A Survey and a Proposed Approach on Robust Tree Overlays

Evangelos Pournaras
Intelligent Interactive Distributed Systems
VU University, Amsterdam
The Netherlands
E.Pournaras@few.vu.nl

Abstract

Tree overlays benefit a wide range of applications as they perform effective and efficient data dissemination and aggregation. However, they are sensitive to failures as local perturbations impact the tree topology globally. Building and maintaining robust tree overlays in distributed environments is the challenge that this paper addresses through an extensive survey of recent approaches.

Inspired by the open issues identified in this survey, this paper also proposes an alternative approach to robust tree overlays, the Adaptive Epidemic Tree Overlay Service (AETOS). The architecture and the fundamental concepts of AETOS are outlined and discussed.

1. Introduction

Hierarchical structures impose effective communication patterns and support the management of distributed systems. In an environment with arbitrarily joins, leaves and failures, such as the Internet, hierarchical topologies are influenced tremendously. The global consequences of local failures in hierarchical topologies can lead to service disruption and underutilization of system resources. This is also the case with tree overlays as a type of multi-level hierarchical topology that this paper discusses.

The robustness of tree overlays is the focus of this paper. The contribution of this work is twofold: (i) an extensive survey on robust, self-organized tree overlays is illustrated and (ii) a proposed approach is outlined, the *Adaptive Epidemic Tree Overlay Service (AETOS)*.

This paper is organized as follows: Section 2 provides an overview of robust tree overlays and related surveys. Section 3 illustrates various robust tree overlays in the context of six defined aspects of investigation. Section 4 discusses the main gap of these approaches and the open issues. Section 5 illustrates an alternative proposed approach on robust tree overlays: the AETOS system. Finally, Section 6 concludes this paper.

2. An Overview of Robust Tree Overlays

Trees are a widely used overlay topology in various applications [25, 29, 17, 11, 6, 9]. The communication cost of search, broadcast and aggregation over trees is low and nodes are based on simple parent-child communication links. In stable environments, or in environments where there is a central control of the topology, trees can be built and maintained relatively easy. However, such central approaches do not scale and the assumption of a stable predictable environment does not represent large-scale distributed systems [9].

Failures in nodes affect their branches underneath resulting in the split of the tree in more than one parts. Consecutive failures can lead to disruption of the whole structure. A failure closer to the root has higher impact than the ones near the leaves. Furthermore, nodes are autonomous and heterogeneous. They can join and leave a system at any time. They have different processing and storing capacity, different bandwidth and can also compete between each other towards satisfying their goals. All these issues affect the stability of the tree overlay tremendously. In addition, links exhibit the delay of the underlying infrastructure, thus messages may experience long delays over the tree overlay.

This paper identifies the lack of a wide-scope investigation on the self-organization of robust tree overlays. Most of the related surveys focus on a specific application. For example, in [5], various methods concerning overlay multicast are examined and classified according to their *cross-link redundancy*, *in-tree redundancy* or *multiple-tree redundancy*. In the survey of [19], overlay multicast technologies are classified according to the node dynamics and the support of an unstructured or structured overlay. In that survey, many of the illustrated approaches are based on tree overlays. Finally, in [32, 21], tree overlays are compared to mesh-based systems in terms of how they handle the network latency in the overlay.

In contrast, this paper aims to identify the crucial aspects of various approaches followed in a range of applications. It also aims to classify them and reach conclusions on what properties are required to build and maintain general-purpose robust tree overlays. In this paper, 22 recent works on robust tree overlays are discussed and classified. Table 1 in Appendix A outlines the illustrated systems of this survey.

3. Aspects of Investigation

Despite the broad scope of the approaches on robust tree overlays, there are some interesting common aspects among recent research work. The *application type* motivates most of the approaches. The self-organization process is driven by *performance metrics* that are defined in the context of each application type. In many cases, the tree overlay is supported by another *complementary overlay* network that provides robustness properties and resilience to failures. The *build and maintenance* is the core aspect in most of the illustrated approaches. Although distributed solutions are the main focus of this work, the *level of decentralization* is examined as well. Finally, an interesting aspect of investigation is how *proactive or reactive* the self-organization process is as it appears that there are different notions of proactiveness and reactiveness in the illustrated approaches.

3.1. Application Type

The majority of the applications concerns *application-level multicast (ALM)* and related video streaming and real-time applications [30, 3, 20, 29]. This fact does not come as a surprise as, in theory, trees appear to be a very effective structure for broadcast. In addition, the restricted spread of IP multicast, due to technical and economical reasons [8] has turned the focus to the use of peer-to-peer tree overlays for multicasting.

Data can be mapped to the nodes of a tree structure for performing efficient queries. This idea was initially adoptoted in database systems. *Complex queries* can be performed over peer-to-peer tree overlays in a fully distributed manner [12, 17]. Building and maintaining a reliable tree overlay is crucial for the data consistency and the extraction of knowledge from a network.

