Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2015) 23: 1338 – 1356
© TÜBİTAK
doi:10.3906/elk-1211-146

Optimal backup parent pools for resilient multicast trees on peer-to-peer networks

Emrullah Turhan TUNALI[1], **Müge SAYIT**[2,*]
[1]Department of Computer Engineering, İzmir University of Economics, Turkey
[2]International Computer Institute, Ege University, Turkey

Received: 26.11.2012 • **Accepted/Published Online:** 18.08.2013 • **Printed:** 28.08.2015

Abstract: Due to churn as well as node and link failures in peer-to-peer (P2P) networks, providing resilient multicast is a challenging issue, particularly for overlay trees. In this work, we first study analytical properties of the backup parent pool that aims at improving the resilience of overlay multicast trees in P2P video streaming. We then present a novel greedy degree-constrained multicast tree construction algorithm that addresses the tradeoffs between maximizing resiliency, maximizing bandwidth utilization, and minimizing delay. The choice of essential design parameters is studied together with seamlessness of the streaming under a variety of fault scenarios. Simulation results indicate that the overhead introduced by the mechanism is negligible if there is sufficient bandwidth in the system. Both analytic and simulation results indicate that the proposed approach improves resiliency, bandwidth utilization, and delay. Comparisons with an algorithm from the literature are carried out and it is observed that the proposed method is superior under a wide range of conditions.

Key words: Capacity-aware multicast trees, multimedia communications, resiliency technique, scalable codec

1. Introduction

Since peer-to-peer (P2P) applications became a promising technology for large-scale or specially aimed networks, much significant research has been done in this area. It is well known that IP multicast is not widely utilized and hence its support for multicast applications is limited; an application layer multicast running on an overlay network has been introduced as an elegant solution to such applications [1]. Since then, many overlay structures have been proposed to address a variety of problems including scalability, resilience, robustness, and bandwidth-delay optimization. The tree-push type of approaches provided smaller overhead in utilization of network resources but failed to provide resilience [2]. Mesh-pull type of approaches [3] introduced efficient solutions for resilience while introducing overhead network traffic due to excessive control messaging and/or redundant data packets.

In a P2P network environment, a peer can experience connection quality distortion caused by four different classes of events: link failure, congestion, parent failure, and churn. From the P2P network node's point of view, these events have the same effect as far as the multicast is concerned. Throughout this paper, we will call the collection of such events node failure without loss of generality. Unfortunately, since churn is high, the node failure rate is considerably high in P2P networks. Hence, the tree-based approach needs to consider this fact. Even though resilience was extensively studied in [2], the fact that nodes have limited bandwidth is not

*Correspondence: muge.fesci@ege.edu.tr

imposed in the implementations. To provide a solution to the problem, the backup parent concept was proposed together with degree-constrained optimal trees [4,5]. In the case of parent failure, a tree node connects to a predetermined backup parent that has adequate idle bandwidth. In this way, the tree is repaired and multicast can resume. Nomination of a backup parent in advance prevents waste of time for backup search. This is classified as a "proactive strategy" in the literature [4,5], as opposed to "reactive strategies". Clearly, the higher the backup capacity reserved, the more resilient the system is and the lower the reserved backup capacity is, with higher bandwidth utilization. Therefore, the bandwidth that a node reserves for backup is to be determined by tailoring the tradeoff between bandwidth utilization and resilience. On the other hand, multicast applications such as video streaming require low initial waiting time and therefore minimizing average delay from root to peers is desirable. Moreover, to minimize total processing overhead at nodes, it is also desirable to send the video data over a smaller number of hops. Therefore, a multicast tree should have minimal average link delay as well as minimal average node depth.

In this study, an algorithm balancing the tradeoffs between resilience, bandwidth utilization, and delay is developed. A theoretical analysis of the proposed algorithm under simplifying assumptions is done and simulation-based comparative evaluation against another algorithm in the field [5] is carried out. The following sections of the paper are organized as follows. In Section 2, related works on multicast tree construction and methods providing error resilience on multicast trees are given. In Section 3, first the optimal choice of backup capacity is addressed, and then the proposed multicast tree construction with backup parent pool is explained. In Section 4, after explanation of the recovery protocol, video streaming simulation results are given together with comparisons with a similar study from the literature. Finally, in Section 5, concluding remarks are made.

2. Related work

The problem of constructing multicast trees simultaneously maximizing bandwidth and minimizing delay was addressed in [6], showing the NP-completeness of the solution. Some other works rephrased the problem as "degree-constrained minimum spanning tree (MST)" and proposed algorithms for the solution [7,8]. Chu et al. [1] implemented a greedy strategy that first picks higher bandwidth nodes and then, among the equal bandwidth nodes, picks the one with lowest delay. None of the above works addressed resilience of the tree when node failure rate is high as in P2P networks. Providing error-resilient dynamic multicast tree construction is one of the most difficult parts of applications requiring real-time processing such as video streaming systems or video conferencing [9]. Kim et al. [10] and Fesci-Sayit et al. [11] addressed the importance of constructing shallow trees in P2P overlay multicast. Other studies [12–14] proposed complete systems that address a variety of issues including scalability, resilience, and bandwidth utilization. Even though these studies reported performance issues such as control overhead, join overhead, failure overhead, and link stress, they did not discuss some of the critical streaming parameters such as average initial startup delay and buffer requirements for seamless streaming. In [12] and [14] systems are presented using a reactive approach, and a good comparison of those reactive systems can be found in [2]. In [5], the authors showed that proactive approaches give better performance than reactive approaches when considering the wide range of node failure patterns.

Proactive techniques in multicast trees can be organized into two categories as single-tree and multiple-tree approaches. The latter approach uses the multiple descriptive coding (MDC) technique [15] and provides error resilience by sending different descriptions of the video on different trees [16,17]. Single-tree approaches concentrate on repairing the tree in case failure happens. For this purpose, some backup mechanism is utilized by considering the capacity of each node. In such systems, the video file does not have to be encoded by using a special technique such as MDC. However, nodes with redundant capacity are necessary for resilience. In [4,18,19], a single-tree approach was used, in which each node reserves a number of backup slots, each slot being able to stream baseline video and nodes in the tree selecting their backup parents according to available backup capacity and round-trip time (RTT) values between candidate backup parent and themselves. In such systems, it can be difficult to maintain updated available slot numbers for each node in the tree, especially if churn is quite high. On the other hand, the ratio of backup slots to service slots was not argued by Jeon et al. [4] or Kusomoto et al. [19]. The latter authors [19] chose the number of backup slots as at least one for each node in the system, while the former [4] did not reserve any backup slots in the root.

