Traffic Engineering in a Multipoint-to-Point Network

Guillaume Urvoy-Keller , Gérard Hébuterne , and Yves Dallery

Abstract—**The need to guarantee Quality of Service (QoS) to multimedia applications leads to a tight integration between the routing and forwarding functions in the Internet. MPLS tries to provide a global solution for this integration. In this context, multipoint-to-point (m2p) networks appear as a key architecture since they provide a cheaper way to connect edge nodes than point-to-point connections. M2p networks have been mainly studied for their load balancing ability. In this paper, we go a step further: we propose and evaluate a traffic management scheme that provides deterministic QoS guarantees for multimedia sources in an m2p network. We first derive an accurate upper bound on the end-to-end delay in an m2p architecture based on the concept of additivity. Broadly speaking, an m2p network is additive if the maximum end-to-end delay is equal to the sum of local maximum delays. We then introduce two admission control algorithms for additive networks: a centralized algorithm and a distributed algorithm, and discuss their complexity and their scalability.**

Keywords—**Multipoint-to-point networks, MPLS, Quality of Service, Deterministic Bounds, Admission Control.**

I. Introduction

Provisioning of Quality of Service (QoS) in high-speed networks has received much attention in the last decade. The ATM community advocated for a connection oriented solution while the Internet community advocated for a connectionless solution. Today, there is a trend to combine these solutions since the backplane of many core routers is an ATM switch fabric. ATM switches provide an high-speed and low cost per port solution for the Internet. However, they are not universally used. Multiprotocol Label Switching (MPLS) [1], has been developed to offer a universal forwarding layer to the Internet. MPLS may inter-operate adequately with ATM [2] or Frame Relay [3] or provide an ad-hoc forwarding service.
An Internet Service Provider (ISP) may use MPLS to establish a set of routes between its ingress nodes and its egress nodes. If point-to-point (p2p) routes are used and there are n edges, then $O(n^2)$ routes are required to connect n nodes. Another possibility to cover the network is to use multipoint-to-point (m2p) connections rooted at the egress nodes. With m2p connections, only $O(n)$ routes are required. The use of m2p Label Switch Paths (LSPs) allows to merge several p2p LSPs: m2p LSP ease traffic management since they reduce the amount of states (corresponding to LSPs) at each node, i.e. not only at edge nodes but also at interior nodes (see Figure 1). In this paper, we propose and evaluate a traffic management scheme for an m2p network that guarantees a deterministic QoS to variable bit rate sources. Sources are assumed to be leaky bucket constrained with a maximal end-to-end delay requirement. We assume a fluid model that closely approximates the behavior of a packet network with a small packet size compared to the service rates of the servers. The fluid model enables us to concentrate on the central issues. The proposed scheme is based on the FIFO scheduling policy, because of its scalability. It is quite likely that more complex policies such as PGPS [4], [5] will be used in the future. However, these policies will not run at a connection level but rather at a class level (to ensure scalability), and a given class will see the m2p network as a FIFO network (with a time-varying service rate). Thus, a first problem to solve is the case of a FIFO m2p network. To our best knowledge, this problem has not been treated previously.
The remainder of the paper is organized as follows. In Section 2, we review the related work in the fields of m2p architectures and end-to-end bounds in FIFO networks. In Section 3, we recall the main results of the Network Calculus [6], [7], [8] that we use to obtain a bound on the end-to-end delay. In Section 4, we emphasize the difficulty to directly derive an end-to-end delay bound for a FIFO network from the concept of service curve introduced in the Network Calculus. In Section 5, we study the maximum end-to-end delay in the case of an m2p network with two servers. The analysis demonstrates that a bounding approach is required for the case of larger networks. We also introduce the concept of *additivity*, which is central to our analysis in the case of larger networks carried in Section 6. In Section 7, we propose and evaluate two admission control algorithms based on our end-to-end delay bound. In Section 8, we conclude and provide some insights for future work.

Manuscript received February 21, 2001; revised November 23, 2001.

G. Urvoy-Keller is with Institut Eurecom, 2229, route des Crêtes, 06904 Sophia-Antipolis, France. E-mail:urvoy@eurecom.fr
G. Hébuterne is with Institut National des Télécommunications, 9, rue Charles Fourier, 91011 Evry cedex, France. E-mail: hebutern@hugo.int-evry.fr
Y. Dallery is with Ecole Centrale de Paris, Avenue des Vignes, 92 295 Chatenay Malabry cedex, France. E-mail: dallery@pl.ecp.fr

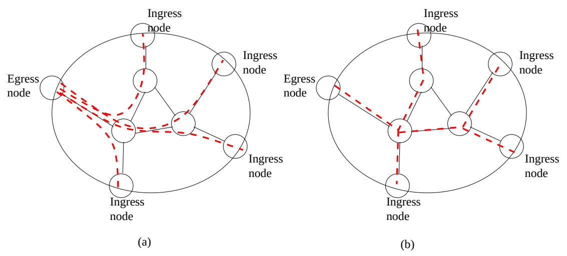


Fig. 1. P2p strategy (a) vs m2p strategy (b)

II. Related Work

A first step toward the provision of QoS service is the ability to balance load in the network. In the traditional Internet, this is achieved with metric-based routing. Network administrators adjust link metrics to balance the traffic. However, this ad-hoc solution is not satisfying in the context of QoS provisioning and there exists a need to explicitly control the routes of the flows in the network. Such a control may be achieved with MPLS. Such a solution has been investigated in [9] in the context of IP over ATM and in [10] in the context of IP over MPLS. In [10], Saito et al. propose a traffic engineering scheme for Internet backbones that tries to provide an optimal load balancing for reliability. The proposed scheme uses multiple m2p Label Switch Paths (LSPs) between each ingress/egress pair to achieve load balancing and reliability. Traffic demands are expressed as service rates. Sources are thus implicitly assumed to be constant bit rate sources.

A first step toward the design of our traffic management scheme is the derivation of an accurate bound on the end-to-end delay for an m2p network. Determining an end-to-end delay bound in a network based on the FIFO scheduling discipline is a challenge since the stability of a FIFO network with a general architecture has not been established yet. Tassiulas et al. [11] proved that the ring architecture is stable under any work-conserving scheduling discipline (and thus under the FIFO scheduling discipline). The result is interesting since the ring architecture is often considered as a "worst-case" architecture due to the high dependency it induces among sessions. However, this result has not yet been extended yet to the case of a general architecture. Chlamtac et al. [12] have focused on FIFO networks with peak rate constrained sources. The authors show that if the peak rate of each source in the network satisfies a constraint related to the number of sources that the source meets on its route, then the network is stable and bounds on end-to-end delays and backlogs exist. The result applies to FIFO networks with a general architecture, but it is restricted to the case of constant bit rate sources. In the present work, we concentrate on a specific architecture, the m2p architecture, however with variable bit rate sources.

III. Network Calculus

The Network Calculus [13], [6], [7], [8], [14] is an analytical method to derive deterministic bounds on end-to-end delays and backlogs. The Network Calculus has been developed both for continuous time [13] and discrete time [8]. We use here the continuous version that is better suited for a fluid-flow analysis. We present, in the following, the basic concepts of the Network Calculus that will be used in the rest of this paper.

A. Sources and Network Elements Modeling

A.1 General Sources

Consider a source S. Let S be a given trajectory of S and Γ_{S} be the set of all possible trajectories for S.

Definition *1:* The cumulative rate function $A_S(t)$ of S is defined as the cumulative amount of bits issued by S in the interval $[0, t]$ (the cumulative rate function of a given trajectory fully characterizes this trajectory).

Definition *2:* A function α is an arrival curve for S if:

$$A_S(t+\tau) - A_S(t) \leq \alpha(\tau), \forall S \in \Gamma_{\mathrm{S}}, \forall \tau \geq 0, \forall t \geq 0.$$

An arrival curve for S provides an upper bound on the number of bits that S can send on any time interval.

Definition *3:* We define Ξ_{S} as the set of arrival curves associated to S:

$$\Xi_{\mathrm{S}} = \{\alpha \,|\, \forall S \in \Gamma_{\mathrm{S}},\, \forall (t,\tau), A_S(t+\tau) - A_S(t) \leq \alpha(\tau)\}.$$

Theorem *1:* (See [13] for proof) Ξ_{S} as a minimum element α^*, called the minimum arrival curve and defined as follows:

$$\alpha^*(\tau) = \max_{S \in \Gamma_{\mathrm{S}}} \max_t \left(A_S(t+\tau) - A_S(t)\right), \forall \tau \geq 0.$$

In the remaining of the paper, and for sake of simplicity, the term "source" may be used to refer to a trajectory.

A.2 Source Model

In the remaining of this paper, we consider sources that are leaky bucket constrained with an additional constraint on their peak rate. A traffic descriptor for a given source S has three parameters (p, R, M) (we note $S \sim (p, R, M)$) that are respectively the peak rate, the mean rate and the maximum burst size of S. Such a source is able to traverse the leaky bucket controller depicted in Figure 2 without experiencing any loss. The size of the token bucket is $M' = M\frac{p}{p-R} > M$ since the peak rate of the source is finite. Let $\Omega(p, R, M)$ be the set of sources S such

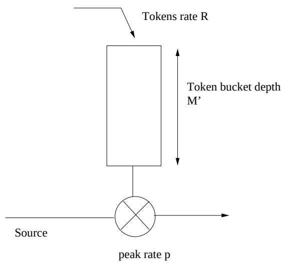


Fig. 2. Leaky bucket controller

as $S \sim (p, R, M)$. The greedy source (trajectory), associated to $\Omega(p, R, M)$, plays an important role in worst case analyses and is defined as follows:

Definition *4:* For a given set $\Omega(p, R, M)$, $G_{\Omega(p,R,M)}$ (or simply G) is the source that consumes tokens as soon as possible. With a token bucket initially full at time $t = 0$, the greedy source G emits at its peak rate during $[0, \frac{M}{p}]$ and then emits at its mean rate R, i.e.:

$$A_G(\tau) = \min\left(p \cdot \tau, R \cdot \tau + M\frac{p-R}{p}\right)$$

The following results hold for the greedy source (proof is left to reader):

- $\alpha^*_G(\tau) = A_G(\tau), \forall \tau \geq 0$ (the minimum arrival curve of the greedy source is its cumulative rate function).
- $\forall S \in \Omega(p, R, M) \forall \tau \geq 0, \alpha^*_S(\tau) \leq \alpha^*_G(\tau)$ (the minimum arrival curve of the greedy source is the minimum arrival curve of all the sources of $\Omega(p, R, M)$)

The minimum arrival curve of a multiplex of leaky bucket constrained sources is the sum of the minimum arrival curves of the sources of this multiplex (see [14]). The resulting source has a concave piece-wise linear arrival curve. We make use of the two notions (source constrained by a single leaky bucket or by a set of leaky buckets) in the remaining of this paper.