Trees are also useful in *publish-subscribe systems* [7, 11] as they can be used to minimize the changes in the events routing. In grid environments, *tasks allocation* is performed by using strong mobility software agents that self-organize nodes in a tree overlay [6]. Finally, robust spanning trees appear useful for *data collection in sensor networks* and *data dissemination in divisible load scheduling* [9], in which performance comes as a a trade-off with data loss due to node or link failures.

Most of the approaches optimize the self-organization process according to the requirements of a specific application type. Despite the various common concepts among the approaches, their characteristics vary significantly.

3.2. Performance Metrics

One of the core characteristics identified in most of the approaches is the fact that the self-organization is based on or driven by one or more performance metrics. These concern either the underlying infrastructure, the application or the performance profile of the individual machines. Some of the most important performance metrics and their influence on the robustness of the tree overlay are discussed below:

- **Delay**: A tree overlay can exhibit long delays when messages are exchanged. For example, if the root of the tree is connected with its children via long-delay links, this results in all the nodes underneath experiencing this delay. Related metrics with the delay are the *hops count* and *eccentricity* that are considered in [9].

- **Bandwidth**: The bandwidth profile of nodes affect the performed operations over the tree overlay. In most cases, high-bandwidth nodes should be placed near the root. In some approaches [10], bandwidth is related to the *node degree*, that is the number of children that each node is connected to. In another related work [34], bandwidth and node degree are considered independent performance metrics.

- **Node degree**: The number of children influences the local state of a node. It can be adjusted according to the bandwidth or the processing capacity of the node. Besides the local influence, there is also a global influence over the tree overlay. Configuring the node degree can lead to a *balanced tree (degree-bounded)* [30], a *fat (wide)* or a *long* one [9, 29]. This global overlay topology configuration can potentially affect the performance of the application significantly and it is related to performance trade-offs [18].

- **Uptime**: This metric refers to the lifetime connectivity of a node. It is also related to its *availability* according to some studies [29, 27]. In [15], the *sojourn probability* is considered as a metric of denoting the availability of nodes. The *join time* of nodes is considered for the optimization of segments transmission in video streaming[20].

Optimizing the tree overlay requires consideration of various metrics and factors that affect the application. For example, a bandwidth-ordered tree overlay is a wide one with many nodes in the top-levels. This may lead to a violation of the node degrees. Furthermore, a tree that is an uptime-ordered one is usually longer with fewer connections in the top levels. In these cases, studies propose the integration of the performance enhancements. The work in [29] discusses how such a bandwidth-ordered tree overlay version can be combined with a time-ordered one towards increasing the overall performance. This integration is performed by locally calculating the *service capability contribution (SCC)*, that is the product of the bandwidth value and the age of a node. In [9], the *path weight* and the *hop count* are weighted towards achieving an effective trade-off between data loss, under node and link failures, and performance. This is how that work deals with the notion of robustness. A similar weighting scheme is proposed in [18], by considering the *relative delay penalty* and the *resource usage*.

In a similar concept, but following a different approach, the work in [15] aims to maximize the minimum link reliability in multiple sessions of Internet broadcasting. The idea of distinct link reliabilities for every session is adopted. The link reliabilities are measured from the sojourn probabilities of the two linked nodes for a specific session.

Weighting schemes are also proposed in the neighboring selection. For example, [31] proposes a scheme with one random and five proximity links in the overlay view of nodes to eliminate the effects of node and link failures. This schemes retains the tree connectivity even after 23% of node failures. Finally, the work in [33] discusses the *Mahalanobis distance*. This metric calculates the correlation of different adaptation attributes towards detecting outliers. The latter is useful in the case of malicious nodes that aim to perform *attraction*, *repulsion* and *disruption* in the self-organization process of the tree overlay.

3.3. Complementary Overlay Support

Many of the approaches discussed in this paper introduce a complementary overlay, individual (central) entities or additional overlay links to increase and support the robustness of the tree overlay.

Mesh overlays are used to support tree overlays in various approaches that concern multicasting. Meshes provide higher robustness as there is redundancy in case of link failures. However, they introduce problems with duplicate messages and suffer from the trade-off between efficiency and latency. For this reason, various approaches try to benefit from both tree and mesh overlays: RESMO [18] builds and maintains a minimum-delay, minimum-resource usage spanning tree over a mesh overlay by considering only links with sufficient bandwidth. Similarly, in [34], the stable nodes of an underlying mesh overlay are organized in a backbone tree overlay. Finally, MeshTree [30] is a mesh overlay that emerges from the insertion of shortcut links over a tree overlay whose nodes are topologically close.