For the single-tree approach, the challenging part of the problem is to develop a mechanism that has low overhead in terms of space, message, and time complexities together with high resilience and streaming properties. It is well known that the streaming systems operate with 2–3 s video buffers, and for seamless streaming, tree repair and resuming packet delivery should fit into this period. Another important issue is bandwidth utilization. ADSL clients have limited upload capacity that limits their use as a nonleaf (intermediate) tree node. Some tree-based studies propose “super-peers” that act as intermediate tree nodes and have higher upload capacity [20]. Even though the introduction of the concept of a super peer might be a deviation from the very definition of a peer, its effectiveness in building better trees makes it attractive. From another point of view, the super peer can be treated as an ordinary peer that is able to measure and declare its high upload capacity. Either way, high-capacity nodes may be placed close to the root. The distribution of the capacities close to the root can then be analyzed to determine the streaming rate. That is, the quality of the video to be streamed will depend on the capacity of the tree constructed. By dividing the upload capacity of each node by the determined streaming rate, a degree can be assigned to each node, thereby formulating the degree- and depth-constrained MST problem. On the one extreme, there may be a number of high-capacity nodes leading to trees of lower height with high quality of video streamed. On the other extreme are the nodes all having upload capacity that is barely adequate for a base layer video, and under such circumstances, it is inevitable to stream base layer video in a pipeline fashion with no resilience.

Jeon et al. [4] proposed a precomputation algorithm run by parent nodes to find a suboptimal backup parent for their children. In case both parent and backup parent fail at the same time, a node requests a new backup parent from its ancestors. A node can recommend a new backup parent for a node applying to it. For this recommendation, the node uses the knowledge of residual available bandwidth of its children and directs the applying node to one of its children. This work has the following features: first, it considers the bandwidth constraints of nodes as well as link delays in their optimization. Second, it assumes that multiple failures happen at the same time. One of the major drawbacks of this approach is that depth of the tree is not considered. That is, a tree may have extensive depth, which is not very desirable. Moreover, if the node failure rate is high, in the case of parent and backup parent failing at the same time, the reconnection time may increase. Locating the higher-capacity nodes at the upper layers of tree strategy was first introduced in [10] and implemented in [11] for allocating bandwidth among multicast trees of hierarchical clusters. Fesci-Sayit et al. [21] also implemented this strategy on hierarchical clusters and introduced the backup parent pool concept together with a comparison with [4] on hierarchical clusters.

Our work proposes a multicast tree for video streaming that considers bandwidth limitations as well as delay optimization. The features of the proposed system can be itemized as follows:

1. Following [22], we assign dependability index to nodes. The system is assumed to be monitoring the churn of nodes and classifying them as high-dependable, medium-dependable, and low-dependable. The highly

dependable higher degree nodes are placed on upper parts of the tree so that there are fewer repairs on the upper tree.

2. Following [11], higher degree nodes are placed towards the root so that the tree formed is as shallow as possible.

3. Among the available nodes, children are chosen to minimize total delay between the layers of the tree.

4. Following [21], the backup parent pool concept is utilized, where each node is assigned a pool of backup parents rather than a single parent.

5. The novelty of our proposed method is the heuristics developed, which considers all of the above features during the multicast tree construction phase. Furthermore, pertinent design preferences are supported by analyses and extensive simulations are used for verification of choices of parameters.

3. Resilient multicasting

In order to construct a multicast tree for a video streaming session, some points need to be stressed. First, we assume that any node with a degree higher than one has no processing power bottleneck as far as forwarding video is concerned. Second, it is well known that video streaming systems try to minimize initial waiting time for video display, which is the sum of tree joining time, initial buffering time, and total delay between a node and root. Therefore, it is desirable to minimize all of these factors, among which only the latter is addressed in this work. We will use the terms "backup" and "service" degrees (or slots) to distinguish the portions of the upload bandwidth that are used for connection to children and reserved for backup, respectively, at the initial construction phase of the multicast tree. Although assigning a backup parent for each node is studied in the literature, the ideal ratio between service and backup degrees is not discussed in detail. Decreasing the number of backup degrees may leave nodes with no backup parent to connect to. On the other hand, decreasing service slots cause a rise in the tree height. Given graph G(V, E) with d_i, $i = 1, \ldots,$ n the degrees of the nodes and $delay_{u,v}$ the delay between nodes u and v, $u, v \epsilon V$. We will assume a fixed store and forward delay $delay_p$ at all of the nodes. We will also assume that node i can fail or leave the system with probability p_i in case of which the respective children should be assigned new parents. Depending on accumulated statistics, we assume that we can roughly estimate p_i. If there are no statistics about a node, we simply assign a high p value to that particular node. We assume that the source does not fail and has a known degree. We would like to build a spanning tree for video streaming with the following properties:

1) the average total delay at the nodes is minimized,

2) the bandwidth utilization is maximized,

3) the resilience is maximized.

Clearly, this is a multicriteria spanning tree optimization problem. To minimize average total delay, average transmission delay from root to other nodes as well as average depth of nodes is to be minimized. Maximizing total bandwidth utilized is essential for maximizing the average video quality streamed and finally maximizing resilience will provide robust and seamless streaming of the video. Maximal bandwidth utilization and maximal resilience are contradicting properties for which we need to address the tradeoff carefully. In the following text, we will assume that the multicast tree is repaired as failure occurs. Following [21], the proposed scheme of this study asks each node to maintain a backup parent pool that is formed from the nodes at least one level up on the tree. This choice preserves the bandwidth hierarchy of the tree as well as meeting the timing constraints during streaming.

Let $(d_1 \geq d_2 \geq \ldots \geq d_n)$ represent the sorted sequence of degrees of nodes and let do represent the degree of the root. Without loss of generality, we assume that all degrees are even. We are to determine c_0, the number of children to be connected to the root, starting from the node that has degree d_1. Let c be the ratio of the allocated upload bandwidth to children. We will call c the children ratio in the sequel. Let $f \leq c_0$ be the number of failing children of the root.

Lemma 1 *If*

$$c\sum_{i=1}^{f}(d_i) \leq f + (d_0 - c_0) + (1-c)\sum_{j=f+1}^{c_0}(d_j), \tag{1}$$

then all of the orphan nodes can find backup parents from their upper layer.

Proof The first thing to note is that Eq. (1) covers the worst-case scenario in which failing nodes have the highest number of degrees. If f children fail, then we need to find backup degrees for the children of the failing nodes (grandchildren of the root) whose sum is the left side of the inequality. Among these grandchildren, f of them may directly connect to the root using their failed parent's bandwidth. This explains the first term on the right side. For the others, we have backup degrees available from the root and siblings of the failed nodes, and if the total available degrees cover all the leftover grandchildren, then all of the orphans can find backup parents from the upper layer. □

Figure 1 illustrates Lemma 1. We assume that $n = 6$ and $\forall i = 0,1,\ldots$n, $d_i = 6$, $c = 0.5$, and $c_0 = 2$. Suppose that both children of the root fail. Then the tree can be repaired as follows: v_1 and v_4 are connected to the root since two service slots of the root are freed after the failing of two children. v_2, v_3, v_5, and v_6 are connected to the root by using its four backup slots. Clearly, with $f = 2$, $c_1 = c_2 = 3$, $d_0 = 6$, and $c_0 = 2$, the left and the right side of Eq. (1) is 6 and the condition is satisfied.

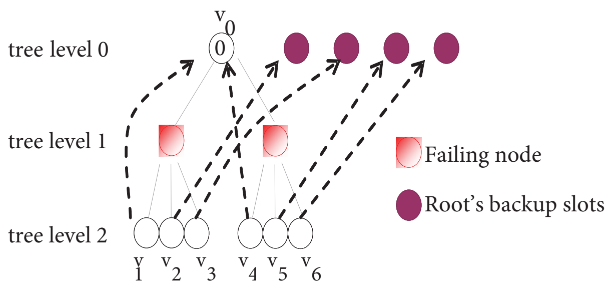


Figure 1. An illustrated failing scenario with root's backup slots.

