FuDyLBA: A Traffic Engineering Load Balance Scheme for MPLS Networks Based on Fuzzy Logic

Joaquim Celestino Júnior, Pablo Rocha Ximenes Ponte, Antonio Clécio Fontelles Tomaz, Ana Luíza Bessa de Paula Barros Diniz

Communication Networks and Information Security Laboratory (LARCES) – Universidade Estadual do Ceará (UECE)
Av. Parajana, 1700 – Itaperi – 60.720-020 – Fortaleza – CE – Brasil
{celestino,pablo,clecio,analuiza}@larces.uece.br

Abstract. This article describes a new traffic engineering scheme based on a load balance reactive method for congestion control in MPLS networks that makes use of local search and fuzzy logic techniques. It is an algorithm based on the "First-Improve Dynamic Load Balance Algorithm" (FID) and it works with linguistic values while performing load balance decisions. Computer simulation experiments have shown a better traffic distribution over the links of a network if compared to the behavior of the original version of the algorithm under the same circumstances.

1. lntroduction

One of the most interesting applications of MPLS in IP based networks is Traffic Engineering (TE) [2]. The main objective of TE is to optimize the performance of a network through an efficient utilization of network resources. The optimization may include the careful creation of new Label Switched Paths (LSP) through an appropriate path selection mechanism, the re-routing of existing LSPs to decrease network congestion and the splitting of traffic between many parallel LSPs. The main approach to MPLS Traffic Engineering is the so-called Constraint Based Routing (CBR) [4]. In this article we present a new scheme to control the congestion in an MPLS network by using a load balancing mechanism based on fuzzy logic entitled FuDyLBA. FuDyLBA was built upon the ideas of the First-Improve Dynamic Load Balance Algorithm (FID) [4]. The main weakness of FID is that it does not take into account the link utilization intensity to define how many LSPs it will re-route which makes necessary in some cases several iterations of the whole algorithm. The proposed scheme adds a fuzzy controller to the structure of FID, thus eliminating the problems concerning its limitations. The article is organized as follows: Section 2 highlights general aspects about the proposed scheme; Section 3 describes in details the pseudocode of the proposed algorithm; in Section 4 we discuss the experiments performed to analyze FuDyLBA and their results; and finally in Section 5 we make some final considerations and sugestions for possible future works.

Joaquim Celestino Júnior, Pablo Rocha Ximenes Ponte, Antonio Clécio Fontelles Tomaz,
Ana Luíza Bessa de Paula Barros Diniz

1. Overview of FuDyLBA

Our algorithm is named FuDyLBA, because it is an improved version of First-Improve DyLBA (FID) by means of fuzzy logic techniques, or Fuzzified DyLBA (FuDyLBA).

In our method, the core of FID was modified in a way that enabled it to work with intensity levels, thus creating a link unload function. This function gets as input an intensity level that indicates how much a certain link should be released of its load. Such function performs in a way very similar to FID, but it does not stop after finding the first suitable LSP and rerouting it. In fact, it keeps restarting after every finding, accounting the amount of traffic load that has been released. When the released traffic load reaches the level defined by the input parameter, the local search stops and the next link is then examined. That goes on until all the links in the network are examined.

To calculate the input parameter of the link unload function, a fuzzy controller was designed. This controller has the percentage of link utilization as input parameter and the intensity level in which the link should be released of load as an output.

To represent the level of link utilization three linguistic input values were defined, as follows: “low congested”, “medium congested” and “high congested”. Each one of these values is defined by its own triangular membership function, hereby named input memberships. To understand linguistically the intensity level necessary for the unload function, three non-continuous membership functions were defined, as follows: “low unload”, “medium unload”, “high unload”, hereby named output memberships.

Every link in the network is measured in terms of percentage of link utilization and when a certain measurement fits any of the input memberships the link is considered congested. The unload function is then executed having the output of the fuzzy controller’ s defuzzification process (center of gravity) as an input.

2. The Pseudocode

The pseudocode of our algorithm is shown in figure 1. Let us define the notation. Three pseudocode blocks are defined. The first block shows the general structure of the algorithm, while the second and third constitute the fuzzified link unload procedure. On the first block the variable *LinkSet* receives the set of all the links in the network, and a loop is started. Inside this loop, one element of *LinkSet* is taken and registered in the variable *link* per iteration. After that, the unload function is called with *link* as input. The loop ends after the variable *LinkSet* has no more elements left.

