Survivability Mechanism for Multicast Streaming in P2P Networks

Rober Mayer, Manoel C. Penna, Marcelo E. Pellenz and Edgard Jamhour
Graduate School of Computer Science
Pontifícia Universidade Católica do Paraná - PUCPR
Rua Imaculada Conceição, 1155, 80215-901, Curitiba, Brazil
email: {rober, penna, marcelo, jamhour}@ppgia.pucpr.br

***Abstract*—This paper proposes a mechanism for the survivability of multicast streams on Peer-to-Peer (P2P) overlay networks, capable to protect the network against failures of source and intermediary nodes. Based on a mixed integer programming model, an algorithm is proposed to build the main multicast tree and the necessary backup trees. The mechanism also includes a recovery protocol that identifies the failures and recovers the multicast stream by activating the appropriate backup tree. The proposed mechanism is evaluated by means of numerical experiments, which demonstrate its effectiveness with respect to the quality of the backup multicast trees, to recovery time and to the bandwidth overhead. It is concluded that the proposed mechanism can improve the reliability of multicast streams on static P2P networks with efficiency and quality.**

***Keywords*—*Static P2P Networks*, *Resilient P2P Multicast*.**

I. Introduction

Multicast transmission is an efficient approach for supporting applications involving group communication. The main characteristic of multicast transmission is its ability to eliminate most of the redundant packets necessary to the transmission to multiple destinations. It is particularly useful for video streaming applications, that is, the distribution of video streams from one source to multiple destinations [1].

Multicast protocols are responsible to route the application packets in multicast transmissions. In response to the difficulties of implementing multicast transmission in the IP network layer, several recently proposed protocols adopt the Peer-to-Peer (P2P) communication model, in which all multicast features are implemented exclusively in user hosts instead of routers [2][3]. P2P is a self adaptive communication model implemented in the application layer, where the participating peers are configured in an overlay network. Multicast protocols deployed on P2P networks, called P2P multicasting, are applications protocols that implement multicast features over the P2P unicast links.

In dynamic P2P networks, a peer joins the system when a user starts the application, contributes with some resources while making use of some resources provided by others, and leaves the system when the user exits the application. Such join-participate-leave cycle is a key characteristic in dynamic P2P networks. The independent arrival and departure of peers create a collective effect called churn. On the other hand, in static P2P networks, all participating peers are interested in staying in the system. Consequently, they do not leave the application as frequently as in dynamic P2P networks, reducing considerably the churn effect. In this case, a peer leaving the system can be considered as a failure event. Examples of static P2P multicasting applications are streaming systems applied for dissemination of critical information, Internet Protocol TeleVision (IPTV) system using set-top boxes, and content Delivery Network (CDN).

Since the transmitted content is assumed to be of interest to the applications, the P2P multicasting should provide with delivery guarantees. In general, two basic approaches can be considered to provide network survivability: protection and restoration. The first assumes that backup trees are pre-computed while the second applies dynamic routing, i.e., new backup routes are computed only when a failure occurs. The main advantage of protection is quick reaction to failures, usually providing smaller recuperation time.

The aim of this work is to propose a protection based survivability mechanism for multicast streams on static P2P networks, capable to protect P2P multicasting against failures of both the source peer and intermediary peers. In the first case, a traditional model of flow conservation is used to construct a set of multicast trees, one to be used as the main tree, and the others capable of protecting the multicast stream in case of failure of the source peer. The model is extended to build another set of backup trees that will be used to recover from failures of intermediary peers. A recovery protocol is also necessary in this case, to detect eventual failures and recover the streaming application by switching the transmission to the appropriate backup tree.

The main objective of this study is to propose and validate the recovery protocol. For this reason, in the current state of our proposal, the backup trees are computed off-line according to a model of Mixed Integer Programming (MIP). Although the MIP model provides the optimal set of backup trees, it severely limits the size of the network. We are developing a heuristic algorithm to generate dynamically the backup trees, but it discussion is not in the scope of this work.

A. Motivation and Contributions

In this study, it is assumed that the content of the flows requires delivery guarantees, and thus, survivability mechanisms are necessary. It is also assumed that the participating peers do not leave the streaming application voluntarily and independently. The study is motivated by the growing needs for fast deployment of this kind of streaming services in a reliable and cost effective way. Examples of applications requiring resilient multicast streams in static P2P scenarios are weather forecasts, security notifications, software updates, and traffic information.