The idea of retaining additional, special-purpose links is adopted in some other cases as well. For example, besides the *parent and children links*, BATON* [12] also retains: (i) *adjacent links* that map adjacent range of values for the support of complex queries and (ii) *neighbor links* that point to nodes in the same level of the tree. TAG [20] and PRM [4] use gossip and random links respectively in order to deal with data loss and discontinuous playback in real-time applications. GoCast [31] also uses and combines a random and a proximity-based gossiping protocol to increase resilience.

Structured peer-to-peer overlays can also support tree-based ones. In [7], a case with a DHT is illustrated that benefits from controlling the number of neighbors.

Some other approaches introduce central entities to manage the tree overlay. For example, DPOCS [1] proposes a dynamic proxy architecture containing an *overlay control server (OCS)* that manages the topology of the tree overlay and distributes video streams from the source. Similarly, OMNI [3] introduces *multicast server nodes (MSNs)* for managing multiple mutlicasting groups. BulkTree [2] groups and manages nodes in clusters. Each of them represents a *super-node* in the tree overlay. The tree becomes more scalable and robust but there is an additional overhead to maintain the super-nodes.

Finally, this survey identifies two special interesting cases: (i) DPTree [17] is a peer-to-peer overlay that is aware of the tree structure but fully decoupled from it. This is achieved by building a skip graph and matching the naming scheme of peers among the skip graph and the tree overlay. (ii) The approach in [14] proposes the exploitation of the underlying *Internet Indirection Infrastructure (i3)* [28] for reliable data delivery in the upper tree overlay.

3.4. Build and Maintenance

In the investigated approaches illustrated in this paper, the building process of tree overlays is either integrated with their maintenance, for example in [16], or it serves as a bootstrapping mechanism for the maintenance that follows, e.g. [3].

The main method that is used for building a tree, or an initial version of it, is the consecutive *joins* to candidate parents and children [30] or to the leaves of the tree [29]. These candidates are derived randomly [15] or from their proximity to the local node [20]. The proximity is defined by performance metrics related to the ones discussed in Section 3.2.

After the initial joins, nodes either aim to improve their position in the tree or they cooperate to optimize the tree topology. In the first case, nodes perform *shift-up* operations [29] by moving to an upper level in the tree, whereas, in the second case, a parent and one of its children *swap* their positions [1, 12].

Plumtree [16] combines *eager and lazy push* gossiping strategies to build and maintain a tree overlay. The node-key mapping of the underlying DHT in [7] is used to form the tree overlay. Alternative methods for the distributed building of a tree overlay include the top-down approach proposed in [18], the Bellman Ford [9] and Prim's [10] algorithm.

The maintenance of tree overlays, either proactive or reactive, is based on various strategies of reconnections. Usually, nodes monitor the connectivity of their neighbors by sending heartbeats [20, 17]. In case of a failure, they try to connect with another node. TreeOpt [23] improves the tree connectivity by performing two types of children moves as an evolutionary optimization of the tree overlay. In [11], a candidate parent is selected by applying and combining different repair strategies related to the application requirements. Similarly in [6], ancestor lists are retained in case of failures. In contrast, the proposed approach in [10] defines a *parent-to-be* for every node (besides the root) before a failure occurs. Thus the repair is faster.

Other techniques propose link redundancy in order to satisfy alternative connectivity in case of failures [12, 34]. Load-balancing also supports the maintenance of tree overlays by aiming to retain the load in the nodes between root and leaves equal [17, 12].

3.5. Decentralization Level

This survey focuses on distributed approaches, thus in most of them nodes are based on local and partial knowledge of their environment. They autonomously connect with or disconnect from other nodes. They are also able to react to the perturbations of the environment [16, 6, 33]. From this viewpoint, the global system is self-organized.

However, some of the illustrated systems are hybrid. DPOCS [1] is based on the *overlay control server (OCS)* that assists nodes to join the multicast groups. OMNI [3] and TAG [20] follow a similar concept by introducing the *multicast server nodes (MSNs)* and a *content server* respectively. mTreebone [34] utilizes only stable nodes for video multicasting. BulkTree [2] groups the nodes to *super-nodes* in order to increase the stability of the tree. Finally, the approach of [15] is based on a video broadcasting source node that centrally collects and calculates statistics based on which the self-organization process performs.

3.6. Proactiveness vs. Reactiveness

Defining the level of proactiveness and reactiveness is hard. Most of the approaches are based on operations that incorporate both notions.

The cases that introduce an underlying complementary robust overlay [30, 34, 31, 16] are a form of proactive approach towards the robustness in the upper tree overlay. Link and data redundancy [12, 20, 4, 17] is also a proactive approach for the resilience of the tree to failures. The same also holds in case of an ordered tree based on a robustness metric [29]. In [10], a more conceptually proactive approach is proposed. Nodes calculate the new parents for their children before a failure occurs and without violating the node degrees.

