# Real-Time Fault-Tolerant Routing in High-Availability Multicast-Aware Video Networks

Roman Messmer
ORF Austrian Broadcasting Corporation
Broadcast Production Systems
A-1136 Vienna, Austria
roman.meszmer@orf.at

Jörg Keller
FernUniversität in Hagen
Dept. of Mathematics and Computer Science
58084 Hagen, Germany
joerg.keller@fernuni-hagen.de

**Abstract:** Live-videostream networks based on multimedia switches are the most recent products used in television production and distribution facilities to transport the live signal from sources like cameras or microphones to dedicated sinks like video monitors, loudspeakers and transmission lines. To switch signals from a single source to several destinations multicasting or point-to-multipoint technology is considered. To compute multicast trees for multimedia communication, constrained shortest paths algorithms are needed. They are fundamental to important network functionality such as Quality of Service (QoS) routing or Multiprotocol label switching (MPLS) path selection and the problems they attempt to solve are known to be NP-complete. In previous work, we have introduced a heuristic called Multimedia Multicast algorithm (MulMic), which delivers nearly optimal multicast trees in a short time. Here, we propose the combination of MulMic and two models for fault-tolerant routing: disjoint paths and reservation of backup paths. Furthermore we introduce a realtime algorithm we call ZirkumFlex to handle one or even several simultaneous node or line failures in a multicast network by a local search to bypass the failed node or line. We also apply our algorithm to example graphs to demonstrate its feasibility.

## 1 Introduction

Implementing distributed media applications like high-resolution video routing on switched packet networks faces several obstacles. The most difficult but essential problem concerns ensuring quality of service (QoS). One precondition is knowing the constrained shortest path, i.e. the cheapest path that satisfies a number of constraints, especially the number of routing hops. Depending on the hardware each single hop adds up several microseconds to the total delay time of the signal path and leads to problems in terms of signal synchronization or even causes frame delayed videos which have to be processed together in follow-up working steps in the video and audio domain. The scale of the problem is better understood when the hop time of a switch is compared with the duration of a video line at the usual high definition video format 720p/60 which lasts about 23 microseconds. Most devices can cope with a time window of several lines without having to synchronize the input signal to an external clock and increase the total transfer time of the video signal additionally. Depending on the video product a delay of up to 200 microseconds is acceptable which makes the minimization of hops an important factor. In most cases one

source has to distribute a signal to a multitude of destinations, e.g. a video signal from a camera is transported to studio monitors and live video mixers at the same time. Thus, we have a multicast or point-to-multipoint routing problem with constraints. We will use the two terms interchangeably. In previous work, we have proposed an algorithm for this problem called MulMic that combines preprocessing with very fast path calculation upon connection requests [MK09].

The failure of nodes in switched packet networks often leads to unacceptable delays or total failure of running processes. Therefore, we investigate fault-tolerance measures in this scenario. We present three algorithms to tolerate node or link failures: (1) by allocating with every multicast tree a backup multicast tree on disjoint paths, (2) by reserving shareable backup trees and (3) introduce an online algorithm that on occurrence of a failure performs a graph search from surrounding graph nodes to bypass the failed node as quickly as possible.

To the best of our knowledge all previous work that investigates fault tolerance without extensive a priori resource consumption refer to point-to-point routing only. Of most interest in our paper is the avoidance of perceptible interruptions of the affected media signals with as little overhead as possible. This is what we mean by "real-time" in our scenario.

The remainder of this article is organized as follows. In Section 2 we discuss related work. In Section 3 we briefly review the MulMic algorithm. In Section 4 we present the various fault-tolerance algorithms, and in Section 5 we apply our algorithm to example graphs. We conclude in Section 6.

