Lightweight Monitoring of Label Switched Paths for Bandwidth Management

P. Vilà, J.L. Marzo, E. Calle, L. Carrillo
Institut d'Informàtica i Aplicacions
Universitat de Girona
Girona (Spain)
{perev | marzo | eusebi | lilianac}@eia.udg.es

Abstract

The purpose of Resource Management is the efficient and effective use of network resources, for instance bandwidth. In this article, a connection oriented network scenario is considered, where a certain amount of bandwidth is reserved for each Label Switch Path (LSP), which is a logical path, in a MPLS or GMPLS environment. Assuming there is also some kind of admission control (explicit or implicit), these environments typically provide Quality of Service (QoS) guarantees. It could happen that some LSPs become busy, thus rejecting connections, while other LSPs may be underutilised. We propose a distributed lightweight monitoring technique, based on threshold values, the objective of which is to detect congestion when it occurs in an LSP and activate the corresponding alarm which will trigger a dynamic bandwidth reallocation mechanism.

1. Introduction

Nowadays, telecommunication networks are evolving quickly and continuously. Transmission speeds and the number of users and services have been growing more and more in recent years. However, there is a noticeable lack of dynamic management tools for resource configuration. Current network management systems are not able to monitor and manage such networks in a centralized way. Centralized management results in a scalability problem because the network management centre is responsible for collecting and processing all the monitoring data from all the network elements being managed.

Pure centralised management evolved into hierarchical and hybrid architectures in order to alleviate the management overload on the central manager. Management by Delegation (MbD) [1, 2] proposed that not only the monitoring but also management functions could be distributed to the management agents by the download of scripts. In recent years, two trends have begun to appear in the literature. One is the distribution of the decision making to the network elements and the other is the automation of the management functions. Many systems have been proposed in both cases, most of them based on Distributed Artificial Intelligence (DAI) techniques, i.e. multi-agent systems and mobile agents [3, 4, 5, 6].

One of the management areas is the dynamic resource management, which has the objective of maximising the network resource utilization. To achieve this, the network technology needs to have the appropriate resource reservation mechanisms. This paper focuses on the logical or virtual network paradigm, i.e. a dynamically configurable network [7]. In MPLS [8] and GMPLS this is carried out by means of Label Switch Paths (LSPs), which can have resources reserved. Connections are established through this set of LSPs.

Several dynamic bandwidth management systems have been proposed in the literature (e.g. [9]). These systems are usually based on a centralized optimisation algorithm, which is executed periodically (e.g. every hour). The main drawbacks of these methods are that the reconfiguration is usually not applied when the problem actually exists and that many changes are required in the whole network at the moment of reconfiguration.

To overcome these problems, we proposed a system [10] which balances the number of changes by making only small rearrangements when a problem is detected. The main objective of that system was to maintain a good network utilisation while ensuring good scalability. In the environment of connection oriented logical networks, this is achieved by minimizing the number of blocked connections for every LSP in the

0-7803-8623-X/04/$20.00 ©2004 IEEE

network. Therefore the proposed system monitored all the LSPs in the network at their origin. When congestion is detected in an LSP, then an alarm is triggered in order to activate a mechanism which, if possible, performs an adjustment in the logical network in order to solve the problem.

Note this paper only focuses on the monitoring function and the mechanism that detects congestion when it occurs in an LSP and triggers the alarm. We call it the Triggering function. We propose and evaluate three different Triggering functions that can be used in different cases. The main characteristics of these functions are that they use only few input values easily obtained from the router or the admission control system, and that the Triggering functions themselves are very simple lightweight processes, which do not overwhelm the network elements.

1.2. Network resource management

Resource Management can be viewed as the management of the bandwidths assigned to the LSPs, i.e. changing their bandwidth in order to better adapt the Logical network to the traffic offered. The ultimate objective is to maximise network utilisation. At times, due to unforeseen changes in the offered load and/or because Logical Network design is not optimal, some LSPs can become underutilised and others congested.

