A Hybrid Fault-Tolerant Algorithm for MPLS Networks

Maria Hadjiona, Chryssis Georgiou, Maria Papa, Vasos Vassiliou

Department of Computer Science, University of Cyprus, CY – 1678, Nicosia, Cyprus
{cs02cm1,chryssis,cs03pm,vasosv}@cs.usy.ac.cy

Abstract. In this paper we present a new fault tolerant, path maintaining, algorithm for use in MPLS based networks. The novelty of the algorithm lies upon the fact that it is the first to employ both path restoration mechanisms typically used in MPLS networks: protection switching and dynamic path rerouting. In addition, it is the first algorithm to adequately satisfy all four criteria which we consider very important for the performance of the restoration mechanisms in MPLS networks: fault recovery time, packet loss, packet reordering and tolerance of multiple faults. Simulation results indicate the performance advantages of the proposed hybrid algorithm (with respect to the four criteria), when compared with other algorithms that employ only one of the two restoration mechanisms.

Keywords: MPLS, fault tolerance, algorithms, rerouting, protection switching.

1 Introduction

The explosive increase of data circulation over the Internet in conjunction with the complexity of the provided Internet services have negatively affected the quality of service and the data flow over this global infrastructure. The Multi-Protocol Label Switching (MPLS) [21] combines the scalability of the IP protocol and the efficiency of label switching to improve network data circulation.

The protection of data flows in the case of link or router failures is very important, especially for real time services and multimedia applications. MPLS employs two basic techniques for network recovery: (i) protection switching, where a pre-computed alternative path, which is usually disjoint from the working path, is set up for every flow and (ii) rerouting, where an alternative path is dynamically recomputed after a fault is detected. For both techniques, the alternative path can be either global or local [22].

The recovery of the MPLS network is based on the algorithm that is applied in order to detect the faults and route the data flow in an alternative path. There are various algorithms that have been proposed in the bibliography. However, each algorithm employs only one of the two basic techniques.

The main motivation for this work is to overcome the drawbacks of the previously proposed schemes for the restoration mechanism in MPLS networks during link/node failure. As already mentioned, existing algorithms use either rerouting [2, 6, 8, 10, 13,

14, 16, 23] or protection switching [1, 3, 4, 5, 7, 10, 12, 17, 18, 20] to reroute traffic fast when a fault occurs in the MPLS domain. Each technique presents both advantages and disadvantages depending on the application or the topology of the network they are employed upon. Protection switching provides fast restoration when compared to the rerouting technique, since the alternative path is already established and the switching to it is performed immediately after the fault is detected. Alternatively, the rerouting technique appears to be better in handling multiple faults, since a new alternative path, if needed, is computed dynamically for each fault. A question driven by the comparison of the two techniques is whether the combination of rerouting and protection switching will give better results.

We consider fault recovery time, packet loss, packet reordering and the ability to tolerate multiple faults as the most important criteria to evaluate a fast restoration algorithm in MPLS networks. To the best of our knowledge there is no current algorithm, either protection switching or rerouting, able to perform well in all four performance criteria. The challenge is to find an efficient way to combine the two restoration mechanisms in order to exploit each method's strengths and obtain a new hybrid algorithm that would perform best in all four criteria.

In this paper we propose and evaluate such a hybrid fault-tolerant path-maintaining algorithm for use in MPLS based networks. It satisfies all four abovementioned performance criteria and deploys effectively, in a non-trivial manner, both mechanisms based on the conditions of the fault, thus exploiting the advantages of each technique. Simulation results demonstrate the effectiveness of the new approach.

2 Related Work

2.1 Performance Criteria

Several criteria to compare the performance between different MPLS-based recovery schemes are defined in [20]. These are: packet loss, additive latency, re-ordering, recovery time, full restoration time, vulnerability, and quality of protection. Fault recovery time is the time elapsed between the fault detection and the time when the first packets are rerouted using the alternative path. Recovery time includes additive latency and sometimes is the equivalent to full restoration time. Packet loss is the percentage of packets lost until the fault is recovered. Packet reordering is whether the packets delivered during the recovery period are delivered out-of-order or not. Vulnerability is the time that the protected LSP (Label Switching Path) is left unprotected and quality of protection is the probability of a connection to survive the failure.