In contrast, reactive nodes monitor their neighbors [17] and perform reconnections to other nodes when a failure occurs. Usually the selection of the nodes is based on various strategies [11] that balance performance trade-offs similarly to the ones discussed in Section 3.2.

Proactive approaches benefit from the fact that they aim to decrease the complexity and time of the repair actions or the impact of failures. However, proactive approaches introduce: (i) a usually constant but (ii) significant communication and processing cost. How comparable this cost is with the one of reactive approaches is something that must be considered in systems that choose one approach over the other.

4. Discussion and Open Issues

The *main gap* that this survey identifies in almost all of the illustrated approaches *is the lack of a generic mechanism that could optimize a tree overlay for any type of application*. The illustrated approaches *do not manage to keep the self-organization process independent of the application requirements*. However, there are plenty of interesting characteristics on them that could be used towards this effort.

Performance metrics provide the notion of robustness for different types of applications. In cases where more than one opposing metrics should be used, weighting schemes such as the ones proposed in [9, 18] appear to be effective to deal with the performance trade-offs. This paper proposes two alternative approaches for the future research directions: (i) the input of a singe performance metric to the self-organization mechanism. This single metric may represent a set of different application requirements. It should be a result of a utility function calculated in the application level and not in the level of the tree self-organization. (ii) the support of multiple criteria in the self-organization process. Howerer, it is not clear how this approach could be realized effectively without increasing the complexity.

Unstructured overlays have a higher robustness than trees. Integrating or setting up two overlays to work together and cooperate can enhance the robustness of the tree considerably. However, quantifying the gain, in matters of robustness, and the losses, in matters of complexity, is hard and it is something that future research must shed light on.

In most of the approaches illustrated, effort was put on maintaining the tree overlay rather than building it. Swaps, and shifts-up operations appear effective to optimize and organize the tree even under the effect of churns. However, more effort is required to clarify how fast these methods perform in large-scale distributed environments especially when catastrophic failures occur. In the latter case, a quantitative indication of the resilience in failures should be provided, such as the one presented in [31].

A high decentralization level is hard to be achieved in some certain cases. For example, when a tree is built and used by the application, the termination of the self-organization process may be required. Nodes have only local control. ”Freezing” the tree overlay for providing consistency to the application can be performed relatively easy in central points. However, distributed local control is still a challenge and open issue, especially for enabling the application to use and improve a tree overlay on-demand.

Ordering the tree according to performance metrics can improve the proactive resilience of trees to failures and their impact. Furthermore, highly proactive approaches, such as the one in [10], can enable the system to react fast to perturbations, thus they should be adopted and investigated further in future work. Finally, self-organized trees can exhibit the same vulnerability of peer-to-peer systems to various types of attacks [33, 24], thus security is an open issue as well.

5. An Alternative Approach: AETOS

Inspired by some of the open issues and the lack of a generic mechanism for robust tree overlays, this paper proposes *AETOS, the Adaptive Epidemic Tree Overlay Service*. AETOS builds and updates a tree overlay *on-demand*. It is based on a single metric based on which nodes are *sorted* in the tree overlay. It is supported by *two gossiping protocols* that provide high robustness and connectivity. The building and maintenance process is realized by *four types of exchanged messages*. It is a *highly distributed system* with *both proactive and reactive* components.

This section provides a brief overview of AETOS. For a detailed illustration readers are referred to the complementary works in [26].

5.1. Architecture Overview

AETOS is based on a 3-layer architecture illustrated in Figue 1. It is placed between the application and the underlying network infrastructure.

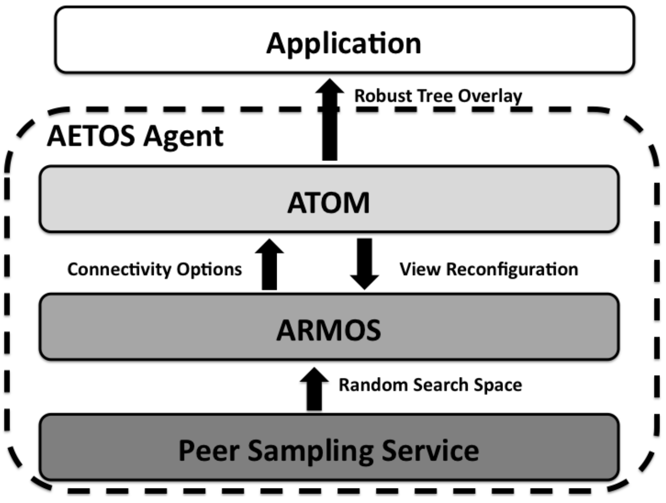

Figure 1. The 3-layer architecture of AETOS.

The bottom layer of AETOS is consisted of the *Peer Sampling Service* [22]. This is a gossiping protocol that maintains highly connected robust random graphs. This choice is made to introduce the robustness properties of random graphs in AETOS and increase the connectivity of the nodes that participate.

