Generic Platform for Failure Recovery in Survivable Trees

V. Dynda

Failure recovery is a fundamental task of the dependable systems needed to achieve fault-tolerant communications, smooth operation of system components and a comfortable user interface. Tree topologies are fragile, yet they are quite popular structures in computer systems. The term survivable tree denotes the capability of the tree network to deliver messages even in the presence of failures. In this paper, we analyze the characteristics of large-scale overlay survivable trees and identify the requirements for general-purpose failure recovery mechanisms in such an environment. We outline a generic failure recovery platform for preplanned tree restoration which meets those requirements, and we focus primarily on its completeness and correctness properties. The platform is based on bypass rings and it uses a bypass routing algorithm to ensure completeness, and specialized leader election to guarantee correctness. The platform supports multiple, on-line and on-the-fly recovery, provides an optional level of fault-tolerance, protection selectivity and optimization capability. It is independent of the the protected tree type (regarding traffic direction, number of sources, etc.) and forms a basis for application-specific fragment reconnection.

Keywords: fault tolerance, failure recovery, tree restoration, distributed algorithms.

1 Introduction

Failure recovery is a fundamental task of the dependable systems needed to achieve fault-tolerant communications, smooth operation of system components and a comfortable user interface. Increasingly popular distributed applications providing data sharing, content distribution or stream data delivery services include many different computers, often at distant geographical locations. To communicate between their nodes, these applications build tree-topology overlay structures to connect the nodes and distribute information.

The failure recovery schemes for overlay trees use the underlying network to build a completely new tree or to restore the tree keeping its original structure. While delay-prone creation of a new tree from scratch is usually possible using the same technique as for creation of the original tree, local tree restoration keeping the rest of the tree intact is a relatively unexplored area of research. Moreover, the large-scale and dynamic nature of tree-based structures in the rapidly evolving area of overlay communications requires the recovery mechanisms to exhibit several key properties such as scalability, independence of location and number of message sources, optional level of fault-tolerance and support for application-specific tree optimization requirements. It is becoming increasingly apparent that a generic failure recovery platform providing a fragment-location and reconnection framework with these properties would be beneficial for many emerging applications.

The problem of the failure recovery of overlay trees considers graph $S = (\mathcal{N}, \mathcal{L})$ of arbitrary topology, where $\mathcal{N}$ is a finite set of vertices representing nodes and $\mathcal{L}$ is a finite set of edges representing links between the nodes. Graph S acts as an underlying network for a tree-topology overlay network modeled as graph $T = (\mathcal{TM}, \mathcal{CE})$, where $\mathcal{TM} \subset \mathcal{N}$ represents tree nodes, $\mathcal{CE}$ is a finite set of core tree edges representing overlay communication links connecting individual nodes $\mathcal{TM}$. The goal of failure recovery is to protect a given tree network T against failure of the faulty cluster $\mathcal{FC} \subset \mathcal{TM}$ of one or more adjacent nodes in the tree (see Fig. 1). Its task is to locate the tree fragments caused by the failure to restore the distributed knowledge of the topology and reconnect the tree, omitting the failed nodes, to enable communication in the tree to continue.

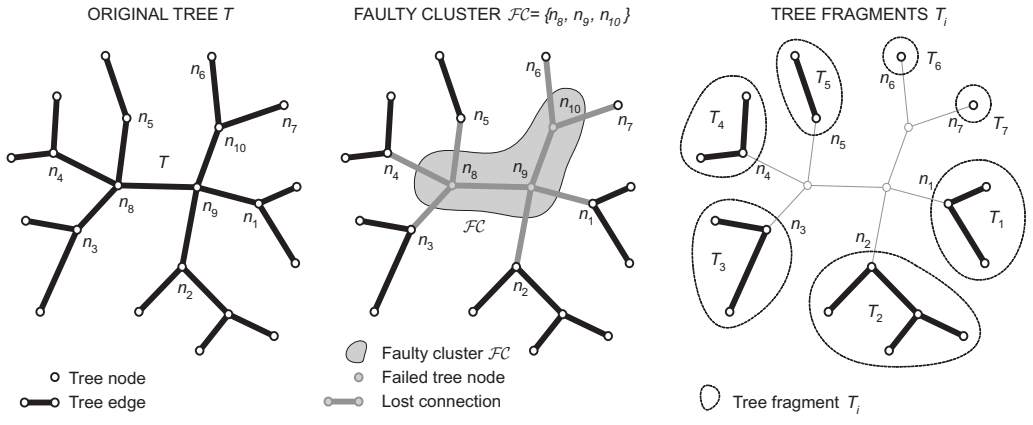

Fig. 1: Failure in the tree network and its partitioning into fragments

In this paper, we analyze the environment of general overlay tree networks and identify the requirements for failure recovery in survivable trees ([1]). We outline a failure recovery platform for preplanned tree restoration based on bypass rings ([2], [3]) – cyclical redundant structures to be used in the event of failure to locate and reconnect the tree fragments. The rest of this paper is structured as follows. In the next section, the related work is summarized. Sections 3 and 4 summarize the requirements for generic failure recovery and relate the qualities that the corresponding recovery scheme is expected to possess. Section 5 briefly describes the proposed platform, and sections 6 and 7 deal with two main issues of recovery – completeness and correctness. Section 8 outlines the elementary possibilities of application-specific fragment reconnection based on the platform. Section 9 discusses the achieved results, and section 10 concludes and sets some future directions.