## 2 Related Work

In multicast-aware IT networks the fault tolerance aspect differs slightly from high speed media networks in terms of the importance of continuity of streams. Video streams may not be interrupted, frozen displays are very unwanted events. In [UZ06] a heuristic called *Adaptive distributed Concurrent Shortest Path heuristic* is presented which generates delay-aware constrained multicast trees. After removing the failed subtree the source node is informed about the former covered subtree and its destinations. Then a new loop-free subtree from (subtree-)source to destination nodes is established. This procedure delivers a nearly optimal routed multicast tree, but its calculation is very time-consuming. A very effective distributed algorithm was introduced in [Gu03], where parts of the network were analyzed for backup routes which could be used in case of a node failure. This approach extends the resource reservation protocol (RSVP, RFC2208) to improve resource utilization, higher call acceptance rate and better quality of service (QoS). The level of fault tolerance of each connection can be controlled separately. Earlier works as [Gr89], [Ch93a] and [Ch93b] used distributed algorithms to restore point-to-point connections after network failures. Their work does not mention point-to-multipoint connections. [ZS92] presented algorithms for realtime communication together with fault tolerance aspects using the multiple paths between a pair of communicating nodes to make more efficient use of the network. While most of the mentioned papers treat link failures, [Ko90] also an-

alyzes node failures. Communication for restoration bases on network flooding and path route monitoring. [RM03] analyzed multiple segmented backup scenarios for mission critical applications. In [YH88] failure immunization technology for the use in the fiber optic domain is presented and analyzed.

## 3 Realtime Routing Algorithm

In [MK09] we proposed an algorithm called Multimedia Multicast Algorithm (MulMic) which calculates a multicast tree in $O(|V| + |E| + |M| \cdot \log |M|)$ time where $V$, $E$ and $M$ represent the set of vertices, edges and multicast destination nodes, respectively. Experiments with different graphs demonstrated not only the favorable runtime of the algorithm but also another important property of the calculated routing. The result of a rerouting after eliminating a failed node is in many cases very similar to the original routing. So there are only few routing commands necessary to restore a multicast path.

In Fig. 1 this property is illustrated. A $10 \times 10$ grid network[1] was set up with 14 multicast destination nodes in different portions of the graph. Some are far apart, e.g. (10) or (36), and some form groups, such as (77, 78, 87, 88). Then a MulMic run was started with source node (55). The resulting routing to node (10) starts at (55), leading upwards to (5) and to the right to (10). For destination (91) the routing starts at (55) leading left to (51) and down to (91). Then a node failure at (7) and a line failure (61-71) were simulated. Failure (7) brought up a rerouting of the path to (10) using (15) to (18), then (8) to (10) to reach its destination. The new routing even leaves the routing from (5) to (6) untouched. A similar case appears with the path from (51) to (91). The new routing results in a path (52) to (72), (71) to (91) which leaves the majority of line routings untouched. Combined with the fault tolerance procedures in Section 4, most of the nodes affected by failure of a node can be supported by a new signal path without noticeable interruption. We now present the algorithm in detail. To achieve the necessary realtime behavior, the proposed algorithm is divided into a time-consuming offline part and a fast online part.

The offline part constructs a routing table for each node of the graph beforehand. The local routing table is then computed in parallel at each switch of the network. The data is updated by accessing a common database containing the local network topology information. The switches listen for node-state and line-state information from other links to update the local routing table and bandwidth consumption. After processing a point-to-multipoint connection request, the common database and the local routing information are updated synchronously. The Floyd-Warshall algorithm, an all-pairs shortest path algorithm with complexity $O(n3)$, can be applied to compute the distances and paths between all nodes of the network graph. In our example a cost of 1 is assigned to every link between routing nodes, because the constant major signal delay occurs during the opto-electronic conversion in the switches (several microseconds), not on the fibre-optic line (nanoseconds, less than a microsecond even for long distance links). After finishing the first part every router has knowledge about the network, e.g. the number of nodes and their connections.

[1] The grid topology is used throughout the paper for the sake of simplicity.