On top of it, ARMOS lies, this is *A Rank-based Middleware Overlay Service*. ARMOS is a proposed variation of the T-MAN gossiping protocol. In contrast to T-MAN that uses a static ranking function, ARMOS builds various topologies based on dynamic proximity criteria. The motivation for the use of ARMOS in AETOS is the clustering of nodes according to a robustness metric. This serves the sorting process of the tree overlay. Section 5.2 illustrates how nodes use dynamic proximity criteria.

Finally, the top-lays is the *Adaptive Tree Overlay Management (ATOM)*. ATOM is responsible for the configuration of the parent-child connections. It receives connectivity options from ARMOS and also provides feedback to it. Section 5.3 illustrates the role of the ATOM layer.

5.2. Myopic View Reconfiguration

ARMOS is based on the *myopic view*. The myopic view consists of the sorted sets of *candidate parents and children*. The sorting is performed according to the value of robustness for every node that belongs to these sets. Figure 2 illustrates this concept. From the set of robustness values, the local node chooses the more robust nodes for its potential neighbors. The two bold curved arrows in Figure 2 depict the neighboring selections.

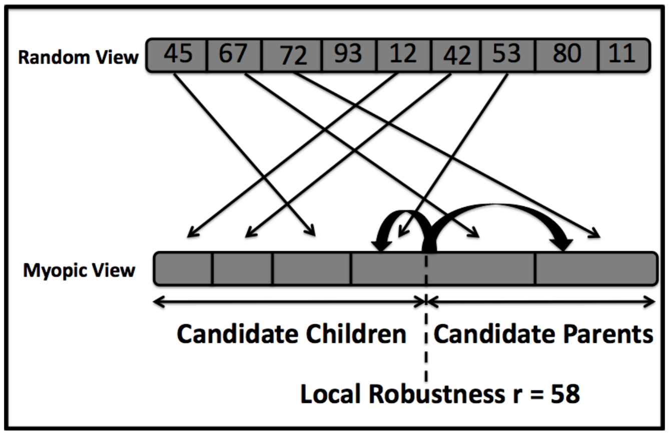

Figure 2. The formation of the myopic view and the default neighboring selections.

Nodes perform *reconfigurations* in their myopic view in order to make it dynamic and provide more flexibility in the proximity criteria. *The concept of a reconfiguration is the exclusion or inclusion of areas from or to the myopic view*. In other words, the range of robustness values is shifted right or left. The number of nodes in the myopic view can vary but without violating the maximum length of it. This enables nodes to find their best neighbors despite their local view of the environment and despite changes in the values of robustness.

More information about the defined reconfiguration of the myopic view are illustrated in [26].

5.3. Adaptive Tree Overlay Management

In the top ATOM layer, the neighboring connections are configured by using 4 exchanged messages: the *request*, *rejection*, *acknowledgment* and *removal* messages. After an interaction, nodes reconfigure their myopic view for either: (i) find a neighbor to connect to if the tree neighboring list is not filled or (ii) improve their position in the tree by changing tree neighbors.

A detailed illustration of the ATOM functionality is illustrated in [26].

5.4. On-demand Building and Maintenance

In contrast to the existing approaches that this paper illustrates, AETOS provides full *local* control of the *bootstrapping and termination* of the self-organization process to the application. Every application client calls the local AETOS service through the local *AETOS proxy*. The call is a request for a new tree overlay or the improve of an existing one. The application is also the one that parametrizes the self-organization process by giving as input: (i) the number of children c that can be supported, (ii) the value of the local robustness metric, (iii) QoS parameters such as how long AETOS should spend trying to optimize the tree overlay. Figure 3 depicts the proposed model.

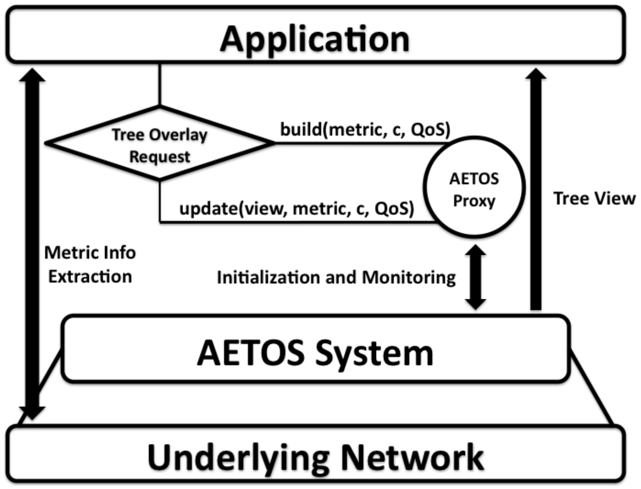

Figure 3. The AETOS proxy and the interaction between the application and AETOS.