2 Related work

Reconstruction of tree-topology graphs without starting from scratch is a relatively new area of research in the field of failure recovery. So far, on-demand restoration schemes building at least the affected subtree anew have been used in a number of applications. Several preplanned special-purpose protocols based on pre-computed backup paths have also been proposed, aiming at some of the mentioned properties while neglecting others. Although there are many possible applications, nearly all of the previous solutions are designed for specific network-layer or overlay multicast schemes. The important property of local recovery is mostly achieved only in single-source multicast trees, and the scalability of many solutions is limited by dependence of the control or memory overhead on the group size k_N.

There are several basic straightforward preplanned methods for multicast tree recovery (based on the group leave operation in [4]). In the *Grandfather* method, each node maintains a backup link to its grandparent in the rooted tree. When a node (except the root) fails, its child nodes contact the grandparent, which either accepts the connection or redirects them down the tree. Subtrees of the affected nodes remain unchanged. In the *Root* method, the children of the failed node try to recover the tree by connecting directly to the root node, which uses the same strategy as in the Grandfather approach above. In the *Grandfather-All* and *Root-All* methods, all descendants of the failed node try to recover by contacting the grandparent or root, respectively, in order to build the whole affected subtree of the failed node anew using the requested optimizations.

All the nodes maintain respective knowledge of their ascendants in the multicast tree – the grandparent node in Grandfather, the root node in Root and Root-All, and all ascendants from grandparent of n to root node in the Grandfather-All method. Except for the Grandfather, the methods do not perform local failure recovery, as the affected nodes contact ascending nodes far up the tree. The scalability of these methods is limited either – in the Grandfather-All method, each node maintains a link to the number of ascendants proportional to the size of the group. The Root and Root-All methods also load the root node with extensive communication proportional to the group size. These two methods also rely on a single root node whose failure breaks down the whole scheme. The Grandfather method is scalable and keeps locality, but it does not cope with multiple adjacent failures in the tree. Moreover, all these methods are designed for single rooted directed multicast trees only. However, these simple methods represent four classes of a number of other approaches based on the same principles.

For example, *Proactive Reconstruction* [5] belongs to the Grandfather-All category, *EFTMRP* [6] for recovery in network-layer CBT multicast uses a principle similar to the Grandfather method, LFR *core recovery* [7] resembles the Root method.

A different approach is chosen in specialized multicast protocols that use administrative control topologies for group management in addition to the data delivery tree. *Narada* [8] is a protocol designed for small multicast groups, where each node keeps a periodically refreshed state about all other group members and uses this information to locate and reconnect the fragments. Due to the state exchange inducing the control overhead $\mathcal{O}(k_N^2)$, the Narada protocol is effective only when the group is small. However, this is an explicit design choice where the high overhead is traded off for greater robustness in recovery from node failures. *Yoid* [9] and *HMTP* [10] are examples of protocols using a dedicated node called the *rendezvous point* to arbitrate the failures and locate the fragments. In addition, cached links to several periodically discovered member nodes are used. HMTP nodes also maintain an ancestor list similarly to the Grandfather-All method. These methods are capable of recovery from large failed clusters in their trees; however, the rendezvous point may become a bottleneck and a single point of failure.

Another solution is used in overlay protocols for media streaming *Nemo* [11], *Nice* [12], *FatNemo* [13] and *ZigZag* [14]. They all first construct an administrative highly connected hierarchy among the nodes, and the data delivery tree is then built using this structure. The hierarchy is organized into layers divided into clusters, where each cluster has a leader node which then also belongs to a cluster in a higher level. When a node fails, the leader of its highest layer or the leader of the cluster of its children (depending on the protocol) is responsible for finding another node to take over the traffic and reconnect the disconnected subtrees. After the recovery, the administrative hierarchy is reorganized to adapt to the topology changes. The failure recovery of these protocols is efficient in heavy traffic multicasting, where the costs of the highly connected hierarchy are amortized by the huge amount of data. On the other hand, for less loaded trees, the memory and control traffic overhead may be significant. The data delivery trees are source-specific, the control overhead of a node is $\mathcal{O}(\log_k k_N)$, where k is a constant proportional to the size of the administrative clusters.

A *Dual-tree* network-layer protection scheme [15] constructs a node-disjoint secondary tree connecting tree leaves in addition to the primary delivery tree. After failure, the scheme identifies disconnected subtrees and reconnects them to the rest of the tree using the secondary tree. The construction of redundant trees increasing network-layer multicast reliability is also studied in [16], and it is employed in overlay multicast as well in *CoopNet* [17]. *Link-protection* [18] and *path-protection* [19] for network-layer multicasting in ATM

networks propose an individual backup path for each link in the tree and for each source-destination pair, respectively.

3 Main issues of overlay failure recovery

From the point of view of the design of failure recovery mechanisms for survivable networks, the characteristics of the environment as well as the properties of the protected tree networks and the group model are essential. Recent trends in a networked computing environment point towards large-scale unbounded network infrastructures with theoretically unlimited number and size of groups interconnected with overlay trees.