The source model presented above encompasses the case of an IP source declared with a TSPEC and the case of a VBR ATM source. An ATM VBR source is constrained by a pair of GCRA algorithms with parameters (T, τ) and $(T', \tau' + \tau)$. Let δ be the cell size in bits. A minimal arrival curve α for a VBR source is (see [15]):

$$\alpha(t) = \min(p \cdot t + b_p, R \cdot t + b_M)$$

where $p = \frac{\delta}{T}$ is the peak rate of the source in bit/s, $R = \frac{\delta}{T'}$ is the sustainable rate in bit/s, $b_p = p \cdot \tau + \delta$ corresponds to the

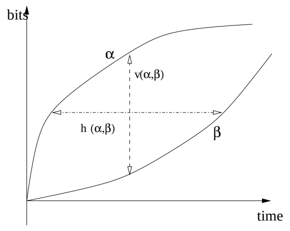


Fig. 3. Upper bounds on backlogs and delays

cell jitter (in bits) and $b_M = \tau' R$ corresponds to the maximum burst size of the source (in bits).

Similarly, an IP source described with a TSpec has a minimal arrival curve $\alpha(t) = \min(M + p \cdot t, b + r \cdot t)$ where M is the maximum size of a packet of the source, p its peak rate, r its sustainable rate and b its bucket depth.

A.3 Network Elements

Within the Network Calculus framework, a network element is characterized by its service curve that intuitively represents a lower bound on the service it provides. Several definitions of a service curve exist. We use the extended service curve defined by Le Boudec [13].

Definition *5:* A network element offers an extended service curve β to a given source S if:

$$\forall S \in \Gamma_{\mathrm{S}}, \forall t \geq 0, \exists t_0 \leq t : A_{S_{out}}(t) - A_S(t_0) \geq \beta(t - t_0)$$

where $A_{S_{out}}$ is the cumulative rate function of S seen at the output of the network element.

Example: a service curve β for a work-conserving server with a service rate C is $\beta(\tau) = C \cdot \tau$. The proof is straightforward: t_0 is equal to the beginning of the busy period that t belongs to, or $t_0 = t$ if there is no backlog at time t.

B. Advanced Results

B.1 Bounds on Delays and Backlogs

Consider a system, seen as a black box. Let $R(t)$ (resp. $R^*(t)$) be the cumulative rate function seen at the input (resp. output) of the system. The backlog at time t is $b(t) = R^*(t) - R(t)$. Cruz [8] has introduced the virtual delay $d(t)$ defined as follows:

$$d(t) = \inf \{T : T \geq 0,\ R^*(t+T) \geq R(t)\}$$

Theorem *2:* Consider a source with an arrival curve α traversing a system that offers a service curve β. Then:

$$b(t) \leq v(\alpha, \beta) \text{ and } d(t) \leq h(\alpha, \beta)$$

where $v(\alpha, \beta)$ and $h(\alpha, \beta)$ represent respectively the maximum vertical and horizontal distances between α and β (see Figure 3):

$$v(\alpha, \beta) = \sup_{s \geq 0}(\alpha(s) - \beta(s))$$
$$h(\alpha, \beta) = \sup_{s \geq 0}(\inf \{T : T \geq 0,\ \alpha(s) \leq \beta(s+T)\})$$

B.2 Output Characterization

Theorem *3:* (See [13] for proof) An arrival curve α^{out} for a source seen at the output of a system that offers a service curve β is:

$$\alpha^{\mathrm{out}}(\tau) = \sup_{v \geq 0}(\alpha(\tau + v) - \beta(v)), \forall \tau \geq 0$$

where α is the arrival curve of the source seen at the input of the system.

B.3 Network Service Curve

A straightforward way to obtain end-to-end delay bounds is to apply Theorems 2 and 3 at each stage in the network and sum the obtained local bounds. It is however possible to obtain a tighter result with the network service curve paradigm:

Theorem *4:* (See [13] for proof) Consider a source S traversing p network elements. Each network element is characterized by an extended service curve $(\beta_j)_{j \in \{1,\ldots,p\}}$. S may see these p network elements as a single network element characterized by a network service curve β that is the convolution of $(\beta_j)_{j \in \{1,\ldots,p\}}$:

$$\beta(t) = \inf_{t_1 + \ldots + t_p = t} (\beta_1(t_1) + \ldots + \beta_n(t_n))$$

The strength of Theorem 4 is that the obtained end-to-end delay bound is smaller than the one obtained through summation of local delay bounds (using Theorems 2 and 3).

IV. End-to-end Delays: A Service Curve Approach

To define a complete traffic management scheme for m2p networks, we first need to derive an accurate bound on the end-to-end delay. As seen in the previous section, the Network Calculus provides a way to derive end-to-end bounds using network service curves. In the present section, we investigate this approach.

A. Service Curve Offered by a FIFO Server

Consider 2 sources, S_1 and S_2 (with respective arrival curves α_1 and α_2) and a server that implements the FIFO scheduling policy with a service rate C. Let λ_C be the function such that: $\forall t \geq 0$, $\lambda_C(t) = C \cdot t$, and R_i (resp. R_i^*) be the cumulative rate function of S_i at the input (resp. output) of the server. For a given time t, let s_0 be the last time instant with no backlog in the server ($s_0 \leq t$). Thus, $R_1^*(s_0) = R_1(s_0)$ and $R_2^*(s_0) = R_2(s_0)$. Since the scheduling policy is work conserving, this yields:

$$R_1^*(t) - R_1^*(s_0) + R_2^*(t) - R_2^*(s_0) = C \cdot (t - s_0). \quad (1)$$

Causality implies that $R_2^*(t) \leq R_2(t)$. Thus:

$$R_2^*(t) - R_2^*(s_0) \leq R_2(t) - R_2(s_0)$$

Since S_2 is constrained by α_2 and the server adds a constraint on the peak rate of the output source, we obtain:

$$R_2^*(t) - R_2^*(s_0) \leq \min(C \cdot (t - s_0), \alpha_2(t - s_0)). \quad (2)$$

From equation (1) and (2), we obtain:

$$R_1^*(t) - R_1(s_0) \geq C \cdot (t - s_0) - \min(C \cdot (t - s_0), \alpha_2(t - s_0)).$$

Let us define $(x)^+$ as $max(0, x)$. Then, $\beta_1 = (\lambda_C - \alpha_2)^+$ is a service curve for S_1, since $C \cdot t - \min(C \cdot t, \alpha_2(t)) = (\lambda_C(t) - \alpha_2(t))^+$.

B. Discussion

The service curve β_1 is conservative. Indeed, if S_2 were preemptive over S_1, the obtained service curve would be the same since, in this case, S_1 receives only the extra capacity unused by S_2. Besides, assume that S_1 and S_2 transit in a second server where they mix with a third source S_3. To derive a service curve for S_1 in the second server, we need an arrival curve for S_2 at

the input of the second server. This arrival curve may be obtained by applying Theorem 3 to the arrival curve of S_2 at the input of server 1 and its service curve at server 1. However, the arrival curve for S_2 at the second server is also pessimistic since the service curve for S_2 at server 1 is pessimistic. Thus, the conservative aspect of the result increases with the size of the network. This approach leads inevitably to pessimistic results. For instance, consider a single server and assume S_1 and S_2 have the same traffic descriptor, namely (p, R, M). The following relation exists between the delay bound D_{SC} obtained with the service curve approach and the maximum delay D_{max}: $D_{max} = \frac{C-R}{C} D_{SC}$. Thus, when $R \to \frac{C}{2}$ (stability requires that $C > 2R$), $D_{SC} \to 2D_{max}$.
The weaknesses of this service curve approach demonstrates the necessity of a new approach the problem. Note, however, that a better (more accurate) service curve than β_1 for a FIFO server may exist. The determination of such a service curve remains an open issue.

V. End-to-end Delay in a Tandem Network

In this section, we study the end-to-end delay in a network with two servers in sequence, called a tandem network (except in the first part where the results hold for p servers in sequence) and stress the complexity of an exact analysis.

A. *Single Source / p Servers in Sequence*

Let S be a source traversing p servers in sequence. The service rate of server j is C_j. We assume that S is constrained by a concave piece-wise linear arrival curve α (i.e. S is constrained by a set of leaky buckets). We also assume that $\forall (i,j) \in \{1,\ldots,p\}^2, i \leq j, C_i \geq C_j$, without any loss of generality, since if C_j is greater than C_i the traffic outgoing of server i experiences no delay in j since its peak rate is lower than the service rate of j.

A.1 Worst-Case Analysis

The analysis addresses two dual problems: computation of the maximum end-to-end delay and computation of the buffer requirement at each server. Note that the latter is equivalent to compute the local maximum delay at each server in the case of FIFO scheduling policy.

A.1.a End-to-end Delays.

Lemma *1:* Consider a source S traversing p FIFO servers with respective service rates $(C_j)_{j\in\{1,\ldots,p\}}$. The end-to-end delay of a bit of S is the same as if the network were restricted to a single server with a service rate $\min_{j\in\{1,\ldots,p\}}(C_j)$.
Proof : As noted previously, we can assume that the servers rates are decreasing. Let us also assume that server j is backlogged during $[0, T_j]$. This means that during $[0, T_j]$, the output process of server j has a constant rate C_j. Since $C_j \geq C_{j+1}$, server $j+1$ is also backlogged during $[0, T_{j+1}]$ with $T_{j+1} \geq T_j$. As a consequence, any backlog period of a given server j is included in a backlog period of any server k with $j \geq k \geq p$.
Now consider a bit that experiences some delay in the network. Let j be the first server where it experiences delay. It will also experience some delay at server $j+1,\ldots,p$. The backlog period at server p has begun at a certain time in the past that we choose to be time zero. The bit has entered the network at time $t \geq 0$. Since the backlog periods are included into each other, the bit that enters at time t, has to wait for all the bits sent in $[0,t]$ to be served by server p. Thus its end-to-end delay is the same as if the network would only comprise server p. $\square$

Theorem *5:* The maximum end-to-end delay of a source constrained by a concave piece-wise linear arrival curve α traversing p FIFO servers in sequence is achieved when the source is greedy.
Proof : The maximum end-to-end delay in the network is the same as if the network would only comprise the slowest server (Lemma 1). For the case of a single node, the maximum delay is achieved when the source is greedy since the cumulative rate function of the greedy source is equal to the arrival curve of the source(Theorem 2). This proves the result. $\square$

A.1.b Buffer Requirements. Let us now compute the minimum buffer capacity required at each server to ensure a zero loss rate. We first establish a relation between $S \in \Gamma_{\mathrm{S}}$ and G at the output of the first server. Let $S_{in,i}$ and $S_{out,i}$ be the input and output sources at server i for trajectory S ($A_S = A_{S_{in,1}}$). The following result holds:

Lemma *2:* $\forall t \geq 0, A_{G_{out,1}}(t) = \min(A_{G_{in,1}}(t), C_1 \cdot t) = \min(\alpha(t), C_1 \cdot t)$
The proof is straightforward (see Figure 4). From Lemma 2, we can deduce that $A_{G_{out,1}}$ is a concave piece-wise linear function. Moreover, $A_{G_{out,1}}$is an arrival curve for the outgoing traffic of server 1 for any source $S \in \Gamma_{\mathrm{S}}$.

