Path Computation Algorithms in NS2

Davide Adami
CNIT Research Unit
Dept. of Information Engineering
University of Pisa (ITALY)
davide.adami@cnit.it

Christian Callegari
Dept. of Information Engineering
University of Pisa (ITALY)
christian.callegari@iet.unipi.it

Stefano Giordano
Dept. of Information Engineering
University of Pisa (ITALY)
s.giordano@iet.unipi.it

Michele Pagano
Dept. of Information Engineering
University of Pisa (ITALY)
m.pagano@iet.unipi.it

ABSTRACT
Originally designed to improve the efficiency of packets forwarding, MPLS provides support for Traffic Engineering and network resilience. Constrained-based path computation is a key building block for Traffic Engineering in MPLS networks, since it allows to set-up LSPs along paths that satisfy QoS constraints. This paper deals with two distinct categories of path computation algorithms: on-line path computation algorithms adn on-line multi path algorithms. All these algorithms have been implemented in NS2 as an extension of OSPF-TE\ns and integrated with RSVP-TE\ns.

Keywords
MPLS-TE, NS2, Path computation, Wang-Crowcroft, CSPF, DBCTE, EBMP, MPBF

1. INTRODUCTION
In the last years, new advanced network architectures have been introduced to satisfy the Quality of Service (QoS) requirements of multimedia applications. In the meantime, Multiprotocol Label Switching (MPLS) [1], originally developed as a fast packet forwarding technique, has been deployed by a great number of Internet Service Providers (ISPs) to exploit, within their network infrastructures, its Traffic Engineering (MPLS-TE) capabilities [2][3].

One of the most important "building blocks" of such network architecture is the Path Computation Element (PCE) [4], which enables the creation of constraint routed Label Switched Paths (LSPs) along explicit paths. A PCE can be realized according to either a centralized or a distributed computation model. The former refers to a model whereby all paths in a domain are computed by a single, centralized PCE. This may be a dedicated server or a designated router in the network. Conversely, the latter refers to a domain or network that may include multiple PCEs, and where the computation of paths is shared among the PCEs. A given path may in turn be computed by a single PCE ("single PCE path computation") or by multiple PCEs ("multiple PCE path computation"), but, often, the computation of an individual path is entirely performed by a single PCE. For example, this is usually the case within a single IGP area of an MPLS-TE network, where the ingress LSR node is responsible for computing the path or for contacting the PCE. A centralized PCE can take into account global information on network resources and existing connection paths to implement optimal computation procedures. In the centralized scenario, off-line path computation algorithms, which have the knowledge of the entire set of demands and therefore make more efficient use of network capacity, are typically used. Since a single centralized PCE may be a performance bottleneck, a distributed PCE is sometimes preferred. Moreover, the ability to perform distributed path computation significantly increases network resilience. Distributed PCEs usually adopt on-line path computation algorithms. Therefore, the path is computed by the ingress LSR which determines a single path to the egress LSR taking into account both links metrics and administrative constraints.

As powerful and flexible simulation tools may help network engineers to design MPLS-TE networks, in the last years, the authors of this paper have developed a new simulation environment, based on NS2 [5], which enhances an MPLS node with the control plane capabilities necessary to set-up LSPs with QoS requirements [6][7][8].

In this framework, the choice among different path computation algorithms and models is a really difficult issue to address, because the efficiency of a path computation scheme has to be evaluated with respect to its ability to satisfy a set of LSPs allocation requests as well as to optimize the utilization of network resources.

More specifically, the focus of this paper is on a performance comparison among two distinct classes of path computation algorithms: on-line path computation algorithms and multi path on-line algorithms, that may be used when a single path can not satisfy the bandwidth requirements The paper is organized as

Conference name: SIMUTools, March 03 – 07, 2008, Marseille, France.

ISBN 978-963-9799-20-2

follows. Sections 2 and 3 discuss the different path computation algorithms we have taken into account in this work: respectively distributed path computation algorithm and distributed multi path algorithms. Section 4 provides a short overview about the implementation of those algorithms in NS2, while section 5 presents the results of the simulations carried out to validate the proposed algorithms and to compare their performance. Finally, section 7 concludes the paper with some final remarks.

2. DISTRIBUTED PATH COMPUTATION ALGORITHMS

This section provides a short overview concerning the three algorithm for distributed path computation that we have implemented in NS2.