To design a practical and efficient failure recovery scheme for these systems, the following characteristics of the computing environment must be taken into account:

- *Characteristics of a distributed system* – asynchronous communication, no global clock and autonomous behavior of the nodes.
- *No central authority* controlling proper functionality of the system; all nodes are peers.
- *No global knowledge* of the number of nodes, their identities and the network topology.
- *Unrestricted failure pattern*; nodes fail arbitrarily at any time.
- *Unlimited size of the system* and the underlying network S.

The most significant attributes of the overlay tree structures involve:

- *Traffic direction*. In single-source trees, the message traffic flows in a single direction from the root (source) node to other member nodes. However, in many emergent applications, the message source may change in the runtime or even several nodes become a message source simultaneously disseminating the traffic to the tree.
- *Tree adaptation*. As individual nodes may arbitrarily join or leave the group, the respective tree is either expanded or shrunk. Moreover, the tree may adapt its topology in order to satisfy potential extern optimization requirements.
- *No global knowledge* and *unlimited size of the tree*. The size of the group is not limited, and the number and identity of the tree member nodes is not fully known. Each tree member keeps only the information about its neighbors in the tree for routing and possibly for tree adaptation purposes.
- *Real-time operation*. The tree operates in real time, such that the traffic in the tree cannot be suddenly turned off or switched to an off-line or stalled mode.
- *Self-containment*. Due to the scale of the system, the possible number of groups and the overlay nature of the trees, the information pertaining to a particular tree is required to be kept solely at the tree member nodes, and no other node is capable to hold even auxiliary information concerning this tree.

It shapes up that a generic platform for failure recovery available for different applications in this environment would be profitable perhaps as a part of the middleware architecture to increase reliability and cut down the costs. The listed attributes represent the restrictive characteristics of the environment and the protected tree. Of course, not all applications employing tree communication structures employ this kind of trees, and the characteristics are somehow relaxed. However, in many other applications, particularly large-scale data sharing or data storing peer-to-peer systems, the tree networks exhibit all these attributes, which then must be reflected by the respective properties of the platform.

4 Survivable trees

A survivable tree is a general tree-topology communication network capable to deliver information to all its correct member nodes even in the presence of failures. Consider $T = (\mathcal{TM}, \mathcal{CE})$ to be an overlay tree network, $\mathcal{FC} \subset \mathcal{TM}$ and $\langle \mathcal{FC} \rangle_T$ to be a connected vertex-induced subgraph of T. Failure of faulty cluster $\mathcal{FC}$ causes T to be partitioned into fragments

$$T_i,\ i = 1, 2, \ldots, \mathrm{N} = card\,(A_T(\mathcal{FC})),$$

where $card\,(A_T(\mathcal{FC}))$ is cardinality of a neighbor set of $\mathcal{FC}$ in T. If $card\,(\mathcal{FC}) = 1$ then $\mathcal{FC}$ is called a *single failure*; if $card\,(\mathcal{FC}) > 1$ then $\mathcal{FC}$ is a *multiple failure* of adjacent nodes in T.

A survivable tree T is required to deliver messages even in the event of several single or multiple failures. A failure recovery is a process of reconnecting fragments T_i in a single restored network $T' = (\mathcal{TM} \setminus \mathcal{FC}, \mathcal{CE}')$, allowing the traffic to continue. We focus on two principal properties that each recovery mechanism employed in a survivable tree must have – *correctness* and *completeness*. Correctness is based on the essential requirement to keep the tree topology of the network even after failure, since the correctness of most applications depends precisely on the acyclic property of the graph. Completeness requires all the fragments of the original tree to be connected in a single restored tree, allowing all correct nodes to participate in T'.

The following extra qualities of the recovery scheme are needed to address the characteristics of the large-scale unbounded environments of the targeted applications and properties of the possible protected trees, and should form the design subject of failure recovery in survivable trees.

- *Scalability* with the size of the protected tree and the underlying network.
- *Multiple failure recovery*. Capability to recover T from multiple failures.
- *Locality*. The recovery affects only the tree nodes in the closest neighborhood of the failed nodes, keeping the rest of the tree in its original structure.
- *On-line recovery*. Ability to recovery several simultaneous failures in a single tree.
- *Computational symmetry*. There is no arbiter node, all $\mathcal{TM}$ nodes are peers.
- *On-the-fly recovery*. The failure recovery is performed while the traffic in the tree goes on, even through the nodes performing the recovery.
- *Optional level of fault-tolerance* provided by the scheme for the survivable tree, allowing an optimal trade-off between survivability and costs to be found.
- *Protection selectivity* allows the fault-tolerance level to be chosen individually for each node. The survivable tree may then provide stronger protection against failure of less reliable or functionally more important nodes in the tree.

- *Traffic direction independence*. The recovery success depends neither on the traffic direction in the tree nor on the link orientation in multi-source multi-rooted trees.
- *Optimization capability*. The scheme takes into account the application-specific requirements regarding the restored tree.

