A Distributed Algorithm for Weighted Max-Min Fairness in MPLS Networks

Fabian Skivée[1] and Guy Leduc[1]

Research Unit in Networking University of Liège (Belgium)
Fabian.Skivee@ulg.ac.be Guy.Leduc@ulg.ac.be

Abstract. We propose a novel distributed algorithm to achieve a weighted max-min sharing of the network capacity. We present the Weight Proportional Max-Min policy (WPMM) that supports a minimal rate requirement and an optional maximal rate constraint and allocates network bandwidth among all aggregates based on a weight associated with each one. Our distributed algorithm achieves this policy for IP/MPLS networks using the RSVP-TE signalling protocol. The algorithm uses per-LSP accounting in each node to keep track of the state information of each LSP. It uses a novel explicit bottleneck link strategy and a different control architecture in which we update the control packet in the forward path. Simulations show that these two elements improve substantially the convergence time compared to the previous algorithms developed to achieve this policy in the ATM context.

1 Introduction

Traffic engineering aims at using information about traffic entering and leaving the network to optimize network performance. An ISP using MPLS [10] would like to optimize the utilization of its available network resources between all the LSPs. The choice of a network bandwidth sharing policy among competing LSPs is a key performance issue.

Consider a set of LSPs, each carrying many TCP connections, creating congestion. Without explicit policing, more aggressive LSPs (with more flows) get more that their fair share, independently of their reservations.

The problem of fair sharing of the network bandwidth has been widely treated in the past and especially in the ATM context ([5] [8]). The classical max-min rate allocation has been widely accepted as an optimal network bandwidth sharing criterion among user traffic flows [1]. Informally, the max-min policy attempts to maximize the network use allocated to the users with the minimum allocation [4].

To provide differential service options, [7] proposes a generic weight proportional network bandwidth sharing policy, also called Weight-Proportional Max-Min (WPMM). The WPMM policy generalizes the classical max-min by associating each flow with a generic weight, which is decoupled from its minimum rate and supports the minimum rate requirement and the peak rate constraint for each flow.

Since a centralized algorithm for the max-min rate allocation requires global network information, which is not scalable to flood, we must develop distributed

algorithms to achieve the same rate allocation in the absence of global knowledge about the network and without synchronisation of different network components. We focus on an edge-to-edge rate-based feedback control scheme, where special control packets are used in both forward and backward paths.

Hou, in [7], provides a good distributed solution for computing the WPMM policy but has two major drawbacks. Firstly, this solution is implemented in the ATM context and uses a lot of Resource Management (RM) ATM cells that create a substantial overhead. And secondly, the convergence is usually slow as illustrated later.

Our proposal solves these two limitations by adding an explicit bottleneck link information in each control packet and by using a different control packet update architecture. These two additions addition change radically the dynamics of the protocol and improve the convergence time by a factor 2 or 3 depending on the precision (see simulation results).

Another contribution is the adaptation of the solution to the actual MPLS architecture using the widespread RSVP-TE protocol [2] instead of RM cells. With our proposed integration in RSVP, we improve the scalability of the protocol by decreasing the overhead added by control packets.

As a possible application of this work, the weighted max-min fair rate allocated to an LSP could be used by a marker at the ingress router to mark the traffic using three colours : green (under reserved rate), yellow (between reserved rate and the fair rate) and red (above the fair rate). In case of congestion, core routers discard the red packets first and possibly, during transient periods, some of the yellow packets by using a WRED policy for example. The objective of the algorithm is to compute the fair rate of each LSP to obtain a network in which only the red packets are discarded and all the green and yellow packets successfully get through.

2 The generic weight-proportional allocation policy

A unified definition of the max-min fairness is provided by [9]. Consider a set $\mathcal{X} \subset \mathbb{R}^N$, the definition of the weighted max-min fair vector with respect to set $\mathcal{X}$ is defined as follows :

Definition 1. *A vector $\overrightarrow{x}$ is "weighted max-min fair on set $\mathcal{X}$" if and only if*

$$(\forall\ \overrightarrow{y} \in \mathcal{X})\ [\ (\exists s \in \{1,\ldots,N\})\ y_s\ >\ x_s \Rightarrow\ (\exists t \in \{1,\ldots,N\})\ \frac{y_t}{w_t} < \frac{x_t}{w_t} \leq \frac{x_s}{w_s}\ ]$$

where w_i is the weight associated with the element i

We can use this theoretical definition to share the free bandwith available in an MPLS network. Our work is based on a previous work in the ATM context provided by [7]. We adapt their definition to the MPLS context.