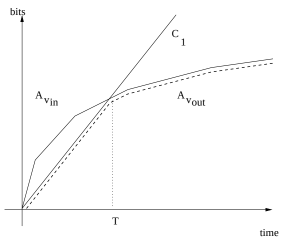


Fig. 4. Output cumulative rate function for a greedy source

Lemma *3:* $\forall S \in \Gamma_{\mathrm{S}}, \alpha^*_{S_{out,1}} \leq A_{G_{out,1}}$
Proof: To prove that $A_{G_{out,1}}$ is an arrival curve for any trajectory S, we use the definition of an arrival curve: an upper bound on the amount of traffic emitted on any time interval. Suppose

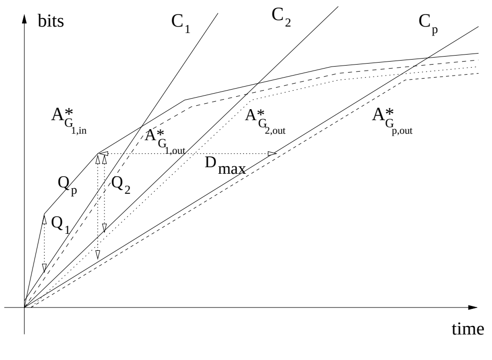


Fig. 5. End-to-end Delay (D) and Buffer Requirements (Q_1, Q_2, Q_3)

that there exists a trajectory S, a time instant t and a time interval τ such that: $x = A_{S_{out,1}}(t+\tau) - A_{S_{out,1}}(t) > A_{G_{out,1}}(\tau)$. Then, necessarily, $\tau > T$ where T (see Figure 4) is the maximum time where the server is backlogged (and thus, its effective output rate is C_1). For $S_{out,1}$ to produce x during τ, $S_{in,1} = S$ must have at least produced x during a time interval of at most τ in the past, since the scheduling policy is work-conserving:

$$\exists t' \le t, A_{S_{in,1}}(t'+\tau) - A_{S_{in,1}}(t') \ge x > A_{G_{out,1}}(\tau)$$

From Lemma 1 and $\tau > T$, we have:

$$A_{G_{out,1}}(\tau) = A_{G_{in,1}}(\tau) = \alpha(\tau)$$

Combining the last two equations, we obtain:

$$\exists t' \le t, A_{S_{in,1}}(t'+\tau) - A_{S_{in,1}}(t') > \alpha(\tau)$$

We thus have a contradiction since S is constrained by α. □
A recursive application of Lemma 2 indicates that the worst-case source for each server is generated by the greedy source. Thus, the greedy source yields the maximum backlogs.

Theorem *6:* The maximum backlog at each server for a source S with a concave piece-wise linear arrival curve α traversing p servers in sequence, is obtained when the source is greedy (see Figure 5).
Note that the arrival curve obtained from Lemma 2 is better (smaller) than the arrival curve that could be obtained with Theorem 3. However, the result holds only for FIFO servers whereas Theorem 3 holds for any scheduling discipline.

B. Multipoint-to-Point Tandem Networks

Consider a tandem m2p network, i.e. a tandem network where sources may enter at node 1 or 2 but exit at node 2 only. Sources that enter the network at node i may be aggregated since they have the same route in the network. Let S_i be the resulting source at node i. S_i is constrained by n_i leaky buckets, where n_i is the number of sources entering at node i. Let α_i be the minimum arrival curve of S_i. Unlike the single source case, buffer requirements and end-to-end delay bounds must be estimated separately.

B.1 Buffer Requirements

The results of this part are obtained for m2p with p servers in sequence. Let Q_j be the minimum buffer requirement at server j that guarantees a zero loss rate. Using Theorem 2, we obtain an upper bound for Q_j with the minimum arrival curve of the input flow at server j. This yields the minimum buffer requirement, provided that one can prove that there exists a trajectory of the system such that the input flow at server j has a cumulative arrival curve equal to this minimum arrival curve.
A consequence of Lemma 2 is that the minimum arrival curve ($\alpha^*_{S_{out}}$) at the output of a FIFO server for an aggregation of leaky bucket constrained sources is maximum when the sources are greedy and strictly synchronous (i.e. they start emitting at the same time instant). Moreover, this minimum arrival curve is a concave piece-wise linear function. Thus, S_{out}, the source seen at the output of the server, is multi- leaky bucket constrained. If this source is to be mixed with a second (leaky bucket constrained) source S_2 and injected in a second server, the maximum backlog is achieved when S_{out} and S_2 are greedy and synchronous. This result can be extended to m2p networks of any size:

Theorem *7:* For a given m2p network with leaky bucket constrained sources, the maximum backlog Q_j at server j is achieved when all the sources are greedy and strictly synchronous, i.e. when the sources start emitting at the same time instant.

B.2 End-to-end Delay

For the single server case, the maximum backlog corresponds to the maximum delay. We prove here that for the case of a tandem m2p networks, achievement of the local maximum delays does not necessarily results in the maximum end-to-end delay. For the single server case, two conditions must be met to obtain the maximum delay: greediness and a strict synchronization (they start emitting at the same time instant) of the sources. For the case of a tandem m2p network, we prove that greediness property is still mandatory (this is intuitively logical since "being greedy" means generating traffic at the maximum possible rate during a maximum period of time, a basic condition to create some backlog), while the synchronization between sources is no more a strict one.

B.2.a Synchronization: A Simple Counter-example. Consider a system with two servers with respective service rates C_1 and C_2 ($C_1 < C_2$) and two sources S_1 (entering at server 1) and S_2 (entering at server 2). We assume that S_1 and S_2 have the same leaky bucket parameters (p, R, M). We also assume that $0 \le R \le C_2 - C_1$. According to the results obtained in the previous section, the maximum backlogs are obtained when S_1 and S_2 are strictly synchronous. This corresponds to the trajectories depicted in Figures 6 and 7. Let d_{imax} ($i \in 1, 2$) be the maximum delay at server i (achieved when the sources are greedy and synchronous) and $d_{max_{S_1,S_2}}$ the maximum end-to-end delay for a given trajectory of S_1 and S_2. Since $C_2 \ge C_1$,

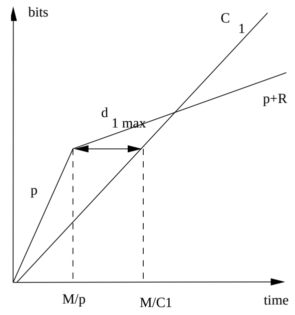


Fig. 6. Maximum Backlog Generation (Server 1)

and since the 2 sources have the same traffic descriptor, the bit that experiences d_{1max} does not experience d_{2max}. Thus, the maximum end-to-end delay is strictly less than the sum of the local maximum delays, i.e. $d_{max_{S_1,S_2}} \le d_{1max} + d_{2max}$. Also, since, $R \le C_2 - C_1$, the maximum backlogs are experienced by the bits that enter the two servers at time $t = \frac{M}{p}$.
Let us now delay the beginning of emission of S_2: S_1 starts

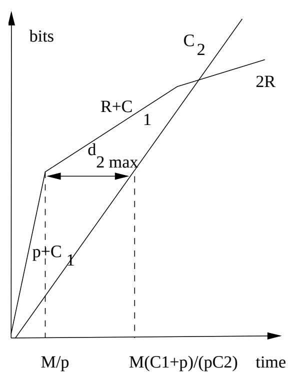


Fig. 7. Maximum Backlog Generation (Server 2)

at time zero and S_2 starts at time $t = \frac{M}{C_1} - \frac{M}{p}$, S_1 and S_2 still being greedy. The bit that experiences d_{1max} enters the network at time $t_0 = \frac{M}{p}$. It exits the first server at time $t_0 + d_{1max} = \frac{M}{p} + \frac{M}{C_1}$. During $[\frac{M}{C_1} - \frac{M}{p}, \frac{M}{p} + \frac{M}{C_1}]$, S_1 has produced $C_1 \cdot \frac{M}{p}$. Thus, the maximum local delay d_{2max} is achieved at server 2 at time $t_1 = \frac{M}{p} + \frac{M}{C_1}$, which is the time instant when the bit that experienced d_{1max} at server 1 reaches server 2. Thus, this bit experiences an end-to-end delay equal to $d_{1max} + d_{2max}$, which is strictly greater than in the strictly synchronous case. This example clearly emphasizes the impact of the synchronization among the sources on the end-to-end delay.

B.2.b Worst Case Conditions. We now investigate the conditions yielding the maximum end-to-end delay. The bit that experiences the maximum end-to-end delay is chosen as the reference bit. There are two cases:
1. the reference bit experiences delay in servers 1 and 2.
2. the reference bit experiences delay in server 2 only.

Note that the case “delay in server 1 only” is not possible since if the reference bit experiences some delay in the first server, this means that the server is in a backlog period. Thus, the output process of server 1 has a rate C_1 during a certain time interval. If we mix this flow with S_2 emitting at its peak rate (we assume $p_2 + C_1 \geq C_2$, otherwise server 2 would be transparent to the flow), this resulting flow would create some backlog at server 2 and thus the reference bit would necessarily also experience some delay at server 2.
The case “delay in server 2 only” is easy to solve since the problem transforms into determining the maximum delay in a single server with two leaky bucket constrained sources: S_1, with a peak rate equal to C_1 and S_2. We can thus apply Theorem 5: the maximum delay is achieved when the two sources are greedy and synchronous.
The case “delay in the two servers” is far more complex as we now see.

B.2.c Delay Equations. We adopt the following notations:
1. θ_1 and θ_2 are the epochs of beginning of the backlog periods where the reference bit enters server 1 and 2 respectively. We set $\theta_1 = 0$. θ_2 might be positive or negative.
2. $d_1(t)$ (resp. $d_2(t+d_1(t))$) is the delay experienced by the bit entering server 1 at time t (resp. $t+d_1(t)$). Note that $t+d_1(t) \geq \theta_2$ since we focus on the delay of bits of S_1 experiencing some delay in the two servers.
3. $D(t)$ is the end to end delay of the bit that enters server 1 at time t: $D(t) = d_1(t) + d_2(t + d_1(t))$.
The following equations hold:

$$\begin{aligned} d_1(t) &= \frac{A_{S_{1,in}}(t) - C_1 t}{C_1} \\ d_2(t + d_1(t)) &= \frac{A'_{S_{1,out}}(t + d_1(t)) + A_{S_{2,in}}(t - \theta_2 + d_1(t))}{C_2} \\ &\quad -(t - \theta_2 + d_1(t)) \end{aligned}$$

where $A'_{S_{1out}}$ is the amount of bits generated by S_1, which have already reached server 2 when the bit emitted at time t arrives at server 2. Since $D(t) = d_1(t) + d_2(t + d_1(t))$, we obtain:

$$\begin{aligned} D(t) &= \frac{A'_{S_{1,out}}(t + d_1(t)) + A_{S_{2,in}}(t + d_1(t) - \theta_2)}{C_2} \\ &\quad -(t - \theta_2) \end{aligned} \tag{3}$$