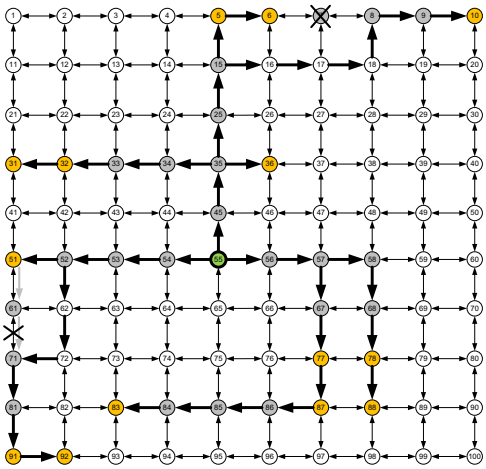

Figure 1: Property of MulMic routing algorithm

The online part starts with a request for a point-to-multipoint path calculation. Let $G = (V, E)$ be the graph representation of the network, $V$ the set of all nodes, $E$ the set of all edges, $M = \{v_1, \ldots, v_m\}$ the set of destination nodes, and $s$ the source node. The online part of the algorithm proceeds as follows:

1. Retrieve all distances $d(s, v_1), \ldots, d(s, v_m)$, i.e. the distances of all destinations from the source, from the routing table, together with the corresponding paths. and sort the distance values in ascending order. From now on we will assume $v_1, \ldots, v_m$ to represent this order, i.e. $v_1$ has the shortest distance from $s$, and $v_m$ has the largest distance from $s$.
2. Start the Multipoint path $MPATH$ with $MPATH = path(s, v_m)$. Let $i$ be a loop index counting down from $m = |M|$ to 2, to add node $v_{i-1}$ to the multicast tree.
   (a) If $dist(v_i, v_{i-1}) < dist(v_{i-1}, s)$
   then add path$(v_i, v_{i-1})$ to $MPATH$
   else add path$(s, v_{i-1})$ to $MPATH$. Information about the path length between $v_i$ to $v_{i-1}$ is retrieved from the local routing table.
   (b) (optional) If other destination nodes $v_j$, where $j < i - 1$, are already covered by the existing path, they are no longer considered for future calculation.
   (c) A variable which calculates the number of hops is increased when the if-case has been taken, i.e. if the shortest path to the next destination is shorter than to the source. The variable is reset to zero otherwise. If the maximum hop count is reached, the next destination node must be connected to the source. So an unacceptable delay time due to the accumulation of hop time is avoided. This way the routing propagation is limited.
3. As there is the possibility of loops in the constructed graph $MPATH$, a general spanning-tree algorithm realized by a depth-first search is used to eliminate them.

The implementation of this algorithm delivers a single point-to-multipoint routing result in record low time $O(|V| + |E| + |M| \cdot \log |M|)$. See [MK09] for a proof.

## 4 Fault-Tolerance Concepts

We now present three concepts to add fault tolerance to the routing method we mentioned above. The concepts differ in the amount of additional bandwidth that they allocate in the fault-free case, and the amount of calculation necessary in the presence of a fault.

### 4.1 Backup path model

This approach tries to establish redundancy by a backup multicast tree with paths disjoint from the multicast tree calculated by the MulMic algorithm, as far as is possible, since the graph is not required to be 2-connected. Its advantage is that in case of a failure, shifting transmission to the backup tree is all that needs to be done. This comes at the cost of doubling the bandwidth consumption. First the primary routing path is to be calculated. The MulMic algorithm from Section 3 is used for realtime calculation of the point-to-multipoint path. The calculated path is transformed into routing commands, which are then sent to the routers along the multicast path. Next the backup routing path is calculated. A graph with modified edge cost is used to calculate disjoint paths between source and destination. Links used in step 2 get a high cost value (e.g. edge cost set to number of total graph edges), so that a shortest-path algorithm calculates paths disjoint from the primary path whenever possible. If there is no disjoint path alternative, the original path must be used to transport the media stream. At least the source router as well as the destination routers are always examples of such non-disjoint paths if they don't provide multiple network interfaces. The calculated secondary path is then transformed into routing commands and sent to the routers. After positive switch confirmation the bandwidth management (BM) is called to update the database. Next we concern the failure management of the backup path model. Immediately (usually within microseconds) after a router or a link failure the destination nodes detect an error[2], the redundant secondary channel immediately is being activated for resuming the transfer of media over the backup path. The defective lines or nodes of the graph are removed from the routing table (locally and in the remote database) and an all-pairs shortest path algorithm with complexity $O(n3)$ is called again afterwards, i.e. offline, to update the topological information base for the multicast algorithm. To select disjoint paths we recommended to set the link cost of the primary path to a certain high value. This is also used to recalculate a connected primary path. The algorithm presented in section 3 is appropriate because in most cases it leaves the already working routed links in place and only presents backup routings for the defective line or node.

