IntelliDyLBA: A Load Balance Scheme for MPLS Networks with Genetic Algorithm Based Auto-Learning and Fuzzy Logic Techniques

Eduardo Guimarães Nobre, Pablo Rocha Ximenes Ponte, Marcial Porto Fernandez, Joaquim Celestino Jr.

***Abstract*— This paper describes IntelliDyLBA, a novel traffic engineering scheme based on a load balance reactive method for congestion control in MPLS networks that makes use of genetic algorithm based learning and fuzzy logic techniques. It is an improvement of the "Fuzzified Dynamic Load Balance Algorithm" (FuDyLBA) with the addition of auto-learning techniques. Computer simulation experiments have shown a better traffic distribution over the links of a network when compared to its predecessor, while showing intelligent adaptation to traffic and structural network changes.**

***Index Terms*— MPLS, Traffic Engineering, Fuzzy Logic, Genetic Algorithms.**

I. Introduction

One of the most important applications of Multi-Protocol Label Switching (MPLS) for IP based networks is Traffic Engineering (TE) [2]. The main goal of TE is to optimize the network's performance by the efficient usage of its resources. MPLS is performed through the Label Switched Paths (LSP) which are paths within the network where labels are accounted for forwarding choices. The most important characteristic of a LSP for TE is the possibility a LSP has of establishing explicit routes within the network.

The main mechanism used by the MPLS framework for traffic engineering is the Constraint-Based Routing (CBR) technique [11]. In this paper, an alternative traffic engineering mechanism that makes use of Fuzzy logic and genectic algorithm based auto-learning techniques is presented. It is the Intelligent Dynamic Load Balance Algorithm (IntelliDyLBA), an improvement of our previous work, the Fuzzified Dynamic Load Balance Algorithm (FuDyLBA) [4]. CBR algorithms are based on a preventive mechanism, whereas IntelliDyLBA only acts when congestion is detected by its fuzzy system, being then a reactive method.

Congestion in an MPLS network can be detected in basically two ways: either a certain link's usage is close to its full capacity, or a new LSP request can not be met due to lack of resources. IntelliDyLBA works in the first situation monitoring the level of utilization of each link in a network through its fuzzy control system. One other eminent characteristic of IntelliDyLBA is its genetic learning capacity which gives the algorithm a strong means of adaptation both independently and intelligently. This adaptation ability solves a common problem found on TE schemes which is directly linked to their inability to adapt properly to network changes (traffic shape, topology, capacity of the links, etc).

This work is organized as follows: in section two, a brief discussion on TE techniques for MPLS networks can be found; in section three, the importance and motivation of our proposal is shown; in section four, we present IntelliDyLBA, the proposed mechanism; in section five we demonstrate some experiments used for the model validation; and finally, in section 6, we conclude briefly with some pointers to future works.

II. Traffic Engineering in MPLS Networks

According to IETF RFC 3272, Traffic Engineering schemes for congestion control can be classified according to their response time scale, their traffic attendance policy (supply or demand); and their congestion response approach (reactive or preventive) [2].

The majority of the proposed TE schemes found in literature are preventive. The two most mentioned ones are CBR and Traffic Splitting [11].

One most cited CBR schemes is the Minimum Interference Routing Algorithm (MIRA). The weak points of MIRA are found in the maximum flow calculation between the ingress and egress nodes (which is used in the "critical" links identification), as well as the unbalanced network utilization [11]. As demonstrated in [12], MIRA is incapable of estimating bottlenecks on links that are "critical" for clusters of nodes. This inability to cope with network dynamics is not

Manuscript received January 21, 2005. This work was supported in part by the following Brazilian Institutions: FUNCAP and CNPq.

Eduardo Guimarães Nobre is with the Communication Networks and Information Security Laboratory (LARCES) in Ceará State Univerty (UECE) Av. Parajana, 1700 – Itaperi – 60.720-020 – Fortaleza – CE – Brazil (e-mail: eduardo@larces.uece.br).