Most of previous research concentrates on recovering dynamic P2P networks [4]. They focus on finding many disjoint parents for each peer and on the protocols to recover the multicast routes after the occurrence of events caused by a peer failure or by the churn effect. Usually in this case, the path diversity is reduced by assuming that the set of ancestors of a node in each tree should be as disjoint as possible.

Some few studies consider the resilience of the multicast tree in static P2P networks by applying resilience mechanisms of kind 1:1, in which streaming flows are delivered simultaneously through multiple multicast trees rooted in different source peers [7][10]. Because the recovery actions switch the source peer and the multicast tree when a failure occurs, recovery protocols are not necessary.

In this article, P2P multicasting is investigated at the level of streams as well at the level of network control. At the stream level we present extensions to an integer programming model used to compute the multicast trees. At the network control level we specify and evaluate a recovery protocol. On the other hand, we do not deal neither with multicast streams at the packet level, nor with the join and leave procedures of peer nodes, because the assumption of static P2P network. Note, however, that the proposed method can be applied in conjunction with other proactive and/or reactive methods proposed literature. To the best of the author's knowledge, there is no study addressing the recovery of intermediary node failures in static P2P multicast trees.

B. Organization of the Paper

The remaining of the paper is organized as follows: In Section 2, we present previous research on P2P multicasting with a special focus on survivability. Section 3 presents the proposed recovery mechanism, including the protection model (the mixed integer programming model) and the recovery protocol. In Section 4, the proposed scheme is evaluated, and the results compared with other published studies. Finally, the last section concludes the paper.

II. Related Work

A review of resilient approaches to peer-to-peer multicast is presented by Hongyun et al. [4], where they are organized into three classes, according to overlay construction approach: tree-based, mesh-based and data-driven. Tree based approaches organize the participating peers into a single or multiple logical trees over which the multicast data is transmitted. The studies in [5], [6] and [7] are examples of studies that can be included in this class. In mesh-based P2P multicast the participating peers form a mesh overlay network and the stream content is delivered through routing protocols. An example included in this class is [8]. In data driven protocols, the necessity of data defines the streaming routing, that is, a peer always forwards data to others that are expecting for it. An example in this class is [9].

Our approach can be classified as a tree-based approach, and will be compared to [6], where Probabilistic Resilient Multicast (PRM) is introduced. It is a multicast recovery scheme based in two basic components. The first is a proactive component in which each peer randomly selects a constant number of other peers, forwarding data to each of them with a small probability. This random forwarding occurs in parallel to the usual data forwarding along the multicast tree, leading to a small number of duplicate packets, which are properly detected and suppressed. The second is a reactive mechanism to handle eventual data losses. According to the authors, these mechanisms can together provide high resilience guarantees and can be used to significantly improve the data delivery ratios of application-layer multicast protocols.

In our proposal, multicast trees and backup trees are computed off-line according to a model of Mixed Integer Programming (MIP). We define extensions to the work proposed by Walkowiak et al. [10] [7]. In the first, the authors addressed multicast transmission in static P2P networks by formulating an optimization problem that builds disjoint multicast trees to protect the system against failures of root nodes and Internet Service Provider (ISP) links. The goal is to minimize streaming costs and maximize transmission throughput. Walkowiak et al. [7] presented an extension of the previous work by also considering the delay metric. By evaluating the results from the proposed MIP model they concluded that the P2P multicasting systems can be enhanced with additional protection methods, without significant reduction of system performance. They also presented a heuristic algorithm to solve the problem.

Our proposal differs from [6] because the reconfiguration of routes in the multicast trees is not probabilistic, but based on optimal backup trees computed off-line. Unlike the proposals found in [7] and [10], our approach provides resilience from backup trees that are activated in the event of failures, and not through the simultaneous transmission of multiple streams through the trees for protection.