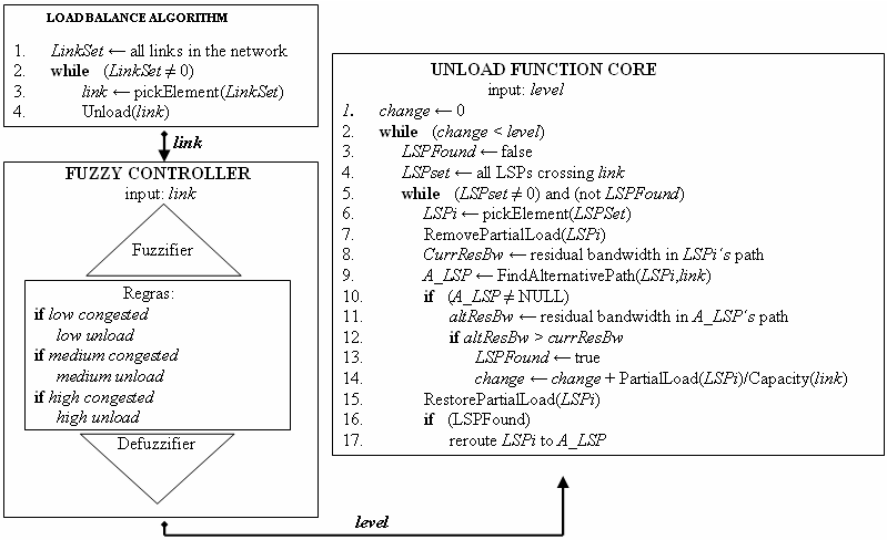

Fig. 1. Pseudocode for FuDyLBA

The unload function is a fusion of the second and third blocks. Actually, this function starts with a fuzzy controller, illustrated in the second block, that feeds the **core** of the unload function with the level which the link should be uloaded, based on the percentage of the link utilization. In the third block, we have the core of the unload function. In this portion of the pseudocode, first the variable change is initialized with zero. This variable will hold the accounting for how much the link was released of load. After that, the first loop is started and it will only stop when the variable change is greater than or equal to the variable level (which holds the intensity wich the link should be unloaded). Inside this first loop, we initialize the variables LSPFound (with false) and LSPSet (with the set of all the LSPs crossing the link) and a new loop is then started. This second loop will only stop when LSPFound is equal to the logical value "true" or when there is no more elements left in LSPSet. Inside this loop the local search is performed first by taking an element from LSPSet and placing it in the variable LSPi. This is the LSP to be examined. After that, the contribution of the LSP stored in *LSPi* in terms of bandwidth consumption is removed from the network through the function *RemovePatialLoad*. The residual bandwidth for the current LSP is, then, calculated and stored in *currResBw*. After that, the algorithm tries to find a new path that does not include the current link and tries to store it in *A_LSP*. If it succeeds, the residual bandwidth for this alternate path is calculated, stored in *altResBw*, and compared to *currResBw*. If it is greater, *LSPfound* is set to "true" and change is increased with the percentage of link capacity that was released. Finally, the partial load of the LSP stored in *LSPi* is restored to the network and if *LSPFound* is "true" the LSP defined by *LSPi* is rerouted to the path defined by *A_LSP*.

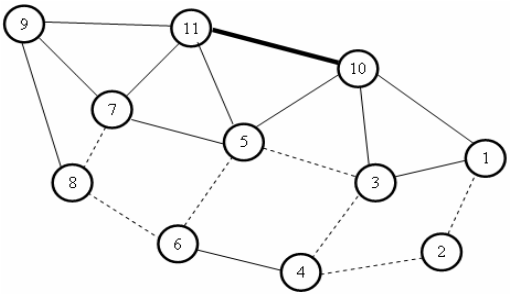

Fig. 2. Topology for the Simuled Network Used in the Experiments

4. Simulations and Results

The objective of the computer simulations was mostly to evaluate the improvements of FuDyLBA over FID, since FID was very well documented and simulated in [3], with comparisons to other load balance techniques that are very much mentioned in the literature, thus making its behavior well known.

4.1. The Experiments

To accomplish the experiments, the topology defined in the figure 2 was used. The thin traced lines represent links with capacity of 1024 units, the thin non-traced lines represent links with capacity of 2048 units and the thick line represent a link with capacity of 3072 units. The LSP requests are limited to the pair of LSRs 1 and 9, always starting at the LSR 1.

Two experiments were performed, both with the duration of 30 seconds, reproducing the exact same network characteristics for the two algorithms.

The first experiment used low granularity LSPs while the second one used higher granularity LSPs. The LSPs varied in a round-robin style in terms of bandwidth consumption, path and duration while they were being injected in the network at the rate of one at every 0,05 seconds. Six hundred LSPs were injected for both experiments. And the paths used were 1-3-10-11-9 and 1-10-11-9.