For the purposes of our work, which focuses more on the global performance and fault-tolerance of the MPLS network, we consider the first three criteria (recovery time, packet loss and packet disordering) and instead of vulnerability and quality of protection, we consider, as a more suitable fourth criterion, the ability of the network to tolerate multiple faults.

2.2 Comparison

Several service restoration algorithms are proposed for MPLS networks, each employing one of the two restoration mechanisms. Particularly, protection switching technique is employed by Haskin [12], Makam [17], Gonfa [10], Two Path [3, 4, 5], RBPC [1], Dual [7], MBAK [20], and SPM [18] algorithms. On the other hand, rerouting technique is employed by Dynamic Routing [10], A.J.C [2], Yoon [2], [23], Chen & Oh [2, 8], Otel [19], MIRA [6, 14], Hongs [13], and Lin & Lui [16] algorithms. We studied each of these algorithms and evaluated them (in a theoretical manner) based on the abovementioned four criteria.

The characterization and evaluation of the existing fault tolerance algorithms based on the selected criteria is shown in Table 1. The table is divided with two horizontal parts; the upper level contains protection switching algorithms, whereas the lower level contains rerouting algorithms. The symbol "+" indicates that the specific algorithm satisfies adequately the corresponding criterion. Due to lack of space we do not present the justification of whether an algorithm adequately satisfies a criterion or not, but the interested reader can obtain this information along with a detailed description of each algorithm in [11].

One can observe that the two algorithms which satisfy the most criteria are Gonfa [10] and Otel [19]. The Gonfa algorithm is a protection switching algorithm which performs well with respect to recovery time, packet loss and packet reordering criteria. On the other hand, Otel algorithm is a rerouting algorithm which performs well with respect to recovery time, packet loss and tolerance of multiple faults. In addition it can be observed that the two algorithms are not able to satisfy all four criteria (only three each). Based on these observations we decided to develop a new algorithm that makes use of these two algorithms and is able to satisfy adequately all four criteria. The new algorithm is presented in the next section.

Table 1. Comparison of existing algorithms based on the four selected criteria.

Algorithms	Local(L), Global(G) Restoration	Recovery Time	Packet Loss	Packet Reordering	Multiple Faults Tolerance
Makam	G			+	
RBPC (Local)	L				+
Two Path	L				+
MBAK	G				+
RBPC (Global)	G			+	+
Dual	L	+	+		
Haskin	L	+	+		
Gonfa	L,G	+	+	+	
Dynamic Routing	L				+
Hongs	L				+
MIRA	G			+	+
Lin & Lui	G			+	+
A.J.C	L			+	+
Chen & Oh	L	+	+		
Yoon	L	+	+		
Otel	L	+	+		+

3 The Hybrid Algorithm

In this section we give a brief description of the new algorithm, referred as Hybrid, and describe its execution through an example.

3.1 Description of the Algorithm

The hybrid algorithm maintains four data structures: (i) a Shortest Path Tree (SPT) where the root is the node that will execute the calculations, (ii) an array of lengths which contains the length of the shortest paths between the SPT root and all other nodes, (iii) a priority queue for nodes, (iv) a list maintained by ingress LSR (Label Switching Router) and contains the working and alternative LSP. The first three data structures are the ones also used by the Otel algorithm (details in [19] or [11]). Hence, the additional state information compared to [19] is the fourth structure.

First, the establishment of the working LSP and the alternative LSP which protects the whole working LSP is fulfilled. Afterwards the segment protection domains are determined along with their backward LSPs. Then the backward LSPs are established. The alternative and backward LSPs are established based on the Gonfa algorithm.