III. Recovery Mechanism

In this paper, we consider node failures and loss of control messages. Three types of failures are considered and explained in the following: the root node, leaf node and intermediate node. The loss of control messages are handled by the recovery protocol through KEEPALIVE messages and timeout mechanism. Flows are propagated through the multicast tree, and the rooting peer is responsible for the transmission. Intermediary peers are responsible for forwarding the flows and leaf peers only consume the flows. Intermediary peers are also flow consumers. Failure of the root peer is critical, since it completely interrupts flow transmission. To solve this problem one can use multiple backup trees with different roots. In case of failure, the resilience is ensured by switching the responsibility transmission to the root of a backup tree. The failure of intermediary peers is also critical, because all its successors are disconnected from the tree. The solution here is to use multiple backup trees having the same root. The backup

```
1: for each (ROOT in the problem) do
2:     Compute the main tree MT rooted in ROOT according to the standard MIP model
3:     for each (failed peer FP in MT) do
4:         Compute the backup tree BT rooted in ROOT according to the modified MIP model
5:     end for
6: end for
```

Figure 1. Main tree and backup tree computing

trees exclude the failed intermediary node, and are activated only when the corresponding failure occurs. Handling the failure of a leaf node is trivial. Nothing needs to be done because it only leaves a vacant place in the lowest hierarchy.

Failures of root peers are addressed by providing multiple main trees routed on different nodes. Main trees are calculated by using the MIP model proposed by Walkowiak et al. [7], referred in this paper as the standard MIP model. Failures of intermediary nodes need an extension to the standard MIP model in order to allow the construction of backup trees that exclude the failed peer. Also, a recovery protocol is needed. The advantage of this approach is to avoid the waste of bandwidth caused by simultaneous transmissions across multiple main trees, but requires the recovery protocol.

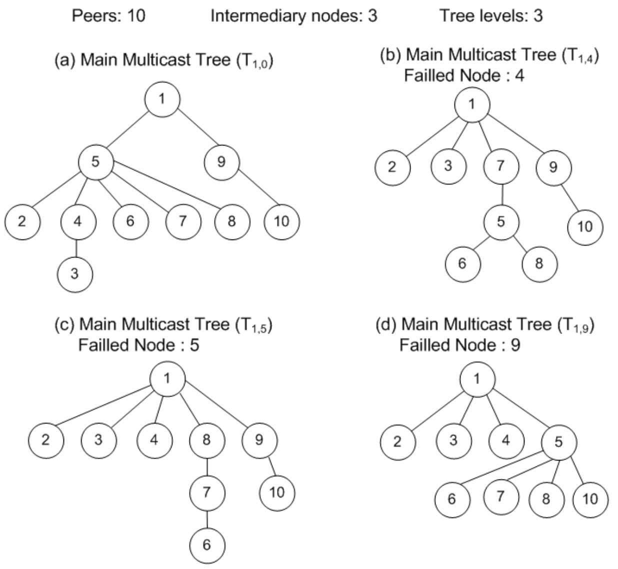
The computation of backup trees is performed before the streaming application starts. The lowest cost backup trees are calculated and stored in recovery tables. Each peer stores a recovery table for each possible intermediary node failure. Because the calculation of recovery tables is performed off-line, the computing time is not critical. The recovery mechanism, named tree switching recovery (TSR) is described is the sequence.

A. Computing the Main Tree and the Backup Trees

A set of multicast trees (main trees) are computed, for the original root and for each backup root, according to the standard MIP model. To protect the streaming application against failures of intermediary nodes, for each main tree, a set of backup trees are computed. For this, the standard MIP model is modified to prevent the failed node to be present in the solution. The modified MIP model is then applied considering the exclusion of every intermediary node present in the main tree, as presented in the algorithm of Figure 1. When more than one failure occurs, the modified MIP model should be executed for each failure combination. The standard MIP model can be reduced to the hop-constrained minimum spanning tree problem, which is NP-complete. This severely limits the size of the network and a heuristic method for backup tree generation is necessary, but it is out of the scope of this study.