There are only three methods capable of failure recovery in a single multi-source tree – Yoid [9], HMTP [10] and network-layer link-protection [18]. However, link-protection recovers the network from link failures only, and Yoid and HMTP are not fully distributed, as they need the RP node for their operation. Other schemes are designed either for single-source trees or for single-rooted shared trees, and they usually do not provide optional failure recovery and protection selectivity. Moreover, the overhead of several recovery methods is proportional to the group size, which degrades their scalability. Local recovery is performed only in the Grandfather type of restoration and in implicit multicast schemes (e.g., Nemo [11], Nice [12], ZigZag [14]).

5 Bypass ring platform

When designing a failure recovery scheme for a survivable tree, we face two main challenges to be solved while keeping in mind the requirements for failure recovery in survivable trees:

- How to locate all the fragments and route messages among them
- How to avoid creating cycles during fragment reconnection

Our solution is based on *bypass rings* – virtual cyclic structures appended to the tree and providing alternative paths to eliminate the failed nodes, locate the fragments and reroute the traffic in the tree ([3]). Each bypass ring is identified by its center node and diameter; its edges connect individual tree branches of the center node in a distance proportional to the diameter. Several concentric bypass rings of increasing diameter form a *bypass framework*.

It is the responsibility of the *bypass routing algorithm* to locate all $N = A_T(\mathcal{FC})$ nodes and route among them cyclically in a uniform direction and order, regardless of the source and destination of the messages, using edges of bypass frameworks. This cyclical path bypassing cluster $\mathcal{FC}$ is referred to as *bypass cycle* $BC(\mathcal{FC})$. The bypass cycle connects all N fragments of the tree so that they can communicate and join together to restore the tree. However, it is not possible to sequentially join all the fragments on $BC(\mathcal{FC})$ since a cycle would occur in T'. Instead, a single bypass edge on $BC(\mathcal{FC})$ that does not participate in the reconnection is to be identified in a distributed way. This is the task of the *leader link election* (LLE) process, which is based on comparing the *hierarchical identifiers* of the fragments ([2]). A hierarchical identifier is a unique compound value inferred from the structure of the failed cluster found out by the routing algorithm.

The overall operation of the scheme involves several fundamental steps – *scheme initialization*, *failure detection*, *designated nodes discovery*, *leader link election*, *tree fragment reconnection* and *bypass ring reconfiguration*. In the initialization phase, the bypass frameworks are set up centered at selected tree nodes against whose failure the tree is to be protected, with the diameter depending on the desired protection level of the particular node. In the event of failure, the failure detecting nodes initiate the recovery immediately and use the bypass routing to discover designated nodes, $DN(\mathcal{FC}) \subset A_T(\mathcal{FC})$ – the bypass cycle nodes with distinct properties related to the type of protected tree allowing them to coordinate the rest of the recovery process. Bypass cycle edges incident to the designated nodes become candidates for the LLE process. As the leader election proceeds along the bypass cycle, the pairs of fragments are successively joined together, forming greater connected components until all the fragments are reconnected into the restored tree T'.

Various fragment reconnection methods respecting the results of LLE can be designed to consider application-specific constraints and requirements regarding the tree properties (e.g., degree constraints, weight functions, latency or bandwidth limitations, etc.).

6 Bypass routing

Bypass routing is one of the key components of the proposed scheme, as it ensures its completeness. Clearly, the success of the routing depends on the availability of the respective bypass rings in the event of failure. To achieve uniform direction and fragment order of the routing, the bypass rings are to be set up systematically in the tree. For this purpose, we introduce the *partial order* of the tree, which unambiguously specifies the sequence of neighbors $Seq_T(n)$ of each tree member node n (e.g., according to their identifiers).

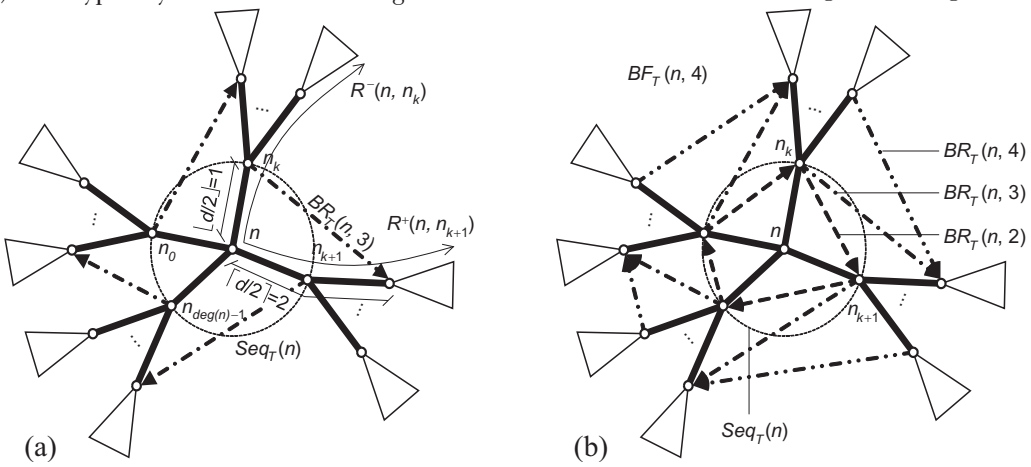

Fig. 2: Bypass ring $BR_T(n, 3)$ (a) and framework $BF_T(n, 4)$ (b)

The partial order defines the arrangement of each bypass ring in the tree.