The brute force algorithm of Figure 2 determines c$_0$ in such a way that orphans can always find backup from the upper layer. The outmost loop of the algorithm of Figure 2 checks all children possibilities, whereas the inner loop fails children in an increasing number and determines the maximum number of failing children for which the tree survives. The algorithm then chooses the index for which the maximum number of children fail and yet the tree survives. In the case of more than one index, the largest index is chosen to maximize bandwidth utilization. This maximizes the resiliency of the topmost region of the tree. The following lemma states that, for the equal degree case, there is no need to reserve backup at the root if d satisfies a specific bound.

```
1  for $i = 1$ to $d_0$                                   /* assign i children to root
2    for $f = 1$ to $i$                                   /* fail children one after the other
3      if $(c\sum_{i=1}^{f}(d_i) > f + (d_0 - c_0) + (1 - c)\sum_{j=f+1}^{c_0}(d_j))$
4      then
5          $f = f - 1$;
6          break;
7       end if
8    end for
9    $f\_array(i) = f$;
10 end for
11 $c_0$ = ind{max $f\_array(i)$, $i = 1, ..., d_0$}/* # children is the  index of the maximum
                                                    element of the array
```

Figure 2. "Children of the root" algorithm.

Lemma 2 *When all nodes have degree d, if $d > 1/(1-c)$, then children of the root algorithm connect d children to the root.*

Proof We can rewrite Eq. (1) as

$$cfd \leq f + (cd - c_0) + (1-c)(c_0 - f)d,$$

which can be reorganized as

$$f \leq c_0(d - cd - 1) + cd.$$

Since $d > 1/(1-c)$, $(d - cd - 1) > 0$ and the value of c_0 that maximizes f in the children of root algorithm is d. $\square$

Having worked out the top layer of the tree, we can proceed by forming the lower layers. The multicast tree construction algorithm of Figure 3 first groups nodes according to their failure probabilities. It then sorts the nodes, first according to their failure group in descending probabilities and then according to their degrees. Recall that all the degrees are assumed as even integers. It then calls the children of root algorithm of Figure 2 to determine the number of children to be connected to the root. Starting from the highest degree node on the leftmost position as child, the root connects its children. Figure 4 gives a consensus algorithm that is developed to optimally assign children. Given a set of parent nodes and a set of children to be connected, the consensus algorithm that minimizes the overall delay between the parents and the children is called and the process goes on until all the requestor nodes are connected. Note that the proposed algorithm favors the choice of more dependable nodes first and then higher degree nodes and finally lower delay nodes at each layer of the tree. Our final algorithm in Figure 5 is the one that is called when a node determines that it needs to find a new parent. It is clear that the message complexity of this algorithm is $O(n)$.

```
given  $c$, the children ratio, root $v_0$, and its degree $d_0$ together with $n$ requester nodes $v_i$, $i = 1,...n$ with their respective degrees $d_i$, and their respective failure probabilities $p_i$.
group  nodes according to $p_i$ to form three sets of nodes $R_1$, $R_2$, and $R_3$ with $R_1$ containing lowest $p_i$ nodes, etc.
sort   nodes $v_i$, $i = 1,...n$ first according to $R_i$ in ascending order and then according to $d_i$ in descending order to form $R = \{ v_i, i = 1,...n \}$ with their respective degrees $d_i$, $i = 1,...n$ ;
call   children of the root algorithm of Figure 2 to determine number of children of root $c_0$ ;
connect    $v_i \in R, i = 1,... c_0$, to root
$k = 1$ ;
update     $Q_k = \{v_i \in R, i = 1,... c_0\}, R = R \setminus Q_k, NL_k = c_0$
while  ( $R \neq \emptyset$ )
    call    consensus algorithm of Figure 4 to optimally connect children $v_i \in R$,
    $i = 1,..., c\sum_{i=1}^{NL_k} c_i$ among parents $v_i \in Q_k$, $i = 1,... NL_k$
    $k = k + 1$ ;
    update     $Q_k = \{v_i \in R, i = 1,.., d\sum_{i=1}^{NL_{k-1}} c_i\}, R = R \setminus Q_k, NL_k = c\sum_{i=1}^{NL_{k-1}} c_i$
end while
```

Figure 3. The multicast tree construction algorithm.

Figure 6 illustrates the resilience of the tree constructed by the algorithm of Figure 3. We assume that $\forall i = 0, 1, \ldots n$, $d_i = 6$, and $c = 0.5$. When the algorithm of Figure 2 is executed, c_0 is determined as 6 with $f = 3$. Then the algorithm of Figure 3 constructs the tree shown in Figure 6a. The lower layers of tree are not drawn for simplicity. Suppose now that 3 children of the root, v_1, v_2, and v_3, fail. Then, as can be seen in Figure 6b, v_7, v_{10}, and v_{13} connect to their grandparent, i.e. the root. The other children of the failing nodes connect to the siblings of the failing nodes as shown in Figure 6 and the tree survives. It is easy to see that if 4 children of the root fail, there is no adequate backup capacity and the backup searching node needs to proceed to searching lower layer backup pools. The following lemma establishes a sufficient condition for tree repair by using one layer up the backup pool when the nodes have equal degrees:

Lemma 3 *Suppose that all the degrees of nodes are equal with value d. Further suppose that f nodes fail at level $i > 0$. If*

$$f \leq \frac{d^i c^{i-1}}{cd - 1}(d - c), \tag{2}$$

then the tree can be repaired just by using layer i backup pool.

```
given  children set of nodes $C = \{v_i, i = 1,...m\}$, parent set of nodes $P= \{v_j, j = 1,...n\}$ with
their respective degrees $d_j$ , $c$, the children ratio, and $\{delay_{ij}$ , $i = 1,...m, j = 1,...n\}$ ;
sort   each column of delay matrix in ascending order ;
mark   in each column $j$ of the sorted delay matrix, mark the top $cd_j$ many respective children
for connection to $v_j$ ;
while  $(i \leq m)$
       while (child $i$ is marked in more than one column)
       erase  among these marks, erase the marks of the columns $j$ that do not contain the
maximum value among their $(cd_j + 1)^{st}$ elements from the top
       re-mark in each mark-erased column $j$, mark the next lowest delay children
       end while
end while
```

Figure 4. The consensus algorithm.

```
if       grandparent allocates dead parent's slot then stop
else if  polling one layer up backup pool is successful then stop
else if  polling upper layer backup pools is successful then stop
else if  sibling that has connected to grandparent allocates a slot then stop
else if  polling other siblings and lower layer nodes is successful then stop
else     return error.
```

Figure 5. The backup search algorithm.