B. Backup Trees and Recovery Tables

Multicast trees are identified according to its root peer and failed peer. Assuming that there are R root peers, and I intermediary nodes, main trees and backup trees are identified by $T_{i,j}$, where i is the identifier of the root peer ($i = 1, ..., R$), and j is the identifier of the failed peer ($j = 0, ..., I$). When no



Peers: 10
Intermediary nodes: 3
Tree levels: 3
(a) Main Multicast Tree (T1,0)
(b) Main Multicast Tree (T1,4) Failled Node : 4
(c) Main Multicast Tree (T1,5) Failled Node : 5
(d) Main Multicast Tree (T1,9) Failled Node : 9
1
2
3
4
5
6
7
8
9
10

Figure 2. Example of Main Tree and Backup Trees.

intermediary node is failed, $j = 0$. For example, $T_{1,0}$ identifies the main tree rooted in 1 and $T_{5,3}$ identifies the backup tree rooted in 5 with the intermediary peer 3 failed.

Figure 2 illustrates the multicast trees for a network with ten peers, limited to three levels. The root peer is 1 and there are three intermediary nodes (4, 5 and 9). The main multicast tree $T_{1,0}$ is depicted in Figure 2a, and backup trees $T_{1,4}$, $T_{1,5}$ and $T_{1,9}$ are depicted in Figures 2b, 2c and 2d, respectively.

Recovery tables are identified by R_i, where i is the identifier of the root node. The recovery table of the tree rooted at peer 1 (R_1) in the example of Figure 2 is shown in Table I. Recall that it will be necessary one table for each root peer. As it can be seen, the recovery tables are represented by a matrix structure, where the columns represent the failed peer (0 representing no failures) and the rows represent a peer. Each cell stores at the list of children of the peer in backup tree.

TABLE I . RECOVERY TABLE EXAMPLE

	Failure	0	4	5	9
Node					
1		5,9	2,3,7,9	2,3,4,8,9	2,3,4,5
2		-	-	-	-
3		-	-	-	-
4		3	-	-	-
5		2,4,6,7,8	6,8	-	6,7,8,10
6		-	-	-	-
7		-	5	6	-
8		-	-	7	-
9		10	10	10	-
10		-	-	-	-

C. Recovery Protocol

The recovery protocol includes fault detection and recovery mechanism. Fault detection is based on exchanging KEEPALIVE messages and on a timeout scheme with retries. Every peer (except the root) periodically sends a KEEPALIVE message to its parent. After a period of time without receiving KEEPALIVE messages from one of its children (timeout expiration) and having exhausted the number of retries, a fault is identified. The peer then switches to the backup tree corresponding to the failed peer and sends a reconfiguration message to his father. The message is sent recursively until it reaches the root, which in turn sends an activation message down to their children, informing the recovery table to be activated, which is recursively propagated to the leaf peers.

The recovery protocol is a multi-threaded algorithm. Thread 1, described in the algorithm of Figure 3, implements the recovery procedure, while Threads 2 and 3 (showed in the algorithms of Figures 4 and 5) are responsible for fault detection. The following global variables are defined: the identification of the peer executing the protocol (PiD), the current root of multicast tree (Root), the current recovery table ($RecoveryTable$). Each child peer has a timer (Timer) and a counter of the number of retries (NoR). All messages have three fields, the sender id, the message type and the identification of the failed peed (FiD), if it is the case. Three types of messages are defined: KEEPALIVE, RECOVER and ACTIVATE.

Thread 1 synchronously receives messages (Line 2). For KEEPALIVE messages it just restarts the timer end the number of retries of the corresponding child. When a RECOVER message is received (Line 7), the peer activates the recovery table of the faulty child (Line 8), then propagates a RECOVER message to it is parent (except the root peer) and waits for an ACTIVATE message (Lines 15-17). When the ACTIVATE message arrives, the peer activates the recovery table of the faulty peer (Line 26) and sends an ACTIVATE message to each of its children (Lines 30-34).The flow of RECOVER and ACTIVATE messages corresponds to a tree traversal, with the computational complexity equals to $O(\log n)$.

Thread 2 sends KEEPALIVE messages. Line 10 defines a PAUSE interval between two KEEPALIVE messages. This is a parameter of the recovery protocol, and its setting is discussed in section IV.