If $\theta_2 \geq 0$, $C_2 \geq C_1$, since during $[0, \theta_2]$ server 1 is backlogged and thus emits at a constant rate C_1 and no backlog appears at server 2. As a consequence: $A'_{S_{1,out}}(t + d_1(t)) = A_{S_{1,in}}(t) - C_1\theta_2$.
If $\theta_2 \leq 0$, S_1 may emit some traffic during $[\theta_2, 0]$ as long as its instantaneous emission rate is smaller than C_1. Let $a^-_{S_{1,in}}(\theta_2)$ be the amount of traffic sent by S_1 during $[\theta_2, 0]$. Note that S_1 can emit at least at a constant rate R (the sum of the mean rates of the sources composing S_1). Indeed, a source constrained by a leaky bucket always receives tokens at a rate R (the arrival rate of tokens in its token pool) and thus can emit bits at this constant rate without affecting its ability to send later at a rate greater than R. As a consequence : $A'_{S_{1,out}}(t + d_1(t)) = A_{S_{1,in}}(t) + a^-_{S_{1,in}}(\theta_2)$.
Equation (3) may thus be rewritten as follows:

$$\begin{aligned} D(t) &= \frac{A_{S_{1,in}}(t) + a^-_{S_{1in}}(\theta_2) + A_{S_{2,in}}(\frac{A_{S_{1,in}}(t)}{C_1} - \theta_2)}{C_2} \\ &\quad -(t - \theta_2) \quad \text{if } \theta_2 \leq 0 \qquad (4) \\ &= \frac{A_{S_{1,in}}(t) - C_1\theta_2 + A_{S_{2,in}}(\frac{A_{S_{1,in}}(t)}{C_1} - \theta_2)}{C_2} \\ &\quad -(t - \theta_2) \quad \text{if } \theta_2 \geq 0 \qquad (5) \end{aligned}$$

B.2.d Greediness. Consider the case $\theta_2 \geq 0$. The end-to-end delay D is a function of the cumulative rate functions $A_{S_{1,in}}$ and $A_{S_{2,in}}$. These cumulative rate functions are upper bounded by the corresponding arrival curves: $\forall i \in \{1, 2\}, \forall t \geq 0, A_{S_{i,in}}(t) \leq \alpha_i(t)$. Since D is an increasing function of $(A_{S_{i,in}})_{i\in\{1,2\}}$, D is maximized when the sources are greedy. We can conclude that, when $\theta_2 \geq 0$, the maximum end-to-end delay is achieved for sources that are greedy starting at a certain time.
When $\theta_2 \leq 0$, we have that:

$$A_{S_{1,in}}(t) + a^-_{S_{1in}}(\theta_2) \leq \alpha_1(t - \theta_2) \tag{6}$$

since S_1 is constrained by α_1 and the considered time interval has a duration $t - \theta_2$. The rhs and lhs of equation (6) are not necessarily equal since during $[\theta_2, 0]$, S_1 must have an instantaneous emission rate less or equal than C_1. We make use of the following lemma to prove that equation (6) is an equality:

Lemma *4:* The maximum end-to-end delay is achieved at a time instant t such that after t, $S_{1,in}$ is not able to emit at a rate greater than C_1.

Proof: Let us prove the result by contradiction. Suppose that the maximum end-to-end delay is achieved for a bit sent at time t and suppose that, after time t, S_1 is still able to send at a rate greater than C_1, say during $[t, t+\delta]$. The end-to-end delay at time t is given by equation (3):

$$\begin{aligned} D(t,\theta_2) &= \frac{A_{S_{1,out}}(t+d_1(t)) + A_{S_{2,in}}(t-\theta_2+d_1(t))}{C_2} \\ &\quad -(t-\theta_2) \end{aligned}$$

Let us now delay the beginning of emission of S_2 by an offset δ and compute the delay at time $t+\delta$ (with the assumption that S_1 emits at rate C_1 during $[t, t+\delta]$). Equation (3) gives:

$$\begin{aligned} D(t+\delta,\theta_2+\delta) &= \frac{A_{S_{1,out}}(t+\delta+d_1(t+\delta))}{C_2} \\ &\quad + \frac{A_{S_{2,in}}((t+\delta)}{C_2} \\ &\quad + \frac{(\theta_2+\delta)+d_1(t+\delta))}{C_2} \\ &\quad -((t+\delta)-(\theta_2+\delta)) \end{aligned}$$

Since $d_1(t+\delta) = d_1(t)+\delta$ (during $[t, t+\delta]$, the backlog at server 1 is increased by $C_1 \cdot \delta$), we obtain:

$$D(t+\delta,\theta_2+\delta) > D(t,\theta_2)$$

The result is thus proved by contradiction. □

A consequence of Lemma 4 is that equation (6) may be an equality for the instant of interest (where the maximum delay is achieved): it is possible to send $\alpha_1(t)$ during $[0,t]$ and, $\alpha_1(t-\theta_2) - \alpha_1(t)$ during $[\theta_2, 0]$, since during this period of time, the emission rate of S_1 is less than C_1.

As a consequence, D is an increasing function of $A_{S_{1,in}}(t) + a^-_{S_{1in}}(\theta_2)$ (and also of $(A_{S_{iin}})_{i\in\{1,2\}}$) in the case when $\theta_2 \le 0$. Thus, the end-to-end delay is maximized when the sources are greedy starting at a certain time.

To summarize, the greediness of the sources is mandatory in any case (backlog at server 2 only or in the two servers with $\theta_2 \ge 0$ and $\theta_2 \le 0$):

Lemma *5:* For any tandem m2p network, the maximum end-to-end delay is achieved when the sources are greedy with different starting times.

Equations (5) and (7) can be rewritten using Lemma 5:

$$\begin{aligned} D(t) &= \frac{\alpha_1(t-\theta_2) + \alpha_2(\frac{\alpha_1(t)}{C_1} - \theta_2)}{C_2} \\ &\quad -(t-\theta_2) \quad \text{if } \theta_2 \le 0 \qquad (7) \\ &= \frac{\alpha_1(t) - C_1\cdot\theta_2 + \alpha_2(\frac{\alpha_1(t)}{C_1} - \theta_2)}{C_2} \\ &\quad -(t-\theta_2) \quad \text{if } \theta_2 \ge 0 \qquad (8) \end{aligned}$$

B.2.e Delay Function Study. We consider sources with piecewise linear concave arrival curves. D, as defined in equation (8), is thus a concave function since concavity is preserved by summation and composition. It has a bell-shaped curve, which starts from zero at time $t=0$ and goes back to zero at time $t=T$, where T is the duration of the network backlog. There is thus only one local maximum. For the remaining of this section, we assume that $\theta_2 \ge 0$. A similar study (though not obvious) could be carried out for $\theta_2 \le 0$.

B.2.f Synchronization. We want to study the influence of θ_2. To do so, we consider the derivative function $\frac{dD}{d\theta_2}(t)$. Equation (8) (f' stands for the derivative of f) gives:

$$\frac{dD}{d\theta_2}(t) = \frac{(C_2-R) - \alpha_2'(\frac{\alpha_1(t)}{C_1} - \theta_2)}{C_2} \quad \text{if } \theta_2 \ge 0 \qquad (9)$$

D is maximized for $\theta_2 = \theta_{2max}$ and $t = t_{max}$. We study, for t set to t_{max}, the influence of θ_2. Since $\theta_2 \ge 0$, $C_2 \ge C_1$. Also, since the bit that experiences the maximum end-to-end delay experiences some delay in the two servers, S_2 must start emitting no later than at the arrival time of the reference bit in the second server, i.e. at time $\theta_{2max} = \frac{\alpha_1(t_{max})}{C_1}$. $D(t)$ has thus one maximum in $[0, \frac{\alpha_1(t_{max})}{C_1}]$. The derivative function can be interpreted as follows: it is all benefit to trigger S_2 sooner than a given θ_2 (say at time $\theta_2 - \delta$) if the value of α_2' is greater than $C_2 - C_1$ after $\frac{\alpha_1(t_{max})}{C_1} - \theta_2$ (arrival time of the reference bit at server 2, which is backlogged from time θ_2), that is if the amount of work done by server 2 in $[\theta_2 - \delta, \theta_2]$, i.e. $C_2 \cdot \delta$, is less than what S_2 and S_1 can produce during the interval $[\theta_{2max} - \delta, \theta_{2max}]$, i.e. $C_1 \cdot \delta + \alpha_2'(\frac{\alpha_1(t_{max})}{C_1} - \theta_2)\delta$.

B.2.g Conclusion. We usually do not know the conditions leading to the maximum end-to-end delay (whether the reference bit experiences some delay in the two servers or in the second server only). For the two server cases, it is possible to derive the value of θ_2 (see [16]). However the analysis does not scale easily to larger networks. A bounding approach is thus necessary for larger networks. A first step toward this objective is the introduction of the concept of *additivity*.

B.2.h Delay Additivity. We say that the delay in a tandem m2p network is additive if the maximum end-to-end delay D_{max} is equal to the sum of the local maximum delays d^G_{imax}. Note that it is not the case in general. Indeed, if d_{imax} is the maximum delay at server i for the trajectory leading to the maximum end-to-end delay, we have: $\forall i$, $d_{imax} \le d^G_{imax}$ and thus $D_{max} \le \sum_i d^G_{imax}$.

B.2.i Additivity conditions. Let us first remark that the only chance for the end-to-end delay to be equal to the sum of the maximum local delays is that the bit that experiences the maximum delay in server 1 also experiences the maximum delay in server 2.

We adopt the following conventions:

1. $t=0$ is the time instant corresponding to the beginning of the activity period at the first server.
2. t_{1max} is the arrival time of the reference bit that experiences the maximum delay in the first server ($t_{1max} = \max(t \mid \frac{d\alpha_{S_1}(t)}{dt} > C_1)$).

3. $t_{1max}+d_1(t_{1max}) = t_{1max}+d_{1max}$ is the arrival time of the bit at the second server (d_{1max} is the maximum delay at server 1. $d_{1max} = \frac{\alpha_1(t_{1max})}{C_1} - t_{1max}$).