Proof We need to have a sufficient number of backup slots in layer l to connect the children of the failing nodes. Since d is the fixed degree of all nodes, by Lemma 2, the algorithm of Figure 2 connects d children to the root and all other nodes have cd children. We first note that layer i of the tree constructed contains $c^{(i-1)}d^i$ many nodes and $(1-c)c^{i-1}d^{i+1}$ many backup slots. Among orphans, f of them connect to their grandparent and $(1-c)c^{i-1}d^{i+1}$ many can find backup at layer i. Since the total number of orphans looking for backup is fcd, the condition that the tree can be repaired just by using one layer up the backup pool can be written as

$$f + (1-c)c^{(i-1)}d^{(i+1)} \geq fcd,$$

which can be reorganized to yield Eq. (2). □

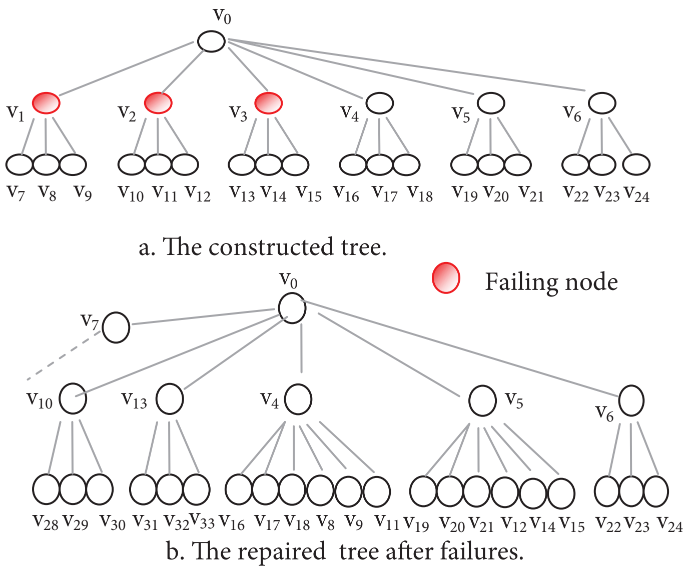

v0
v1
v2
v3
v4
v5
v6
v7 v8 v9 v10 v11 v12 v13 v14 v15 v16 v17 v18 v19 v20 v21 v22 v23 v24
a. The constructed tree.
Failing node
v7
v10
v13
v5
v28 v29 v30 v31 v32 v33 v16 v17 v18 v8 v9 v11 v19 v20 v21 v12 v14 v15 v22 v23 v24
b. The repaired tree after failures.

Figure 6. Optimal tree construction and repair after failure.

Lemma 4 *Excluding the root, suppose that all the degrees of nodes v_i are equal with value d. Let c, with $0 < c < d$, be the fixed number of children to be assigned to all nonroot nodes. Then the total backup capacity of the children of v_i, BC_i, is maximized by choosing $c = d/2$.*

Proof $BC_i = c(d - c)$, and differentiating with respect to c and equating to zero yields the result. □

It should be noted that Lemma 4 establishes a local criterion for the equal degree case that does not necessarily yield global resilience features such as percentage of nonroot nodes that were not able to find a backup. However, it provides a starting point for the unequal degree case in crafting the tradeoff between bandwidth utilization and tree resilience.

Lemma 5 *Suppose that all the degrees of nodes including the root are equal with even value $d \geq 4$, $c = 0.5$, and let N be the total number of nodes excluding the root. Then the multicast tree construction algorithm of Figure 3 has complexity $O(N^2)$.*

Proof This sort operation has complexity $O(NlogN)$. The children of root algorithm is $O(d^2)$. The crucial part is the consensus algorithm called in the while loop. Note that for n_p parents and n_c children, the consensus algorithm is $O(n_p n_c)$. It should be noted that, in the worst case, the number of nodes in layer i of the multicast tree is $d(d/2)^{i-1}$, $i = 1, \ldots, h$, where h is the depth of the tree. Hence, the consensus algorithm runs with values $n_p = d(d/2)^{i-1}$, $i = 1, \ldots, h-1$, $n_c = n_p(d/2)$.

We note that

$$N = d\sum_{i=1}^{h}\left(\frac{d}{2}\right)^{i-1} \tag{3}$$

and the inner while loop in the consensus algorithm runs

$$M = \frac{d^3}{2} \sum_{\substack{i=0 \\ i\ even}}^{m} \left(\frac{d}{2}\right)^i,$$

where

$$N = d\sum_{i=1}^{h} \left(\frac{d}{2}\right)^{i-1} \tag{4}$$

$m = (h\text{- } 1) + (h\text{- } 2) - 1 = 2h\text{- } 4.$ %4

Since

$$\sum_{i=1}^{n} q^{i-1} = \frac{q^n - 1}{q-1}, q \neq 1, \tag{5}$$

applying Eq. (5) to Eq. (3) with $q = d/\ 2$, we get

$$N = d\frac{\left(\frac{d}{2}\right)^h - 1}{\left(\frac{d}{2}\right) - 1}, \tag{6}$$

and applying Eq. (5) to Eq. (4) with $q = (d/\ 2)^2$, we get

$$M = \frac{d^3}{2}\frac{\left(\frac{d}{2}\right)^{2(h-1)} - 1}{\left(\left(\frac{d}{2}\right)^2 - 1\right)}, \tag{7}$$

From Eq. (7), we can write

$$M = \frac{d^3}{2}\frac{\left(\frac{d}{2}\right)^{2h-2}}{\left(\frac{d}{2}-1\right)\left(\frac{d}{2}+1\right)} = 2d\frac{\left(\frac{d}{2}\right)^{2h}}{\left(\frac{d}{2}-1\right)\left(\frac{d}{2}+1\right)}. \tag{8}$$

From Eq. (8), we can write

$$M \leq 2d\frac{\left(\frac{d}{2}\right)^{2h}}{\left(\frac{d}{2}-1\right)\left(\frac{d}{2}-1\right)} \leq d^2\frac{\left(\frac{d}{2}\right)^{2h}}{\left(\frac{d}{2}-1\right)\left(\frac{d}{2}-1\right)}. \tag{9}$$

For $N \geq d$, $h \geq 1$ and Eq. (9) can be rewritten as follows.

$$M \leq d^2\frac{\left(\frac{d}{2}\right)^h\left(\frac{d}{2}\right)^h}{\left(\frac{d}{2}-1\right)\left(\frac{d}{2}-1\right)} \leq d^2\frac{\left(\left(\frac{d}{2}\right)^h-1\right)\left(\left(\frac{d}{2}\right)^h-1\right)}{\left(\frac{d}{2}-1\right)\left(\frac{d}{2}-1\right)} = N^2.$$

This completes the proof. □

Let p be node failure probability in a time slot, h be tree depth, a_l be the number of nodes at layer l, and b_l be the number of failing nodes at layer l. For the equal degree case, we will now compute the probability that all of the layer l backup parent slots are used up. Recall that in such a case, the backup search algorithm polls upper layer pools. This probability is computed as

$$\sum_{j=1}^{a_l-b_l} \binom{a_l}{b_l+j} p^{b_l+j}(1-p)^{a_l-b_l-j}, l=1,..,h, \tag{10}$$