2.1 Constrained Shortest Path First

Constrained Shortest Path First (CSPF) is the "default" path computation algorithm used by OSPF-TE [8]. Every link *(i,j)* is characterized by two attributes: *bij* (residual bandwidth) and *cij* (cost). The algorithm aims at finding a path with bandwidth $B \geq BMIN$ and minimum cost, taking also into account the administrative constraints (link colours).

CSPF executes the following steps:

1. set $cij = \infty$ if $bij < BMIN$ (or whose colour is different from the selected colour)
2. compute the path P with the minimum cost (applying the Dijkstra algorithm).

2.2 Wang Crowcroft

Unlike CSPF, the Wang-Crowcroft (WC) path computation algorithm [9] aims at finding a path which satisfies multiple QoS constraints, given in terms of bandwidth (BMIN) and delay (DMAX).

Every link *(i,j)* is characterized by two metrics: *bij* (residual bandwidth) and *dij* (propagation delay). Unlike the original version of the algorithm, which takes into account only the propagation delay, our implementation considers also the queuing and transmission delays (see [10] for further details).

The algorithm consists of the following steps:

1. set $dij = \infty$ if $bij < BMIN$
2. compute the path P with the minimum delay D^* (applying the Dijkstra algorithm)
3. compare D^* with $DMAX$. If $D^* < DMAX$ select the path, otherwise the request is rejected.

2.3 Delay and Bandwidth Constrained with TE objectives

The Delay and Bandwidth Constrained with TE objectives (DBCTE) is a new path computation algorithm, which extends the WC algorithm with load balancing capabilities. To support such functionalities, a new metric, BTE, is introduced.

For each link, *BTE* is defined as the ratio between the link bandwidth (*Bij*) and the link residual bandwidth:

$$BTE(i, j) = \frac{Bij}{bij}$$

Hence, BTE is a quantity that grows when the residual bandwidth decreases and is always greater or equal to one.

The algorithm consists of the following steps:

1. set $dij = \infty$ if $bij < BMIN$
2. compute the path P with the minimum BTE (applying the Dijkstra algorithm)
3. calculate the delay D^* of P
4. compare the delay D^* with $DMAX$. *If* $D^* < DMAX$ select the path, otherwise use the WC algorithm.

It is relevant to highlight that whereas CSPF takes into account only one constraint, WC and DBCTE consider both bandwidth and delay constraints.

3. DISTRIBUTED MULTI PATH ALGORITHMS

When a LSP can not be established because the bandwidth constraint (required bandwidth *BMIN*) can not be satisfied along any path, a solution can be represented by the use of a multi path algorithm. In this case the traffic flow is split in several distinct traffic flows, which are then forwarded along distinct LSPs.

In the following we will only consider the possibility of splitting each flow in a maximum of two distinct flows. This assumption is justified by the need of maintaining a low volume of signaling traffic over the network when a LSP establishment request has to be satisfied.

Thus, the problem becomes to find two distinct path P_1 and P_2 with available bandwidth B_1 and B_2 , such that $B_1 + B_2 = B_{min}$.

In this section we present the two multi path algorithms, that have been implemented in the simulator. Before providing their description, in the next subsection we focus on the discussion of a problem that arises when this kind of algorithm is adopted in a network.

3.1 Buffer Dimensioning

When a multi path algorithm is used, instead of a "simple" path computation algorithm, one additional problem should be taken into account. The use of two distinct paths to forward a single flow usually takes to a situation where one portion of flow (forwarded along P_1) experiments a delay D_1, while the other portion experiments a delay D_2. As it appears clear the difference between the two delays should be such that packet reordering at destination is possible without any packet loss. This leads to the need of a correct dimensioning of the destination node buffer.

To solve this problem we have considered the network delay difference to dimension the buffer, according to the Network Calculus theory.

The delay associated to a LSP is computed taking into account all the three components of the delay [10], thus:

$$D = Dpropagation + Dtransmission + Dqueueing\_max$$

which represents a worst case upper bound for the delay experienced along a LSP.

Moreover we also have considered a lower bound, which is the case in which the packets do not experience any queuing delay:

$$D^* = Dpropagation + Dtransmission$$

Thus, the network delay variation is given by

$$\Delta = max\ (|D_1 - D_2^*|, |D_2 - D_1^*|)$$

At this point, considering a traffic flow, with cumulative function $R(t)$, we can say that, to avoid packet loss, at the destination node, the dimension of its buffer should be at least $R(2\Delta)$. In particular for a Constant Bit Rate (CBR) flow, with cumulative function $R(T)=rt$, the buffer size should be at least $2\Delta r$.