The alternative LSP and backward LSPs are used by data flows with low priority. (When a fault occurs, the LSPs will be needed for restoration of the fault. Hence the low priority flows will stop routing via those paths in order to forward the influence data flow with high priority). In addition, all the abovementioned data structures are created. Once the initialization phase is complete, the algorithm begins its path maintenance operation using the Gonfa algorithm. Depending on the nature and location of a fault as well as the current state of the network topology, the algorithm might divert in using the Otel algorithm and back (along with some additional calculations), as can be observed by the Hybrid algorithm's outline, given in Figure 1. The data structure SPT is updated with the use of any Single Source Shortest Part algorithm (SSSP) [15]. Full details of the hybrid algorithm can be found in [11].

3.2 Example of an Execution of the Algorithm

For a better understanding of the algorithm we describe a specific execution of the algorithm and make use of the network topology given in Figure 2. The working path is established between LSR1, LSR3, LSR5, LSR7, LSR9 and LSR11 and the alternative path is established between LSR1, LSR2, LSR4, LSR6, LSR8, LSR10 and LSR11. The first backward LSP is LSR3, LSR1, the second backward LSP is through LSR7, LSR5, LSR3 and the third backward LSP is formed by LSR11, LSR9 and LSR7.

The first link failure occurs between LSR5-LSR7 and based on the Hybrid algorithm's description the fault is recovered using the Gonfa algorithm. The data flow is routed via the backward LSP which it is formed by LSR5 and LSR3 and then follows the alternative path LSR4, LSR6, LSR8, LSR10 και LSR11. Afterwards the traffic is routed directly to the alternative path by LSR3, as shown in Figure 2.

Begin:
Establish working, alternative and backward LSPs
Compute:
1. SPT
2. Array of lengths
3. Array with the pre-established paths in Ingress LSR

Set _working_LSP as available
Set _alternative_LSP as available
Run the Gonfa algorithm

When failure occurs check:
If (failure is in working path)
    Set _working_LSP as NOT available
If (failure is in alternative path)
    Set _alternative_LSP as NOT available
If (_working_LSP IS available && _alternative_LSP IS available)
    Update SPT using SSSP and array of lengths
If (_working_LSP IS NOT available && _alternative_LSP IS available)
    Step1: Recover from fault using the Gonfa algorithm
    Step2: Update SPT using SSSP and array of lengths
If (_working_LSP IS available && _alternative_LSP IS NOT available)
    Update SPT using SSSP and array of lengths
If (_working_LSP IS NOT available && _alternative_LSP IS NOT available)
    Recover from fault using the Otel algorithm

When repair of a failure occurs check:
Step1: Update SPT using SSSP and array of lengths
Step2: Check: working LSP is repaired?
If (compare array with pre-established paths in Ingress LSR and SPT)
    Step1: Reroute the traffic in the working LSP
    Step2: Set _working_LSP as available
Else (compare array with alternative paths in Ingress LSR and SPT)
    Set _alternative_LSP as available

Figure 1. Outline of the Hybrid Algorithm

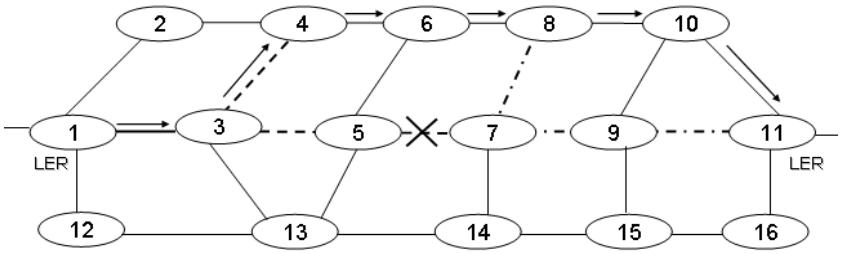

Figure 2. Recovery from the fault using the Gonfa algorithm

The next step is to update all data structures of every node. In this example we concentrate on LSR3, as LSR3 is the upstream LSR of the next fault and consequently LRS3 will become responsible in finding the alternative LSP and route the flow. In Figure 3(a) the SPT before the failure is shown and in Figure 3(b) the SPT after the failure is shown.