Let A_2 be the cumulative rate function of S_2 and θ_2 its instant of beginning of emission. Since the maximum delay at server 2 must be achieved at time $t_{1max} + d_{1max}$, the following conditions must hold:

$$\frac{d(\alpha_{1out} + A_2)(t)}{dt} > C_2 \text{ for } t \in [\theta_2, t_{1max} + d_{1max}] \quad (10)$$

$$\frac{d(\alpha_{1out} + A_2)(t)}{dt} < C_2 \text{ for } t > t_{1max} + d_{1max} \quad (11)$$

In the interval $[0, t_{1max} + d_{1max}]$, the output rate of server 1 is C_1 (backlog period). Thus, equations (10) and (11) become:

$$\frac{dA_2(t)}{dt} > C_2 - C_1 \text{ for } t \in [\theta_2, t_{1max} + d_{1max}] \quad (12)$$

$$\frac{dA_2(t)}{dt} < C_2 - C_1 \text{ for } t > t_{1max} + d_{1max} \quad (13)$$

For the previous conditions to hold, a necessary condition is $C_2 \geq C_1$. We also know that a necessary condition to generate d_{2max} is that S_2 is greedy. Let $t_{2max} = \max(t \mid \frac{d\alpha_2(t)}{dt} > C_2 - C_1)$. A necessary and sufficient condition for the bit arriving at time $t_{1max} + d_{1max}$ (and experienced d_{1max}) to experience d_{2max} is that $t_{2max} \leq t_{1max} + d_{1max}$ (since then $\theta_2 = t_{1max} + d_{1max} - t_{2max}$). We obtain the following theorem:

Theorem *8:* In a tandem m2p network, sources can be synchronized so as to generate an end-to-end delay equal to the sum of the local maximum delays if and only if:

$$\begin{cases} C_2 \leq C_1 \text{ and } t_{1max} + \frac{\alpha_1(t_{1max})}{C_1} \geq t_{2max} \text{ , with:} \\ t_{1max} = \max(t \mid \frac{d\alpha_1(t)}{dt} > C_1) \\ t_{2max} = \max(t \mid \frac{d\alpha_2(t)}{dt} > (C_2 - C_1)) \end{cases} \quad (14)$$

B.2.j Lower Bound. Let us now assume that the conditions of Theorem 8 are not fulfilled, i.e. $t_{1max} + \frac{\alpha_1(t_1max)}{C_1} \geq t_{2max}$ where:

$$t_{1max} = \max(t \mid \frac{\alpha_1(t)}{dt} > C_1) \quad (15)$$

$$t_{2max} = \max(t \mid \frac{(\alpha_{1out} + \alpha_2)(t)}{dt} > C_2) \quad (16)$$

Note that the definitions of t_{2max} given in equations (14) and (16) are equivalent since in equation (14), C_1 is the rate of the greedy source seen at the output of server 1. The key idea is to build a trajectory of S_1 where the burst leading to d_{1max} is delayed in such a way that the reference bit (experiencing $d_{1\ max}$) exits server 1 at time t_{2max} (we still assume that S_2 is greedy during $[0, t_{2max}]$). This trajectory is defined as follows:

1. S_1 is silent during $[t_{2max} - d_{1max}, t_{2max}]$
2. S_1 is greedy during $[t_{2max} - d_{1max} - t_{1max}, t_{2max} - d_{1max}]$

We still have to define the trajectory of S_1 during $[0, t_{2max} - d_{1max} - t_{1max}]$. We assume that it is maximal, i.e. that S_1 produces as much traffic as possible, under the constraint that it remains able to generate a burst leading to d_{1max} for $t \geq t_{2max} - d_{1max} - t_{1max}$. S_1 is thus able to generate as many bits as the greedy source during $[0, t_{2max} - d_{2max}]$, i.e. $\alpha_{1out}(t_{2max} - d_{1max})$. If the sources were greedy and synchronous, the second server would receive $\alpha_{1out}(t_{2max})$ in $[0, t_{2max}]$. Thus, one “looses” (as compared to the strictly synchronous case) the difference Q between these two quantities, that is $Q = \alpha_{1out}(t_{2max}) - \alpha_{1out}(t_{2max} - d_{1max})$. Thus the bit of S_1 that experiences d_{1max} in the first server experiences $d_{2max} - \frac{Q}{C_2}$ in the second server. The end to end delay of this bit is thus :

$$D = d_{1max} + d_{2max} - \frac{\alpha_{1out}(t_{2max}) - \alpha_{1out}(t_{2max} - d_{1max})}{C_2} \quad (17)$$

Since α_{1out} is given by Lemma 1, we can easily compute D. We have thus obtained a lower bound for the end-to-end delay since D corresponds to a trajectory of the system. If D is close to the sum of the local maximum delays, this would prove that the sum of maximum local delays provides a good approximation of the end-to-end delay. We further investigate this approach in the next section to obtain a bound on the end-to-end delay for m2p networks of arbitrary sizes.

VI. General Multipoint-to-Point Networks

In the previous section, we proved that a tandem m2p network is additive if and only if $t_{2\ max} \leq t_{1max} + d_{1max}$. We also characterized the corresponding additive trajectory: S_1 and S_2 greedy respectively from times $\theta_1 = 0$ and $\theta_2 = t_{1\ max} + d_{1\ max} - t_{2\ max}$. In the following, we call additive bound, the sum of the local maximum delays along the route of an m2p network. We generalize the approach of the previous section to the case of p servers in sequence. First note that any m2p network with p servers in sequence can be partitioned in a set of subnetworks for which the following property either holds or not:

Property *1:* For any two adjacent servers j and $j+1$, we have: $t_{(j+1)\ max} \leq t_{j\ max} + d_{j\ max}$.

A. Additive Networks

Consider an m2p network with p servers in sequence for which Property 1 holds. Let $(\theta_j)_{j \in \{1,\ldots,p\}}$ be defined as follows:

1. $\theta_1 = 0$
2. $\theta_{j+1} = \theta_j + (t_{j\ max} + d_{j\ max} - t_{(j+1)\ max})$, $j \in \{1, \ldots, p-1\}$

If S_j is greedy, starting from time $t = \theta_j$, (note that $\theta_{j+1} \geq \theta_j$), the bit experiencing $d_{1\ max}$ at server 1 experiences $d_{j\ max}$ at server j for all $j \in \{1, \ldots, p\}$. The end-to-end delay of this bit is thus: $D_{max} = \sum_{j=1}^{p} d_{j\ max}$. An m2p network for which Property 1 holds is thus additive. Besides, since the only way for a bit to experience $\sum_{j=1}^{p} d_{j\ max}$ is to experience $d_{j\ max}$ at server j ($j \in \{1, \ldots, p\}$), it follows that a network that does not fulfill Property 1 is not additive. This means that Property 1 is a necessary and sufficient conditions for m2p networks with p servers in sequence to be additive.

B. Non-additive Networks

In this section, we generalize the lower bound approach initiated in the tandem network case. We then use this lower bound to test the accuracy of the additive bound in the case of non-additive networks. A straightforward generalization would hide

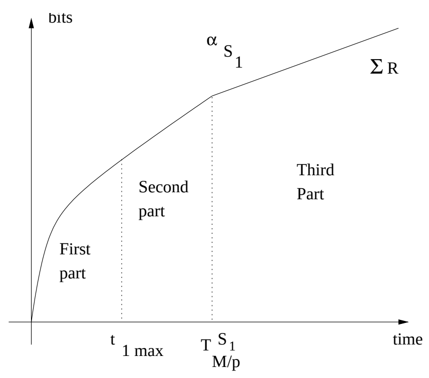


Fig. 8. S_1 initial trajectory

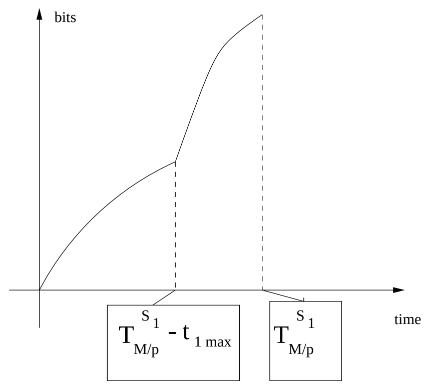


Fig. 9. S_1 modified trajectory

the difficulty of the construction of the trajectory. Therefore, we first present the three-server case.

B.1 Three-server Case

B.1.a Lower Bound. First, consider a two-stage network. If it is non-additive, this means intuitively that the burst leading to $d_{1\ max}$ is not sufficient to obtain $d_{2\ max}$ at server 2 (considering the greedy trajectory of the system), since when all the bits from this burst have reached server 2, the local delay on this server is less than $d_{2\ max}$. The idea behind the lower bound approach is to delay the burst at server 2 so as to synchronize local maximum delays. Obviously, the delay at the second server will necessarily be less than the delay in the greedy synchronous case.
Consider now an m2p network with three servers and three sources $(S_i)_{i\in\{1,\dots 3\}}$ (S_i entering at node i). The trajectories of the sources are chosen so as to maximize the amount of bits in buffer j when the reference bit (the one experiencing $d_{1\ max}$ at server 1) arrives.

B.1.b Trajectory of Sources. Consider the greedy trajectory of S_1 (see Figure 8). It can be divided into three parts. The first part corresponds to the part of the trajectory necessary to achieve the local maximum delay $d_{1\ max}$. The second part corresponds to the time interval necessary for the last bucket of the sources composing S_1 to empty. In the last part, all the sources composing S_1 emit at their mean rate.

Now consider the trajectory of S_1 given in Figure 9. S_1 is a multiplex of n_1 sources. The traffic descriptor of source k is (p_k, R_k, M_k) ($k \in \{1, \dots n\}$). For the greedy trajectory of the system, the source with index k emits at its peak rate p_k during $[0, \frac{M_k}{p_k}]$ and then emits at its mean rate R_k. Let us define:

$$T^{S_1}_{\frac{M}{p}} = \max_{k\in\{1,\dots,n_{S_1}\}} \left(\frac{M_k}{p_k}\right) .$$

$T^{S_1}_{\frac{M}{p}}$ corresponds to the beginning of the third part defined in Figure 8. The modified trajectory is built by changing the beginning of emission of the sources composing S_1 as follows:
1. if $\frac{M_k}{p_k} \le t_{1\ max}$ then S_k:
 (a) emits at its mean rate during $[0, T^{S_1}_{\frac{M}{p}} - t_{1\ max}]$,
 (b) becomes greedy for $t \ge T^{S_1}_{\frac{M}{p}} - t_{1\ max}$ (this is possible since its bucket is still full at this time).
2. if $\frac{M_k}{p_k} \ge t_{1\ max}$, then S_k:
 (a) emits at its mean rate during $[0, T^{S_1}_{\frac{M}{p}} - \frac{M_k}{p_k}]$,
 (b) becomes greedy for $t \ge T^{S_1}_{\frac{M}{p}} - \frac{M_k}{p_k}$.

The modified trajectory has two parts (see Figure 9):
1. A first part where some sources emit at their peak rate whereas others emit at their mean rates. This part corresponds to the second part of the initial greedy trajectory with a slight modification: if a source emits at its peak rate during τ_1 and then at its mean rate during τ_2 in the initial trajectory, then, in the modified trajectory, it first emits at its mean rate during τ_2 and then at its peak rate during τ_1. Due to this inversion between τ_1 and τ_2, we term this part the 'inverted part' of the modified trajectory.
2. A second part, strictly equivalent to the first one in the initial trajectory.
Note that, as with the initial trajectory, the last bucket empties at time $t = T^{S_1}_{\frac{M}{p}}$. A modified trajectory is built for S_2 and S_3 using the same method. We now set the synchronization parameters.

B.1.c Synchronization of Sources. With the modified trajectory described above for S_1, the last bit of the burst (reference bit) experiences a delay $d_{1\ max}$ at the first server. S_2 is triggered so that the end of its burst corresponds to the arrival of the reference bit. This bit will then experience $d_2 \le d_{2\ max}$ in the second server. Since, a priori, $T^{S_1}_{\frac{M}{p}} \ne T^{S_2}_{\frac{M}{p}}$, the previous synchronization method leads one of the sources to start emitting before the other. Assume that S_2 starts emitting before S_1. To maximize the number of bits backlogged at server 2 at the time where the reference bit arrives, it is possible to modify the trajectory of S_1 such that it emits at its mean rate before the beginning of the modified trajectory, in an interval of length $T^{S_2}_{\frac{M}{p}} - T^{S_1}_{\frac{M}{p}}$. This trajectory of S_1 is valid with respect to its leaky bucket constraint. The same method is applied to synchronize S_3, as shown in Figure 10.