3.2 Modified Equal Bandwidth Multi Path

In this subsection we present a modified version of a classical multi path routing algorithm, the Equal Bandwidth Multi Path (EBMP) [11], that has been implemented in the simulator.

The version we have implemented differs from the original one, because the maximum number of paths that can be used by a single traffic flow is equal to two, while in the original EBMP a single flow can be forwarded on an arbitrary number of paths. Another difference is that in the original version bandwidth should be divided equally among the different LSPs, while in our version we split the bandwidth in different portion between the two LSPs.

In this case each network link is described by three metrics: *bij* (residual bandwidth), *dij* (propagation delay), and *Uij* (link utilization). This last metric is computed as:

$$Uij = \frac{Ballocated + Brequest}{Bij}$$

The algorithm imposes that the link utilization should not exceed a threshold *UMAX*, set by the network administrator.

The algorithm consists of the following steps:

1. set $dij = \infty$ if $Uij > UMAX$
2. set $dij = \infty$ if $Bij < BMIN$
3. compute the path P with the minimum delay D^* (applying the Dijkstra algorithm)
4. if $D^* < DMAX$ select the path
5. else for i=2;10
    a. set $BMIN\_1 = BMIN/i$ and $BMIN\_2 = 1-BMIN/i$
    b. set $dij = \infty$ if $Uij < BMIN\_2$
    c. compute the path $P2$ with the minimum delay $D2^*$ (applying the Dijkstra algorithm)
    d. set $dij = \infty$ if $Uij < BMIN\_1$
    e. compute the path $P1$ with the minimum delay $D1^*$ (applying the Dijkstra algorithm)
    f. if $D1^* < DMAX$ and $D2^* < DMAX$ select the paths
6. end;

The algorithm ends when the for cycle ends without any result (the request can not be satisfied) or when the paths are selected.

To be noted that we have decided to stop the cycle after nine iterations so as to avoid an excessive execution time.

3.3 Modified Maximum Path Bandwidth First

The maximum path bandwidth first (MPBF) [11] algorithm is a multi path algorithm, which aims at maximizing the resource utilization in a network. As in the previous case we have implemented a modified version of the classical algorithm, so that the maximum number of paths, that can be used by a single traffic flow, is equal to two.

Each network link is described by two metrics: *bij* (residual bandwidth) and *dij* (propagation delay).

The algorithm consists of the following steps:

1. compute the path P with the minimum delay D^* (applying the Dijkstra algorithm)
2. compute the available path bandwidth B^*
3. if $D^* < DMAX$ and $B^* > BMIN$ select the path
4. else if $D^* < DMAX$
    a. set $BMIN\_2 = BMIN - B^*$
    b. compute the path $P2$ with the minimum delay $D2^*$ (applying the Dijkstra algorithm)
    c. compute the available path bandwidth $B2^*$
    d. if $D2^* < DMAX$ and $B2^* > BMIN\_2$, select the paths P and $P2$
5. else the request can not be satisfied

As it appears clear, differently from the previous algorithm the MPBF aims at allocating the maximum quantity of available resources along the computed path instead of splitting the requested resources in an arbitrary fashion.

4. NS2 SOFTWARE MODULES

In this section, we shortly describe the new features added to the NS2 simulator. The main enhancement concerns the possibility of establishing a constraint routed LSP along the path computed by one of the algorithms described in the previous subsections.

These functionalities are enabled by inserting the following command in the simulation script:

- <Ingress-LER> **create-crlsp** <Algorithm> <Source> <Egress-LER> <SessionID> <FlowID> <TunnelID> <Bandwidth> <MaxDelay> <Buffer> <TTL>

The command is inserted in the simulation script when a distributed path computation approach is adopted. In this case, the ingress node uses the selected <Algorithm> to compute a path (or two paths in the case of multi path algorithm) which satisfies the constraints (*BMIN* = <Bandwidth>, *DMAX* = <MaxDelay>). If a path is found, the nodes list is passed to the RSVP-TE agent,

which inserts it in the Explicit Route Object (ERO) of the Path message. Then, an LSP, with reserved bandwidth <Bandwidth> and identified by the tunnel ID <TID>, is created between the <Ingress-LER> and the <Egress-LER>. If a multi path algorithm has been used the LSPs are established only if the destination node respects the constraint on the buffer dimension (section 3.1).