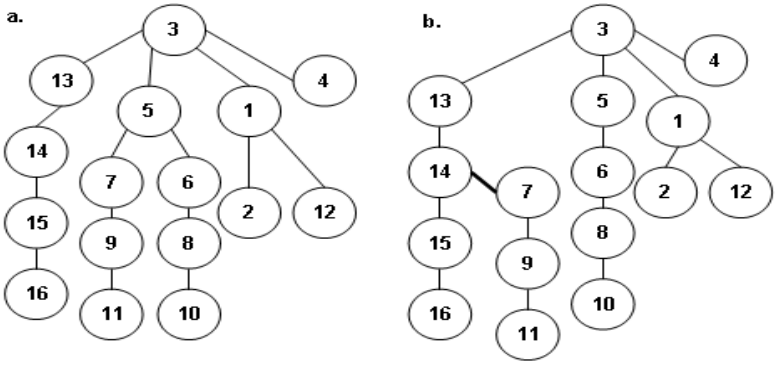

Figure 3. (a) SPT before link failure. (b) SPT after link failure.

The next fault occurs on the link connecting LSR3 and LSR4. Per the Hybrid algorithm's description the fault can be repaired with the use of the Otel algorithm. The SPT of LSR3 after the first fault is shown in Figure 3(b). Hence the next step is to consider the SPT subtree rooted at the disconnected downstream LSR and then starting with the subtree root, all the nodes in this subtree are marked as "unreachable". In this case the unreachable node is only LSR4 and the destination node is a reachable node. Therefore there is a path which can be used to route the data to the destination node. The local path is LSR3, LSR13, LSR14, LSR7, LSR9, and LSR11. Figure 4 shows how the data flow is routed after the second failure.

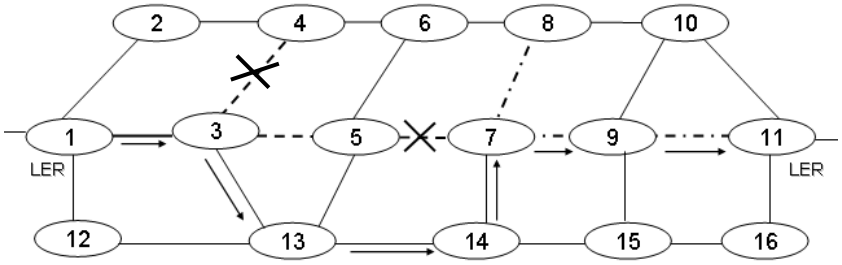

Figure 4. Recovery from the second fault.

After routing the flow via the local alternative LSP, the Otel algorithm continues in order to update its data structures. The SPT is updated by deleting the branch linking LSR4 and its parent LSR3 and adding as an SPT branch the linking LSR2 and LSR4, as shown in Figure 5.

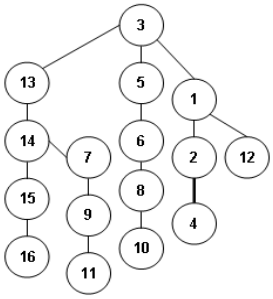

Figure 5. SPT after second link failure.

4 Simulations and Analysis

We have performed simulations that experimentally evaluate and contrast the performance (based on the four criteria) of the Hybrid algorithm with the performance of Otel and Gonfa (i.e., the best representatives of each restoration mechanism). The simulations were performed on ns2 [9]. We experimented using several different simulation scenarios, considering different network topologies (including mesh, star, hierarchical with single- and multi-homed nodes, and common telecommunication networks in USA and UK). The simulation results were analyzed by grouping the topologies in categories depending on the number of nodes and links they have. We call a topology with a small number of nodes (<10) as *simple*; otherwise we call it *complex*. Also, depending on the density of the topology (small/large number of links) it is called *sparse* or *dense* (per the standard graph-theoretic definitions). Hence, we have four different topology categories: simple and sparse, simple and dense, complex and sparse, complex and dense. Due to lack of space, here we present and analyze simulation results for the category of *complex and dense* topologies (which includes the hierarchical multi-homed and the USA telecommunication network topologies). Readers are referred to [11] for further details and for the results of other categories.