[2]Physical detection via Loss Of Light (LOL) or CRC error.

### 4.2 Reservation model

Instead of actually establishing backup paths one can also simply reserve such paths by means of Resource ReSerVation Protocol (RSVP) [RS06]. The advantage of this variant is that backup path reservations can be shared between trees that use disjoint links that never will both be affected by a node or link failure. The disadvantage is the overhead to establish the reserved backup path in case of a failure. The network design which uses a number $n$ of backup paths for $m$ signal paths is called $n : m$ p-cycle and has been studied extensively.

There are concepts like shared link risk groups (SLRG) [Ru05] which consider a set of links that may fail simultaneously because of a common risk they share on the network. For example, they may all be less prioritized destinations. In [RC08] a dual link failure resiliency is considered in a project called backup link mutual exclusion (BLME). P-Cycle in multidomain networks are surveyed in [Dr09].

### 4.3 Online failure management model

This proposal does not assume any additional reserved or switched redundancy path. Instead it assumes that there is additional bandwidth available in the network that can be used for a failure-triggered routing. The basic idea of this model which we will call *ZirkumFlex* is a combination of the MulMic routing algorithm together with a rapid calculation of a path alternative in case of a node or link failure. The alternative path changes only those routes directly affected by the failure.

In Fig. 2 there is an example for the ZirkumFlex algorithm operating in a common graph. The upper image shows a part of the network graph, the lower the multicast tree only. The node $s$ represents the source node, routing starts from there. Destination nodes $d_1, \ldots, d_6$ are in dark gray. The figure shows the moment when a failure occurs at the node with the flash symbol. Because the multicast tree is loop-free per definition there is only one single predecessor of any node in the tree. We begin at the already established multicast path. An additional feature of the MulMic algorithm sets the variable for the distance from the source node $d$ in each node without extra cost in a depth-first search. Every node holds its number/name, distance $d$, the multicast call number (to distinguish between several multicast routings per node) and finally a linked list with its source name/number initially filled in plus a rendezvous field containing encountered successor names. A node list holds node names not yet connected to the predecessor.

After a failure has been detected the predecessor and the successors of the failed node(s) is/are determined. A following test of the predecessor (pre) node being a member of a unicast or multicast routing determines if rerouting activity is to be set. If the pre node is not a member, only the central database is to be informed about the failure and no other action is taken. Else if the local node indeed is a member, start the ZirkumFlex-algorithm detailed in Fig. 4. To achieve link and path restoration on large graphs in short time and avoiding the $O(n^3)$ all pairs shortest path algorithm over the whole graph we propose