Thread 3 is invoked when a timeout occurs for one of the children (in Child variable) of the peer executing the protocol. Line 2 defines a LIMIT on the number of retries before considering Child has failed. This LIMIT is also a parameter discussed in section IV. In line 8 the peer sends a RECOVER message to itself, informing the failure of Child.

```
1:  while TRUE do
2:      RECEIVE (Message)
3:      if Message.Type = KEEPALIVE then
4:          Child = Message.Sender
5:          Timer.Child = $time\_now$ + TIMEOUT
6:          NoR.Child = 0
7:      else if Message.Type = RECOVER then
8:          Activate the $RecoveryTable$ for Messge.FiD
9:          if PiD $\neq$ Root then
10:             Retrive Parent in the active $RecoveryTable$
11:             Message1.Sender = NiD
12:             Message1.Type = RECOVER
13:             SEND (Message1) to Parent
14:             RECEIVE (Message1)
15:             while Message1.Type $\neq$ ACTIVATE do
16:                 RECEIVE (Message1)
17:             end while
18:         end if
19:         Retrieve Children in active $RecoveryTable$
20:         Message1.Sender = NiD
21:         Message1.Type = ACTIVATE
22:         for each Child in Children do
23:             SEND (Message1) to Child
24:         end for
25:     else if Message.Type = ACTIVATE then
26:         Activate the $RecoveryTable$ for Message.FiD
27:         Retrieve Children in active $RecoveryTable$
28:         Message2.Sender = PiD
29:         Message2.Type = ACTIVATE
30:         for each Child in Children do
31:             SEND (Message2) to Child
32:             Timer.Child = $time\_now$ + TIMEOUT
33:             NoR.Child = 0
34:         end for
35:     end if
36: end while
```

Figure 3. Recovery Protocol - Thread 1

```
1:  if (PiD = Root) then
2:      exit
3:  end if
4:  while TRUE do
5:      Retrieve Parent in the active $RecoveryTable$
6:      Message.Sender = PiD
7:      Message.Type = KEEPALIVE
8:      Message. FiD = NONE
9:      SEND (Message) to Parent
10:     SLEEP(PAUSE)
11: end while
```

Figure 4. Recovery Protocol - Thread 2

```
1: NoR.Child = NoR.Child + 1
2: if (NoR.Child $<$ LIMIT) then
3:     exit
4: else
5:     Message.Sender = PiD
6:     Message.Type = RECOVER
7:     Message.FiD = Child
8:     SEND (Message) to PiD
9: end if
```

Figure 5. Recovery Protocol - Thread 3

IV. Evaluation

In this section we evaluate the performance of the proposed resilience mechanism. The evaluation of the application requirements is outside the scope of this study. The resilience mechanism is evaluated according to three distinct aspects: quality of backup trees, recovery time, and bandwidth overhead. The quality of the backup trees is evaluated with respect to their costs. In the standard MIP model, the cost of a solution is given by the delays introduced in all branches of a tree. The same metric is used to evaluate the quality of backup trees, whose costs are compared to the main tree.

To evaluate the recovery time and the bandwidth overhead, the recovery protocol was implemented in an event driven simulator, based on the actors-messages paradigm. According to this paradigm, a simulation model is composed by a set of actors or tasks that communicate among them using messages. Nodes are implemented as task and communication channels are implemented as queues. Regarding the topology of the evaluated networks, it was assumed that all nodes are interconnected by virtual (Internet) links. An important issue was to model the transmission delay in links of the overlay network. Several studies have addressed this point and the following two models for Internet link delay were considered. Hongli [11] used a maximum likelihood estimation procedure to find the best fit distribution to a large set of data measured over the Internet. They found the link delay follows the Gamma distribution. Kaune et al. [12] built a model for Internet delay based on geographical location. This model has two components, the minimum RTT and the jitter. Both models were implemented in the simulation, the first for distances less than 200 km and the other for lager distances. In this simulation scenario the actual position of the nodes is irrelevant and just the distance between each pair of nodes in the backup tree should be considered, which was modeled as a random variable uniformly distributed between 10 and 600 km. The recovery time is evaluated with respect to the amount of peers in the P2P network.

Bandwidth overhead is computed by measuring the extra bytes necessary to provide the survivability feature. We analyzed the bandwidth overhead by measuring the bandwidth consumed for the purpose of reconfiguration, that is, the average number of bytes sent per second in the multicast tree, in addition to normal transmission rate of data. For this case, TSR and PRM are compared. Considering that this study only addresses control of PDP networks, evaluations are made considering only the control plane messages. The traffic model for control messages in the evaluation scenarios is provided in the subsections that follow. All simulations are executed 30 times and the results provided for a confidence level of 99%.