Pablo Rocha Ximenes Ponte is with the Communication Networks and Information Security Laboratory (LARCES) in Ceará State Univerty (UECE) Av. Parajana, 1700 – Itaperi – 60.720-020 – Fortaleza – CE – Brazil (Phone/Fax: +55(85)32992676; e-mail: pablo@larces.uece.br).

Marcial Porto Fernandez is with the Communication Networks and Information Security Laboratory (LARCES) in Ceará State Univerty (UECE) Av. Parajana, 1700 – Itaperi – 60.720-020 – Fortaleza – CE – Brazil (e-mail: marcial@larces.uece.br).

Joaquim Celestino Jr. is with the Communication Networks and Information Security Laboratory (LARCES) in Ceará State Univerty (UECE) Av. Parajana, 1700 – Itaperi – 60.720-020 – Fortaleza – CE – Brazil (e-mail: celestino@larces.uece.br).

only related to MIRA, but can be found in all CBR mechanisms [11].

Few congestion control mechanisms (reactive schemes) can be found in the literature, whereas most of them can be found in [7] and [11]. One of them is the First-Improve Dynamic Load Balance Algorithm (FID) [11], an algorithm that implements an optimized local search. For each link in a network which is considered congested, FID tries to find the first LSP crossing the link that, if re-routed to an alternative path, is capable of augmenting the link's capacity without worsening the network's load balance. FID is an improvement of the Dynamic Load Balance Algorithm (DyLBA) [3], and the main difference lies in the level of the local search.

Another worth mentioning scheme is FuDyLBA [4]. This scheme analyzes each link of a network performing a classification through a fuzzy controller that will indicate whether or not the link is considered congested based on the level of link utilization (input). After matching this input to any of the linguistic values defined in the input membership functions, the fuzzy controller returns a proper unload level (defuzzification). This value will be used to unload this particular link. The unloading procedure performs in way similar to FID's general behavior, but instead of re-routing only a single LSP, it keeps trying to reroute as many LSPs as necessary, until the amount of released load reaches the level indicated by the fuzzy controller.

III. Motivation and Problem Definition

As discussed in [4], the fuzzy controller's membership functions of FuDyLBA are fundamental to the behavior of the algorithm. In fact, well designed membership functions can improve very much the algorithm performance. There is, on the other hand, a key element for the good design of those membership functions which lies in the network dynamics. It is impossible, though, to model a single ideal set of membership functions to cover all possibilities in network characteristics.

This difficulty to comply with the dynamics of a network is not exclusive to fuzzy algorithms, but it unleashes itself as a challenge in all load balance methods for MPLS networks, especially in the preventive approaches.

An efficient way of solving the adaptability problem in load balance algorithms for MPLS networks is the use of genetic algorithms. Such technique can improve a load balance scheme, during its own functioning, in a way that a number of its characteristics will evolve, converging to an optimal behavior representing the learning acquired from the changes of the network.

IV. IntelliDyLBA

A. General Overview of IntelliDyLBA

IntelliDyLBA extends the basic behavior of FuDyLBA [4] through changes in the fuzzy control system and the addition of genetic algotihm based auto learning techniques.

Firtstly, the single fuzzy controller in FuDyLBA was substituted by a set of 12 newly designed fuzzy controllers. Their design was based on an experimental study on the relationship between their individual set of membership functions versus several dynamic network characteristics. Twelve sets (input/output) of membership functions were selected. Each set had the best performance in a given network characteristic. Based on those 12 sets of membership functions, 12 new fuzzy controllers were designed to represent each network scenario, constituting the basis for IntelliDyLBA's fuzzy controlling system.

IntelliDyLBA's load balance rounds are similar to FuDyLBA's. The difference lies in the fact that for each IntelliDyLBA's round, a different fuzzy controller is used. That makes a normal FuDyLBA's round to be expanded into 12 different IntelliDyLBA's load balance rounds, each one using one of the 12 fuzzy controllers mentioned above.