Each bypass ring is referred to as $BR_T(n, d)$, where n is its *center node* and d is its *diameter*. $BR_T(n, d)$ consists of $\deg_T(n)$ bypass edges connecting each pair of tree branches $B_T(n, n_k)$ and $B_T(n, n_{k+1})$ neighboring in $Seq_T(n)$. We define the *positive* and *negative ordered rays*, $R_T^+(n, n_k)$ and $R_T^-(n, n_k)$ of each branch $B_T(n, n_k)$ according to the partial order as its leftmost and rightmost path, provided that T is drawn as a planar graph where $Seq_T(n)$ of each node follows a clockwise direction. Each bypass edge of $BR_T(n, d)$ connecting $B_T(n, n_k)$ and $B_T(n, n_{k+1})$ is then initiated on $R_T^-(n, n_k)$ at distance $\left\lfloor \frac{d}{2} \right\rfloor$ from n and terminated on $R_T^+(n, n_{k+1})$ at distance $\left\lfloor \frac{d}{2} \right\rfloor$, as shown in Fig. 2a. The bypass framework is defined as the union of concentric bypass rings (see Fig. 2b):

$$BF_T(n, d_{\max}) = \bigcup_{d=2}^{d_{\max}} BR_T(n, d).$$

With this arrangement, all the bypass rings keep the same direction, allowing the bypass routing algorithm to route between branches of a given center node using its rings, and to employ rings of lower diameters centered in particular branches to route through those branches, while preserving the direction.

Supposing that frameworks $BF_T(n, d_{\max})$ are set up around each node $n \in \mathcal{TM}$ in T, there are $d_{\max}$ bypass edges initiated at each node and terminating at nodes at increasing distances (up to $d_{\max}$) on $R_T^+(n, n_k)$ of each of the node's branches $B_T(n, n_k)$, $n_k \in A_T(n)$. The routing itself is then based on the fact that each faulty cluster is an intersection of the respective tree branches of the nodes neighboring with the cluster (as illustrated in Fig. 3):

$$\langle \mathcal{FC} \rangle_T = \bigcap_{n_i \in A_T(\mathcal{FC})} B_T(n_i, n_j), \text{ where } n_j \in A_T(n_i) \cap \mathcal{FC}$$

At each node $n_i \in A_T(\mathcal{FC})$, the routing algorithm systematically browses $B_T(n_i, n_j)$ using the bypass edges initiated at n_i to find another node neighboring with $\mathcal{FC}$, node n_{i+1}, which is the next on $BC(\mathcal{FC})$. The branch lookup is performed sequentially by checking the nodes on $R_T^+(n_i, n_j)$ at an increasing distance until the first non-faulty node, n_{i+1}, is found. The sequence of nodes on $R_T^+(n_i, n_j)$ is kept for further use by LLE. This process is shown in Fig. 4a. Fig. 4b demonstrates a practical example of routing in the network from Fig. 1.

It can be shown [2] that bypass routing is feasible provided that the length of the positive ordered ray between every two bypass cycle neighbors is less or equal to $d_{\max}$. The lower bound of the maximum recoverable failure is thus $\left\lfloor \frac{d_{\max}}{2} \right\rfloor$ nodes in arbitrary clusters and $d_{\max-1}$ nodes in internal clusters of the tree (i.e., the clusters not containing leaves of the protected tree). The memory needed to keep routing information at a node is equal to

$$\mathcal{O}(\deg_T(n) * d_{\max}).$$

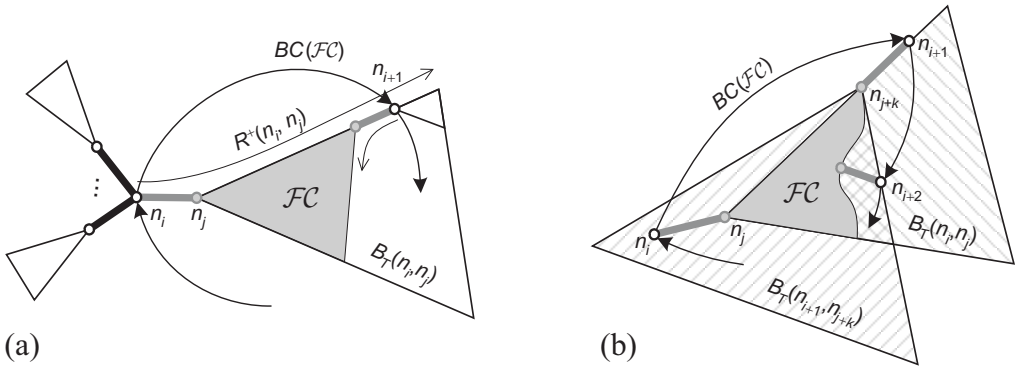

Fig. 3: Relation between nodes $n \in A_T(\mathcal{FC})$ and faulty cluster $\mathcal{FC}$