5. PERFORMANCE COMPARISON

This section describes the simulation tests carried out to validate the considered path computation algorithms and to compare their performance.

A random topology has been generated by means of BRITE, a public domain universal topology generator downloadable from [12]. The considered topology consists of 20 nodes and the generation model for interconnecting the nodes is based on the Waxman's probability model, given by the following formula:

$$P(u, v) = \alpha e^{\frac{-d}{\beta L}}$$

where $P(u, v)$ is the probability that a link between the nodes u and v is created, α $(0 < \alpha \leq 1)$ and β $(0 < \beta \leq 1)$ are Waxman specific parameters, d is the Euclidean distance between the nodes, and L is the maximum distance between any two nodes (in our simulations $\alpha=0.15$, $\beta=0.2$, d and L assume their default values). Moreover, the link propagation delay has been modeled as a random variable uniformly distributed in [1, 30] ms, whereas the link capacity may assume one of the following values: 155 Mbps, 622 Mbps, 2.5 Gbps, and 10 Gbps. The resulting topology is described in table I.

Table 1. Network topology

From node	To node	Bandwidth	Delay
1	9	155 Mbps	10 ms
2	0	155 Mbps	5 ms
3	0	10 Gbps	18 ms
4	1	622 Mbps	21 ms
5	3	155 Mbps	11 ms
6	1	2.5 Gbps	11 ms
7	6	622 Mbps	10 ms
7	2	622 Mbps	8 ms
8	6	622 Mbps	20 ms
9	3	622 Mbps	4 ms
10	5	155 Mbps	4 ms
10	0	622 Mbps	21 ms
11	8	622 Mbps	8 ms
12	11	155 Mbps	6 ms
12	8	622 Mbps	6 ms
13	3	622 Mbps	6 ms
13	4	622 Mbps	6 ms
14	4	622 Mbps	19 ms
14	6	155 Mbps	4 ms
15	4	2.5 Gbps	12 ms
15	11	622 Mbps	25 ms
16	13	622 Mbps	10 ms

The number of LSPs set-up requests has been varied between 3 and 11 and a set of 250 simulations has been carried out for each LSP set-up requests value.

Moreover, in each simulation:

- the egress and the ingress LSRs have been chosen randomly among the network nodes
- for each LSP, the requested bandwidth has been assumed as a random variable uniformly distributed in [5,50] Mbps
- for each LSP, the maximum delay has been assumed as a random variable uniformly distributed in [150, 400] ms

The following parameters have been considered to compare the different algorithms:

- mean number of allocated LSPs
- success probability, defined as the probability that there are enough available resources to accept a new LSP set-up request
- resource utilization, defined as the ratio of the allocated bandwidth and the total link bandwidth
- execution time, defined as the time necessary for an algorithm to compute all the requested paths
- bandwidth rejection ratio, defined as the ratio of the overall bandwidth of the rejected LSP set-up requests and the total required bandwidth

Figures 1 to 7 show the results of such comparison, to be noted that, in fact, EBMP and MBPF respectively refer to the modified version of the two algorithms, we have implemented in the simulator.

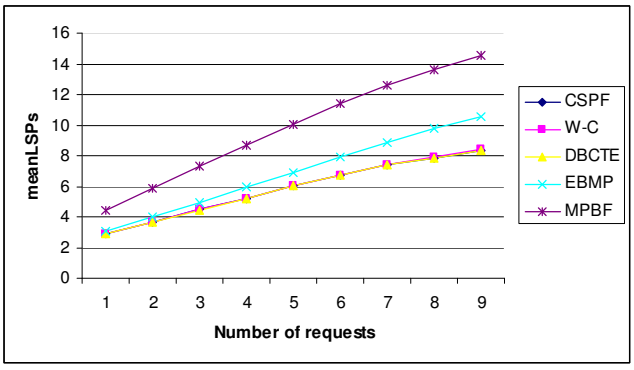

Figura 1: Mean number of allocated LSP

It's worth noticing that, as expected, the overall performance are improved in the case of multi path algorithms, with respect to "simple" path computation algorithms. Indeed we achieve

improvements of at least 10% in all the parameters used for the performance comparison.

In more detail, concerning the success probability and the bandwidth rejection ratio, with the multi path algorithms, we respectively achieve about 10% and about 20% of improvements.