During each load balance round, IntelliDyLBA accounts the change in network's load balance quality. This is done by the calculation of the standard deviation for the residual bandwidth for all the links in the network. Every action of the current fuzzy controller in IntelliDyLBA sets off a change in this standard deviation, which is them stored in the "learning matrix" together with the input and output values for the fuzzy controller the was responsible for that change. After the 12 fuzzy controllers were used, the data collected in the "learning matrix" are them processed. The goal is to approximate IntelliDyLBA's general behavior to the best examples found in the "learning matrix". For that, the twelve fuzzy controllers of IntelliDyLBA's fuzzy controlling system are improved by means of genetic algorithms, using the example set as a standard for the improvement. The learning matrix processing and the genetic algorithm based improvement is done by a special part of IntelliDyLBA called "Genetic Learning Module".

B. The Pseudo Code

The pseudo code of IntelliDyLBA is demonstrated in fig. 4.1. It is possible to identify trhee basic blocks: the first one demonstrates the algorithm's general structure; the second represents the unload function, responsible for IntelliDyLBA's core mechanism; and the third block represents the "Genetic Learning Module".

In the first block of the pseudocode, it is possible to understand the general structure of the algorithm. Let's define its element and notations.

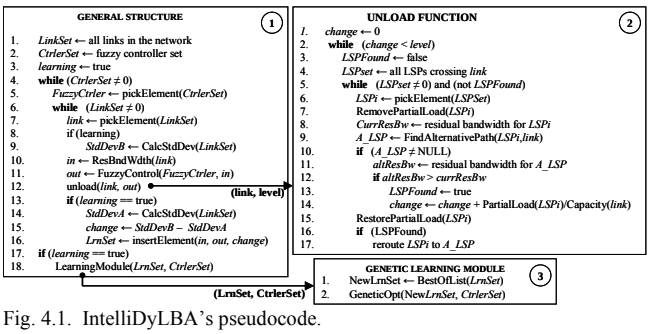

Fig. 4.1. IntelliDyLBA's pseudocode.

The algorithm starts classifying each link in the network using one of the controllers from its fuzzy system. This controller will receive the intensity of residual bandwidth as

input and will return the level in which the link should be unloaded as output. After that, the standard deviation for the residual bandwidth for all the links in the network is calculated. The unload function is called and it will unload the link in examination based on the level received as input. The standard deviation for residual bandwidth for all the links in the network is again calculated. The difference between the standard deviation before and after the unload procedure (variable *change*) is then calculated and stored in the "learning matrix" together with the input (variable *in*) value given to the fuzzy controller and the output (variable out) value returned by it. After analyzing all the links in the network and performing the same control operations with each one of them, the next fuzzy controller is selected, and a new load balance round begins. After all the 12 fuzzy controllers were used to balance the load in the network, the "Genetic Learning Module" is activated.

C. Fuzzy Control System

IntelliDyLBA's fuzzy control system is composed of 12 different fuzzy controllers which are used sequentially in a round robin style. Each controller is used during an entire load balance round, which consists of classifying and unloading every link in a given network. Each controller's input membership functions (fuzzification) represent values for the linguistic variable "residual bandwidth intensity". In the case of the output membership functions (defuzzification) for each controller, the linguistic variable which is represented is "unload intensity".

The controllers are divided into three groups concerning their membership functions. The first group uses a set of three triangular input membership functions and a similar set to represent the output membership functions. The second group

GROUP 1	GROUP 2	GROUP 3
If ResidualBandwidth = *Low* UnloadLevel ← *Intense* *If* ResidualBandwidth = *Average* UnloadLevel ← *Medioano* *If* ResidualBandwidth = *Intense* UnloadLevel ← *Low*	*If* ResidualBandwidth = *Very Low* UnloadLevel ← Very *Intense* *If* ResidualBandwidth = *Low* UnloadLevel ← *Intense* *If* ResidualBandwidth = *Intense* UnloadLevel ← *Low* *If* ResidualBandwidth = *Very Intense* UnloadLevel ← *Very Low*	*If* ResidualBandwidth = *Very Low* UnloadLevel ← Very *Intense* *If* ResidualBandwidth = *Low* UnloadLevel ← *Intense* *If* ResidualBandwidth = *Average* UnloadLevel ← *Average* *If* ResidualBandwidth = *Intense* UnloadLevel ← *Low* *If* ResidualBandwidth = *Very Intense* UnloadLevel ← *Very Low*