An MPLS network is a set of IP/MPLS routers interconnected by a set of links $\mathcal{L}$. A set of LSPs $\mathcal{S}$ traverses one or more links in $\mathcal{L}$ and each LSP $s \in \mathcal{S}$

is allocated a fair rate r_s. Denote $\mathcal{S}_l$ the set of LSPs traversing link $l \in \mathcal{L}$. The aggregate allocated rate F_l on link l in $\mathcal{L}$ is

$$F_l = \sum_{s \in \mathcal{S}_l} r_s$$

Let C_l be the capacity (maximum allowable bandwidth) of link l. A link l is saturated or fully utilized if $F_l = C_l$. Denote RR_s and MR_s, the reserved rate requirement and the maximal rate constraint for each LSP $s \in \mathcal{S}$, respectively. For feasibility, we assume that the sum of all LSPs' RR requirements traversing any link does not exceed the link's capacity, i.e. $\sum_{s \in \mathcal{S}_l} RR_s \leq C_l$ for every $l \in \mathcal{L}$. This criterion is used by admission control at LSP setup time to determine whether or not to accept a new LSP on a link.

Definition 2. *We say that a rate vector $r = \{r_s \mid s \in \mathcal{S}\}$ is **MPLS feasible** if the following two constraints are satisfied :*

$$RR_s \leq r_s \leq MR_s \text{ for all } s \in \mathcal{S}$$
$$F_l \leq C_l \text{ for all } l \in \mathcal{L}$$

In the generic weight-proportional max-min (WPMM) policy, we associate each LSP $s \in \mathcal{S}$ with a weight (or priority) w_s. Informally, this policy first allocates to each LSP its RR. Then from the remaining network capacity, it allocates additional bandwidth for each LSP using a proportional version of the max-min policy based on each LSP's weight while satisfying its MR constraint. The final bandwidth for each LSP is its RR plus an additional "weighted" max-min fair share. Formally, this policy is defined as follows and directly derives from Def 1.

Definition 3 (WPMM-feasible vector). *A rate vector r is weight-proportional max-min (WPMM) if it is MPLS-feasible, and for each $s \in \mathcal{S}$ and every MPLS-feasible rate vector r' in which $r'_s > r_s$, there exists some LSP $t \in \mathcal{S}$ such that $\frac{r_s - RR_s}{w_s} \geq \frac{r_t - RR_t}{w_t}$ and $r_t > r'_t$.*

Definition 4 (WPMM-bottleneck link of a LSP). *Given an MPLS-feasible rate vector r, a link $l \in \mathcal{L}$ is a WPMM-bottleneck link with respect to r for a LSP s traversing l if $F_l = C_l$ and $\frac{r_s - RR_s}{w_s} \geq \frac{r_t - RR_t}{w_t}$ for all LSP t traversing link l.*

The following proposition links the relationship between the above WPMM policy and the WPMM-bottleneck link definitions.

Proposition 1 (WPMM vector). *An MPLS-feasible rate vector r is WPMM if and only if each LSP has either a WPMM bottleneck link with respect to r or a rate assignment equal to its MR.*

The centralized Water Filling algorithm computes the fair rate for each LSP according to this policy [4]. This centralized algorithm for the WPMM rate allocation requires global information which is not scalable to flood. It is intended to be used as the network bandwidth sharing optimality criterion for our distributed algorithm, which will be presented in the next section.

3 Proposed distributed WPMM algorithm

In this section, we propose an algorithm that converges to the WPMM policy quickly through distributed and asynchronous iterations.

3.1 Basic algorithm

Our distributed solution uses the RSVP signalling protocol to convey information through the network.

The PATH and RESV packets contain the following parameters :

- RR (Reserved Rate [2]) : provided at the creation of the LSP
- W (Weight [2]) : provided at the creation of the LSP
- ER (Expected Fair Rate) : the fair rate that the network allows for this LSP
- BN (BottleNeck) : id of the LSP's bottleneck link

Periodically, the ingress sends a PATH packet. Each router in the path computes a local fair share for the LSP and updates the ER and BN fields if its local fair rate is less than the fair rate present in the PATH packet. Upon receiving a PATH packet, the egress router sends a RESV packet, initialized with the data obtained in the PATH, to the ingress router.

In the backward path, each router updates its information with the RESV parameters (ER,BN). Upon receiving a RESV packet, the ingress obtains the information about its allowed fair share.