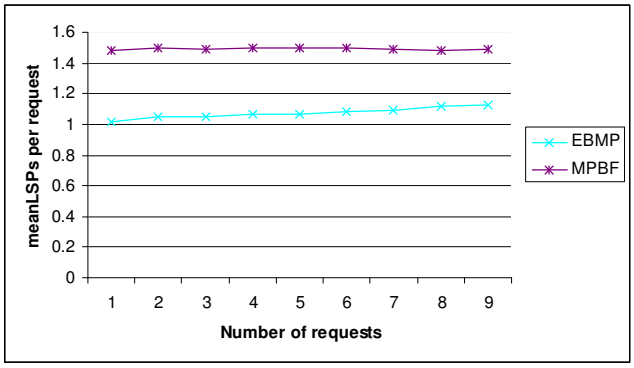

Figura 2: Mean number of allocated LSPs per request (Multi Path case)

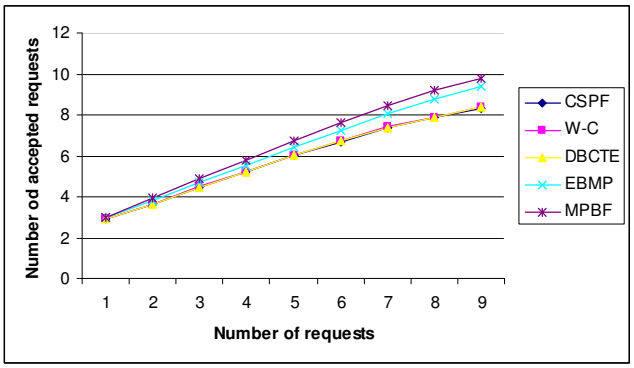

Figura 3: Mean number of accepted requests

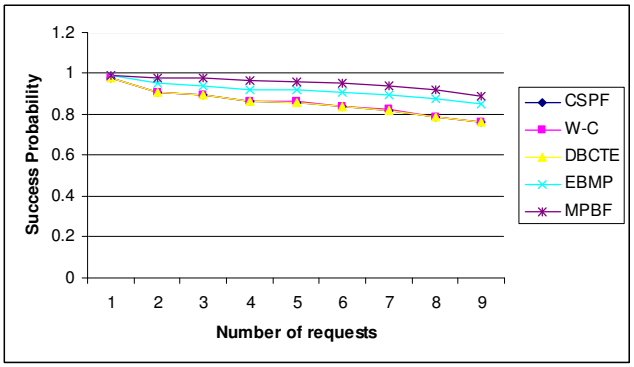

Figura 4: Success probability

To be noted that the use of a multi path approach leads to some apparently ambiguous results. In figure 1, we can see that the mean number of established LSPs is, in some cases, bigger than the number of LSP requests (figure 1) , despite of a success probability which is much lower than one (see figure 4).

These behaviors are justified by the fact that for each required LSP one or two LSPs are allocated so as to satisfy the required constraints, as shown in figure 2, where the mean number of allocated LSP per request is shown.

Hence to produce a more fair comparison we can consider figure 3, where we show the mean number of accepted requests.

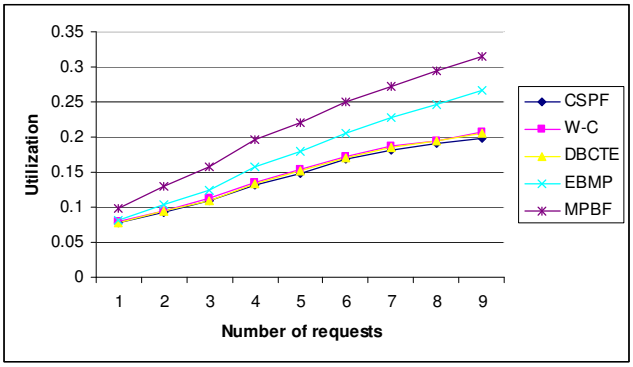

Figura 5: Resource utilization

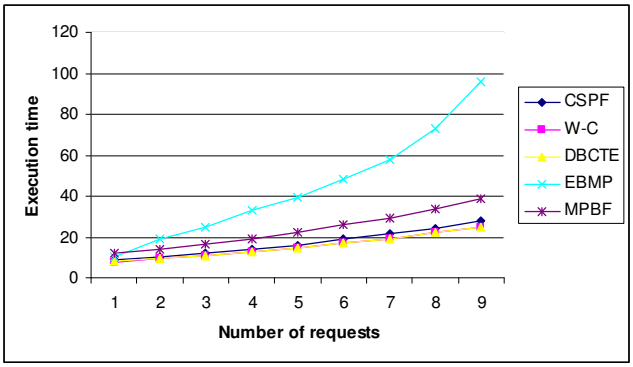

Figura 6: Execution time

Moreover, regarding the execution time of each algorithm, we can notice that, in general, multi-path algorithms have higher computational times, even though BMPF has an execution time which is much lower than EBMP and is comparable with the three "simple" distributed path computation algorithms.