Fig. 4.2. Fuzzy controlling system's inference rules.

uses a set of four triangular functions for input and output membership. Finally, the third group uses 5 triangular functions for input and output membership. Each one of these three groups is composed of four different fuzzy controllers. Each one of these controllers has a sparser or more concentrated set of membership functions, so that the distribution of these is the distinguishing element.

It is important to notice that this description of IntelliDyLBA's fuzzy controllers indicates only their initial state, since they will suffer changes in their membership functions during the functioning of the algorithm.

The inference rules for IntelliDyLBA's fuzzy control system follow a general structure and can be found in fig. 4.2.

D. Unload Function

This function aims to unload a certain link indicated by the input variable *link* based on a percentage indicated by the variable level. The pseudocode for the *unload* function is presented in the second block of fig. 4.1. This functions performs in the same way of FuDyLBA's *unload* function which is described in details in [4].

E. Genetic Learning Module

The third block in fig. 4.1 represents the algorithm's "genetic learning module". The genetic algorithm based optimization of the twelve fuzzy controllers takes place in this part of the algorithm.

In this block, two basic steps can be observed. The first one processes the data collected in the "learning matrix" by creating an objective sample set (objective function examples) constituted by the 50 best examples. To choose these best examples, this step bases the selection on the variation of standard deviation (variable *change*), described in subsection 4.2. In the second step, the genetic algorithm based optimization routine is called. This routine will approximate the behavior of the twelve fuzzy controllers to the objective sample set. That means that these 12 controllers, after suffering the optimization, will have input-output responses closer to the 50 examples collected from the "learning matrix". This process will make IntelliDyLBA's fuzzy control system to converge to an optimal behavior that reflects the network's current scenario.

The use of genetic algorithms to optimize fuzzy controllers was demonstrated in [6]. The parameters used for the genetic algorithm based optimization of the twelve fuzzy controllers are the same described in [5].

V. Simulations and Results

A. Overview

This section presents the results of simulations of the behavior of the proposed algorithm with comparisons to its predecessors FuDyLBA [4] and FID [11].

All simulations were carried out under MNSv2 (*MPLS Network Simulator version 2*) [1], an extension of NS-2 (*Network Simulator 2*) [8]. The fuzzy control system and the genetic algorithm based optimization routine were simulated through *JFS Fuzzy System* [9].

B. Experiments

All simulations were based on the three topologies indicated in fig. 5.1 where thin pointed lines represent links with 1024 units capacity, thick pointed lines represent links with 1536 units capacity. Thin traced lines represent links with 1664 units capacity, thick traced line represent links with 2048 units capacity, thin full lines represent links with 2560 units capacity, and thick full lines represent links with 3072 units capacity. The main distinction between topologies is their distribution of link capacity. The first topology possesses the standard deviation for this distribution equal to zero. The second topology has this standard deviation equal to 338, and the third has it equal to 577.

In all simulations, 600 LSPs were injected during a 50 seconds time period. LSP injection during simulations followed a pseudorandom distribution for LSP injection interval, LSP hold time, LSP bandwidth capacity. LSP injection intervals varied between 0.05 and 0.1 second, LSP hold times varied between 3 and 8 seconds and LSP bandwidth consumption (capacity) varied between 8 and 32 units.

A total of 30 finite independent simulations [10] with 30 different pseudorandom seeds were performed. During the simulation, the topology also varied so that each group of ten simulations (10 out of the 30 seeds) used the one of the topologies topology in fig. 5.1. This way, besides assuring an independent variation in traffic distribution, the experiments also assured a topological variation that made possible a more complete evaluation of the adaptability characteristics of all three algorithms.