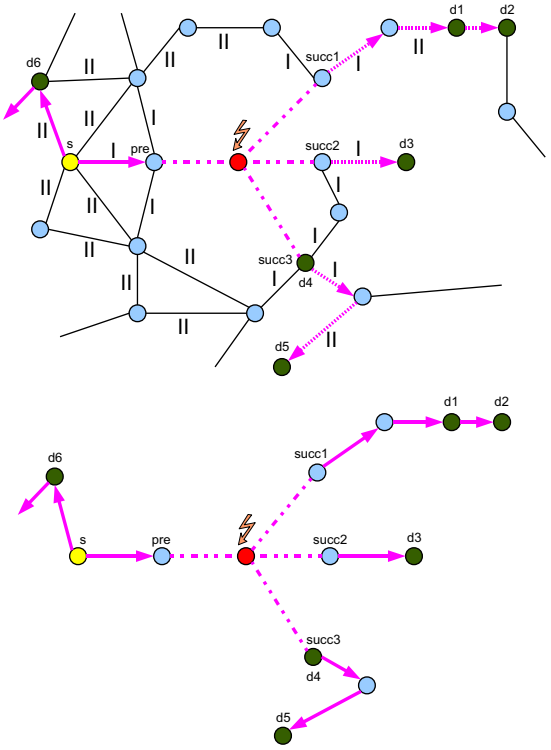


Figure 2: Heuristics example: standard graph and multicast-path

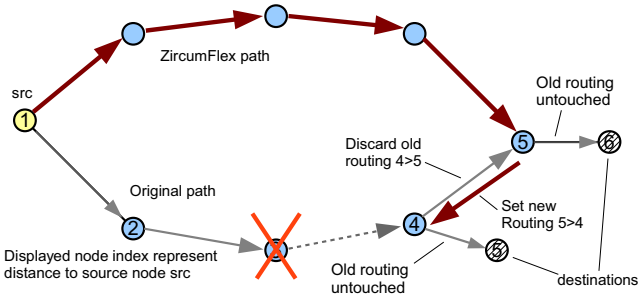


Figure 3: Online Failure Graph

using local graph breadth first searches (BFS) running only on the predecessor node of the failed node/link. As long as there are unreached successors do the following:

- do a simple BFS-run (only direct neighbors) and mark the linked list of the reached

nodes with the path to the origin, i.e. the first detected node gets the predecessor or successor node name in its linked list, respectively. In the next run there are two entries in the list and so on.

- if a previously predecessor-BFS-marked node is reached, the path from the predecessor to the affected successor is taken into an emerging *regional graph*[3]. Then the successor node is marked as reached. Running BFSs of known successor nodes are terminated, their corresponding node names are taken out of the node list. Already visited nodes/edges are then excluded from further BFS-runs.
- if another successor-BFS-marked node is reached the path between both successors is taken into the regional graph again. Already visited nodes/edges are again excluded from further BFS-runs.

This procedure puts up a local subgraph of the network graph (regional graph) and delivers a spanning tree containing shortest paths between successors and predecessor within the regional graph in a few microseconds, therefore we speak of real-time property. Our preliminary tests indicate that only a few BFS steps are necessary in general graphs.

ZirkumFlex allows to eliminate links of pathological (after a failure no longer signal-supported) nodes by iteratively starting at predecessor and successor nodes and eliminating the connected edges as long as the outdeg of the referring nodes equals one[4] or until a multicast-destination is reached. A following DFS-run within the regional graph starting at the pre node sets the correct routing direction in the regional graph.

In some network layouts there appears the effect of the signal path having to be routed forward and backward for a minor number of hops, which consumes unnecessary bandwidth to support nodes which lie closer to the failed node than the bypass. Figure 3 shows the details. This problem can easily be solved by utilizing the already mentioned source-distance information gathered during the MulMic-DFS-run. Routing from a higher node number to a lower one means that the old routing has to be terminated and a reverse link direction has to be applied for this graph edge when applying the new routing. The major advantage for this model is that the successor node and therefore the destination nodes get the original signal as quickly as technically possible and the not directly affected nodes are not disconnected for rerouting purposes.

## 5 Example Graphs

As an example of the proposed algorithm, Fig. 5 shows a relevant part of a large network graph (hundreds of nodes) and the resulting regional graph with predecessor, failed node and some multicast destinations. In this case (grid-topology) one BFS step (12 links affected) only is necessary to get the whole regional graph. This rerouting can be done in a few microseconds.

[3] A simple concatenation of the reached node's path lists (linked list) and the node-BFS's path lists in inverted visit order delivers the path.