A. Quality of the Backup Trees

The quality of the backup trees is assessed by the average delay costs introduced by the tree (considering all branches). Figure 6 shows the average tree costs with respect to the amount of peers. It can be observed that the costs remain close to those of the main tree, even with the modified restrictions and with one less peer in the backup tree (the failed peer). This is important, because depending on the faulty intermediary peer, the cost of the backup tree could increase due to the lower amount of available network resources.

B. Recovery Time

The recovery time evaluates how fast the recovery mechanism recovers from failures. Recovery time is composed by fault detection time and propagation time. Detection time is the time it takes for a peer to realize that one of its children has failed. Propagation time is the time it takes to the RECOVER message to arrive at the root, plus the time it takes to the ACTIVATE messages to be disseminated along the backup tree.

Detection time depends on the values of parameters PAUSE and LIMIT in the algorithms of Figures 3, 4 and 5. LIMIT is set to 3, allowing at most 2 retries. PAUSE is set as a multiple of the maximum delay time, named here the maximum delay time factor (*MDTF*). The maximum delay time is computed to cover the 95 percentile of the distribution of the delay time.

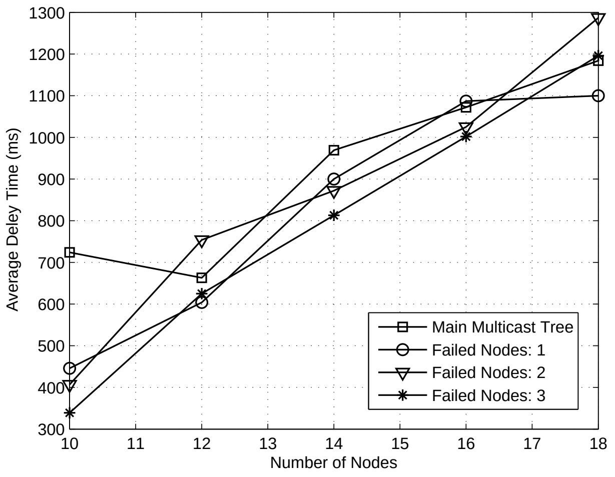


Figure 6. Quality of Backup Trees.

Figure 7 shows the normalized recovery time and the normalized bandwidth overhead for delays varying from 10 to 130ms. The normalized values are plotted for *MDTF* varying from 0.5 to 2.0. We can observe that $MDTF = 1.5$ leads to good values for both recovery time and bandwidth overhead. As a result we defined $MDTF = 1.5$, that is, the PAUSE parameter equals 1.5 times the maximum message time.

In Figure 8, we can see the average recovery time for the size of network from 10 to 18 peers. It can be observed that the recovery time remains stable as the number of peers increases. Although one could expect increasing the recovery time with the network size, since the total cost of the system increases, it remains almost constant from the fact that when the quantity of peers increases, the transmission means (upload and download links) also become more abundant. As a result, peers have more children in optimal trees, decreasing the number of levels and therefore reducing the delay. The graph in Figure 8 shows the average recovery time taken for a sample of 30 simulations, shown with 99% confidence level.

C. Bandwidth Overhead

We call bandwidth overhead, the control information transmitted by the recovery protocol, and we analyzed it by measuring the bandwidth consumed by the recovery protocol. We measured the average number of bits per second sent in the multicast tree for the transmission of control messages. In TSR, control messages are initiated by parents of intermediary peers that failed, and are targeted recursively at their new parents until reaching the root node. In PRM, recovery is achieved by the redundant probabilistic forwarding of data.