The evaluation of the three algorithms was based on different scenarios. The type and order of the faults were different in each simulated case so to cover different events that may occur at any time. All scenarios include multiple (two) faults. Here, we present and examine the following four scenarios:

Scenario 1: The first fault occurs on the working path and the second fault occurs on Gonfa's alternative path (which is different from the Otel's alternative path).
Scenario 2: The first fault occurs on the working path and the second fault occurs on Otel's alternative path (which is different from the Gonfa's alternative path).
Scenario 3: The first fault occurs on the working path and the second fault occurs on a common link for the two alternative paths (Otel and Gonfa).
Scenario 4: The first fault occurs on Gonfa's and Hybrid's pre-established alternative path and the second fault occurs on the working path.

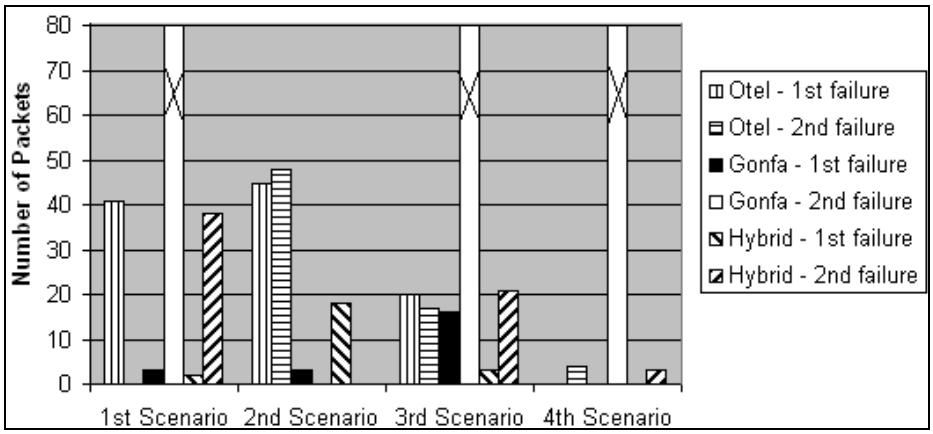

Figure 6. Packet loss in complex and dense topology.

Figure 6 presents the results for packet loss. The Hybrid algorithm is able to recover from faults with lower packet loss compared to the other two algorithms in all but one of the scenarios. The 1st and 3rd scenarios are the scenarios with the highest packet loss for the Hybrid algorithm. This is mainly due to the fact that the topology is complex and dense and the SPT is much larger, thus more time is needed to calculate an alternative path. As expected, the Gonfa algorithm fails to recover traffic from the failures where both working and alternative paths were affected. In those cases, enormous packet loss was observed (as the algorithm cannot deliver packets anymore), which is shown by a cross in the respective chart bar. Also high packet loss is observed for the Otel algorithm in the two cases where the faults occur both in the working and the alternative path and the algorithm must reroute the traffic twice.

In Figure 7 the number of re-ordered packets is shown. Generally, packet reordering is approximately the same for the three algorithms, in cases where all three algorithms are able to reroute the traffic. There is a small increase on packet reordering in the 3rd scenario in all algorithms. Once more we experience high packet reordering for the Gonfa algorithm in the 1st, 3rd and 4th scenarios due to its failure to recover from the second fault. In all cases the number of re-ordered packets for the hybrid algorithm is less or at least the same with the lowest numbers in the other two algorithms.