The objective of Bandwidth Management can also be seen as the minimisation of the Connection Blocking Probability (CBP), i.e. the probability that an offered call is rejected due to insufficient capacity. There are two actions usually performed by the bandwidth management systems in order to increase the bandwidth of a congested LSP: re-allocation of bandwidth and re-routing of LSPs [11]. Re-allocation is preferable to re-routing because it is less traumatic for the already established connections.

2. Lightweight monitoring architecture

The system architecture is based on a fully distributed and independent set of monitoring processes, placed on every node in the network. There is a main process on every node responsible for monitoring the creation and destruction of LSPs beginning in that particular node. This process has the ability to create and destroy threads (lightweight processes) and each one of these threads is assigned to monitor a single LSP. Figure 1 shows these different processes in a single node. We call these processes Node Monitor and LSP Monitor. On the other hand LSP Monitors are usually halted and only awake every time the monitoring period expires; this results in a very lightweight monitoring processes. Note that LSPs that pass through or end on a particular node are not monitored at all on that particular node.

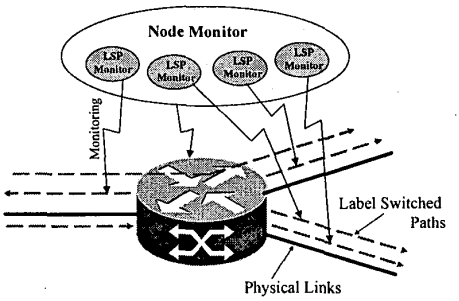

Figure 1. Light monitoring architecture: detail of a node

When the Triggering Function detects a congested LSP, an alarm message is sent to a Bandwidth Management System, which can be a centralised system or a distributed one. This system is responsible for the readjustment of the logical network, typically trying to increase the capacity of the congested LSP. We have also developed a distributed system which receives the alarms from the monitoring system and tries to increase the capacity of the congested LSP by several means [10].

Another possible function of the Node Monitor could be to offer the Bandwidth Management System a particularised monitoring for every single LSP. This would require enhancing the communications between the Node Monitor and the Bandwidth Management System. In this situation, each LSP could be monitored using a different Triggering function and different parameters according to, for instance, the LSP traffic characteristics.

One of the main ideas behind distributed architectures is the proximity of the decision making to the managed elements of the network. Also, the distribution of the processing load becomes more balanced since it is not concentrated in a single point. Another advantage of a distributed system is its robustness against failures. Therefore, the monitoring processes of our proposal can be placed on the same nodes and communicate directly with the node control system.

3. Triggering function models

In this section, we define the three proposed Triggering functions. Every time the LSP Monitoring process monitors its assigned LSP, the Triggering

function is executed. The Triggering function determines whether the monitored LSP is to be considered congested or not. If it is considered to be congested, an alarm message will be sent to the corresponding LSP Management System, the details of which are beyond the scope this work. With regard to the Node Monitoring processes, we assume that every router/switch always informs its corresponding Node Monitor as to when a LSP creation/release occurs.

The Node and LSP monitors obtain the values of three monitored variables from the Node Control System directly or through an SNMP agent. In fact, the monitored variables are closely related to the Admission Control mechanism. More specifically, they are related to the number of accepted connections, the number of rejected connections and the bandwidth already assigned to the currently established connections of a given LSP. We assume that these variables are very common at the Node Control Systems and/or the SNMP MIBs, and that they are usually available. We also think this information could easily be collected and maintained by the Admission Control mechanism and made available for management purposes. The monitored variables (for a given LSP) are:

- The total number of offered connections (*OC*). This is a counter that keeps track of the number of offered connections from the beginning of the LP operation (e.g. $OC = 1233$).
- The total number of rejected connections (*RC*). This is a counter that keeps track of the number of the offered connections that have been rejected. Always, $RC \leq OC$ (E.g. $RC = 125$).
- The current LSP load (*L*). This is the amount of bandwidth assigned to the established connections of a LSP at that precise moment, given as a percentage of the total amount of bandwidth assigned to the LSP.