Many prior efforts in ATM networks have been done on the design of ABR algorithms to achieve the classical max-min ([5] [8]). The work of Charny [6] was one of the few algorithms that were proven to converge to the max-min in the ATM context. Hou extends Charny's technique to support the minimum rate, peak rate and weight for each flow. Our proposition uses Hou's work but improves the performance by modifying the update mechanism. In Hou's algorithm, the routers update their information in the forward path. So, they cannot have the information computed by the downstream routers. These routers have the correct information in the next cycle. In our approach, we update the router in the backward path, so they have the correct information like the ingress node. Moreover, we add a new parameter (BN) that conveys explicitly the bottleneck link in the path. This information improves considerably the convergence time.

The following are the link parameters and variables used by the algorithm.

- $\mathcal{L}$: Set of links
- $\mathcal{S}_l$: Set of LSPs traversing link l
- FR_l^i : fair rate value of the LSP $i \in \mathcal{S}_l$ as known at link l

[2] This information is only useful during the establishment of the LSP or when these values are modified. To reduce the size of RSVP packet, we can remove these parameters from the refresh RSVP packets.

- BN_l^i : bottleneck link id of the LSP $i \in \mathcal{S}_l$ as known at link l (refer to Definition 4)
- $\mathcal{B}_l$: Set of LSPs bottlenecked at link l, i.e. $\mathcal{B}_l = \{i \mid i \in \mathcal{S}_l \text{ and } BN_l^i = l\}$
- $\mathcal{U}_l$: Set of LSPs not bottlenecked at link l, i.e. $\mathcal{U}_l = \{i \mid i \in \mathcal{S}_l \text{ and } BN_l^i \neq l\}$
- We have $\mathcal{U}_l \cup \mathcal{B}_l = \mathcal{S}_l$

The fair rate of a LSP i is composed of the reserved rate of the LSP (RR^i) and an additional fair share allocated by the network. This additional fair share is proportional to the weight of the LSP. On a particular link, we can compute a value φ_l that gives the additional fair share per unit of weight for the LSPs bottlenecked on this link. Algorithm2 computes φ_l :

Algorithm2 : Calculation of φ_l (if $\mathcal{S}_l \neq \emptyset$[3])

$$\mathrm{FB}_l := C_l - \sum_{i \in \mathcal{S}_l} RR^i - \sum_{i \in \mathcal{U}_l} (FR_l^i - RR^i)$$

$$\varphi_l := \left\{ \begin{array}{ll} \frac{\mathrm{FB}_l}{\sum_{i \in \mathcal{B}_l} W^i} & \text{if} \mathcal{B}_l \neq \emptyset \\ \frac{(C_l - \sum_{i \in \mathcal{S}_l} FR_l^i)}{\sum_{i \in \mathcal{S}_l} W^i} + \max_{i \in \mathcal{S}_l} \frac{(FR_l^i - RR^i)}{W^i} & \text{otherwise} \end{array} \right\} \quad (1)$$

The basic case (i.e. $\mathcal{B}_l \neq \emptyset$) occurs when some LSPs are bottlenecked on link l, we compute the free bandwidth on link l (FB_l) by taking the capacity of the link minus the RR of all LSPs traversing l, minus the additional fair share of the LSPs not bottlenecked on l. φ_l is equal to the free bandwidth divided by the sum of the LSP's weights for each LSP bottlenecked on l, i.e. φ_l is the free bandwidth we will allocate to each $i \in \mathcal{B}_l$ per unit of weight.

The second case (i.e. $\mathcal{B}_l = \emptyset$) occurs when all LSPs are bottlenecked at another link than l. In this case, the value of φ_l is chosen as in [7] and [6] to achieve convergence.

The fair rate (FR_l^i) of a LSP i bottlenecked on link l is computed by :

$$(\forall i \in \mathcal{B}_l) \quad FR_l^i := RR^i + \varphi_l * W^i$$

A key element of the algorithm is the strategy to set the BN_l^i information correctly. We use the following definition of "bottleneck consistency" :

Definition 5 (Bottleneck-consistent). *Let $\mathcal{U}_l$ be the set of LSPs not bottlenecked at link $l \in \mathcal{L}$ and φ_l be calculated according to Algorithm2. $\mathcal{U}_l$ is bottleneck-consistent if*

$$(\forall i \in \mathcal{U}_l) \quad FR_l^i \leq RR^i + \varphi_l * W^i$$

This definition derives directly from Definition 4 and means that all LSPs not bottlenecked at a link must have a bottleneck elsewhere or reach their maximal

[3] If there is no LSP using link l (i.e. $\mathcal{S}_l \neq \emptyset$), $\varphi_l := \infty$

rate, so that they have an allocated fair rate less that the one proposed by the current link. If that were not the case, some LSPs in $\mathcal{U}_l$ would have to be moved to $\mathcal{B}_l$ (i.e. would be bottlenecked at l).