To simulate PRM behavior we used the implementation made available by Birrer [13]. We considered from 10 to 18 nodes distributed in sites of PlanetLab [14], with the parameters presented by Birrer et al. [6]. The probability of redundant transmission is set to 0.01. The amount of redundant messages exchanged between peers increases with the number of peers because PRM needs to discover and maintain the list

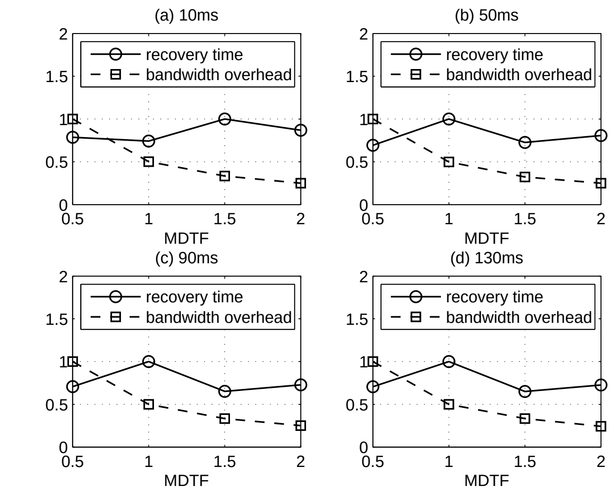


Figure 7. Normalized Recovery Time and Bandwidth Overhead.

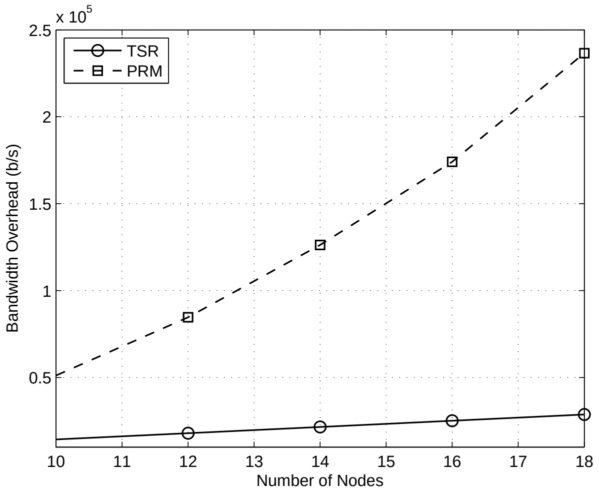


Figure 9. Bandwidth Overhead.

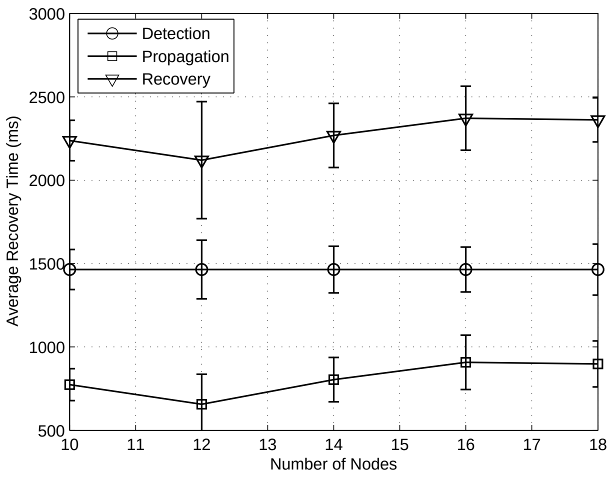


Figure 8. Recovery Time.

of pairs included in random routing. In TSR, the overhead due to control messages is negligible, because a single RECOVER message is sent by the node that detected the fault that runs through part of a branch of the tree to the root. To activate the new configuration, only one ACTIVATE message is sent to each node. On the other hand, the overhead introduced by fault detection is significant and is directly affected by the 2 parameters of the algorithm of Figure 3.

Figure 9 shows how the bandwidth overhead is affected by the size of the P2P network. In both systems (PRM and TSR) we can observe that the bandwidth overhead increases linearly with the number of peers. In PRM this is because the random forwarding of redundant packets, while in TSR the overhead is mainly due to the KEEPLIVE messages. It can be seen that as the size of the network grows, TSR achieves better results when compared to PRM. It may be noted that for 18 peers, the curve of the proposed mechanism is about 8 times lower than the PRM. The reason for this is that the overhead in TSR occurs only by the amount of KEEPALIVE messages needed to test the connectivity between network peers, and because the difference between rates in the control plane and the data plane is very large. The low increase in overhead for TSR indicates that the mechanism has good scalability with respect to bandwidth overhead.