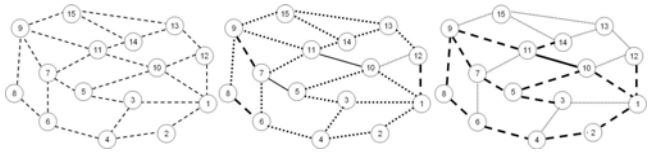

Fig. 5.1. Topologies used for the experiments.

C. *Evaluation Metric*

Each link in a given network can be measured in terms of residual bandwidth. If we group all links in this network, it is possible to notice that their residual bandwidth measures tend either to spread or to approximate to one another as they network receives traffic. To measure the dispersion of a sample set, one uses the standard deviation calculation. Interpreting this mathematical method in the situation where the sample is a set of residual bandwidth measures of the links of a network, the standard deviation may be translated into a load balance measurement technique. A higher standard deviation indicates that the measures for the residual bandwidths of the link of a network are very disperse from each other, meaning that these links are being used in a non equitable way, in other words, a bad load balance quality.

As the main goal of the proposed algorithm is to dynamically balance the load of a network, the metric used to evaluate the performance of the three simulated algorithms was the calculation of the standard deviation for the residual bandwidth of all links in the simulated networks.

D. *Results*

On all 30 simulations, the standard deviation for the residual bandwidth of all links in the network was calculated during time.

We can see the curve of this standard deviation versus time of one of the 30 simulations in fig. 5.2. As it is possible to observe, IntelliDyLBA balances the load more quickly them it peers, besides keeping the network balanced for longer period of time. This behavior repeated for all the 30 experiments, even when topology changed. To represent each experiment by a unique value, we chose to calculate the average standard deviation for each algorithm in a given experiment. With that, we could have a more exact notion of the load balance quality during the experiment. Figure 5.3 shows a curve for the standard deviation averages of all simulated algorithms for the experiments that used the first topology. Based on the analysis of these measures, it is possible to realize that IntelliDyLBA had an average standard deviation quite lower than it peers during all simulations.

Table I shows the confidence interval and confidence level for the standard deviation averages of fig. 5.3.

The experiments for the second and the third topology presented similar results concerning IntelliDyLBA's behavior,

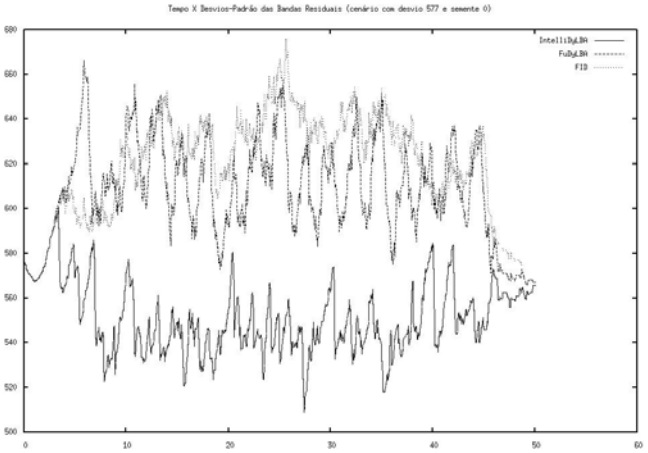

Fig. 5.2. Standard deviation for the residual bandwidth of the links versus time (in seconds) for all three algorithms in one of the 30 experiments.

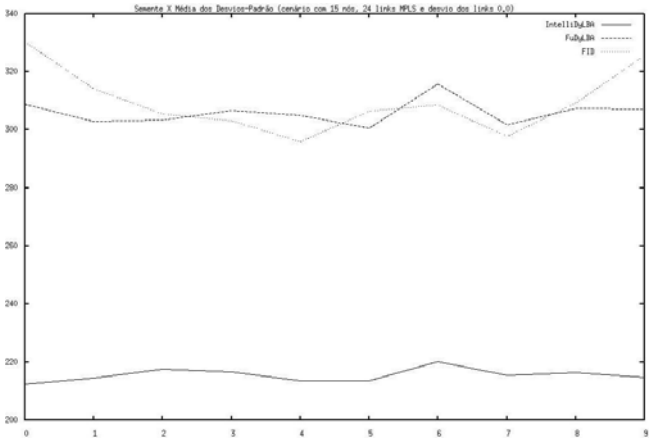

Fig. 5.3. Standard deviation average for theexperiments which used the first topology

confirming the algorithm's capacity to adapt.