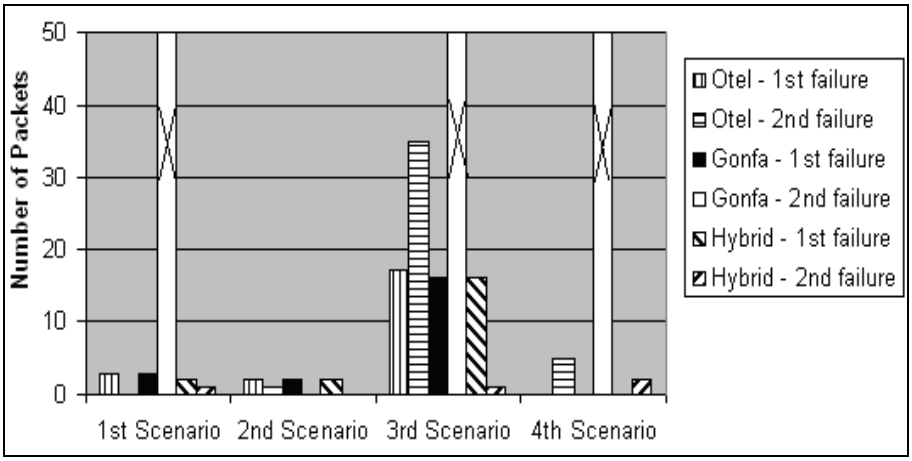

Figure 7. Packet reordering in complex and dense topology.

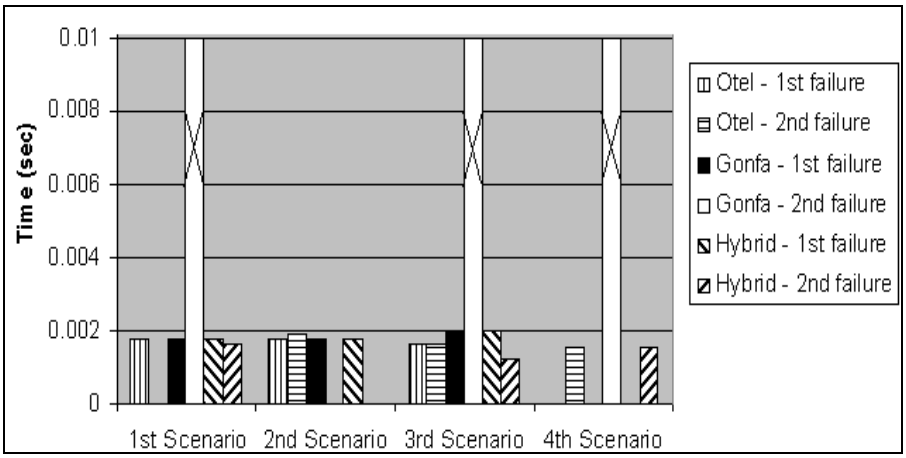

Figure 8. Fault recovery time in complex and dense topology.

The recovery time for each fault is shown in Figure 8. In the first scenario all three algorithms are able to recover from the first failure, as expected. In the second failure the Otel algorithm is not influenced and its recovery time is zero. In contrast, the hybrid and Gonfa algorithms are affected by the failure. The Hybrid algorithm is able to find an alternative path in 0,0016 seconds. However the Gonfa algorithm was not able to recover traffic from the failure (indicated by a cross in its chart bar).

In the second scenario it is observed that all algorithms are able to recover from the first failure with the same recovery time. The second failure only influences the Otel algorithm as the fault occurs on its alternative path; the algorithm is able to recover from the failure in 0,0019 seconds. The Gonfa and Hybrid algorithms are not affected by the fault and the recovery time is therefore zero.

Once again in the third scenario, all algorithms are able to recover from the first failure. As for the second fault, the Gonfa algorithm was not able to recover traffic from the failure. The Otel algorithm needs 0,0016 seconds to recover from the second failure whereas the Hybrid algorithm needs 0,0012 seconds.

In the last scenario of the complex and dense topology, the first fault does not influence the data flow of the algorithms as it is occurred on Gonfa's pre-established alternative path (i.e., the working path is unaffected). For this reason the recovery time for the first fault is zero for all three algorithms. As for the second fault, Gonfa fails to recover. On the contrary, both Otel and Hybrid are able to calculate dynamically an alternative path and switch the traffic in the alternative path in 0,0015 seconds.

Based on the obtained simulation results, we draw the following conclusions regarding the proposed Hybrid algorithm. First of all, each topology category gives, not surprisingly, different results for the same failure scenario. The network topology plays a significant role in the Hybrid algorithm (and Otel) due to the fact that the alterative path is calculated via the SPT, which basically represents the topology. Based on the obtained results a small increase on packet loss, packet reordering and the recovery time is observed while the network topology is becoming more complex and dense. This is due to the fact that when the SPT becomes bigger, it requires more time to be updated (for the computation of a new alternative path). Moreover it appears that the performance of the algorithm depends on two correlated factors: (i) the size of the SPT subtree, affected by the failure and (ii) the number of links originating from nodes outside the subtree but incident to subtree nodes.