B.1.d Result for Delay. The lower bound on the maximum end-to-end delay is obtained as the end-to-end delay of the reference bit in the modified trajectory. Since all the sources are leaky bucket constrained, the initial and modified trajectories correspond to piece-wise linear curves. Computation of the intrinsic parameters as well as the delay of the reference bit is thus straightforward from the algorithmic point of view.

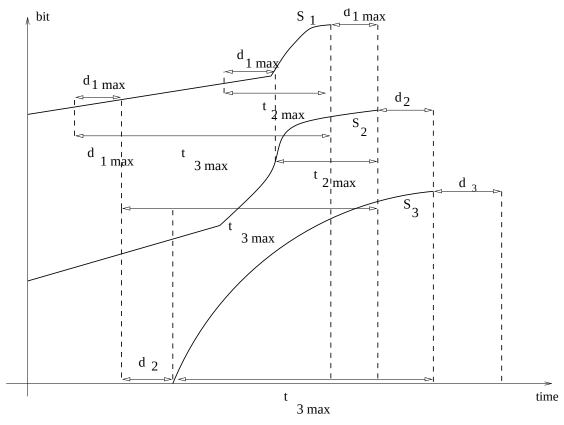


Fig. 10. Synchronization of sources

B.2 Numerical Results

We want to estimate the accuracy of the additive bound in a non-additive m2p network by using the lower bound presented above. Accuracy means here the relative difference between the additive bound and this lower bound. We consider m2p networks with $p = \{4, 5, 8, 10\}$ servers in sequence. For each server, we draw the number of sources entering at this stage in a uniform fashion in the set $\{1, \ldots, 5\}$. Characteristics of the sources are also randomly chosen from Table I using a uniform law. We have to set the capacities of the servers. A necessary

TABLE I

SOURCES DESCRIPTORS

Peak rate p	*Mean rate* M	*Burstiness* M
10	0.1	10
100	1	100
1000	10	1000

condition for a network to be additive is that the rates of the servers increases (from the leaves to the root). Conversely, if capacities decrease, the network is non-additive (sufficient but not necessary). We set the service rate of all servers to be equal to the sum of the mean rates of all the sources times $\gamma = 1.01$ (γ is used to ensure stability). This sum represents the minimum capacity of the last server. Doing so, the most important part of the end-to-end delay is concentrated on the last server of the network. To obtain significant results, we calculate the relative range, defined as the difference between the lower bound and the additive bound divided by the additive bound, for this initial system, i.e. a particular random generation of the sources descriptors and capacities of servers. We then change the input server of some of the sources. The following algorithm is applied:

Step 1: the initial network is built.

Step 2-9: each source is “moved” from node j to node $j - 1$ with probability 0.1.

Applying this algorithm, the m2p network heuristically “worsens” and thus the relative range should increase.

The results, presented in Figure 11, are obtained for 10000 successive random generations of networks. The x-axis is indexed following the steps of the algorithm. For each step of the algorithm and for each network size, we compute the mean relative range.

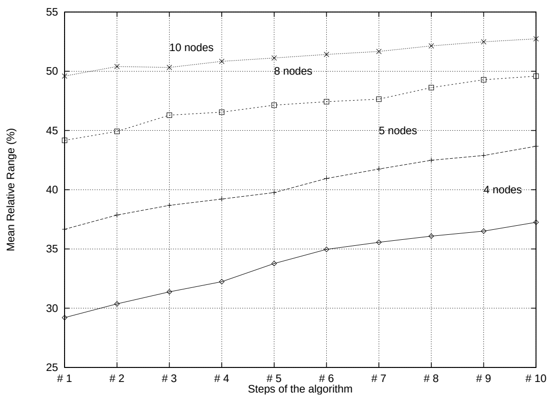


Fig. 11. Accuracy of the additive bound

B.2.a Discussion. For non-additive m2p networks, we have proposed an upper bound on the end-to-end delay, the additive bound, and a heuristically obtained lower bound. The maximum, exact, end-to-end delay over all possible trajectories of the system lies between these two bounds and gives full meaning for considering the relative range as a performance parameter. The obtained results confirm the good accuracy of the additive bound. The mean relative ranges remain reasonable even for large size of networks. The maximum error, not presented here is no more than 67%. It thus remains within the same order of magnitude, which clearly indicates that the additive bound is a valid approximation of the end-to-end delay.

C. Well-formed Multipoint-to-Point Networks

Our expectation is that the additive bound always represents an accurate upper bound on the maximum end-to-end delay for m2p networks. Proving such a statement requires an exhaustive study, which is not possible. We restrict our study to a specific class of m2p networks, that we term well-formed m2p networks. A well-formed m2p network is an m2p network where the following rule applies: capacities of the servers increase from the leaves to the root of the tree. We extend here the previous results to the case of well-formed m2p networks with p servers in sequence, using the same lower bound as in the non-additive case. Indeed, the method used to build the trajectory leading to the lower bound is based only on the set of intrinsic parameters $(t_{j\ max}, d_{j\ max})$. It does not rely on any assumption concerning the additivity of the network. It may thus be applied to the case of well-formed m2p networks.

C.1 Results

The method used to generate a well-formed network is the following:

1. For each server, the number of sources (between 1 and 5) entering at this node and their characteristics are drawn from Table 1 using uniforms laws.

2. The service rate of server j is then set to the sum of the mean rates of the sources served by this server times a coefficient α. α can take one of the three following values $\{1.1, 1.5, 2.0\}$, which, for each set of sources, leads to three different networks.

We present in Table II the numerical results obtained for networks of various sizes (from 3 to 20 servers). For each network size, 10000 networks are drawn. The performance parameter computed for each network is the relative range between the lower bound and the additive bound.

TABLE II
AVERAGE RELATIVE RANGES (IN %) WITH NETWORKS OF DIFFERENT SIZES

	Size=3	*Size=5*	*Size=10*	*Size=15*	*Size=20*
$\alpha = 1.1$	0.72	1.29	2.03	2.89	3.87
$\alpha = 1.5$	1.97	3.36	5.67	8.42	11.23
$\alpha = 2.0$	1.96	3.13	5.06	7.64	10.18

C.2 Discussion

The results in Table II strongly confirm our claim: the additive bound represents an accurate approximation of the end-to-end delay. These results are also interesting since the way well-formed m2p networks are built here is close to a real dimensioning process. Indeed, α^{-1} is the rate of the server divided by the sum of the average rates of the sources it serves. It thus represents the average activity rate of the servers and tuning activity rates at a given value is a common way to dimension networks. Compared to the results obtained in the previous section, the relative ranges obtained here are significantly smaller: for instance, for a network with 10 servers, the relative range was close to 50% whereas, here, it is close to 5%. This is due to the method used to set the server rates in each case. In the case of strictly non-additive m2p networks, all the servers had the same capacity, which lead to a strictly non-additive network, whereas here, the rates increase from one server to another, which is a necessary (though not sufficient) condition to obtain an additive network.

VII. Admission Control Algorithm

In this section, we derive an admission control algorithm based on the additive bound presented in the previous section. We propose two versions of the algorithm, a centralized and a distributed version.

A. Centralized Algorithm

Consider first a single FIFO server and n leaky bucket constrained sources. The maximum delay is obtained when all the sources are greedy and strictly synchronous. An arrival curve of the aggregated source is the sum of the arrival curves of all the sources. Since summation is a commutative operation, the maximum delay does not depend on the order in which sources are introduced in the network. Thus, from the admission control algorithm point of view, the answer to the admission request of a new source in a single-server network with n established sessions is equivalent to the answer to the request of the $n+1$ sources simultaneously.
Consider now an m2p network. With an admission control algorithm based on the additive bound, the admission of a new session requires to compute the local maximum delay at each server along the path of the session up to the root server. Since the root server belongs to the path of all sources, admitting a new source, with n sessions already set-up, is equivalent to admit these $n+1$ sources simultaneously. We make use of this property to simplify the presentation of the centralized algorithm. The problem to solve is the following: "Given n sources with specific QoS requirements, is it possible to admit these n sources simultaneously?". The algorithm has two phases: (i) computation of the additive bounds along each path of the network and (ii) checking the non-violation of the QoS constraint of each session.

A.1 Computing the Additive Bound

So far, the computation of the additive bound as been presented only in the case of m2p networks with servers in sequence. Generalization to a tree m2 networks relies on the following observation: the flow seen at the output of a given subtree of a given m2p network is multi-leaky bucket constrained (see Lemma 2 and 3). As a consequence, (i) maximum local delays are obtained when all the sources are greedy and synchronous, and (ii) computation of these delays can be made starting from the leave servers and moving to the root server.

A.2 Checking the QoS Constraint

Once the maximum local delays are obtained, we can compute the additive bound along each path of the network. We assume that the centralized algorithm has a complete knowledge of the network topology and of the input server of each source (which is equivalent to know its path in the network). We must thus compare, for each session, the required end-to-end delay and the additive bound along its path to accept or reject the new session request.

B. Distributed Algorithm

When executed, with n sources already accepted, the admission control algorithm must process the request from a new source. Let $(S_i)_{i\in\{1,\ldots,n\}}$ be these already accepted sources, $(D_i)_{i\in\{1,\ldots,n\}}$ their delay requirements and $(D_i^{\text{eff}})_{i\in\{1,\ldots,n\}}$, the effective delays, i.e. the additive bounds along the path of the sources.
Since $(S_i)_{i\in\{1,\ldots,n\}}$ have already been accepted, $D_i \le D_i^{\text{eff}}$ ($\forall i \in \{1,\ldots,n\}$). The quantities $\delta_i = D_i^{\text{eff}} - D_i$ ($i \in \{1,\ldots,n\}$) represent safety margins for the sources.
The admission of a new source S_{n+1} requires to re-compute local delays for all servers along the path of S_{n+1}, but not on all the servers in the network. As explained before, this operation can be made sequentially starting from the input server of S_{n+1} and moving down to the root server. To limit the amount of computation, each server stores the arrival curve of the flow seen at each of its inputs. Let I_i be the set of indices of the servers along the path of S_i and $(\Delta_j)_{j\in I_{n+1}}$ the variations of the local maximum delays induced by S_{n+1} at the servers of I_{n+1}. The admission algorithm must check whether the admission of the new source violates the QoS requirements of the other sources,

which can be expressed through the following system of equations:

$$\delta_i \geq \sum_{k \in I_i \cap I_{n+1}} \Delta_k, \, \forall i \in \{1, \ldots, n\} \tag{18}$$

For each source S_i, $i \in \{1, \ldots, n\}$, $I_i \cap I_{n+1}$ is the set of servers where S_i and S_{n+1} meet. This set is never empty: it contains at least the root server. The problem with the distributed algorithm is that checking equations (18) can be made only at the root server, since this is only at this server that all the local delay variations $(\Delta_j)_{j \in I_{n+1}}$ are known. The final admission decision is thus made at this last step. Therefore, we have two options :

1. If we want to limit the amount of messages exchanged between servers, safety margins should be stored in the root server only. Then, to check equations (18), the root server needs to have a complete knowledge of the network topology and of the path of each session. This is possible only for small networks and a small number of sources. Another drawback of this method is that QoS violations are detected at the last possible moment. For instance, if at the input server of S_{n+1}, the delay variation Δ is greater than the safety margin δ of a source served by this server, the admission algorithm could have been aborted at this step of the procedure.
2. On the opposite, if we impose that a server only has a local vision of the network, then, once a source is accepted, the following operations must be performed:
   (a) the safety margin δ must be distributed among all the servers of the path of this source.
   (b) if a safety margin is modified because of a new source has been accepted, it must be transmitted to the next server since the two sources now share the same path and thus the new source will change this safety margin on every server until the root server is reached.
   (c) if a source is accepted, one must ensure that all safety margins are correctly updated.