We can observe the distribution of the standard deviation averages during the experiments which used the second topology in fig. 5.4, whereas we can see the same for the experiments which used the third topology in fig. 5.5.

TABLE I
CONFIDENCE LEVEL AND INTERVAL FOR THE EXPERIMENTS WHICH USED THE FIRST TOPOLOGY

Algorithm	Average Standard Deviation	Confidence Level	Confidence Interval
FID	309.642	95%	+/- 6.89435
FuDyLBA	305.915	95%	+/- 2.6985
IntelliDyLBA	215.557	95%	+/- 1.40914

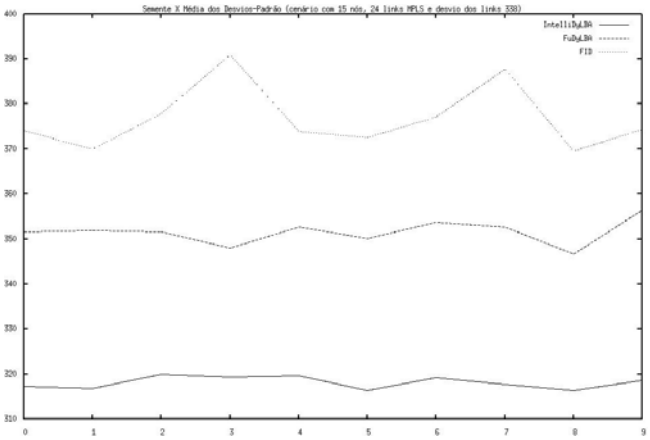

Fig. 5.4. Standard deviation average for theexperiments which used the second topology

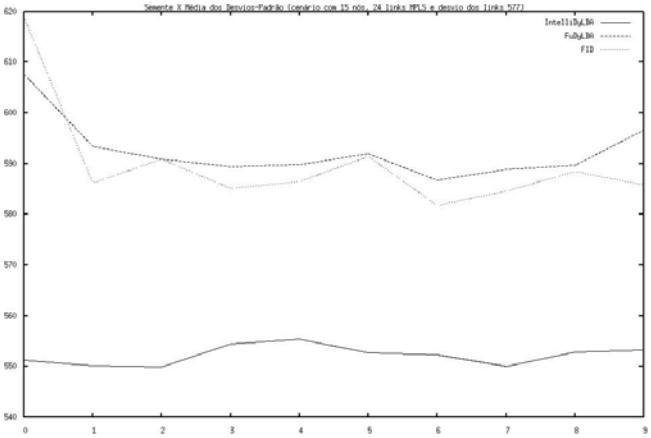

Fig. 5.5. Standard deviation average for theexperiments which used the third topology

TABLE II
CONFIDENCE LEVEL AND INTERVAL FOR THE EXPERIMENTS WHICH USED THE SECOND TOPOLOGY

Algorithm	Average Standard Deviation	Confidence Level	Confidence Interval
FID	376.777	95%	+/- 4.42517
FuDyLBA	351.515	95%	+/- 1.71377
IntelliDyLBA	318.11	95%	+/- 0.864944

TABLE III
CONFIDENCE LEVEL AND INTERVAL FOR THE EXPERIMENTS WHICH USED THE THIRD TOPOLOGY

Algorithm	Average Standard Deviation	Confidence Level	Confidence Interval
FID	589.977	95%	+/- 6.53577
FuDyLBA	592.536	95%	+/- 3.70995
IntelliDyLBA	552.242	95%	+/- 1.15648

Table II clarifies the results of figures 5.4 by showing the confidence level and interval for the average standard deviation. Table III does the same for fig. 5.5.