Note that the offered/rejected information could also be obtained under other forms, such as accepted/rejected (e.g. 1108/125) connections or offered/accepted (e.g. 1233/1108) connections. We also assume that these variables are accumulative and their value never decreases in the life of the LSP. Note also that the LSP load is independent of the real traffic load of the currently established connections. As the values of the above defined variables are obtained periodically, we define them in terms of time: *OC(t)*, *RC(t)*, and *L(t)*.

The three Triggering functions are called *Rejected(t, limit)*, $CBP_{30}(t, limit)$, and *Load(t, limit)*. Their input value, *limit*, is the limit for considering the monitored LSP to be congested. In the first case, *limit* is an absolute value, while in the second and third cases, it is a percentage. These functions also depend on the time because they are evaluated periodically. All these functions have the same output: "**1**" if the LSP is considered congested, and "0" otherwise. If the output is "1", then the Node Monitor sends the corresponding alarm message.

The idea of the *Rejected(t, limit)* function is to count the rejected connections or flows in the successive monitoring periods. If there are rejected connections in the present period (i.e. $RC(t) > RC(t-1)$), then the rejected connections in the present period (i.e. the difference $RC(t) - RC(t-1)$) is accumulated in an internal counter of the LSP Monitor. If in the present period there are no rejected connections (i.e. $RC(t) = RC(t-1)$), then the LSP Monitor counter is reset to zero. When the value of the counter is equal or greater than the given limit, then the LSP is considered to be congested. A formal definition is presented below.

First of all, some prior definitions are needed:

$$\delta_{RC}(t) = RC(t) - RC(t-1)$$

$$count(t) = \begin{cases} count(t-1) + \delta_{RC}(t) & \text{if } \delta_{RC}(t) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Then the Triggering function *Rejected(t, limit)* is as follows:

$$Rejected(t, limit) = \begin{cases} 1 & count(t) \geq limit \\ 0 & \text{otherwise} \end{cases}$$

The idea of this function is to allow a few occasional rejections, but if the rejections persist in time and/or there are many rejections and the limit is exceeded, then the LP is considered to be congested. Table 1 shows a numerical example.

Table 1. Rejected function numerical example

t	OC(t)	RC(t)	Count(t)	Rejected(t, 5)
0	0	0	0	0
1	15	2	2	0
2	22	3	3	0
3	34	3	0	0
4	48	6	3	0
5	62	9	6	1

The idea of the $CBP_{30}(t, limit)$ Triggering function is to evaluate the Connection Blocking Probability (CBP) for the last 30 offered connections or flows, i.e. the ratio between the rejected connections and the total offered connections. To calculate this CBP, we found there was a lack of information, because the only values obtained from the router/switch are *OC(t)* and *RC(t)* at a given time *t*. Therefore, the distribution of the accepted and rejected connections is unknown. For this reason, the $CBP_{30}(t, limit)$ function calculates the

CBP in the worst case, i.e. the case when the accepted connections are all grouped at the beginning of the monitored period and the rejections are all grouped at the end of the monitored period. Some previous definitions are also necessary. We define a sequence of bits a_n, where n is an integer value, as follows:

$$a_i = \begin{cases} 0 & OC(t-1) < i < OC(t) - RC(t) \\ 1 & OC(t) - RC(t) \le i \le OC(t) \end{cases}$$

If there had been offered calls on a monitoring period (i.e. $OC(t) > OC(t\text{-}1)$) then $OC(t) - OC(t\text{-}1)$ bits are added to this sequence, zeros for the amount of accepted connections on that period and ones for the amount of rejected connections, in that particular order. Then the $CBP_{30}(t, limit)$ evaluates the last 30 elements of this sequence, and it is defined as follows:

$$CBP_{30}(t, limit) = \begin{cases} 1 & \frac{1}{30}\sum_{i=OC(t)-29}^{OC(t)} a_i \ge limit \\ 0 & \text{otherwise} \end{cases}$$

This can be implemented easily using the idea of a shift register. It is necessary to consider that $a_{-29},..,a_0$ has been initialised to zero for a proper operation. Although the window size could be an input parameter ($CBP(t, limit, window)$) we fixed it at the value of 30, in order to have a certain amount of offered connections for the CBP calculation and not to have too many open parameters for the experiments. Figure 2 also shows an example.