The termination is similar to the one proposed for T-MAN [13]. However, the local node participates again in the self-organization process if and only if the application requests an improvement of the tree overlay. Otherwise the parent-children connections remain locked.

5.5. An Example of Self-Organization

Calculations were performed in a small-scale environment to investigate the convergence of AETOS. The calculations provide a first indication of the AETOS behavior.

Nodes, 10 in total, are placed in a binary tree ($c = 2$ for every node). The nodes receive random values in the range (0,100). These values express the robustness. Every node retains a random view of 3 other ones, a myopic view with 2 candidate parents and 4 candidate children (its maximum length is equal to 6) and its tree neighboring list, that is the parent and its children.

AETOS runs in discrete rounds. In every round: (i) the peer sampling service protocol updates the random view, (ii) ARMOS updates the myopic view by applying the appropriate view reconfigurations as well, and (iii) one parent and two children requests are sent.

After every round, the connectivity of the tree is evaluated and the ordering of the nodes over the tree as well. The calculations reveal that there is already a fully connected tree at the end of the 1st round. As myopic view reconfigurations are performed, the nodes tend to find the appropriate neighbors. Finally, the system converges to the required tree topology on the 5th round.

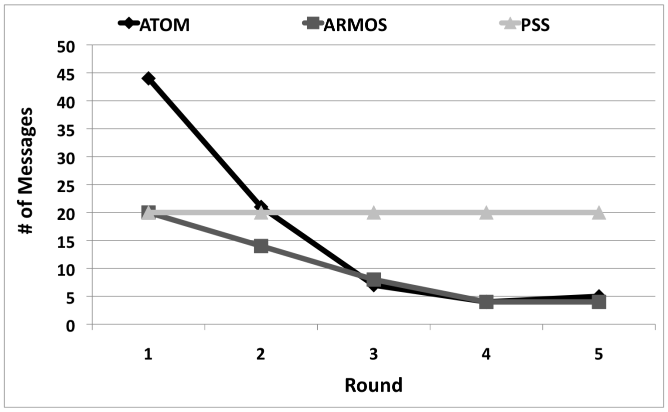

Figure 4. The number of exchanged messages for each layer of the AETOS architecture.

The number of generated messages per round has been calculated as an illustration of the AETOS communication overhead. Figure 4 depicts the number of messages for each layer of the architecture. The graph reveals that, in this example, the system converges. The number of generated messages from the ATOM and ARMOS layers decreases gradually as the myopic views are reconfigured and nodes end with their final best parent and children. In contrast, the Peer Sampling Service has a constant communication cost as it functions independently from the other two upper layers.

Although definite conclusions cannot be reached by this small-scale experiment, they do provide a positive indication and motivation to further perform large-scale simulations in future work.

6. Conclusions

This survey paper illustrates and covers the recent work of the self-organization in robust tree overlays. The proposed approaches are discussed from the viewpoint of 6 aspects: *application type*, *performance metrics*, *complementary overlay support*, *build and maintenance*, *decentralization level* and *proactiveness versus reactiveness*. The main gap identified is the lack of a generic mechanism for robust tree overlays that can satisfy different types of applications. Complementary overlays can increase the robustness if they are used effectively. Finally, dynamic systems should be both proactive and reactive to the unpredictable failures of distributed environments.

This papers also proposes *AETOS, the Adaptive Epidemic Tree Overlay Service*. AETOS comes as an effort towards bridging the main gap that this paper identifies. It is a generic, both proactive and reactive, fully decentralized self-organization mechanism that is based on two gossiping protocols. Further extensive experimental work will provide more insights of the AETOS effectiveness.

Acknowledgments

This project is supported by the NLnet Foundation `http://www.nlnet.nl`.

References

[1] B. Akbari, H. R. Rabiee, and M. Ghanbari. DPOCS: A Dynamic Proxy Architecture for Video Streaming Based on Overlay Networks. In *IEEE MICC & ICON '05*, 11 2005.

[2] G. An, D. Gui-guang, D. Qiong-hai, and L. Chuang. Bulk-Tree: An overlay network architecture for live media streaming. *Journal of Zhejiang University*, 7(1):125–130, 2006.

[3] S. Banerjee, C. Kommareddy, K. Kar, S. Bhattacharjee, and S. Khuller. Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications. In *INFOCOM*, 2003.

[4] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan. Resilient multicast using overlays. *IEEE/ACM Trans. Netw.*, 14(2):237–248, 2006.

[5] S. Birrer and F. E. Bustamante. A Comparison of Resilient Overlay Multicast Approaches. *IEEE Journal on Selected Areas in Communications*, 25(9):1695–1705, 2007.

[6] A. J. Chakravarti, G. Baumgartner, and M. Lauria. The organic grid: self-organizing computation on a peer-to-peer network. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 35(3):373–384, 2005.