V. Conclusions and Future Work

This paper proposed a recovery mechanism in the P2P multicasting networks named TSR. Its goal is ensure the survival of multicast flows in the presence of failures of intermediary nodes. TSR was evaluated according to three aspects: the quality of multicast tree, the recovery time, and the bandwidth overhead. The quality of backup trees is suitable due to the characteristics of the extended MIP model. It was demonstrated that, even with the additional constraints of the extended MIP model, the cost remains similar to the main multicast tree obtained with the standard MIP model. The recovery protocol was evaluated and it can be concluded that it is not severely impacted by the addition of control messages, because when a node resets the multicast tree, only a few reconfiguration messages go through a few links. The cost converges to a stable value as the number of levels in the multicast tree increases. This is because the size of recovery messages is small, because more transmission resources are made available when the number of peers increases, and because the number of levels in the multicast tree decreases when the amount of peers increase. TSR was also evaluated with respect to the bandwidth overhead, being compared to a classical reactive P2P survivable protocol. The increase of bandwidth overhead in both cases is linear, but the slope is smother in TSR. From the presented results we can conclude that the proposed mechanism improves the reliability of multicast streams on static P2P networks with efficiency and quality. However, the limitations of scale related to the size of the P2P network introduced by the MIP model must be addressed in future work.

References

[1] S. Birrer and F. E. Bustamante, “Resilient Peer-to-Peer Multicast without the Cost,” Twelfth Annual Multimedia Computing and Networking (MMCN), 2005, pp. 113-120.

[2] O. C. Kwon, H. Song, and T. Um, “A robust P2P video multicast streaming system under high peer-churn rate,” Proceedings of the 13th International Conference on Communication Technology (ICCT), 2011, pp. 843-848.

[3] E. G. Mora, C. Greco, B. Pesquet-Popescu, M. Cagnazzo, and J. Farah. Cedar, “An optimized network-aware solution for P2Pvideo multicast,” Proceedings of the 19th International Conference on Telecommunications (ICT), 2012, pp. 1-6.

[4] Y. Hongyun, H. Ruiming, C. Jun, and C. Xuhui, “A Review of Resilient Approaches to Peer-to-Peer Overlay Multicast for Media Streaming,” Proceedings of the 4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM), 2008, pp. 1-4.

[5] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, “Resilient Multicast Using Overlays,” IEEE/ACM Transactions on Networking, 2006, vol. 14, no. 2, pp. 237-248.

[6] S. Birrer and F. E. Bustamante, “Resilience in Overlay Multicast Protocols,” Proceedings of the 14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), 2006, pp. 363-372.

[7] K. Walkowiak and M. Przewozniczek, “Modeling and Optimization of Survivable P2P Multicasting,” Computer Communications, 2011, vol. 34, Issue 12, pp. 1410-1424.

[8] N. Magharei and R. Rejaie, “PRIME: Peer-to-Peer Receiver-Driven Mesh-based Streaming,” Proccedings of 26th IEEE International Conference on Computer Communication (INFOCOM), 2007, pp. 1415-1423.

[9] M. Bienkowski, M. Korzeniowski, and F. Heide, “Dynamic load balancing in distributed hash tables,” Proceedings of the 4th international conference on Peer-to-Peer Systems (IPTPS), 2005, pp. 217-225.

[10] K. Walkowiak, “Survivability of P2P Multicasting, ” Proceedings of the IEEE 7th International Workshop on the Design of Reliable Communication Networks, 2009, pp. 92-99.

[11] Z. Hongli, “Modeling Internet Link Delay Based on Measurement,” International Conference on Electronic Computer Technology, 2009, pp. 420-424.

[12] S. Kaune, K. Leng, A. Kovacevic, G. Tyson, and R. Steinmetz, “Modelling the Internet Delay Space Based on Geographical Locations,” Proceedings of the 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing, 2009, pp. 301-310.

[13] S. Birrer, “Addresing the Limitations of Tree-based Approaches to High-Bandwith Streaming Multicast,” UMI Number: 3303707. Evanston, Illinois, June. 2008.

[14] PlanetLab Consortium, Available at: http://www.planet-lab.org. [retrieved: December, 2013].