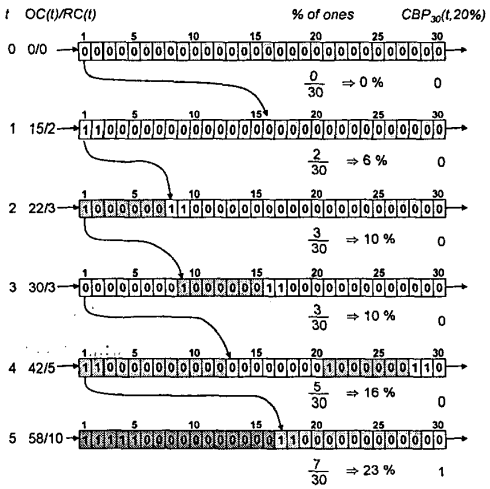

Figure 2. CBP_{30} Function Numerical Example

Both these first two Triggering Functions can be applied using any monitoring period. However, if the monitoring period is too high compared with the offered connection rate, then these functions may lose their meaning. That is, the monitoring period should be defined in a way that the number of offered connections on each monitoring period is not greater than a given limit with a given probability. This supposes that the offered connections rate is known, which is usually not true.

The first two Triggering functions are reactive, i.e. they wait until a certain level of rejection (given by the limit) is produced. The third function, *Load(t, limit)*, can be considered to be preventive, because it tries to solve the problem before it exists. The idea of the *Load(t, limit)* function is simply that if the percentage of occupation of the monitored LSP exceeds the given limit, then the alarm is sent in order to increase the LSP capacity. Therefore, before the LSP is full and begins to reject connections the LSP Management System tries to increase it. In this case the function is defined as follows:

$$Load(t, limit) = \begin{cases} 1 & L(t) \ge limit \\ 0 & \text{otherwise} \end{cases}$$

This is a very simple function, which moreover does not depend on past values, just the instantaneous load. Note that this load is the amount of LSP bandwidth assigned to user connections given as a percentage of the total amount of LSP bandwidth, and it does not reflect the real traffic.

4. Experiments and results

The main objective of the experiments presented in this section is to gain information on the behaviour of the Triggering functions when monitoring a network. Due to the complete independence of the LSP Monitoring processes, we chose to simulate only a two-node network with only one LSP. This also helps to focus our attention to the Triggering function behaviour, as well as simplifying the simulations and facilitating the possibility of performing many more tests.

We implemented both the monitoring and management systems using Java. Each Node Monitoring process is an independent Java process and the LSP Monitors are threads inside each Node Monitor. The Monitoring System monitors a simulated network, which is also implemented (in C++) as a distributed system [11].

In order to evaluate the Triggering functions, we need to provoke congestion in a given LSP in order to cause connection rejections. When the Triggering function sends an alarm indicating that the LSP is congested, we implemented a mechanism that

increases the capacity of the LSP by a fixed amount, which we call Step Size. The simulation is configured in such a way that the LSP can always be increased. Therefore, we are interested in the warm up part of the simulation, when there are rejections and the LSP is adapting its capacity to the offered load. We performed simulations of ten minutes with a high offered load, and the effect was that during these ten minutes, there were many LSP capacity changes.