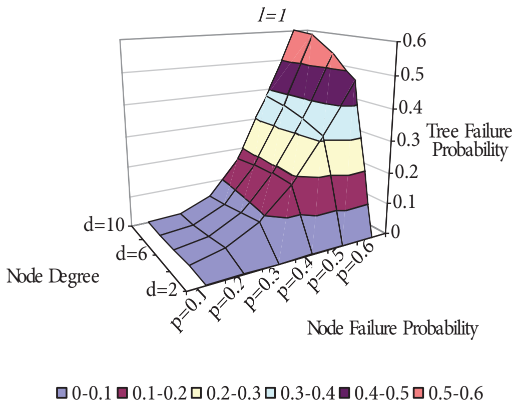
where

$$a_l = d\left(\frac{d}{2}\right)^{l-1}$$

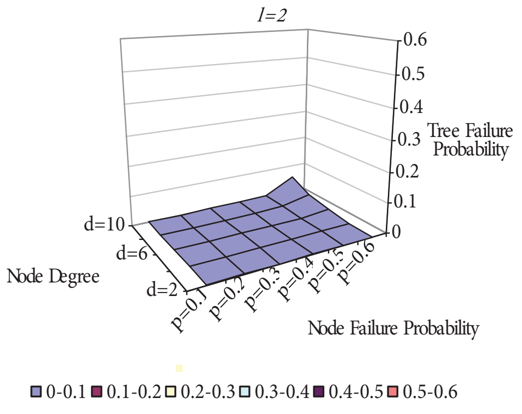
and

$$b_l = \frac{d}{d-1}\sum_{j=1}^{l}\left(\frac{d}{2}\right)^{j}.$$

Figures 7a and 7b plot Eq. (10) for $l = 1$ and $l = 2$, respectively. For a variety of combinations of node failure probability p and node degree d, we see that layer 1 of the tree is more vulnerable than layer 2. However, we see that tree failure probability stays in acceptable limits if all nodes have equal degrees. It should also be noted that when nodes have different degrees, the probabilities may not be as low as in Figure 6a. As an extreme case, consider that there is only one high degree node and all the rest of the nodes in layer 1 have low degrees. In such a case, the loss of this high degree node may cause the backup search algorithm to poll other siblings since one layer up the backup pool is not existent.



a. Tree failure probabilities at layer 1.



b. Tree failure probabilities at layer 2.

Figure 7. Comparative tree failure probabilities for different degrees of nodes and different node failure probabilities.

We will now analyze a worst-case scenario in which any node can only connect to either its grandparent or one of the backup parents at the upper layer of the tree. If no backup is found at the upper layer, we will simply assume that the node disconnects from the tree. This analysis will reveal the effectiveness of the backup parent pool. We will deal with the equal degree case and analytically compute the average number of nodes disconnected from the multicast tree in a time slot. We will use the notation given in Table 1. In the following analysis, we assume that leaf nodes are assigned in a round robin fashion and the backup parent pool of a node at layer j consists only of nodes at layer $j-1$.

Table 1. Notation of disconnection analysis.

d	Degree of all nodes
c	Ratio of degree allocated to children
N	Total number of nodes in the multicast tree excluding root
nn_j	Number of nodes at layer j of original multicast tree (root is assumed to be at layer 0)
h	Depth of original multicast tree
nns_j^k	Number of nodes at layer j of a subtree whose root is at layer k of original multicast tree, $k = 1,.., h-1$
nb_j	Number of backup degrees at layer $j$of original multicast tree
p	Probability of failure of a node in unit time interval
nf_j	Number of failing nodes at layer j of original multicast tree
c_h	Children ratio of leaf layer of original multicast tree, i.e. nn_h/nn_{h-1}
nnb_j	Number of nodes at layer j of original multicast tree that could not find backup
nnb_j^k	Number of nodes at layer j that could not find backup due to failures at layer k
nfs_j^k	Number of failing nodes at layer j of a subtree whose root is at layer k of original multicast tree, $k = 1,.., h-1$
nlb_j^k	Number of nodes at layer j looking for backup due to failures at level k
$ntlb_j$	Total number of nodes at layer j looking for backup

We note the following.

$$nn_j = \begin{cases} d^j c^{j-1} & j = 1, ..., h-1 \\ N - \sum\limits_{k=1}^{h-1} d^k c^{k-1} & j = h \end{cases}$$

$$nb_j = d^{j+1}c^{j-1}(1-c) - d(1-c)(E(nf_j) + E(nnb_j))j = 1, \ldots, h-1$$

$$nns_j^k = \begin{cases} d^j c^j & j = 0, 1, ..., h-k-1 \\ d^j c^j \frac{nn_h}{nn_{h-1}} & j = h-k \end{cases}$$

$$E(nfs_j^k) = \sum_{m=1}^{nns_j^k} m \binom{nns_j^k}{m} p^m (1-p)^{nns_j^k - m}, j = 0, \ldots, h-k, k = 1, ..., h-1$$

For any node failing at layer j, there are cd children at layer $j+1$, among which one will connect to its grandparent and

$$E(nfs_1^j) = \sum_{k=1}^{cd} k \binom{cd}{k} p^k (1-p)^{cd-k}$$

many will be failing themselves. Hence, the expected number of nodes looking for backup at layer $j+1$ due to failures at layer j, denoted $E(nlb_{j+1}^j)$, is

$$E(nlb_{j+1}^j) = (cd - 1 - E(nfs_1^j))E(nf_j)$$

where

$$E(nf_j) = \sum_{m=1}^{nn_j} m \binom{nn_j}{m} p^m (1-p)^{nn_j - m}.$$

Hence:

$$E(ntlb_{j+1}) = \sum_{k=1}^{j} E(nlb_{j+1}^{k}),$$

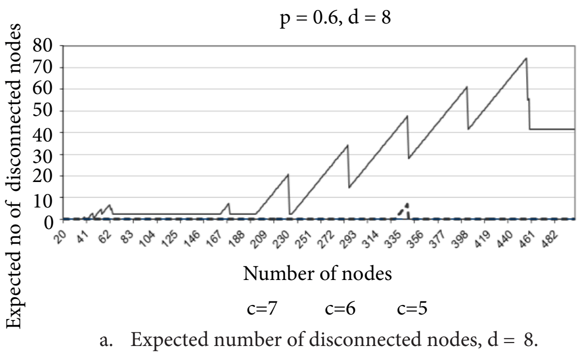
$$E(nnb_{j+1}) = E(ntlb_{j+1}) - nb_j,$$

$$E(nnb_{j+1}^{k}) = E(nlb_{j+1}^{k})E(nnb_{j+1})/E(ntlb_{j+1})k = 1, \ldots, j.$$