Furthermore, two different SPTs can give different results in the same scenarios. This is derived from the different recovery times given by the Otel and Hybrid algorithms in the same scenario. The hybrid algorithm uses SSSP to update the SPT each time a fault occurs and does not affect the flow of the data. This procedure is not included in the Otel algorithm, hence different SPTs are developed for the two algorithms. As explained before, the structure of the SPT is a basic factor for the performance. Therefore different recovery times can be observed.

The Hybrid algorithm can approach the same recovery time as the Gonfa algorithm when the former is at the stage where it employs the protection switching technique. The cases where the Gonfa algorithm is called to restore the flow are when single and multiple faults occur in the working path and the alternative path is still available. In addition, these cases are considered to be the ideal cases for the Hybrid algorithm, since the protection switching technique provides fast restoration of the flow. Moreover, the hybrid algorithm can approach the same recovery time as of Otel's, in the case of repairing a fault using the rerouting technique. (The Otel algorithm is considered to be one of the best rerouting algorithms with respect to fault recovery time). Sometimes it is observed that in the same scenario the two algorithms may have different fault recovery times. The reason is that the two algorithms may develop different SPTs and this leads to different recovery times. Nevertheless, when the same SPT is developed for both algorithms the same recovery times is measured (as expected).

Per the simulation results, high packet loss is observed in the Hybrid algorithm when the Otel algorithm is called to restore the flow, especially when the topology is complex and dense. When the topology is simple, low packet loss is observed, since the SPT is simpler and requires less time to calculate an alternative path. The lowest

packet loss is given when the Gonfa algorithm is applied in order to reroute the traffic to a pre-established path.

As for the packet reordering, the simulations have shown that the number of packets received in out-of-order is relatively small in most scenarios. While the topology is becoming more complex and dense, a continued increase in packet reordering is observed, especially when the Otel algorithm is applied. When the Gonfa algorithm is followed packet reordering is low.

It is evident from the simulation results that the Hybrid algorithm can tolerate multiple faults in the working path as well as in the alternative paths (as it can employ dynamic rerouting) regardless of the topology category (and provided of course that an alternative path exists). To conclude, it appears that the Hybrid algorithm combines effectively the advantages of protection switching and dynamic rerouting restoration techniques and hence it is able to reroute the traffic as many times as the number of failures detected (if needed to do so).

5 Conclusions

The algorithm presented in this paper is the first hybrid algorithm that employs both protection switching and path rerouting restoration mechanisms in an effort to perform well in a number of important criteria. The Hybrid algorithm combines effectively both mechanisms and decreases the fault recovery time, reduces the packet loss and packet reordering in several cases (when compared with algorithms that employ only one of the two mechanisms) and supports multiple link and node failures both on the working and recovery paths.

For future work we plan to enhance the Hybrid algorithm with Quality of Service criteria in its path selection process. This will enable us to provide combined fault tolerance and traffic engineering in VC-based networks.

References

1. Afek, Y., Bremler-Barr, A., Kaplan, H., Cohen, E., Merritt, M.: Restoration by Path Concatenation: Fast Recovery of MPLS Paths. *In Proceedings of the twentieth annual ACM symposium on Principles of Distributed Computing*, pp. 43--52. (2001)
2. Ahn, G., Jang, J., Chun, W.: An Efficient Rerouting Scheme for MPLS-Based Recovery and Its Performance Evaluation. *Telecommunication Systems*, 19(3-4), pp. 481--495. (2002)
3. Bartos, R., Raman, M.: A Heuristic Approach to Service Restoration in MPLS Networks. *In Proceeding of ICC 2001*, pp. 117--121. (2001)
4. Bartos, R., Raman, M.: A Scheme for Fast Restoration in MPLS Networks. *In Proceedings of the Twelfth IASTED International Conference on Parallel and Distributed Computing Systems (PDSC),* pp. 488--493. November (2000).
5. Bartos, R., Raman, M., Gandhi, A.: New Approaches to Service Restoration in MPLS-Based networks. *In Proceedings of EUROCON 2001 International Conference on Trends in Communications*, pp. 58--61. (2001)