Our algorithm employs per-LSP accounting at each output port of a node. That is, we maintain a table to keep track of the state information of each traversing LSP. For each LSP i, we keep $FR_l^i, RR^i, W^i, BN_l^i$. Based on this state information, we compute the explicit rate for each LSP to achieve the WPMM rate allocation.

The following is the node algorithm, with each output port link initialized with $\mathcal{S}_l := \emptyset$ and $\varphi_l := \infty$.

Algorithm3 : Node behaviour

```
Upon receipt of a PATH[4] {
   LSPCreationAndTermination();
   updateER();
}
```

```
Upon the receipt of a RESV[4] {
   FR_l^i := ER; BN_l^i := BN;
   updateφ_l();
   Forward RESV(i,RR,ER,W,BN)
}
```

```
LSPCreationAndTermination() {
    if LSP termination then {
        S_l := S_l − {i};
        updateφ_l();
    } else if LSP creation then {
        B_l := B_l ∪ {i};
        RR^i := RR; W^i := W;
        updateφ_l();
    } }
```

```
updateER() {
    NER := φ_l * W^i + RR^i
    ER' := max(min(ER, NER), RR^i);
    if (ER' < ER) then {
        BN := l;
        ER := ER';
    }
    Forward PATH(i,RR,W,ER,BN);
}
```

```
updateφ_l() {
    use Algorithm2 to calculate φ_l;
    if (U_l ≠ ∅) then {
        repeat {
            // stops when bottleneck-consistency (see Def5) is achieved
            φ_l^1 := φ_l;
            p := argmax_{i ∈ U_l}(FR_l^i − RR^i)/W^i);
            if (((FR_l^p − RR^p)/W^p) > φ_l^1) then {
                Move p from U_l to B_l;
                use Algorithm2 to calculate φ_l;
            }
        } until ((((FR_l^p − RR^p)/W^p) ≤ φ_l^1) or (U_l = ∅));
    } }
```

The update$\varphi_l()$ procedure will first recompute φ_l based on current values of FR_l^i and BN_l^i for all i. The next step, i.e. the repeat-until loop, will ensure

[4] PATH and RESV packets contains the following parameters (i,RR,W,ER,BN)

that, when it terminates, the set $\mathcal{U}_l$ is bottleneck consistent. To do so, LSPs not satisfying $(FR_l^i - RR^i)/W^i) \leq \varphi_l$ should be moved from $\mathcal{U}_l$ to $\mathcal{B}_l$. At each iteration, one LSP, say p, such that $(FR_l^p - RR^p)/W^p) = \max_{i \in \mathcal{U}_l}(FR_l^i - RR^i)/W^i)$ is removed. The new $\mathcal{B}_l$ and $\mathcal{U}_l$ sets may lead to an new φ_l, which needs to be recalculated. This process ends when either $\mathcal{U}_l$ is empty or $\mathcal{U}_l$ is bottleneck consistent.

Finally, the edge behaviour is simple. The ingress is responsible for sending PATH packets and for updating the LSP fair rate information upon the reception of a RESV packet. In our model, the upstream router of a link computes the fair rate associated with that link. The ingress thus computes the fair rate of the first link and so uses the node algorithm defined in Algorithm3.

The Egress behaviour is simple, upon receiving a PATH packet, the egress sends a RESV packet, initialized with the data obtained in the PATH, to the upstream node.

The structure of the Algorithm3 guarantees that for every LSP $i \in \mathcal{S}$, the fair rate (FR_l^i) information in each node is MPLS-feasible, i.e. $RR^i \leq FR_l^i \leq MR^i$.

With our update architecture in the backward path and the explicit bottleneck link information, we decrease the convergence time substantially as shown in the next section.

3.2 Improve algorithm to deal with the RSVP refresh mechanism

The RSVP protocol comes with an optimized mechanism to minimize the processing time of the PATH and RESV packets. If two successive PATH (or RESV) packets are the same, the upstream node only sends a refresh PATH. The downstream node refreshes the LSP entry but doesn't process the whole PATH packet.