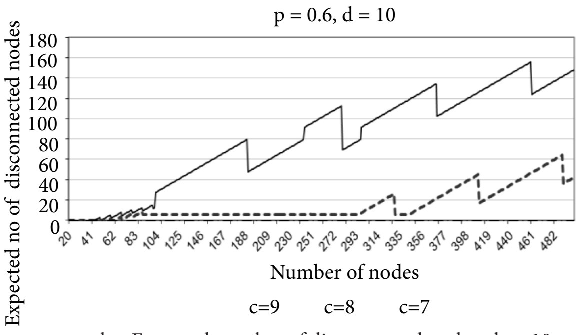
Starting with $j = 1$, we can recursively compute E(nnb_{j+1}) for $j = 1$, 2, ...,$h-1$. We then compute the expected number of nodes disconnected from the multicast tree, denoted N_{disc}, as

$$N_{disc} = \sum_{j=2}^{h} E(nnb_j). \tag{11}$$

We have computed Eq. (11) for a variety of p, c, $d$against N and examined numerous number of plots. Figures 8a and 8b show the cases for $p = 0.6, d = 8$, 10. We should note that c, as indicated in the plot, is the number of children rather than the ratio of children. This is simply because of numerical considerations. For all of the other combinations with lower c values indicated in the plots, we have observed that N_{disc} is almost zero. Similar observations with other combination of parameters yielded the same conclusion. We have not included them due to space considerations. The spiky behavior observed in Figure 8 is expected to be due to the number of nodes at the lowest tree layer increasing to yield a full tree topology. The increase in the magnitude of spikes can be explained with proportionality with the total number of nodes. As the value of c increases above $0.5d$, we start obtaining nonzero values in some of the cases. These observations directly support Lemma 4. Another observation is that, unlike expected, N_{disc} does not increase drastically as p increases as it is shown in Figure 9. This is due to the reason that with nodes leaving the system at a high rate, there will be fewer nodes looking for backup and the tree survives.



a. Expected number of disconnected nodes, d = 8.



b. Expected number of disconnected nodes, d = 10.

Figure 8. Expected number of disconnected nodes versus total number of nodes. **a)** Expected number of disconnected nodes, d = 8. **b)** Expected number of disconnected nodes, d = 10.

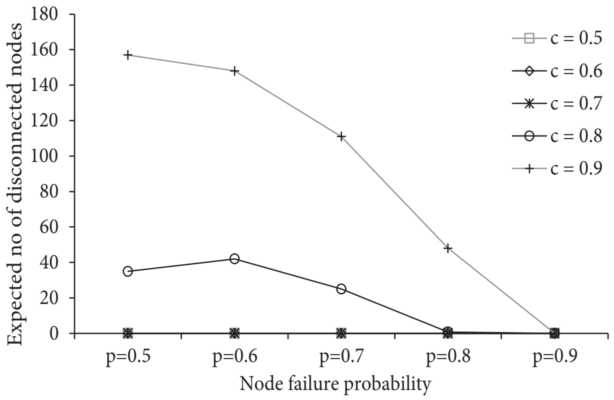


Figure 9. Expected number of disconnected nodes versus node failure probability. $N = 500$ and $d = 10$ in the experiments.

4. Implementation and simulation results

4.1. Implementation notes

Before the video streaming session starts, each node has a list of backup parent addresses that consist of the backup parents allocated at the layers above. In each tree level l including level 0, i.e. the root, one or more backup parents are selected so that nodes belonging to lower layers experiencing parent failure can connect. For level $l + 1$, nodes' backup parent lists consist of backup parent addresses from tree level l to tree level 0. These backup parent addresses are sent by parent nodes to their children.

After the streaming session starts, nodes periodically send heartbeat messages to their backup parents. Hence, if any backup parent leaves the system, the node can easily detect backup parent failure and does not lose any time by trying to connect to the failed backup parent. In order to find a new parent, nodes attempt connection trials by scanning through their backup parent list and send a connection request message with buffer level information to the first backup parent in their list. The backup parent receiving a connection request message may:

1. accept the request if there is available service capacity,
2. reject the request,
3. direct the node to lower levels of the tree if the buffer level of the requesting node is under a certain threshold.

4.2. Simulation testbed

In order to measure the performance of the proposed algorithm, we first carried out simulations with different parameter sets. In our simulations, our network model consisting of 500 nodes is generated by the GT-ITM module [23]. We have used the delays generated by the GT-ITM module. Generally, two nodes that share an edge in the overlay architecture may not actually be directly connected to each other. In this case, the communication path between these nodes is chosen as the shortest path in the network and delay between these nodes is calculated by considering the shortest path between them. In each set of experiments, after multicast trees are constructed, some nodes are forced to fail. Failure times of nodes are distributed according to Poisson distribution. We classified failure rates into three different classes as low, medium, and high according to failure probability of nodes in the system. Recall that this probability also includes churn.

4.3. Optimal backup ratio

In the first set of experiments, tests with different numbers of backup slots are carried out in order to observe which ratio of backup slots to service slots gives the best performance. Node degrees are uniformly generated as 2, 4, 6, 8, and 10. For each node experiencing parent failure, following [5], the recovery time is computed by summing up the latencies of the links that the control message travels for finding a new parent. The average recovery time represents the averaged values of recovery times for all nodes connecting to a new parent during the whole simulation. In Figure 10, four different backup slot selection strategies are compared by considering average recovery time. Again following [5], failure probability of nodes, p, is calculated according to the number of failed nodes determined by failure times obtained from Poisson distribution. According to the distribution pattern, the failure probability of nodes varies from 0.1 to 0.6. It is seen that 50% backup slots and 70% backup slots give the minimum value in terms of recovery time, where the performance of 70% backup slots is slightly better than that of 50% backup slots. Similar behavior is observed when maximum recovery time is examined. Even though 50% and 70% backup slots yield comparable performance as far as backup is concerned, 50% should be preferred due to its higher bandwidth utilization. We see that this preference based on Figure 10 supports the results of Lemma 4.

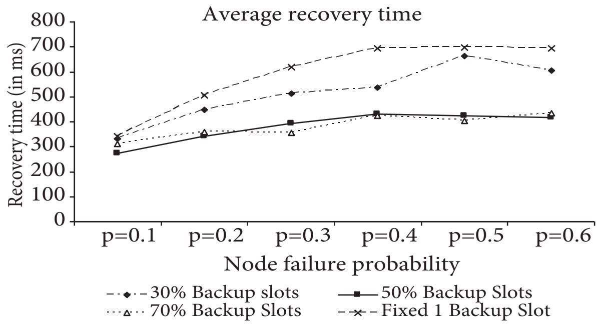


Figure 10. Average recovery time of different backup slot selection strategies.

We define total delay as the time elapsed between departure of a packet from the source and its arrival at the destination. This period includes both link delays and store and forward delays. Table 2 summarizes total delays obtained with different backup strategies with store and forward delay chosen as 5 ms. Average initial waiting time is calculated by adding maximum recovery time to the average total delay. We see that 50% backup strategy yields the best performance in "average hop", "average total delay", "maximum hop", and "maximum total delay", which are all multicast tree parameters. On the other hand, the 70% backup strategy is slightly better in "average initial waiting time". Recalling that the 50% backup strategy provides higher bandwidth utilization than the 70% backup strategy, we conclude that the 50% backup strategy is again the winner, which again supports Lemma 4.

Table 2. Tree performance of different backup selection strategies.