Analyzing the results for the three topologies, it is possible to observe that FuDyLBA, in some experiments, has a worse performance compared to FID. In contrast with the topological characteristics, we can extend this observation with the fact that this happened mostly with the third topology, where the standard deviation for the topology's links capacity is the highest. This observation is important to highlight IntelliDyLBAS's capacity to adapt in relation to FuDyLBA. As we can see, IntelliDyLBA works better in all cases.

VI. CONCLUSION

In this work, we presented IntelliDyLBA, a reactive load balance algorithm for MPLS networks that incorporates fuzzy logic and genetic algorithm based auto learning techniques. Comparing IntelliDyLBA's behavior to its peers' (FID and FuDyLBA), it was possible to realize the algorithm's great capacity to adapt to network changes in an intelligent manner, even in situations where its predecessor (FuDyLBA) shows a decay in performance.

Future works should focus on the "learning module" construction, which could be rebuilt using neural networks techniques, and on the objective sample set assembling, which may become quite a challenge.

REFERENCES

[1] Ahn, G. and Chun, W. (1999). Overview of MPLS Network Simulator: Design and implementation. http:// flower.ce.cnu.ac.kr/~fog1/mns/.

[2] Awduche, D.; Chiu, A.; Elwalid, A.; Widjaja, I. e Xiao, X.. (2002) "Overview and Principles of Internet Traffic Engineering.", IETF RFC 3272, [Online]. Available: http://www.faqs.org/rfcs/rfc3272.html

[3] Battiti, R. and Salvadori, E. (2002) "A Load Balancing Scheme for Congestion Control in MPLS Networks. Technical report", Universitá di Trento, Dipartimento di Informatica e Telecomunicazioni, November.

[4] Celestino Jr., J., Ponte, P. R. X., Tomaz, A. C. F., Diniz, A. L. B. P. B. (2004) "FuDyLBA: A Traffic Engineering Load Balance Scheme for MPLS Networks Based on Fuzzy Logic", in Proceedings of ICT 2004 (pp. 622 - 627), Fortaleza, Brazil, August.

[5] Fernandez, M. P.; Pedrosa, A. C. P.; Rezende, J. F. (2003) "Implementação de Políticas de Gerenciamento com lógica Fuzzy e Algoritmo Genético visando à melhoria da Qualidade de Serviço (QoS)", Revista da SBrT Vol 18-2, october.

[6] Herrera, F.; Lozano, M.; and Verdegay, J. (1995); "Tuning fuzzy logic controllers by genetic algorithms,"International Journal of Approximate Reasoning, vol. 12, pp. 299–315, June.

[7] Jüttner, A.; Szviatovszki, B.; Szentesi, A.; Orincsay, D. e Harmatos, J. (2000) "On-demand Optimization of Label Switched Paths in MPLS Networks.", In: Proceedings of IEEE International Conference on Computer Communications and Networks, p. 107–113, Las Vegas - Nevada, Outuber.

[8] McCanne, S. e Floyd, S. (1998); "NS – Network Simulator - Version 2"; [Online]. Available: http://www.isi.edu/nsnam/ns/

[9] Mortensen, J. E. (1998); "JFS Fuzzy System". [Online]. Available: http://inet.uni2.dk/~jemor/jfs.htm

[10] Pawlikowski, K.; Jeong, H. -D. J. e Lee, J. S. R. (2002); "On Credibility of Simulation Studies of Telecommunication Networks" IEEE Communications Magazine, pp. 132–139, Jan.

[11] Salvadori, E;.Sabel, M e.Battiti, R. (2002) "A Reactive Scheme for Traffic Engineering In MPLS Networks. Technical report", Universitá di Trento, Dipartimento di Informatica e Telecomunicazioni, Dezembro.

[12] Wang, B.; Su, X. e Chen, C.P.. (2002) "A New Bandwidth Guaranteed Routing Algorithm for MPLS Traffic Engineering", In: Proceedings of ICC, volume 2, p.1001-1005, New York – USA.)