[7] P. Costa and D. Frey. Publish-Subscribe Tree Maintenance over a DHT. In *ICDCSW '05: Proceedings of the Fourth International Workshop on Distributed Event-Based Systems (DEBS) (ICDCSW'05)*, pages 414–420, Washington, DC, USA, 2005. IEEE Computer Society.

[8] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen. Deployment issues for the IP multicast service and architecture. *Network, IEEE*, 14(1):78–88, 2000.

[9] D. England, B. Veeravalli, and J. B. Weissman. A Robust Spanning Tree Topology for Data Collection and Dissemination in Distributed Environments. *IEEE Transactions on Parallel and Distributed Systems*, 18(5):608–620, 2007.

[10] Z. Fei and M. Yang. A proactive tree recovery mechanism for resilient overlay multicast. *IEEE/ACM Trans. Netw.*, 15(1):173–186, 2007.

[11] D. Frey and A. L. Murphy. Failure-Tolerant Overlay Trees for Large-Scale Dynamic Networks. In *P2P '08: Proceedings of the 2008 Eighth International Conference on Peer-to-Peer Computing*, pages 351–361, Washington, DC, USA, 2008. IEEE Computer Society.

[12] H. V. Jagadish, B. C. Ooi, K.-L. Tan, Q. H. Vu, and R. Zhang. Speeding up search in peer-to-peer networks with a multi-way tree structure. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 1–12, New York, NY, USA, 2006. ACM.

[13] M. Jelasity, A. Montresor, and O. Babaoglu. T-Man: Gossip-based Fast Overlay Topology Construction. *Elsevier Computer Networks*, 2009. To appear.

[14] K. Lakshminarayanan, A. Rao, I. Stoica, and S. Shenker. End-host controlled multicast routing. *Comput. Netw.*, 50(6):807–825, 2006.

[15] C. Y. Lee and H. Dong Kim. Reliable overlay multicast trees for private Internet broadcasting with multiple sessions. *Comput. Oper. Res.*, 34(9):2849–2864, 2007.

[16] J. Leitao, J. Pereira, and L. Rodrigues. Epidemic Broadcast Trees. In *SRDS '07: Proceedings of the 26th IEEE International Symposium on Reliable Distributed Systems*, pages 301–310, Washington, DC, USA, 2007. IEEE Computer Society.

[17] M. Li, W.-c. Lee, and A. Sivasubramaniam. DPTree: A Balanced Tree Based Indexing Framework for Peer-to-Peer Systems. In *ICNP '06: Proceedings of the Proceedings of the 2006 IEEE International Conference on Network Protocols*, pages 12–21, Washington, DC, USA, 2006. IEEE Computer Society.

[18] Y. Li and W. T. Ooi. Distributed construction of resource-efficient overlay tree by approximating MST. In *ICME*, pages 1507–1510, 2004.

[19] Z. Li and Y. Shin. Survey of Overlay Multicast Technology. June 2002.

[20] J. Liu and M. Zhou. Tree-assisted gossiping for overlay video distribution. *Multimedia Tools Appl.*, 29(3):211–232, 2006.

[21] Y. Liu, Y. Guo, and C. Liang. A survey on peer-to-peer video streaming systems. *Peer-to-Peer Networking and Applications*, 1(1):18–28, 2008.

[22] Márk Jelasity and Spyros Voulgaris and Rachid Guerraoui and Anne-Marie Kermarrec and Maarten van Steen. Gossip-based peer sampling. *ACM Trans. Comput. Syst.*, 25(3):8, 2007.

[23] P. Merz and S. Wolf. TreeOpt: Self-Organizing, Evolving P2P Overlay Topologies Based On Spanning Trees. In *SAKS'07*, Bern, Switzerland, 2007.

[24] J.-D. Mol, D. H. J. Epema, and H. J. Sips. The Orchard Algorithm: Building Multicast Trees for P2P Video Multicasting Without Free-Riding. *IEEE Transactions on Multimedia*, 9(8):1593–1604, 2007.

[25] E. Pournaras, M. Warnier, and F. Brazier. A Distributed Agent-based Approach to Stabilization of Global Resource Utilization. In *Proceedings of International Conference of Complex Intelligent and Software Intensive Systems (CISIS'09)*, March 2009. (to appear).

[26] E. Pournaras, M. Warnier, and F. M. T. Brazier. Adaptive Agent-based Self-organization for Robust Hierarchical Topologies. In *ICAIS '09: Proceedings of the International Conference on Adaptive and Intelligent Systems*. IEEE, September 2009. (to appear).

[27] S. Sen and J. Wang. Analyzing peer-to-peer traffic across large networks, booktitle = IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment. pages 137–150, New York, NY, USA, 2002. ACM.

[28] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet indirection infrastructure. In *SIGCOMM*, pages 73–86, 2002.