	Avg. hop	Average total delay (s)	Max. hop	Maximum total delay (s)	Avg. initial waiting time (ms)
30% backup slots	3.67	1.42	4	3.05	4518
50% backup slots	3.67	1.7	4	3.7	4208
70% backup slots	4.6	2.05	6	4	3540
Fixed 1 backup slot	3.25	1.25	4	2.41	4348

4.4. Comparison with Fei and Yang's [5] approach

In this section, we compare our algorithm with that of Fei and Yang [5] in terms of a variety of aspects including scalable video streaming. Essential simulation parameters are chosen to be the same as in [5] with the only difference being that [5] has total degree of nodes uniformly distributed between 2 and 6 inclusive, whereas our simulation has the same distribution between 2, 4, 6, 8, and 10. The reason for the extra 8 and 10 is due to recent increase in the available Internet bandwidths, particularly in "supernodes". The video sequence consists of a base and two enhancement layers having bit rate values of 500 Kbps, 800 Kbps, and 1 Mbps, respectively. In Figure 11, bandwidth distribution of nodes used in this set of experiments is given. We first examine total delay from source to destination. Figure 12 gives total delay when store and forward delays are assumed as zero, as in [5]. We observe that the algorithm from [5] yields lower total delays (approximately 750 ms difference on the average), as expected. This will cause lower initial waiting time in steaming and hence the video buffer may accordingly be chosen lower. However, in the forthcoming text, we will see that the algorithm from [5] requires a larger average recovery time (approximately 2000 ms more), which, in turn, increases the size of video buffer. This will eliminate the advantage of [5] in total delay. Figure 13 gives number of hops from source to destination. We observe that the algorithm from [5] constructs trees with 20 hops on the average, whereas the same value in the proposed method is only 5.5. Even though a large number of hops does not affect total delay due to the fact that we assumed zero store and forward delay, we will see that average recovery time will increase due to increase in control messaging. Figure 14 gives the expected received bit rate by nodes for the algorithm from [5] and the proposed algorithm. The proposed algorithm yields better results in terms of expected received bit rate since it considers upload bandwidth of nodes. Nodes having low upload bandwidth located near the root cause degradation of received bit rate and we see this negative effect in the results of the

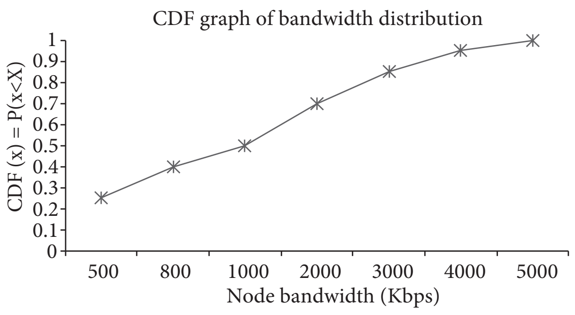


Figure 11. Bandwidth distribution of nodes.

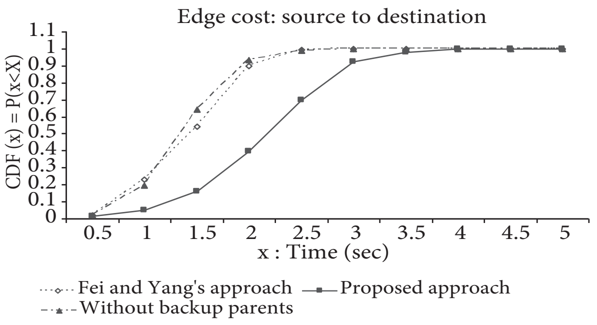


Figure 12. CDF graph of link delay.

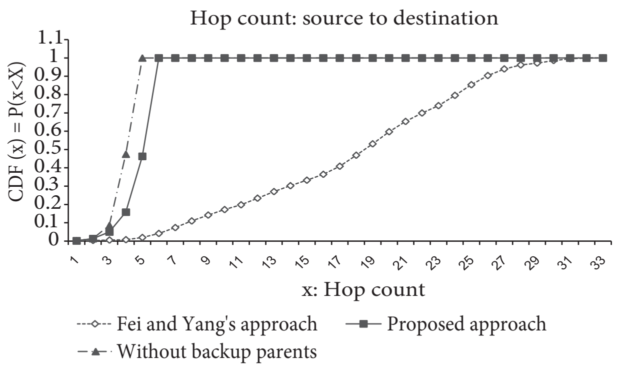


Figure 13. Hop count from source to destination.

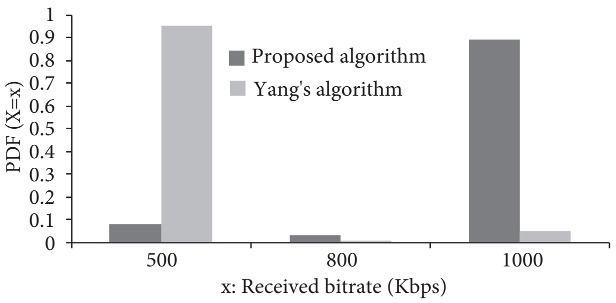


Figure 14. Received bit rates of nodes.

algorithm of [5]. In the proposed system, 90% of the nodes receive video at the highest bit rate (1000 Kbps), whereas this value is 5% in [5].

In video streaming applications, initial buffering is done for possible link or connection failure and nodes continue to play video from their buffer while trying to find a new parent after parent failure is detected. In this respect, maximum recovery time plays an important role to determine initial buffering time, i.e. to play the video in a seamless fashion, the buffer level must be higher than the maximum recovery time. Increasing failure probability of nodes causes increase in tree recovery time since greater numbers of nodes affected by parent failure increase the average time of finding a new parent. In the last set of experiments, we will examine tree recovery times. Following [5], node degrees are uniformly generated as 2, 4, and 6. In Figure 15, tree recovery time for failure rates of $p = 0.5$ and $p = 0.6$ are given in comparative cumulative distribution graphs. More than 90% of recovery times obtained by the proposed algorithm are under 1000 ms, while the same percentage of recovery time of the algorithm proposed in [5] reaches 2000 ms. Similar experiments carried out for $p = 0.1$, 0.2, 0.3, and 0.4 yielded similar conclusions.

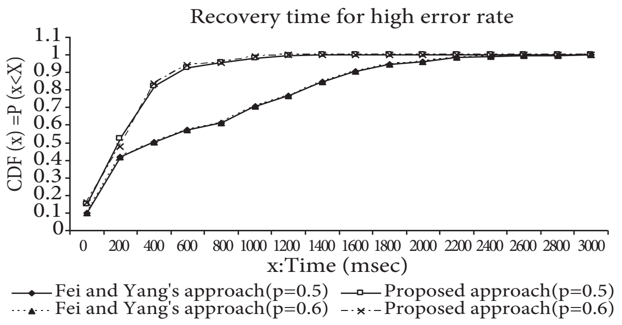


Figure 15. CDF graph of recovery time.

Finally, in Figures 16a and 16b, average and maximum recovery times are given, respectively. It is observed that, for the proposed algorithm, average recovery time stays stable for failure probabilities of nodes higher than $p = 0.3$. From Figure 16b, we observe that the buffer needed for the proposed algorithm can be chosen around 2 s, which is a very reasonable size. On the other hand, the same value should be chosen as 4 s for the algorithm proposed by [5]. This leads to the conclusion that [5] requires a larger buffer to compensate for the higher recovery time, which, in turn, will eliminate the buffer advantage of [5] obtained in the initial waiting time.