Our solution can easily be extended to keep this property. We must develop a strategy to determine if a node must send a full new PATH or just a refresh PATH. On each output port, we associate with each LSP i a special bit (NR^i) that is set if we must send a full new PATH packet. When some value of an LSP changes in the output port table, we set this bit for all the LSPs of this table. When we receive a PATH and we refresh the LSP entry, we clear this bit. (for more information see [3])

When the system is stable, only refresh RSVP packets are used. With this improved algorithm, we can keep the advantages of the RSVP refresh mechanism.

The use of RSVP and its refresh mechanism reduce the overhead needed compared to ATM RM cells. In ATM, one RM cell is send every 32 cells. Therefore, the overhead introduced by ATM depends on the flow rate. On the other hand, our scheme introduces a fixed overhead. Finally, the use of the refresh mechanism reduces the processing needed by the core routers that only need to refresh the LSP entry in their tables.

4 Simulation results

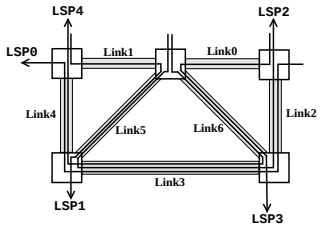
To compare our solution with Hou's, we have developed a specific prototype to simulate the two algorithms. We have implemented Hou's solution adapted to the MPLS context (using RSVP mechanism in place of RM cells).

Our simulation process consist of generating network topology and LSPs on this topology. Next, we add the LSPs one by one and we execute the two algorithms on this topology.

For simulating the RSVP process, we use the concept of iteration and RSVP-cycle. A RSVP-cycle consists of the forwarding of a PATH packet along all the nodes in the path from the ingress to the egress node and the backwarding of a RESV packet along all nodes from the egress to the ingress node. An iteration is the execution of an RSVP-cycle for all the LSPs. At the end of each iteration, we have a vector of LSP fair rates.

We have two possibilities to stop the process. The first is to execute the iterative process until the mean relative error between the last rate vector and the WPMM allocation vector (computed a priori using Water Filling algorithm) is under a fixed precision (e.g 10^{-4}). Another possibility is to stop the process if the mean relative error between two successive rate vectors is under a fixed precision. The first possibility shows us how our solution reaches the optimum and with which convergence speed. The second approach shows us how our algorithm can be used in practice without any a priori knowledge of the WPMM rate allocation. The simulation in the sequel are based on the first approach.

The simple network configuration we use in our simulation is organized like the olympic symbol. There are five MPLS routers connected with 7 links of 100Mb/s and 5 LSPs traversing the topology Fig reftopo2a.



(a)

Id	RR	MR	Weight	FR	BN
0	10	100	10	44.44	3
1	20	100	20	40	6
2	30	100	30	60	6
3	40	100	40	44.44	5
4	50	100	50	55.56	5

(b)

Fig. 1. *(a) Network topology (b) LSP configuration*

Fig. 1b lists the LSP parameters : RR requirement, MR constraint, weight and fair rate allocation for each LSP traversing the network configuration. Fig 2a shows the evolution of the fair rates under Hou's distributed algorithm. The rates converge to their optimal WPMM rate listed in Fig 1b. The algorithm takes 41 iterations before getting sufficiently close to the optimal fair share (i.e. euclidean distance under 0.01 %). Fig 2b shows the evolution of the fair rates under our proposed distributed algorithm. Our solution reaches the optimum in 33 iterations.

Extensive simulation We made extensive simulations on a large number of complex topologies. We created 63 topologies with 20 to 100 nodes and with 20, 50, 100, 200, 300 and 1000 LSPs. We executed the two solutions with different levels of precision. With a precision of 10^{-4}, our solution is in general 2.86 times

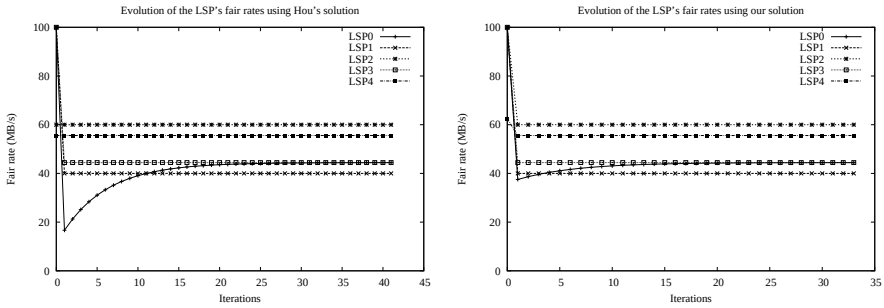


Fig. 2. *(a) Hou's solution (b) our solution on the complex network topology*

faster than Hou's solution. Table 1 presents a short report of the simulation results.