We decided to test the three Triggering functions under several limit values, several monitoring periods and several step sizes. All these variables are specified in Table 2. In addition to the 192 different combinations of these parameters we chose to perform all these simulations with two different traffic models: one with homogeneous connections and the other with heterogeneous connections. The connection distributions used in both cases are presented in Table 3.

Table 2. Simulation parameters

Parameter	Values			
Monitoring Period (s)	2	5	10	20
Step Size (Kbps)	500	1000	2000	4000
Rejected Limit (connections)	1	3	5	7
CBP_{30} Limit (%)	10	30	50	70
Load Limit (%)	85	90	95	99

Table 3. Offered connections

Offered Connections	Initial LSP Capacity (Kbps)	Connection Size (Kbps)	Mean Inter-arrival Time (negative-exp. distributed) (s)	Mean Duration Time (negative-exponential distributed) (s)
Homogeneous	512	64	1	1000
Heterogeneous	8000	64	2	1000
		2000	16	1000

The results obtained can be analysed and compared from several points of view. Even focussing on just one traffic type means there are still too many results to be presented in easily-understood manner. For this reason, we have grouped the results in different ways and we present several graphs showing the behaviour of the Triggering functions with regard to the monitoring interval, the step size and the limits.

By fixing a Triggering function and its limit, it is possible to see that there are no great differences in relation to the monitoring period (Figure 3). That is, the performance of the Triggering function is similar for monitoring periods of 2s, 5s, and 10s. Only the 20s period gives, in all cases, a significantly worse result, which means therefore, that a 20s period is too long. It is already possible to see that the best Triggering function is Load, followed by Rejected and finally the worst is the CBP.

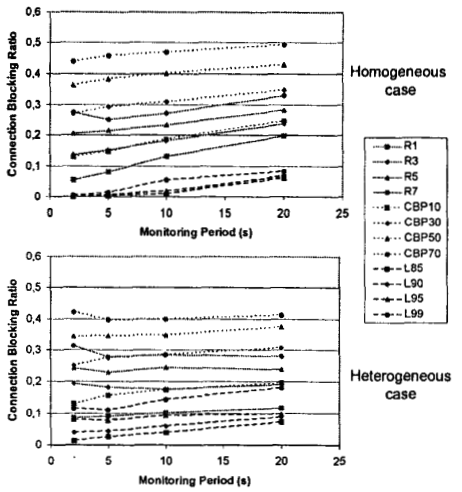

Figure 3. Call blocking ratio in relation to the monitoring period for the different Triggering functions and limits

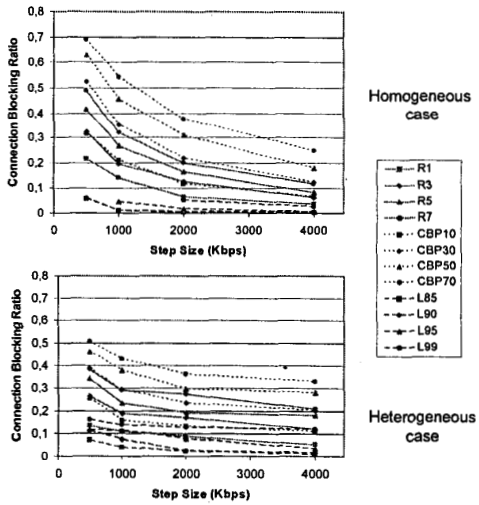

Figure 4. Call blocking ratio in relation to the Step Size for the different Triggering functions and limits

With regard to the Triggering function limits, the general behaviour is that the lower the limit, the better the Triggering function performance. In these Figures, it is also possible to compare specific cases, for

instance CBP 10% performs better than Rejected 7 for all the monitoring periods.

Figure 4 presents the same results as Figure 3, but using the Step Size instead of using the Monitoring Period. In this case, it is clearer that for all the Triggering functions and their limits, the greater the Step Size, the better the performance of the Triggering function. This is because when the Triggering function detects that the LSP is congested, it is then increased by the Step Size, and the greater the Step Size, the longer it takes the LSP to become congested again.