We now present an algorithm that does not rely on the assumption that the root server has a complete knowledge of the network topology and of the routes of the sources. We describe the data structures used at each server and provide a skeleton of the two phases of the algorithm: in the first phase, local delay variations are computed and QoS violations are checked. In the second phase, called the termination phase, the decision to admit or reject the new source is made.

B.1 Data Structures

Each server stores a table with, for each source that it serves, an identifier and its safety margin. Each server must also store the arrival curve of the input flow at each of its interfaces (when a new source arrives, only one arrival curve is modified). An arrival curve is stored as a list of points since with leaky bucket constrained sources and FIFO servers, arrival curves are piecewise concave linear functions. The maximum local delay d_{max} must also be stored.

B.2 First step: admitting a new source

This step of the algorithm is initiated by the input server of the new source and is propagated sequentially until the root server is reached. Let D be the sum of the maximum local delays between the input server of the new source and the current server. D becomes equal to the additive bound when the current server is the root server.

```
Algorithm at server j ($j \in I_{n+1}$):
1. Upon receipt of a new session request: computation of the new arrival curve of the input flow.
2. Safety margins are (temporarily) updated using the list of modified safety margins received from the previous server. They become effective only if the request is accepted.
3. Computation of the new value of the maximum local delay $d_{max}$, which gives the variation $\Delta_j$ and the new value of $D$, i.e. $D + d_{max}$.
4. If $[(\min_{i \mid j \in I_i} \delta_i \geq \Delta_j)$ and $(D \leq D_{n+1})]$ then: (the source is locally accepted)
  (a) if (server $j ==$ root server) then:
    i. A *Confirmation of Acceptance Message* is sent to the previous server in the path of $S_{n+1}$ with the new values of the safety margins of the sources arriving at this interface.
    ii. *Updating Messages* are sent on the other interfaces with the final value of the safety margins (which are known only at this stage)
  (b) else:
    i. Storage of the new (temporary) value of the safety margins: $\forall i, j \in I_i, \delta_i = \delta_i - \Delta_j$
    ii. Transmission to the next server in $I_{n+1}$ of:
      A. the set $(\delta_i)_{j \in I_i}$
      B. the new value of $D$
      C. the arrival curve at the output
5. Else: a *Rejection Message* is sent to the previous server in $I_{n+1}$.
```

B.3 Second Step: Updates

This phase is initiated by the root server or by the server where a QoS violation is detected. There are three cases:

1. Receipt of a *Confirmation of Acceptance Message*. Only servers from I_{n+1} may receive this message. The server must:
   (a) update the safety margins with the received values (including S_{n+1})
   (b) forward this message to the previous server in I_{n+1}
   (c) send *Updating Messages* on the other ingress links with the corresponding safety margins values
2. Receipt of a *Rejection Message*. Only servers from I_{n+1} may receive this message. They have to:
   (a) release the temporary structures (safety margin values, arrival curve and maximum local delay)
   (b) forward this message to the previous server in I_{n+1}
3. Receipt of an *Updating Message*. Only servers that do not belong to I_{n+1} may receive this message. The server must:
   (a) update its current safety margins with the received values
   (b) forward this message with the corresponding safety margins on each of its ingress links.

B.4 Session Termination

The admission control algorithm is also executed at each session termination. The local delays for the servers of the path of this session must be updated as well as the safety margins of the sources. The procedures involved are similar to the ones used for acceptance.

B.5 Discussion

To ensure the correctness of the algorithm, two admission procedures cannot be made simultaneously. They must be sequentialized. If the two admission procedures are initiated on two disjoint paths in the tree network, the root server will have to choose which source is treated first. This will obviously have an impact on the other source since the source that is treated first is more likely to be accepted than the second one. Note, however, that the admission procedure for the second source does not have to be re-initiated. For the case where the two admission procedures are initiated on the same path, the first one that

reaches the first server that the two sources share, gains priority over the other one. This means that the second procedure is delayed until the decision for the first source is made.
Note finally that the algorithm converges as long as no message is lost. A reliable communication channel, such as a permanent TCP connection, may be used to ensure that no messages are lost between adjacent servers.

C. Example

We further illustrate the distributed admission control algorithm presented above for the case of the m2p network of Figure 12. For this figure as well as Figures 13 and 14, we adopt the following conventions:
1. Each source is constrained by a single leaky bucket.
2. S_j is the source entering the network at server j, D_j its required end-to-end delay and δ_j its safety margin.
3. For each server, $d_{j\ max}$ is the current maximum local delay at a given step of the algorithm and $d'_{j\ max}$ the new value of the maximum local delay. Δ_j is the maximum local delay variation, i.e. $\Delta_j = d'_{j\ max} - d_{j\ max}$.
4. Procedures used at each server are represented with squares linked to servers in Figures 13 and 14. Expressions like "$D = d'_1 \leq D_1$?" correspond to tests performed by the server. We suppose that all the tests succeed, which allows to study the acceptance of a new source.
5. Arrows between squares correspond to data exchanges.
6. D represents the value of the additive bound along the considered path at the consider server. It becomes equal to the additive bound at the root server.

We assume that sources S_4, S_5, S_7 and S_8 are already set up and consider the admission of S_1. There are two steps in the algorithm. The first one (Figure 13) corresponds to the computation of the additive bound. It begins at server 1 and moves down to server 3. At each server, the algorithm tries to detect any QoS violation for the already established sessions as well as for the new source. The second step (once S_1 is accepted - Figure 14) corresponds to the updates of the safety margins of all sources.

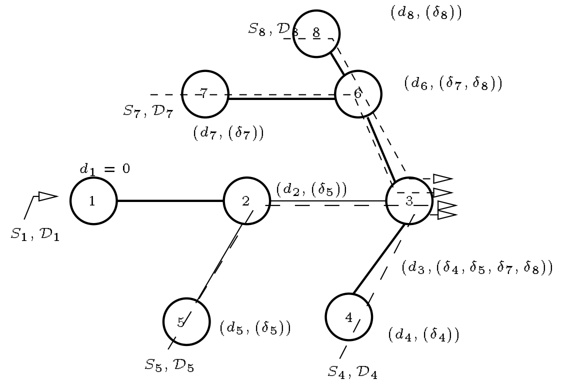
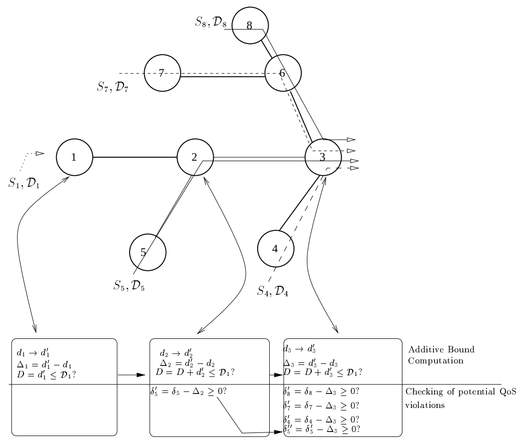


Fig. 12. m2p network with 8 servers



$S_8, \mathcal{D}_8$
8
7
6
$S_7, \mathcal{D}_7$
1
2
3
$S_1, \mathcal{D}_1$
5
4
$S_5, \mathcal{D}_5$
$S_4, \mathcal{D}_4$
$d_1 \to d'_1$
$\Delta_1 = d'_1 - d_1$
$D = d'_1 \leq \mathcal{D}_1$?
$d_2 \to d'_2$
$\Delta_2 = d'_2 - d_2$
$D = D + d'_2 \leq \mathcal{D}_1$?
$\delta'_5 = \delta_5 - \Delta_2 \geq 0$?
$d_3 \to d'_3$
$\Delta_3 = d'_3 - d_3$
$D = D + d'_3 \leq \mathcal{D}_1$?
$\delta'_8 = \delta_8 - \Delta_3 \geq 0$?
$\delta'_7 = \delta_7 - \Delta_3 \geq 0$?
$\delta'_4 = \delta_4 - \Delta_3 \geq 0$?
$\delta''_5 = \delta'_5 - \Delta_3 \geq 0$?
Additive Bound Computation
Checking of potential QoS violations

Fig. 13. First step: bound determination

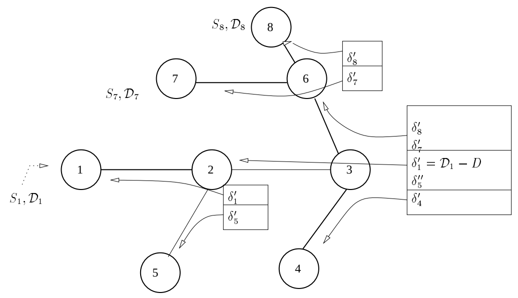


Fig. 14. Second step: updating phase

amount of data to transfer. We study successively the two main phases of the algorithm, namely "Computation of the bound" and "Updates".
Let us consider a source S that traverses p servers (see Figure 15). We adopt the following conventions:
1. The index of the interface where S enters at each server is denoted as (0).
2. For each server j, we define a pair (n_j, k_j), where n_j is the number of sources entering at server j by an interface other than (0) and k_j is the number of servers belonging to the paths of these sources (each server is counted only once).
3. $N = \sum_{j=1}^{p} n_j$ is the total number of active sources in the network and $K = \sum_{j=1}^{p} k_j$ is the total number of active servers.