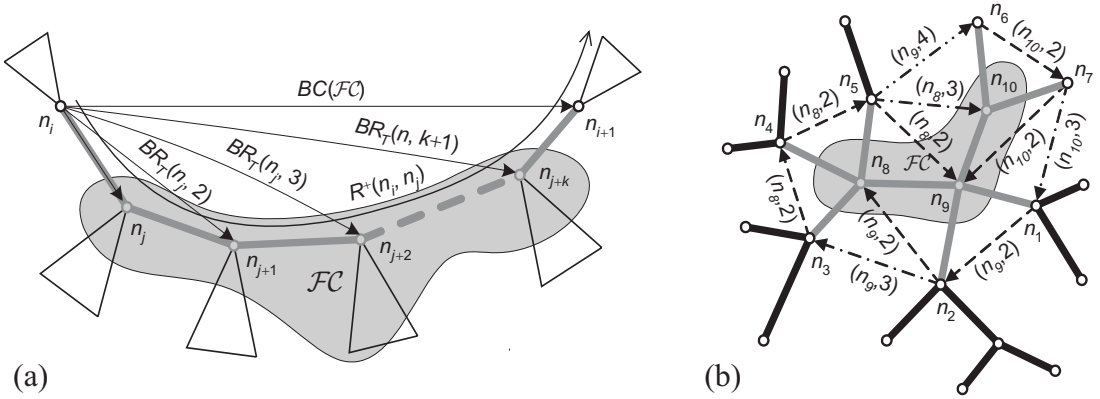

Fig. 4: Routing from $n_i \in A_T(\mathcal{FC})$ to n_{i+1} (a) and routing around $\mathcal{FC} = \{n_8, n_9, n_{10}\}$ (b)

7 Leader link election

The task of the leader link election algorithm is to identify a leader – a single edge on the bypass cycle that will not participate in fragment reconnection in order to ensure correctness of the restored tree while keeping it connected. Moreover, to support on-line and on-the-fly recovery, we look for a solution guaranteeing that once a link loses the election, it remains lost and that there is not a state of the algorithm in which there is no leader (i.e. all the fragments are connected). The completeness property is provided by the bypass routing algorithm; the bypass cycle forms a ring-topology communication structure for the election.

The election is based on identifiers of the bypass cycle nodes; the basic idea is similar to the Chang-Roberts leader election [20], where the maximum known ID is sent around the cycle by means of *ELECTION* messages and compared with the ID of each intermediate node. The LLE algorithm exploits the favorable properties of Chang-Roberts on ordered cycles, where it needs only $N = card(BC(\mathcal{FC}))$ messages and only a single *ELECTION* message to decide whether a given node (the bypass cycle edge incident to it) loses the election or not (see [3] for further details).

Except for single failures $\mathcal{FC} = \{n_f\}$ where $BC(\mathcal{FC}) = A_T(\mathcal{FC}) = BR_T(n_f, 2)$, the bypass cycle nodes are not ordered. For this reason, the hierarchical identifiers that uniquely identify each node relative to another node in the tree are used. The leaves of an arbitrary partially ordered rooted tree are ordered in parts according to the hierarchical identifiers based in their closest common parent node. Applying the principle of Chang-Roberts for an ordered ring, more than one leader can be elected using N *ELECTION* messages. These leaders (except one) are thereafter eliminated by a recursive *sweep process* considering hierarchical identifiers related to the common root node. Fig. 5 illustrates the principle of eliminating multiple leaders (only the leaf nodes are members of $BC(\mathcal{FC})$).

The common root node for bypass cycle nodes is a faulty node $n_r \in \mathcal{FC}$ with the minimum identifier determined together with the respective hierarchical identifiers step by step by the routing algorithm as it browses the relevant ordered rays in the bypass cycle lookup. In this way, the algorithm utilizes a byproduct of the bypass routing – knowledge of the cluster structure – to achieve $\mathcal{O}(N \log_b N)$ average message complexity of the election, where b is the average branching factor in $\mathcal{FC}$. Moreover, the algorithm needs only N messages to elect a leader link in an arbitrary bypass cycle in hierarchically ordered trees (e.g., binary search trees) and also in the bypass cycles of single failures in general partially ordered trees.

8 Fragment reconnection

The platform constituted by bypass routing and leader election forms a basis for various fragment reconnection methods responsible for joining the fragments into a single connected tree T' according to application-specific requirements. Fragment reconnection can be performed together with the LLE process – as the LLE messages travel around the bypass cycle and determine individual nodes not to be leaders, the respective fragments can be joined to the rest of the tree so that the data traffic can be transmitted immediately.

The most straightforward reconnection approach comes directly from the LLE process. The fragments on $BC(\mathcal{FC})$ are sequentially joined, except for the terminal fragments of the leader link. The drawback is the fact that the diameter of the failure recovery area $\langle A_T(\mathcal{FC})\rangle_{T'}$ is always equal to $N-1$, which might be a limitation for some applications because without extern tree balancing, the tree would depreciate to a linear graph after a certain number of recoveries. This is called *LR reconnection method*.