[4] as there is no forking in the network which could be necessary to support another destination node

```
// pre: predecessor node, n: number of successor nodes
// succ(i): i th successor nodes, i=1 to n
// LinkedList PL for the path list information in every node
// nodelist for remaining successor nodes

procedure LBFS (Graph G, Node nod) //LBFS = Labeling BFS

 do a BFS starting at nod; // do only one level of stepping down
 For all nodes discovered during BFS do
   if node has been reached by an already pre-node-derived search
     then build part of MPath by concatenating the meeting PLs.
     // delivers a shortest path from pre to current succ
     Lookup nodes in rendezvous list, terminate their BFS and
     delete them from nodelist.
     Ignore already marked links. If there is no more
     unmarked link available, stop the affected BFS-run.

   elseif reached node is already marked by another succ node then
     build part of MPath by concatenating the meeting PLs.
     Set rendezvous information (local succ-name) in each other node.
     Ignore already marked links. If there is no more unmarked link
     available, stop the affected BFS-run.

   else
       add the already reached nodename(s) to the linked list PL of
       the reached node to setup the routing information.
 od
end LBFS

algorithm online_rerouting(Graph G, Node failed_node, Node pre,
      Node[] succ(i), Integer n) // main algorithm

 remove failed_node from G;
 while nodelist != empty  do
   LBFS(G, pre);  //iterate one step of BFS started from the pre node
   while there are successors j do
        LBFS(G, succ(j));
   od
 od

 // remove pathological edges
 For all succ's and pre do
    remove nodes in direction of old multicast path (in opposite
    direction for pre node) until following node has outdeg != 1
    or until the following node is a destination node.

 Start DFS over MPath to eliminate loops, correct routing directions
 //correct loops
    If the routing for a link would be issued against a falling
    number pair (di, di+1) (i.e. old routing direction was in opp.
dir.)
    then cancel old routing, establish new connection in new routing
dir.

 Calculate routing commands with information gathered in the
linkedLists
 consider that no command necessary for an already switched link
end online_rerouting
```

Figure 4: Pseudo code for ZirkumFlex algorithm

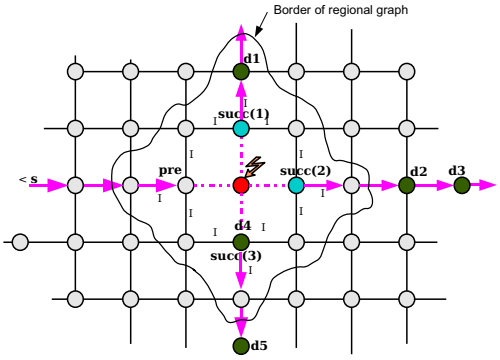


Figure 5: Heuristics example, part of grid graph

The proposed algorithm also handles several simultaneous node or line failures in one pass. In Fig. 6 a grid graph with two simultaneous failures, e.g. a failure of a shared power-supply, is shown. Along the multicast tree path starting at the source node marked "src" there is only one predecessor which is marked with "pre". The possible successor nodes are marked with $d_1, ..., d_3$. Signalling the defective nodes to the predecessor node for instance could be done by a broadcast initiated by the node that detects a loss of light and has been chosen for a multicast route.

In this example, there are three successor nodes visible. It is easy to see that only two BFS steps are needed to connect all of the successor nodes. After the first run which touches 12 nodes there are another runs until all of the common nodes are found. In this example the resulting backup graph is optimal. It follows the shortest path around the failure nodes. The rest of the potential multicast destination paths are untouched which is vital to the overall system performance.

## 6 Conclusion

We have investigated fault-tolerance measures in switched networks for video multicast. The backup path model combining the MulMic algorithm and choosing disjoint paths gives a very robust solution for multicast routing in multimedia applications. In all examples investigated, node failures did not lead to any noticeable error. Disadvantages of this approach are the immense use of resources as well as a communication overhead. Any routing needs to set up nearly twice the needed nodes and lines to be used. Also if there are two failures in neighboring nodes that are used for disjoint paths there will be a total failure of the signal. A more efficient way is the use of the RSVP-protocol to reserve a disjoint path. The reserved paths can be used as a backup of other nodes as well. If a failure occurs, then all other affected reserved paths have to be recalculated. This leads to a considerable