6. Capone, A., Fratta, L., Martignon, F.: Dynamic Routing of Bandwidth Guaranteed Connections in MPLS Networks. *Wireless and Optical Communications*, 1(1), pp. 75--86. (2003)
7. Chen, J., Chiou, C.C, Wu, S.L.: A Fast Path Recovery Mechanism for MPLS Networks. *In Proceeding of ICOIN 2005,* pp 58--65. (2005)
8. Chen T.M., Oh, T.H.: Reliable Services in MPLS. *IEEE Communications Magazine*, vol. 37, pp. 58--62. December (1999)
9. Fall, K., Varadhan, K.: The network simulator -*ns-2. The VINT project. UC Berkeley, LBL, USC/ISI, and Xerox PARC,* `http://www.isi.edu/nsnam/ns/`.
10. Gonfa, L.H.: *Enhanced Fast Rerouting Mechanisms for Protected Traffic in MPLS Networks.* Phd Thesis, Technical University of Catalonia, February (2003)
11. Hadjiona, M., Georgiou, Ch., Papa, M., Vassiliou, V.: *A Hybrid Fault-Tolerant Algorithm for MPLS Networks,* Technical Report TR-07-06, Department of Computer Science, University of Cyprus, December (2007), `http://www.cs.ucy.ac.cy/~chryssis/MPLS-TR.pdf`
12. Haskin, D., Krishnan, R.: A Method for Setting an Alternative Label Switched Paths to Handle Fast Reroute. Internet Draft (2000)
13. Hong, D.W., Hong, C.S.: A Rerouting Scheme with Dynamic Control of Restoration Scope for Survivable MPLS Network. *In Proceeding of ICOIN 2005*, pp. 233--243. (2005)
14. Kodialam, M., Lakshman, T. V.: Minimum Interference Routing with Applications to MPLS Traffic Engineering. *In Proceedings of IEEE Infocom*, pp. 884--893. (2000)
15. Kurose, J.F., Ross, K.W.: *Computer Networking – A Top-Down Approach Featuring the Internet,* 3rd Edition, Addison-Wesley, 2004.
16. Lin, J.W., Liu, H.Y.: An Efficient Fault-Tolerant Approach for MPLS Network Systems. *In proceedings of ISPA 2004*, pp. 815--824.(2004)
17. Makam, S., Sharma, V., Owens, K., Huang, C.: Protection/Restoration Mechanism for MPLS Networks. Internet Draft, (1999)
18. Menth, M., Milbrandt, J., Reifert, A.: Self-Protecting Multipaths - A Simple and Resource-Efficient Protection Switching Mechanism for MPLS Networks. *In Proceeding of IFIP-TC6 Networking Conference (Networking)*, (Athens, Greece), pp. 526--537. May (2004)
19. Otel, F.D.: On Fast Computing Bypass Tunnel Routes in MPLS-Based Local Restoration. *In proceeding of High Speed Networks and Multimedia Communications 5th IEEE International Conference*, pp. 234--238. February (2002)
20. Pu, J., Manning, E., Shoja, G.C.: Reliable Routing in MPLS Networks. *In Proceedings of Communications and Computer Networks*, (2002)
21. Rosen, E., Viswanathan, A., Callon, R.: *Multiprotocol Label Switching Architecture.* RFC 3031, January (2001)
22. Sharma, V., Hellstrand, F: *Framework for Multi-Protocol Label Switching (MPLS)-based Recovery*, RFC 3469, February 2003.
23. Yoon, S., Lee, H., Choi, D., Kim, Y., Lee, G., Lee, M.: An Efficient Recovery Mechanism for MPLS-based Protection LSP. *In Proceedings of Joint 4th IEEE International Conference on ATM (ICATM 2001) and High Speed Intelligent Internet Symposium*, pp. 75--79. April (2001)