Precision	Hou's algorithm	Our algorithm	Gain
0.01	5.46	2.05	2.67
0.001	18.40	6.81	2.70
0.0001	40.43	14.11	2.86
0.00001	66.86	18.98	3.52

Table 1. *Average number of iterations on 63 topologies*

The number of iterations needed by our solution on large topologies is relatively high but if we look at the number of LSPs that reach their fair share, we see that after a few iterations 90% of the LSPs have reached them. So only a few LSPs continue to change. The improved solution adapted to RSVP becomes very useful and will improve hugely the performance and scalability of the algorithm.

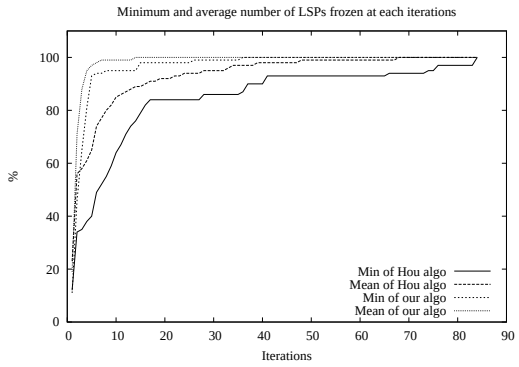


Fig. 3. *Minimum and average LSPs frozen at each iterations*

We also executed the simulations on 50 topologies with 100 to 1000 LSPs and we have plotted the minimum and average percentage of LSPs that have reach their fair share after each iteration for the two algorithms (Fig 3. The average and the minimum are cumulative values and so produce monotonically increasing functions. Normally, at the last iterations, the min and the average values reach 100 %. Hou's solution takes 84 iterations to converge in the worst case and our

solution takes only 36 iterations. We can also see that Hou's solution takes 16 iterations to stabilize 90 % of the LSPs while our solution takes only 4.

5 Conclusion

We presented a novel distributed algorithm to achieve the WPMM rate allocations. This solution provides a scalable architecture to share the available bandwidth among all the LSPs according to their weights. This policy offers an attractive pricing model to the network service providers wishing to introduce priority options to users in a usage-based pricing model.

Our algorithm improves considerably the performance by using a new update plan of the control packet and with the integration of an explicit bottleneck link marking strategy. The utilization of the RSVP protocol to refresh periodically the core node information decreases the overhead present in the ATM-ABR architecture (for more information see [3]).

Our future work will focus on the development of a rigourous convergence proof as well as executing our algorithm on even more complex topologies. Another challenging issue is to improve the weight concept by using utility functions to describe the traffic. The objectives become to maximize the global utility of the clients. And finally, we will investigate the integration of this algorithm in a real Diffserv MPLS network.

Acknowledgments This work was supported by the European project ATRIUM (IST-1999-20675) and the belgian regional project TOTEM

References

1. ATM Forum Technical Commitee. *Traffic Management Specification - Version 4.0*, February 1996. ATM Forum/95-0013R13.
2. D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow. *RSVP-TE : Extensions to RSVP for LSP tunnels*, December 2001. RFC 3209.
3. S. Balon, F. Blanchy, O. Delcourt, G. Leduc, L. Mélon, and F. Skivée. Synthesis and recommendation for intra-domain traffic engineering. Technical Report ATRIUM Deliverable D3.3, University of Liège, Novembre 2003.
4. D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, 1992.
5. Y. Chang, N. Golmie, and D. Siu. *A Rate-Based Flow Control Switch Design for ABR Service in an ATM Network*, August 1995. Twelfth International Conference on Computer Communication ICCC'95.
6. A. Charny. An algorithm for rate allocation in a packet-switching network with feedback. Technical Report MIT/LCS/TR-601, 1994.
7. Y. Thomas Hou, Henry Tzeng, Shivendra S. Panwar, and Vijay P. Kumar. A generic weight-proportional bandwidth sharing policy for atm abr service. *Performance Evaluation, vol 38*, pages 21–44, 1999.
8. R. Jain, S. Kalyanaraman, R. Goyal, S. Fahmy, and R. Viswanathan. *ERICA switch algorithm: A complete description*, August 1996. ATM Forum/96-1172.
9. J. Y. Le Boudec and B. Radunovic. A unified framework for max-min and min-max fairness with applications. October 2002.
10. E. Rosen and al. *Multiprotocol label switching architecture*, January 2001. RFC 3031.