In the first experiment the bandwidth consumption used for the injected LSPs varied between 8 and 32 units as their lasting time varied between 3 and 8 seconds.

In the second experiment the bandwidth consumption used for the injected LSP varied between 32 and 128 units and their lasting time was the same used in the first experiment.

4.2. Results

The technique of choice for the evaluation of the quality of the load balance for each algorithm was the calculation of the standard deviation, during the time, for the percentages of residual bandwidth for each link in the simulated network.

The standard deviation is a well known dispersion measurement technique that is very suitable for the quality quantification of any load balance technique.

The standard deviation measures how much the elements of a sample set are apart from each other. Specifically in our case, those elements are the residual bandwidth for the link. A higher standard deviation would mean, in a final analysis, a worse load balance quality.

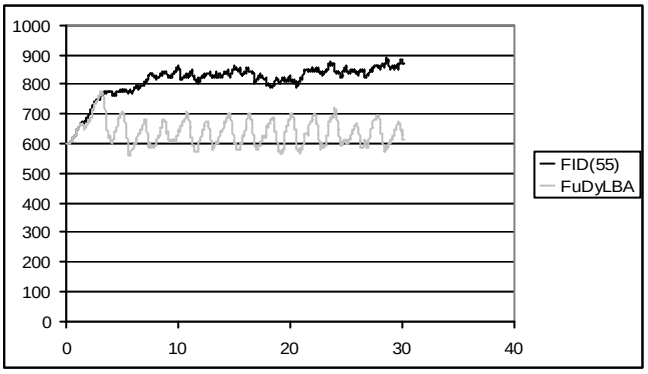

Fig. 3. Standard deviations of the desidual dandwidths for the first experiment, along the time in seconds, respectively.

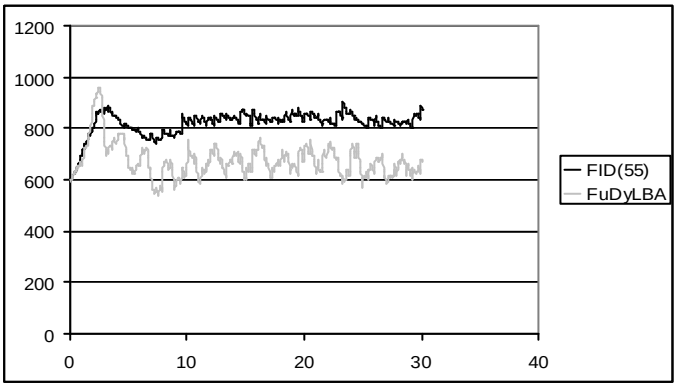

Fig. 4. Standard reviations of the residual bandwidths for the second experiment, along the time in seconds, respectively

In figure 3 we can see the chart for the first experiments. This chart shows the standard deviation of the residual bandwidth for every link in the network along the

time for FID and FuDyLBA. The same measurements for the second experiment can be found in figure 4.

Analyzing the presented charts, we can notice that FuDyLBA performs a better load balance than FID, not only as a final result, but for the most part of the experiment, even under attenuating circumstances.

The second experiment intended to show the effects of LSP granularity in FuDyLBA. By the charts it is possible to realize that the curves for both algorithms get closer, thus showing that under the insertion of high granularity LSPs FuDyLBA performs more similar to FID. This is a result of the link based characteristic of FuDyLBA. This does not constitue a downside because the computational complexity follows the same behavior.

5. Conclusions and Future Works

In this paper we presented the benefits of Fuzzy Logic techniques to traffic engineering reactive congestion management methods. Specially, we presented FuDyLBA, a reactive load balance algorithm based on fuzzy logic based that has its origins in the FID algorithm. Computer simulations were able to demonstrate how better FuDyLBA performs if compared to its predecessor. Is was shown as well that attenuating circumstances, like LSP granularity, can suppress the improvements brought by fuzzy logic techniques, making the functionality of the modified algorithm to be similar to the original one. That, in the other hand, does not constitute a downside, because computational complexity follows the same behavior, thus making the fuzzified algorithm, during attenuating circumstances, to behave like its non-fuzzy version with the same computational cost.

Some aspects of the algorithm are yet to be adressed in future works. This includes a better design of the membership functions and the integration of neural networks and genectic algorithms techniques.

References

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

[2]Awduche, D. O. e Jabbari, B. (2002) "Internet Traffic Engineering using Multi-Protocol Label Switching (MPLS)", Computer Networks, (40):p. 111–129, September.

[3]Battiti, R. e 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]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, December.