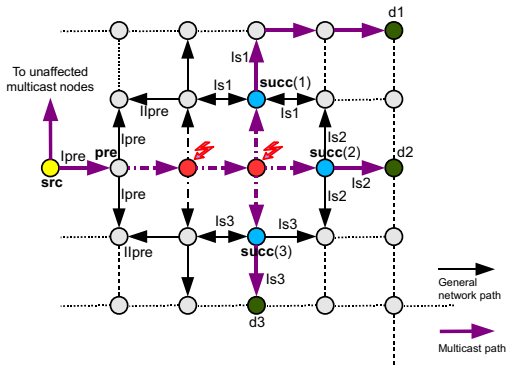


Figure 6: Proposed heuristic handling two simultaneous failures

amount of network communication. Simultaneous failures of two neighboring nodes used for disjoint paths lead to a sudden interruption of the routed multimedia signal as in the backup path model. After calculating and switching another routing the signal can be restored again.

Our proposed *ZirkumFlex* algorithm overcomes this problem. There is no redundant path and no communication overhead. After a failure of one or more neighboring nodes a recalculation of the multicast path is done automatically and in a very short time. Because of using only a so-called regional graph of limited size this algorithm also works very well for large network graphs.

## References

[Ch93a] C.-H. E. Chow et al.: A Fast Distributed Network Restoration Algorithm. Proceedings 12th International Phoenix Conference on Computers and Communications, pp. 261–267, 1993

[Ch93b] C.-H. E. Chow et al.: RREACT: A Distributed Protocol for Rapid Restoration of Active Communication Trunks. Proceedings 2nd Network Management and Control Workshop, pp. 391–406, 1993

[Dr09] H. Drid et al.: A Topology Aggregation Model for Survivability in Multi-Domain Optical Networks Using p-Cycles. NPC '09: Proceedings of the 2009 Sixth IFIP International Conference on Network and Parallel Computing, pp. 211–218, 2009

[Gr89] W.D. Grover: SELFHEALING NETWORKS: A Distributed Algorithm for k-shortest link-disjoint paths in a multi-graph with applications in real time network restoration. Doctoral Dissertation, Department of Electrical Engineering, University of Alberta, Fall 1989

[Gu03] P.K. Gummadi et al.: An efficient primary-segmented backup scheme for dependable real-time communication in multihop networks. IEEE/ACM Trans. Netw., vol. 11, no.1, pp. 81–94, Feb. 2003

[Ko90] H. Komine et al.: A distributed restoration algorithm for multiple-link and node failures of transport networks. Proceedings GlobalCom '90, 403.4.1403.4.5, 1990

[MK09] R. Messmer and J. Keller: Real-Time Computation of Point-To-Multipoint Routes in Optical Networks for Digital Television. Proceedings Int. Conference on Parallel and Distributed Computing and Systems 2009, Nov. 2009

[RC08] S. Ramasubramanian and A. Chandak: Dual-link failure resiliency through backup link mutual exclusion. IEEE/ACM Trans. Netw., vol. 16, no. 1, pp. 157–169, 2008

[RM03] G. Ranjith and C. Siva Ram Murthy: A Multiple Segmented Backups Scheme for Dependable Real-Time Communication in Multihop Networks. Proceedings International Parallel and Distributed Processing Symposium, pp. 121b, 2003

[RS06] M. Karsten: Collected experience from implementing RSVP. IEEE/ACM Trans. Netw., vol. 14, no. 4, pp. 767–778, 2006

[Ru05] C. Ruan et al.: p-cycle design in survivable WDM networks with shared risk link groups (SRLGs). Proceedings 5th International Workshop on Design of Reliable Communication Networks, 2005

[UZ06] H. Ural and K. Zhu: Distributed delay constrained multicast routing algorithm with efficient fault recovery. Networks vol. 47, no. 1 , pp. 37–51, 2006

[YH88] C. H. Yang and S. Hasegawa: FITNESS: Failure Immunization Technology for Network Service Survivability. Proceedings GlobalCom '88, 47.3.1-47.3.6, 1988

[ZS92] Q. Zheng and K.G. Shin: Fault-tolerant real-time communication in distributed computing systems. Proceedings IEEE Fault-Tolerant Computing Symposium, pp. 86–93, 1992