[29] G. Tan, S. A. Jarvis, X. Chen, and D. P. Spooner. Performance Analysis and Improvement of Overlay Construction for Peer-to-Peer Live Streaming. *Simulation*, 82(2):93–106, 2006.

[30] S.-W. Tan, G. Waters, and J. Crawford. MeshTree: Reliable Low Delay Degree-bounded Multicast Overlays. *Parallel and Distributed Systems, International Conference on*, 2:565–569, 2005.

[31] C. Tang and C. Ward. GoCast: Gossip-Enhanced Overlay Multicast for Fast and Dependable Group Communication. In *DSN '05: Proceedings of the 2005 International Conference on Dependable Systems and Networks*, pages 140–149, Washington, DC, USA, 2005. IEEE Computer Society.

[32] K.-H. Vik, C. Griwodz, and P. Halvorsen. Constructing low-latency overlay networks: Tree vs. mesh algorithms. In *LCN*, pages 36–43. IEEE, 2008.

[33] A. Walters, K. Bauer, and C. Nita-Rotaru. Towards Robust Overlay Networks: Enhancing Adaptivity with Byzantine-Resilience. Technical Report CSD TR 05–026.

[34] F. Wang, Y. Xiong, and J. Liu. mTreebone: A hybrid tree/mesh overlay for application-layer live video multicast. In *in IEEE ICDCS*, page 49, 2007.

Appendix

A. Outline of the Survey

Table 1 outlines the self-organized tree overlays that are illustrated in this paper. Note that the AETOS system is also included in the table for comparison with the illustrated systems.

Table 1. Outline of the self-organized robust tree overlays illustrated in this paper.

System	Application	Performance Metrics	Complementary Overlay	Build & Maintenance	Decentralization	Pro/Re-active
[9]	Sensor networks, load scheduling	hop count, path weight	No	Bellman Ford	Central and distributed	proactive
RESMO [18]	ALM	delay, resource usage	mesh	top-down, reconnections	distributed	reactive
DPOCS [1]	video streaming	delay, bandwidth, node degree	Overlay Control Server (OCS)	joins, swaps, reconnections	hybrid	reactive
MeshTree [30]	ALM	delay, node degree	backbone, delivery and mesh links	joins, rewiring	distributed	reactive
OMNI [3]	multicasting and real-time applications	delay, node degree	Multicast Server Nodes (MSNs)	central sorting, swaps, promotions	hybrid	reactive
BATON* [12]	multi-attribute queries	data	adjacent and neighbor links	joins, swaps, load-balancing	distributed	both
mTreebone [16]	video multicasting	delay, bandwidth, node degree	mesh	joins, link redundancy	hybrid	reactive
TAG [20]	real-time asynchronous streaming	time indexing, delay, bandwidth	gossiping links	joins, heartbeats, reconnections	hybrid	both
[29]	live streaming	uptime, bandwidth	No	joins, shifts-up	distributed	both
BulkTree [2]	live media streaming	delay	super-nodes	joins and leaves management	hybrid	reactive
[15]	Internet broadcasting with multiple sessions	delay, link capacity, node degree, node sojourn probabilities	No	random tree, link swaps (Tabu search), reconnections	hybrid	reactive
[33]	ALM	latency, bandwidth	No	detection, response, recovery, local temporal and spatial data correlation	distributed	reactive
[10]	ALM	latency, node degree	No	Prim algorithm, proactive parent-to-be calculation	distributed	proactive
[11]	publish-subscribe	node degree	No	reconnection strategies	distributed	reactive
[7]	publish-subscribe	node degree, node keys	DHT	node-key mapping, joins and leaves management	distributed	reactive
PRM [4]	ALM	latency, losses	random links	randomized forwarding, retransmissions, loss correlations, Ephemeral Guaranteed Forwarding (EGF)	distributed	both
TreeOpt [23]	ALM	delay	epidemic protocol	two types of local children moves	distributed	reactive
Organic Grid [6]	grid load scheduling	delay, bandwidth, node degree	strong mobility software agents	friend-of-friend joins, reconnections (ancestor lists), prefetching	distributed	both
Plumtree [16]	broadcast	No	peer sampling (HyParView)	eager and lazy puch gossiping	distributed	both
GoCast [31]	ALM	delay, node degree	random and proximity-based gossiping	insertions, removals and replacements of random and proximity neighbors	distributed	both
DPTree [17]	complex queries of multidimensional data	No	skip graph decoupled from the tree overlay	assignment of tree branches to nodes, heartbeats, load-balancing and navigation for data insertion and removal	distributed	both
[14]	ALM	latency, node degree	Internet Indirection Infrastructure [28]	joins and refresh messages	distributed	reactive
AETOS [26]	generic	generic	Peer Sampling Service [22], T-MAN [13]	candidate neighbors from the myopic view exchange request, rejection, acknowledgment and removal messages	distributed	both