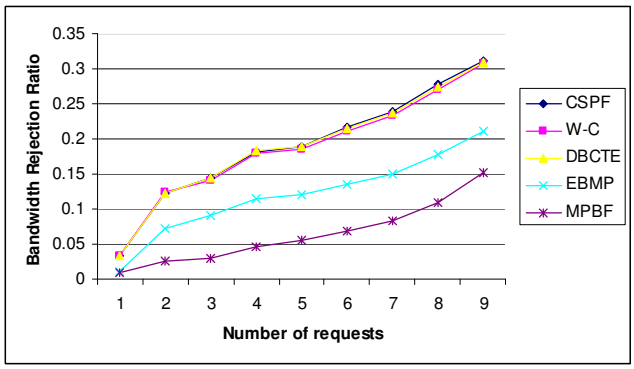

Figura 7: Bandwidth rejection ratio (Multi Path case)

It is worth noticing that the three single path computation algorithms almost achieve the same performance.

Regarding a comparison between the two distinct multi path routing algorithms, we can easily verify that the use of the MBPF algorithms leads to better performance. Indeed it achieves higher values for success probability (about 5% more), and resource utilization (about 5% more). These results are justified by the fact that the BMPF tends to maximize the resource utilization allocating all the available resources along the optimum path *P,* while the EBMP, arbitrarily splitting the required resources on the two paths, tends to waste resources along the path *P1* (see section 3.2 and 3.3).

6. CONCLUSIONS

Constraint-based path computation is a key function in MPLS and GMPLS networks. Several algorithms have been proposed in literature to satisfy the QoS requirements of LSPs allocation requests based on traffic engineering strategies.

The paper describes several path computation algorithms: "simple" distributed path computation algorithms and multi path distributed path computation algorithms.

Such algorithms have been implemented as new modules for the Network Simulator (NS2), so as to provide a new powerful and flexible simulation tool to help in their work network designers and administrators.

Finally, the paper reports the results of the simulations, performed in a network with randomly generated topology, to validate and compare the effectiveness of the described algorithms.

7. ACKNOWLEDGMENTS

The authors would like to thank Aldo Bizzarri for his work in support of the development activities related to the simulator.

8. REFERENCES

[1] E. Rosen et al., *Multiprotocol Label Switching Architecture,* IETF *RFC 3031,* January 2001

[2] Awduche, D., et al.: *Requirements for Traffic Engineering over MPLS*. IETF RFC 2702, September 1999

[3] W. Lai, *Traffic Engineering for MPLS,* Internet Performance and Control of Network Systems III Conference, Boston, Massachusetts, USA, July 2002

[4] A. Farrell, J.P. Vasseur, j. Ash *A Path Computation Element (PCE)-Based Architecture,* IETF RFC 4655, August 2006

[5] The Network Simulator vers.2.26 (NS2) Home Page www.isi.edu/nsnam/dist/

[6] The RSVP-TE Network Simulator Home Page: http://netgroup-serv.iet.unipi.it/rsvp-te_ns/

[7] The OSPF-TE Network Simulator Home Page: http://netgroup-serv.iet.unipi.it/ospf-te_ns/

[8] D. Adami, C. Callegari, D. Ceccarelli, S. Giordano, M. Pagano, *Design and implementation of the OSPF-TE Network Simulator*. IPS-MoMe 2006 Proceedings

[9] Z. Wang and J. Crowcroft, *Quality of Service Routing for Supporting Multimedia Applications,* IEEE Journal on Selected Areas in Communication, September 1996

[10] Adami, D., Callegari, C.. Giordano, S., and Pagano, M., *A New Path Computation Algorithm and its Implementation in NS2,* IEEE International Conference on Communications (ICC 2007)

[11] Lee, J., Kim, B., *Multi-Path Constraint-based Routing Algorithms for MPLS Traffic Engineering.* IEEE Network 2003

[12] BRITE, available at http://www.cs.bu.edu/brite/