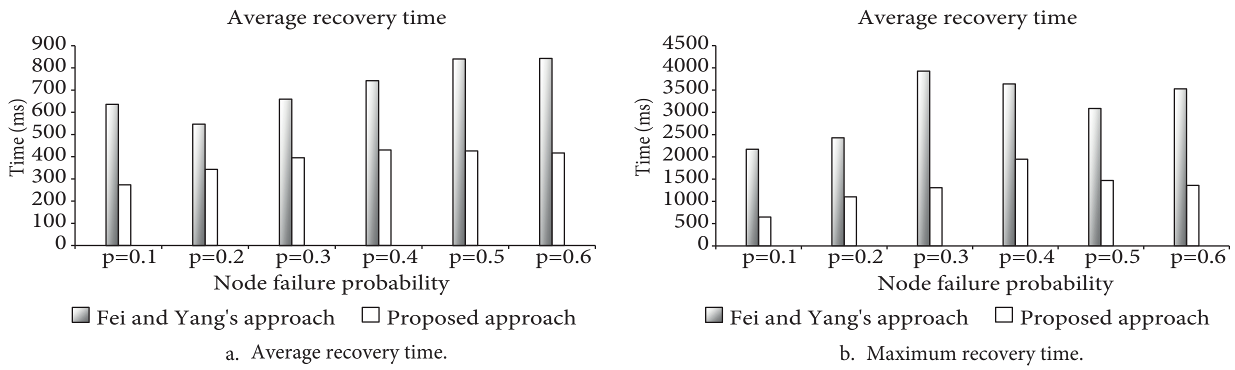


Figure 16. Recovery time: a) average recovery time, b) maximum recovery time.

5. Conclusion

In this paper, we investigated the problem of providing error resiliency in multicast trees. Experimental results show that in terms of initial waiting time, tree recovery time, and hop count, the proposed multicast tree construction with backup parent pools gives better performance than other major works in the literature. Furthermore, simulations show that error resiliency of the proposed system is stable even if failure probability of the nodes in the tree increases. This work is the first study about how to design backup parent related parameters. We proposed an optimal solution to the ratio of backup slots to service nodes and tree height trade-off. According to both theoretical study and simulation results, it is recommended that each node in the tree should reserve 50% of its capacity as backup slots. As for the children of the root, a brute force algorithm is developed to find the optimal number. The proposed algorithm can be used in any application using multicast tree for communication.

Overall, experimental results of the proposed system showed that error resiliency techniques used in a P2P communication system should be adaptive since nodes in the system can join or leave the system unexpectedly. The backup parent pool and ratio of backup slots provide a quick response to the unexpected failures by means of adaptive properties.

References

[1] Y. Chu, S. Rao, S. Seshan, H. Zhang, "A case for end system multicast", IEEE Journal on Selected Areas in Communications, Vol. 20, pp. 1456–1471, 2002.

[2] S. Birrer, F.E. Bustamante, "A comparison of resilient overlay multicast approaches", IEEE Journal on Selected Areas in Communications, Vol. 25, pp. 1695–1705, 2007.

[3] S. Xie, B. Li, G.Y. Keung, X. Zhang, "CoolStreaming: design, theory and practice", IEEE Transactions on Multimedia, Vol. 9, pp. 1661–1671, 2007.

[4] H. Jeon, S.C. Son, J.S. Nam, "Overlay multicast tree recovery scheme using a proactive approach", Computer Communications, Vol. 31, pp. 3163–3168, 2008.

[5] Z. Fei, M. Yang, "A proactive tree recovery mechanism for resilient overlay multicast", IEEE/ACM Transactions on Networking, Vol. 15, pp. 173–186, 2007.

[6] Z. Wang, J. Crowcroft, "Bandwidth-delay based routing algorithms", in IEEE GLOBECOMM, pp. 2129–2133, 1995.

[7] F. Bauer, A. Varma, "Degree-constrained multicasting in point-to-point networks", in IEEE INFOCOM, pp. 369–376, 1995.

[8] G. Raidl, "An efficient evolutionary algorithm for the degree-constrained minimum spanning tree problem", in Congress on Evolutionary Computation, pp. 104–111, 2000.

[9] X. Jin, W.P.K Yiu, S.H.G Chan, Y. Wang, "On maximizing tree bandwidth for topology-aware peer-to-peer streaming", IEEE Transactions on Multimedia, Vol. 9, pp. 1580–1592, 2007.

[10] E. Kim, J. Jang, S. Park, A. Sussman, J.S. Yoo, "Improving resiliency using capacity-aware multicast tree in P2P-based streaming environments", High Performance Computing and Communications, Vol. 4208, pp. 925–934, 2006.

[11] M. Fesci-Sayit, E.T. Tunali, A.M. Tekalp, "Bandwidth-aware multiple multicast tree formation for p2p scalable video streaming using hierarchical clusters", in IEEE International Conference on Image Processing, pp. 945–948, 2009.

[12] S. Birrer, F.E. Bustamante, "Resilient peer-to-peer multicast without the cost", in 12th Annual Multimedia Computing and Networking Conference, 2005.

[13] D.A. Tran, K.A. Hua, T.T. Do, “A peer-to-peer architecture for media streaming”, IEEE Journal on Selected Areas in Communications, Vol. 22, pp. 121–133, 2004.

[14] S. Banerjee, S. Lee, B. Bhattacharjee, A. Srinivasan, ”Resilient multicast using overlays”, IEEE/ACM Transactions on Networking, Vol. 14, pp. 237–248, 2006.

[15] V.K. Goyal, “Multiple description coding: compression meets the network”, IEEE Signal Processing Magazine, Vol. 18, pp. 74–94, 2001.

[16] V.N. Padmanabhan, H.J. Wang, P.A. Chou, “Resilient peer to peer streaming”, in IEEE International Conference on Network Protocols, pp. 16–27, 2003.

[17] P. Baccichet, T. Schierl, T. Wiegand, B. Girod, “Low-delay peer-to-peer streaming using scalable video coding”, in IEEE Packet Video Workshop, pp. 173–181, 2007.

[18] Y. Okada, M. Oguro, J. Katto, S. Okubo, “A new approach for the construction of ALM trees using layered video coding”, in ACM Workshop on Advances in Peer-To-Peer Multimedia Streaming, pp. 59–68, 2005.

[19] T. Kusumoto, Y. Kunichika, J. Katto, S. Okubo, “Tree-based application layer multicast using proactive route maintenance and its implementation”, in ACM Workshop on Advances in Peer-To-Peer Multimedia Streaming, pp. 49–58, 2005.

[20] J. Kim, Y. Lee, S. Bahk, “SALSA: Super-peer assisted live streaming architecture”, in IEEE International Conference on Communications, pp. 1–5, 2009.

[21] M. Fesci-Sayit, E.T. Tunali, A.M. Tekalp, “Resilient peer-to-peer streaming of scalable video over hierarchical multicast trees with backup parent pools”, Signal Processing: Image Communication,, Vol. 27, pp. 113–125, 2012.

[22] F. Wang, Y. Xiong, J. Liu, “mTreebone: A collaborative tree-mesh overlay network for multicast video streaming”, IEEE Transactions on Parallel and Distributed Systems, Vol. 21, pp. 379–392, 2010.

[23] Georgia Institute of Technology, Modeling Topology of Large Internetworks, available at http://www.cc.gatech.edu/projects/gtitm/, date of last access: October 2012.