Two different reconnection methods, called *TRM* and *HRM*, were proposed in [3]. In the TRM method, all the fragments are joined directly to one of the designated nodes, while the HRM method allows the fragments to be joined in longer consecutive sequences. As a generalization of these two approaches, we propose the parameterized *HR-x reconnection method*, where value x affects the length of the successively joined fragments along $BC(\mathcal{FC})$ and thus it can influence the degree of the nodes and the diameter of $\langle A_T(\mathcal{FC})\rangle_{T'}$. The maximum number of new core edges incident to the affected nodes is

$$\min\left(\left\lceil \frac{N-1}{x} \right\rceil + 2,\ N-1\right).$$

The diameter is proportional to x as well. HR-1 thus represents the TRM, and HR-N is equivalent to the LR method.

The possibility to influence the properties of the restored tree may help to balance or optimize it. The particular value of x can even be chosen autonomously by each bypass cycle node, so the local requirements may also be supported. We

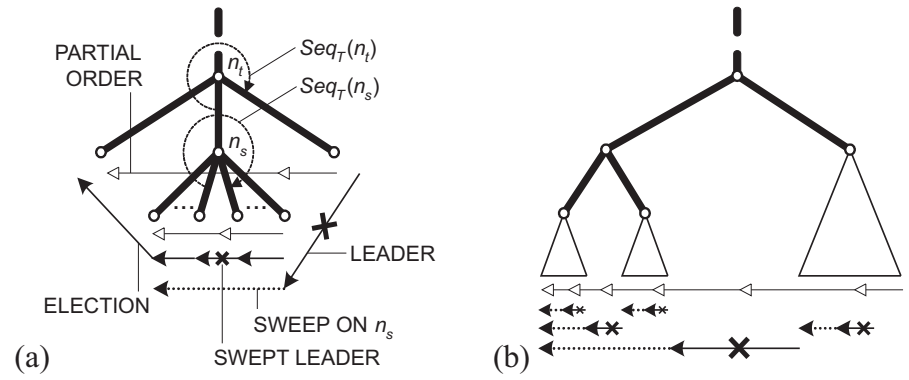

Fig. 5: Leader election and the recursive sweep process

note that the HR-x method is only an illustration of the reconnection process. More sophisticated systematical reconnection methods can be applied, provided that they retain correctness and completeness of the recovery.

9 Discussion

The characteristics of the computing environment in large-scale distributed systems as well as the general properties of overlay trees in this environment place quite specific requirements on tree restoration. The proposed platform for generic failure recovery is designed to meet the mentioned requirements of survivable trees and to form a basis for application-specific fragment reconnection.

The node memory complexity of the bypass ring recovery scheme is $\mathcal{O}(\deg_T(n) * d_{\max})$ and the average message complexity of the recovery is $\mathcal{O}(N \log_b N)$, where $\deg_T(n)$ is node degree, N is number of tree fragments, b is average branching factor of the failed nodes, and $d_{\max}$ is an optional parameter proportional to the provided fault-tolerance. The lower bound of the maximum size of the recoverable failure is $\left\lfloor \frac{d_{\max}}{2} \right\rfloor$ nodes in arbitrary clusters and $d_{\max}-1$ nodes in internal clusters of the tree. The lower bound of the maximum diameter of internal faulty clusters is $d_{\max}-2$.

The scheme is scalable with the group size, as its overhead depends solely on N and $d_{\max}$ and performs local recovery since only the nodes closest to the failed cluster (on $BC(\mathcal{FC})$) are involved in the recovery. The scheme also supports multiple, on-line and on-the-fly recovery – it is capable to recover the tree from several multiple failures with respect to the $d_{\max}$ parameter while communication in the tree continues, and the simultaneous recoveries do not interfere with each other because of the locality property. Protection selectivity and optional fault-tolerance is provided so that a trade-off between survivability and costs can be easily chosen. The scheme is independent of the type of protected tree (regarding traffic direction, number of sources, etc.) and forms a basis for application-specific fragment reconnection.

The simulations and measurements verifying the described behavior of the proposed bypass ring scheme were performed using the GFS file system ([21], [22]) as a test bed. GFS is a peer-to-peer large-scale file system providing a fault-tolerant and highly available file service. It extensively employs vast tree communication structures for replica-based management of its data, and it is a typical application to utilize the bypass ring scheme. GFS has been implemented ([23]) and simulated ([24]) in network simulator *ns2*.

One of the important results is the fact that protection with $d_{\max} = 4$ already provides ample fault-tolerance, and it is fully sufficient for the GFS application; the probability of employing rings in the recovery dramatically decreases with their diameter. The simulations also show the real scales of the recoverable failures. The average size of the recovered cluster in trees with average branching factor 4 is approx. 1.5 $d_{\max}$, and the average diameter is approximately $d_{\max}-2$ for $2 \le d_{\max} \le 10$. This result confirms the possibility to easily choose a trade-off between survivability and costs.

10 Conclusion

In this paper, we summarized the general properties of large-scale environments and identified the requirements for generic failure recovery in survivable trees. The main contribution of the paper is the outline of a scalable platform for local failure recovery that meets all the required properties. The platform is based on bypass rings – the redundant cyclic structures introduced in [3] and specified in detail in [2]. The recovery provided by this platform is generic to the intent that it is independent of specific tree properties and communication pattern, and it enables application-specific tree reconnection to optimize the restored tree according to some extern constraints and requirements. The performed simulations [24] in the GFS file system confirm the theoretical results.

