorithm, we applied both allocation methods, **TVAfB** and the the **MRA** using the NS-2 simulator and the example in Figure 3(b). The four aggregates in the example are composed of 10 TCP[1] connections (each with maximal congestion window size of 100), and use different paths, $r1, .., r4$. Each TCP connection transfers a file of 2MByte.

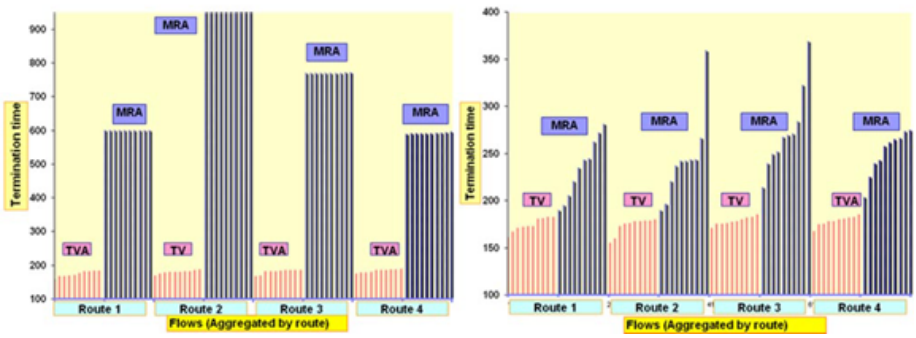


Fig. 5. The height of a per-connection vertical bar indicates the termination time of the appropriate TCP flow. Every ten bars are grouped by aggregate, for *TVA* (Bars group:1,3,5,7) and *MRA* (Bars group:2,4,6,8).

The regulation entities (token buckets) that are located at the ingress nodes, $1, 2, 3$, and 4, perform policing and metering for the arriving aggregates, namely all the 10 TCP connections are policed together. The *MRA* only allows packets that arrive within the maximal rate, $1Mbps$ in this example. We set the tokens fill-rate to be $1Mbps$ and the bucket size to be 1000B (equals to the size of 2 packets). The token bucket parameters for the *Traffic Volume Allocation* (TVA) are the values that are calculated in Section 4.4 and presented in Figure 3(b). In both methods, any 'out-of-profile' packet is dropped, though we allow bursts in the size of the token bucket. Further, we locate weighted queues of 186 packets (equals to 90,000 Bytes) at the output ports of nodes 5 and 8 towards arcs $e5$ and $e8$. We use propagation delay of 20ms for all the links in each direction, except for link $e8$ whose propagation delay is 40ms.

The simulation measures the time it takes for each connection to transmit the 2Mbyte file. We compare the per-aggregate average termination time, computed over all the

[1] TCP was selected due to its bursty nature and its prevalence in today Internet. This enforces us further to discuss the TCP congestion control in the context of our work.

connections within each aggregate, and the number of the dropped packets per-aggregate. Figure 5(a) depicts the simulation termination results for the two allocation methods for all the connections. Clearly, *TVA* gained a $2.5 - 4.5$ speedup in the file transfer time. The reason for this is the higher number of conforming packets, and thus less drops. Indeed, for *TVA* the average drop rate is 2.5%-6%, while for *MRA* it is 16.7%[2]. The file transfer times for the *MRA* are much longer than *TVA* because of the huge 'out-of-profile' dropping, which causes TCP timeouts. Running the same example but with 1/10th of the propagation delay over all the links (see Figure 5(b)) decreases the termination times that are achieved by the *MRA* since it decreases the time the slow-start phase requires to ramp up. It does not affects *TVA* performance since it spends its time in congestion avoidance (due to the small percent of packet drops) and the policer allows it to transmit enough packets, such that it start receiving acknowledgements before it exhausts its window. To further study our algorithm performance, we looked at more scenarios where the loads over the different routes are not even such that the bottleneck link $e8$ is under used. All the TCP connections that participated in a non-even scenario increased their rates related to the even-load scenario[3].