With regard to the Heterogeneous connection case, the results show some small differences with the Homogeneous case. The most notable differences are: first, that the behaviour between the different cases is not so similar; second, that in several Triggering functions, the 2-second Monitoring time produces a worse performance than the 5-second one, and third, that the Load function performs clearly better than the others, but not much better, as is the case in the Homogeneous connection cases.

The main conclusion from these results is that, in general, the Load function has the best performance in most of the situations. This is because this function is preventive. However the Load function has a severe drawback: the misuse of a percentage of the LSP capacity, which will usually not be used. This could prove to be unacceptable, especially with large LSPs or with a great number of LSPs.

5. Conclusion and future work

The Monitoring System presented in this paper is designed to help a Bandwidth Management System (which is beyond the scope of this work) to perform a dynamic management of a Logical Network composed of LSPs. This system represents a lightweight load to the network nodes because of the simplicity of the Triggering functions, the relatively long monitoring period, and the low number of monitored variables, which moreover could be easily available. The use of a distributed system where the processes are completely independent, makes the Monitoring system scalable.

In order to determine the characteristics of the three proposed Triggering functions, we have performed a battery of tests comprising several monitoring periods, several Triggering function limits, several Step Sizes and two different offered loads. The ultimate goal of these experiments is not to select one of the Triggering functions, specific parameters, Step Size or monitoring period, but to gain experience on their behaviour in order to select the best option in every situation.

In addition, we are also studying how to convert these simple monitoring processes into a more complex system with the autonomy to decide the parameters for itself. That is to say, we are looking into the possibility of converting the monitoring processes into autonomous software agents capable of decisions about, for instance, what the best Step Size is in each situation and instant.

6. Acknowledgement

This work was partially supported by the Spanish government under contract TIC2003-05567.

7. References

[1] Y. Yemini, G. Goldszmidt, S. Yemini, "Network Management by Delegation", 2nd International Symposium on Integrated Network Management, Washington DC, April 1991

[2] Germán Goldszmidt, Yechiam Yemini, "Delegated Agents for Network Management", IEEE Communications Magazine, March 1998

[3] S. Albayrak (Ed.), "Intelligent agents for telecommunication applications", Lecture Notes on artificial intelligence vol.1699, Springer-Verlag 1999, ISBN 3-540-66539-0

[4] A.L.G. Hayzelden, R.A. Bourne (Eds.), "Agent technology for communications infrastructure", John Wiley & Sons Ltd. 2001, ISBN 0-471-49815-7.

[5] T. Magedanz, K. Eckardt, "Mobile Software Agents: A New Paradigm for Telecommunications Management", IEEE/IFIP NOMS'96, Network Operation and Management Symposium, Japan, April 1996.

[6] W. Caripe, G. Cybenko, K. Moizumi, R. Gray, "Network Awareness and Mobile Agent Systems", IEEE Communications Magazine, July 1998.

[7] V.J. Friesen, J.J. Harms, J.W. Wong, "Resource management with virtual paths in ATM networks", IEEE Network vol.10 no.5, September/October 1996.

[8] E. Rosen, A. Viswanathan, R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, January 2001

[9] T-H. Cheng, Y-K. Sze, C-W. Tan, "A heuristic algorithm for allocating virtual path bandwidth in an ATM network", Computer Communications vol.22 no.9, 1999.

[10] P. Vilà, J.L. Marzo, A. Bueno, "Automated Network Management Using a Hybrid Multi-Agent System", In proceedings of Artificial Intelligence and Applications (AIA 2002), September 2002. Málaga, Spain. ACTA Press.

[11] J.L. Marzo, P. Vilà , L. Fàbrega, D. Massaguer, "Distributed Simulator for Network Resource Management Investigation", to appear in Computer Communications Journal - Special issue on Recent Advances in Communication Networking, Volume 26, Issue 15, September 2003, Pages 1782-1791.