Future research in this area may include specification of particular mechanisms for (autonomous) management of fault-tolerance level and protection selectivity with respect to the state of individual nodes, or a proposal of more sophisticated reconnection methods tailored exactly to specific application requirements.

Reference

[1] Dynda, V.: "A Concept of Survivable Trees and its Deployment in a Fault-Tolerant Multicast." In: Workshop 2004, CTU, Prague, 2004, p. 234–235.

[2] Dynda, V.: "A Bypass-Ring Scheme for a Fault-Tolerant Multicast." *Acta Polytechnica*, Vol. **43** (2003), No. 2, p. 18–24.

[3] Dynda, V.: "A Simple Scheme for Local Failure Recovery of Multi-Directional Multicast Trees." In: IFIP / IEEE ISCIS 2003, LNCS, Vol. **2869**, Springer-Verlag, Germany, 2003, p. 66–74.

[4] Deshpande, H., Bawa, M., Garcia-Molina, H.: "Streaming Live Media over a Peer-to-Peer Network." Report No. CS-2001-31, CS Dept., Stanford University, 2001.

[5] Yang, M., Fei, Z.: "A Proactive Approach to Reconstructing Overlay Multicast Trees." In: IEEE Infocom '04, 2004.

[6] Jia, W. et. al.: "An Efficient Fault-Tolerant Multicast Routing Protocol with Core-Based Tree Techniques." *IEEE Transactions on Parallel and Distributed Systems*, Vol. **10** (1999), p. 984–999.

[7] Manimaran, G., Chakrabarti, A.: "A Scalable Approach for Core Failure Recovery in Multicasting." In: *Advanced Computing and Communications*, 2000, p. 191–196.

[8] Chu, Y. H., Rao, S. G., Seshan, S., Zhang, H.: "Enabling Conferencing Applications on the Internet Using an Overlay Multicast Architecture." In: ACM SIGCOMM '01, 2001.

[9] Francis, P.: "Yoid: Extending the Multicast Internet Architecture.", 2000 http://www.icir.org/yoid.

[10] Zhang, B., Jamin, S., Zhang L.: "Host Multicast: A Framework for Delivering Multicast to End Users." In: IEEE Infocom '02, 2002.

[11] Birrer, S., Bustamante, F. E.: Nemo: "Resilient Peer-to--Peer Multicast without the Cost." Report No. NWU--CS-04-36, Northwestern University, 2004.

[12] Banerjee, S., Bhattacharjee, B., Kommaredy, C.: "Scalable Application Layer Multicast." In: ACM SIGCOMM '02, 2002.

[13] Birrer, S. et. al.: FatNemo: "Building a Resilient Multi-Source Multicast Fat-Tree." In: WCW '04, LNCS, Vol. **3293**, Springer-Verlag, Germany, 2004, p. 182–196.

[14] Tran, D. A., Hua, K. A., Do, T.: "ZigZag: An Efficient Peer-to-Peer Scheme for Media Streaming." In: IEEE Infocom '03, Vol. **2**, 2003, p. 1283–1292.

[15] Fei, A. et. al.: "A Dual-Tree Scheme for Fault-Tolerant Multicast." In: ICC '01, Vol. **4**, 2001.

[16] Medard, M. et. al.: "Redundant Trees for Preplanned Recovery in Arbitrary Vertex-Redundant or Edge-Redundant Graphs." *IEEE/ACM Transactions on Networking*, Vol. **7** (1999), No. 5, p. 641–652.

[17] Padmanabhan, V. N., Wang, H. J., Chou, P. A.: "Resilient Peer-to-Peer Streaming." In: IEEE ICNP '03, 2003, p. 16–27.

[18] Wu, C. et. al.: "A New Preplanned Self-Healing Scheme for Multicast ATM Network." In: IEEE ICCT '96, Vol. **2**, 1996, p. 888–891.

[19] Wu, C., Lee, W., Hou, Y.: "Back-up VP Preplanning Strategies for Survivable Multicast ATM Networks." In: IEEE ICC '97, Vol. **1**, 1997, p. 267–271.

[20] Chang, E. G., Roberts, R.: "An Improved Algorithm for Decentralized Extrema-Finding in Circular Configuration of Processors." *Communication of the ACM*, Vol. **22** (1979), No. 5, p. 281–283.

[21] Dynda, V., Rydlo, P.: "Large-Scale Distributed File System Design and Architecture." *Acta Polytechnica*, Vol. **42** (2002), No. 1, p. 6–11.

[22] Dynda, V., Rydlo, P.: "Fault-Tolerant Data Management in a Large-Scale File System." In: IEEE ISADS 2002, Mexico, 2002, p. 219–235.

[23] Zradička, L.: "A Distributed File System Model." Master Thesis, Department of Computer Science and Engineering, CTU Prague, 2003.

[24] Řehák, P.: "Fault-Tolerance in a Distributed File System." Master Thesis, Department of Computer Science and Engineering, CTU Prague, 2005.

Ing. Vladimír Dynda
phone: +420 224 357 616
fax: +420 224 923 325
e-mail: xdynda@sun.felk.cvut.cz

Department of Computer Science and Engineering

Czech Technical University in Prague
Faculty of Electrical Engineering
Karlovo náměstí 13
121 35 Praha 2, Czech Republic