A common real-world architecture that can benefit from using the *TVAfB* algorithm is an access or a metro network. In a common metro architecture, a set of paths from the clients (modem pools, T1 lines, etc.) forms a tree towards the ISP Internet gateway. The link capacities in this network are the same due to a homogeneous usage of technology, e.g., 1Gbps Ethernet. Thus, the link to the gateway router becomes a bottleneck and an fbn in the *TVAfB* algorithm. This link capacity, $1Gbps$, is shared by the sustained rates of all the paths. Obviously all the preceding links have an excess bandwidth that can be added to the rate of the paths. Furthermore, the needed buffering resource in the gateway router are modest[4].

6 Concluding Remarks

The solutions presented in this paper can be used by network administrators as a design tool. The algorithm assumes the knowledge of the traffic rate demands across the network and the ability to lay a set of fixed routing paths. It can be performed as often as any *keep-alive* algorithm in a connection-oriented network. Beside the fact that all the algorithms runs in a polynomial number of steps, we verified the practicality by examining issues such as required buffer size and shaping algorithms. It is a fast and easy-to-deploy algorithm that can be used over one or more network domains, in order to find the bottleneck links, buffering needs, and SLA parameters.

Acknowledgments: We thank Danny Dolev for many helpful discussions.

[2] Note that the *TVA* transfer time is only 50% higher than TCP theoretical achievable rate.

[3] This framework can use a model that sizes the buffer of a bottleneck link considering the parameters of the TCP sources [13].

[4] Assuming this router has 16 1Gbps input-ports which are aggregated into one 1Gbps output link, and a burst period of 1ms, the cumulative burst size $BS = (1Gbps \cdot 16 - 1Gbps) \cdot 1ms = 15,000,000bits \simeq 2MB$, meaning only $2MB$ to be shaped, in case of congestion.

References

[1] Allalouf, M., Shavitt, Y.: Centralized and distributed approximation algorithms for routing and weighted max-min fair bandwidth allocation. (In: IEEE HPSR'2005)
[2] Nichols, K., Jacobson, V., Zhang, L.: A two-bit differentiated services architecture for the internet – RFC no. 2638. Internet RFC, Internet Engineering Task Force (1999)
[3] Cruz, R.L.: A calculus for network delay part II: Network analysis. IEEE Trans. on Information Theory **37** (1991) 132–141
[4] Parekh, A.K., Gallagher, R.G.: A generalized processor sharing approach to flow control in integrated services networks: the multiple node case. IEEE/ACM Transactions on Networking **2** (1994) 137–150
[5] Georgiadis, L., Guérin, R., Peris, V., Sivarajan, K.N.: Efficient network QoS provisioning based on per node traffic shaping. IEEE/ACM Transactions on Networking **4** (1996)
[6] Yaron, O., Sidi, M.: Generalized processor sharing networks with exponentially bounded burstiness arrivals. In: INFOCOM (2). (1994) 628–634
[7] Christin, N., Liebeherr, J., Abdelzaher, T.: A quantitative assured forwarding service. In: IEEE INFOCOM 2002, New York, NY, USA (2002)
[8] Liebeherr, J.: A note on statistical multiplexing and scheduling in video networks at high data rates (2002)
[9] Clark, D., Lehr, W., Liu, I.: Provisioning for bursty internet traffic: Implications for industry structure. In: MIT ITC Workshop on Internet Quality of Service. (1999)
[10] Guerin, R., Ahmadi, H., Naghshineh, M.: Equivalent bandwidth and its application to bandwidth allocation in high-speed networks. IEEE Journal on Selected Areas in Communications **9** (1991) 968–981
[11] Biton, E., Orda, A.: Qos provision and routing with stochastic guarantees. (In: QShine'04)
[12] Allalouf, M., Shavitt, Y.: Achieving bursty traffic guarantees by integrating traffic engineering and buffer management tools. Technical report, Dept. of EE, Tel Aviv University (2005) EES2005-50.
[13] Dhamdhere, A., Jiang, H., Dovrolis, C.: Buffer sizing for congested internet links. (In: INFOCOM'05)