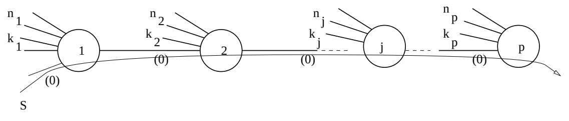


Fig. 15. Reference Configuration for estimating the complexity

D. Complexity

To study the complexity of the admission control algorithm presented above, we evaluate its storage requirement and the

D.1 Phase 1: Bound Computation

D.1.a Storage Requirement Q. Each server stores the arrival curve for the incoming flow and the safety margins of each source that it serves. An arrival curve is stored as a list of points. Considering a single server with n input sources, the number of

points to store is upper bounded by $n+1$ since each greedy source adds one point corresponding to the time where its emission rate decreases from its peak rate to its mean rate and the server adds one point corresponding to the time instant where it clears the backlog. Applied to server j of Figure 15, we obtain that the total number of points of the arrival curve for the incoming flow is upper bounded by the sum of the number of sources crossing this server and the number of servers that have already treated these sources. For each point, the total amount of data to store is constant (λ). The storage capacity required to store the arrival curve of the input flow is:
1. At server 1:
$\lambda((n_1 + p_1 + \underbrace{1}_{S}) + \underbrace{1}_{\text{server}})$
2. At server 2: $\lambda((n_1 + 1 + p_1 + 1) + (n_2 + p_2))$
3. $\ldots$
4. At server p, $\lambda(\sum_{j=1}^{p}(n_j + k_j + 1) + 1)$
The total amount Q_1 of memory required to store the arrival curves can be upper bounded by p times the amount of data required at server p. We obtain:

$$Q_1 \leq p\lambda(\sum_{j=1}^{p}(n_j + k_j + 1) + 1) \leq \lambda p(N + K + p + 1). \quad (19)$$

Thus ,

$$Q_1 = O(p(N + K + p)). \quad (20)$$

As for safety margins, each server stores the safety margins of all the sources that is serves. Let γ be the size of the memory used to store a safety margin. Then, server 1 has to allocate a memory of size $(n_1+1)\gamma$, server 2 has to allocate $(n_1+1+n_2)\gamma$ and server p, $(\sum_{j=1}^{p} n_j + 1)\gamma$. The total amount Q_2 of memory required to store the safety margins may be upper bounded by p times the amount of memory required at server p. We obtain:

$$Q_2 \leq p\gamma(\sum_{j=1}^{p} n_j + 1) \leq \gamma p(N + 1). \quad (21)$$

Thus,

$$Q_2 = O(pN). \quad (22)$$

Eventually, we obtain:

$$Q = Q_1 + Q_2 = O(p(N + K + p)) \quad (23)$$

D.1.b Amount of Transmitted Data X_1. Each server provides its neighbors with its output arrival curve and its set of safety margins. As a consequence, the amount X_1 of data to transfer is of the same order of magnitude as Q.

D.2 Phase 2: Updates

In the second phase of the admission control algorithm (when a new session is accepted), the root server provides each server with the new values of the safety margins of the sources that it serves. There are N safety margins and a given server must at most transmit all these safety margins. Since there are K servers, the total amount of data to transfer X_2 in the second phase is such that:

$$X_2 = O(KN) \quad (24)$$

E. Discussion

Q, X_1 and X_2 depend on the number of sources, the number of servers and the length of the path of S. To provide orders of magnitude, helping at discussing complexity issues, we use the following assumptions :
1. the length of the path is equal to the mean length of a path in a binary tree network, i.e. $p \sim log_2 K$.
2. the network is dimensioned so that the number of servers is proportional to the number of sources, i.e. $K \sim O(N)$.
With the two above assumptions, equations (23) and (24) become $Q = O(N log_2 N)$ and $X_1 + X_2 = O(N^2)$. Thus, when the number of sources is increased by a factor of two, the amount of data to be transferred increases by a factor of four. This non-linearity indicates that the admission algorithm may not scale well. A detailed analysis of the algorithm indicates that while the first phase of the algorithm (bound computation) is computationally intensive, the lack of scalability is mainly due to the second phase (updates). Indeed, this second phase results in a flooding of the network so as to ensure the exactness of the admission algorithm. However, we should keep in mind that the concentration of the traffic at the edge servers results from their role as interfaces between backbones of different ISPs. It is not due to the m2p architecture. The m2p architecture is used to reduce the cost in terms of number of connections (or LSPs) required to cover the network ($O(n)$ rather than $O(n^2)$).
We are now at the point where we can provide some guidelines for the design of an operational admission control algorithm for our traffic management scheme:
• If the backbone has a moderate size or, more precisely, if there is a moderate number of edge routers, then a centralized solution is a good option. A dedicated server, connected to all the edge routers via an m2p LSP, acts as an admission control server, the so-called bandwidth broker in the DiffServ terminology [17], [18]. The bandwidth broker must have a complete knowledge of the paths of each source (and thus the topology of the networks) with their characteristics and their safety margins. Note that the bandwidth broker only interacts with edge routers, not with interior routers (but it must keep track of the changes in the topology, which can be done through the routing protocol for instance).
• If the backbone is large, a distributed algorithm should be used. However, in this case, some additional means must be used to guarantee the scalability of the traffic management scheme. Note that in the distributed case, not only edge routers but also all interior routers are engaged in the admission control procedure. A way to ensure scalability could be to minimize the frequency of execution of the admission control algorithm. This may be achieved with an adequate grouping of sessions at the ingress servers. For instance, the set of sources issued by an other ISP with the same QoS constraint, can be grouped. This could be done by the bandwidth broker of a given domain or by the clients of an ISP who would rent VBR trunks.

VIII. Conclusion and Outlook

Traffic engineering is getting more and more important with the emergence of applications with QoS constraints and potentially a highly varying emission rate. Current routing algorithms do not take QoS constraints into account while ISPs need to have more control over the routes followed by packets in their network. A mixed solution that combines routing and forwarding, as proposed by MPLS, is very appealing. In the context of MPLS, the multipoint-to-point architecture is a key architecture. In this paper, we have discussed the fundamental problem of designing a complete traffic management scheme for multimedia applications and for m2p networks. The first problem is to obtain an accurate upper bound on the end-to-end delay in an m2p architecture. A bounding approach is required as demonstrated in the study of a tandem m2p network. Therefore, we introduce the concept of additivity. A path in an m2p network is additive if its maximum end-to-end delay is equal to the sum of the local maximum delays. We show that there exists a whole class of m2p networks that are additive. For the most intricate case of non-additive networks, we show that the additive bound represents an accurate approximation of the maximum end-to-end delay.

We next propose two admission control algorithms based on the additive bound. The first algorithm is a centralized one, the second algorithm is a distributed one. We discuss the key aspects of the two algorithms and especially their complexity and their scalability. This enables us to provide some guidelines concerning the design of a complete traffic management scheme. The choice of a distributed or centralized version depends heavily on the number of routers (edge routers and interior routers) in the backbone. All in all, it seems that a centralized version with an admission control server acting as the so-called bandwidth broker in DiffServ, is an appealing solution.

Future work should concentrate on practical experiments to compare the centralized and distributed versions. Another important research issue is the use of multiple m2p Label Switch Paths among a pair of ingress/egress routers. This method allows to achieve reliability and load balancing [10]. Our traffic management scheme could be extended to the multiple m2p LSP case with each m2p LSP representing a given class of service.

References

[1] E. Rosen, A. Viswanathan, and R. Callon, “Multiprotocol label switching architecture,” *Internet Request for Comments*, vol. RFC 3031, Jan. 2001.

[2] B. Davie, J. Lawrence, and K. McCloghrie, “MPLS using LDP and ATM vc switching,” *Internet Request for Comments*, vol. RFC 3035, Jan. 2001.

[3] A. Conta, P. Doolan, and A. Malis, “Use of label switching on frame relay networks specification,” *Internet Request for Comments*, vol. RFC 3034, Jan. 2001.

[4] A. K. Parekh and R. G. Gallager, “A generalized processor sharing approach to flow control in integrated services networks: The single node case,” *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 344–357, June 1993.

[5] A. K. Parekh and R. G. Gallager, “A generalized processor sharing approach to flow control in integrated services networks: The multiple node case,” *IEEE/ACM Transactions on Networking*, vol. 2, no. 2, pp. 137–150, Apr. 1994.

[6] R. L. Cruz, “A calculus for Network Delay, Part I: Network Elements in Isolation,” *IEEE Transactions On Information Theory*, vol. 37, no. 1, pp. 114–131, Jan. 1991.

[7] R. L. Cruz, “A calculus for Network Delay, Part II: Network Analysis,” *IEEE Transactions On Information Theory*, vol. 37, no. 1, pp. 132–141, Jan. 1991.

[8] R. L. Cruz, “Quality of service guarantees in virtual circuit switched networks,” *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 6, pp. 1048–1056, Aug. 1995.

[9] H. Ahmed, R. Callon, A. Malis, and J. Moy, “IP switching for scalable IP services,” *Proceedings of the IEEE*, vol. 85, no. 12, 1997.

[10] H. Saito, Y. Miyao, and M. Yoshida, “Traffic engineering using multiple multipoint-to-point LSPs,” in *Proceedings of IEEE Infocom*, Tel Aviv, Israel, Mar. 2000.

[11] L. Tassiulas and L. Georgiadis, “Any work-conserving policy stabilizes the ring with spatial reuse,” in *Proceedings of IEEE Infocom*, Toronto, Canada, June 1994.

[12] I. Chlamtac, A. Faragó, and H. Zhang, “A deterministic approach to the end-to-end analysis of packet flows in connection-oriented networks,” *IEEE Transactions on Networking*, pp. 422–431, August 1998.

[13] J.-Y. Le Boudec and P. Thiran, *Network Calculus*, Springer Verlag LNCS 2050, June 2001.

[14] C.-S. Chang, *Performance Guarantees in Communication Networks*, Springer-Verlag, 2000.

[15] J.-Y. Le Boudec, “An application of Network Calculus to guaranteed service networks,” *IEEE Transactions on Information Theory*, vol. 44, no. 3, May 1998.

[16] G. Urvoy, G. Hébuterne, and Y. Dallery, “Delay-constrained VBR sources in a network environment,” Tech. Rep. 98-10-04, Institut National des Télécommunications, 1998.

[17] S. Blake, D. Black, M. Carlson, E. Davies, and W. Weiss Z. Wang, “An architecture for Differentiated Services,” *IETF RFC 2475*, 1998.

[18] X. Xiao and L.N. Ni, “Internet QoS : A big picture,” *IEEE Network*, pp. 8–18, March/April 1999.

Guillaume Urvoy-Keller received the Engineer Degree from the Institut National des Télécommunications in 1995 and the Ph.D. in Computer Science from the University of Paris VI in 1999. In 1999-2000, he was an Assistant Professor at the University of Versailles. He is currently an Assistant Professeur at Institut Eurecom. His interests are in the Quality of Service provisioning and traffic engineering for the Internet.

Gérard Hébuterne holds a “Doctorat” (PhD) and an “Habilitation à diriger les Recherches”. He was with CNET (France Telecom research lab) from 1973 to 1994. He has first specialised in traffic studies in SPC switches and then participated actively in performance studies for broadband systems (ATM, FR). He is with the Institut National des Tlcommunications since July 1994, where he leads the Networks Department. He has specialised in traffic studies, and especially overload control in telecommunications systems and broadband networks. His present work focuses on the Quality of Service aspects in multiservices broadband networks.

Yves Dallery received his Ph.D. and the degree of “Habilitation à Diriger des Recherches” from the Institut National Polytechnique de Grenoble (INPG) in 1984 and 1989, respectively. He is currently Professor of Manufacturing and Logistics at Ecole Centrale de Paris. Before that, he was Directeur de Recherche at the Centre National de la Recherche Scientifique (CNRS). In 1984-1985, he was a post-doctoral fellow at Harvard University. In 1991-1992, he was a visiting scientist at M.I.T. and in 1992-1993 he was an Associate Professor of Manufacturing Engineering at Boston University. His research interests are in operations management, supply chain management, and stochastic models.