

Adaptive Self-organization in Distributed Tree Topologies

Evangelos Pournaras¹, Martijn Warnier² and Frances M.T. Brazier²

¹Chair of Sociology, in particular Modeling and Simulation

ETH Zurich, Zurich, Switzerland

²Faculty of Technology Policy and Management

Delft University of Technology, Delft, The Netherlands

Tree topologies are often deployed in large-scale distributed systems to structure a hierarchical communication. Building and maintaining overlay networks self-organized in tree topologies is challenging to achieve in dynamic environments. Performance trade-offs between resilience to failures and message overhead need to be considered. This paper introduces eight adaptation strategies that provide a higher abstraction, modularity and reconfigurability in the tree self-organization process. Performance can be further enhanced by dynamically changing strategies during system runtime. Experimental evaluation illustrates the performance trade-offs and properties of adaptation strategies.

Keywords: tree topology, overlay network, self-organization, software agent, adaptation

Introduction

Distributed systems and their applications often require an organizational structure to perform their operations efficiently. Certain topologies of overlay networks are defined by graph properties that enable or enhance a specific application. Tree topologies are one of them. Their properties, e.g., path uniqueness, and the hierarchy they reflect enable operations such as decentralized search, decision-making, aggregation or information dissemination. These operations are fundamental in various application domains such as distributed databases [González-Beltrán et al., 2008; Zhuge and Feng, 2008], application-level multimedia multicasting [Tan et al., 2005a] and grid computing [Chakravarti et al., 2006].

However, trees suffer from lack of redundancy and therefore their structure is sensitive to single node or link failures. Moreover, a tree experiences heterogeneous load, meaning that nodes close to the root of the topology receive messages forwarded from all the other nodes at the bottom part of this tree. This effect is also related to the impact of a failure that is much higher for nodes close to the root of a tree. All of these issues make the use of trees in dynamic distributed environments challenging and sometimes infeasible.

Self-organization in trees is required to overcome these limitations. A robust tree requires both building and maintenance during runtime of a distributed application. Nodes self-organize themselves in a tree topology that ideally minimizes the impact of their failures. One way to achieve this optimization is by ordering a tree according to specific application criteria, i.e., the performance profile of nodes. An optimized tree may also have constraints such as the number of children with which each node can connect. All of these criteria should be met during self-organization.

This paper studies the adaptation strategies of AETOS, the *Adaptive Epidemic Tree Overlay Service* Pournaras et al. [2010b,a]. Self-organized tree topologies with different graph properties are formed by simply using certain adopting adaptation strategies. This paper shows how these strategies can be combined dynamically, providing a powerful meta-level of adaptation and abstraction in the self-organization of tree topologies based on which a wide range of performance trade-offs can be explored as confirmed by the experimental findings of this paper.

Some of the basic system components of AETOS are described in our earlier work [Pournaras et al., 2010b,a]. In contrast, the contribution of this paper is three-fold [Pournaras, 2013]:

1. The problem of building and maintaining robust tree topologies is illustrated as a more generic graph theoretic problem. The selection of graph properties studied in this paper is grounded to applications that perform distributed operations based on tree topologies.

2. The design of adaptation strategies is improved to achieve higher performance and abstraction as shown in new extensive experimental results studied in this paper. Moreover, this paper extends the adoption of adaptation strategies from static to dynamic resulting in improved performance.

3. A survey of related methodologies is illustrated that distinguishes the more generic features of AETOS compared to other application-dependent mechanisms for building and maintaining tree topologies.

The organization of this paper is outlined as follows: ‘Problem Description’ illustrates graph properties of tree topologies and their relation to distributed applications. It also defines performance metrics for self-organization. ‘Sys-

tem Overview' provides a high-level overview of AETOS. 'Dissemination and Collection' illustrates the bottom level of the AETOS architecture. 'Clustering' discusses the input and adaptations of the middle level. 'Building and Maintenance' shows the establishment of parent-child links in the top level. Moreover, 'Adaptation Strategies' illustrates a number of generic strategies engaged in the middle level of AETOS. 'Experimental Evaluation' follows. 'Comparison with Related Work' compares AETOS with related methodologies. 'Conclusions' concludes this paper.

Problem Description

Trees are connected undirected acyclic graphs [Baldwin and Scragg, 2004; Gross and Yellen, 2005] with some graph properties such as degree-bounding, ordering, balancing and completeness that influence the performance of distributed applications, making a tree more robust to node failures or balancing the load between nodes. This paper focuses on building optimized tree topologies that tune and reflect specific graph properties to meet requirements of various application types.

Graph properties and applications

Trees may have graph properties that influence the performance of several distributed applications:

- *Degree-bounding*: The degree of a vertex is the number of (non-directional) edges attached to this vertex. The degree of a vertex that belongs to a tree is the number of its children plus one for its parent. Degree-bounding of a tree topology refers to the maximum number of children that a node supports and is related to various application constraints. For instance, bandwidth, storage and processing capacity of a node are related to the maximum number of nodes to which multimedia content can be disseminated [Fei and Yang, 2007]. Degree-bounding should be respected during building and maintenance of tree topologies. If the degree of a node is significantly lower than the upper bound, then the node is underloaded or the resources allocated for its application are not fully exploited. In contrast, if the degree of a node exceeds the upper bound, then the node is overloaded resulting in a bottleneck or unavailability of this node. In this case, the topology may disconnect and the performance of the application degrades.

- *Ordering*: An ordered tree is a rooted tree in which the children of each vertex are assigned a fixed ordering. This ordering is performed using assigned weights that rank vertices of a tree. These weights are usually related to an application. For example, multimedia multicasting applications [Birrer and Bustamante, 2007] compute the bandwidth and availability [Bhagwan et al., 2003] of nodes. Information about these metrics can be used for the maximization of content delivery and speed that is crucial for real-time applications such as multimedia multicasting. Similarly, social

recommender systems [Manouselis and Costopoulou, 2007] capture the profiles of users based on their preferences about an online product or service. Based on preference matching, these systems increase engagement of users and the quality of recommendations.

This paper focuses on the level-order scheme [Gross and Yellen, 2005] that is described as a top-down and left-to-right traversal. However, note that in practice the actual communication between the nodes always occurs via the parent-children links. Because of this fact, the distributed applications to which this paper refers to mainly require a top-down order rather than a left-to-right order. Positioning nodes with high availability [Bhagwan et al., 2003] close to the root reduces the impact of a node failure as the disconnected branch is smaller in size. Similarly, node close to the root of a tree with low bandwidth is a bottleneck. Therefore, ordering provides a proactive robustness by minimizing the impact that a low-performing node causes to the topology and its applications.

- *Balancing*: A balanced tree with a degree-bounding for each vertex is a rooted tree from which the leaves have the same depth or their depth difference is '1', if degree-bounding of the vertices does not permit having the same depth. An imbalanced tree experiences an unequal load among different branches with different path lengths. Therefore, distributed applications may experience higher latency and communication overhead due to the higher number of hops to reach each node of a tree [Jagadish et al., 2006a].

- *Completeness*: A complete tree is a rooted balanced tree with all of its vertices, except the leaves, having the maximum possible number of children as defined by the degree-bounding. Complete trees have the minimum possible height given the degree-bounding of the vertices. Overlay networks organized in complete tree topologies make effective resource allocation for their applications as nodes exploit the maximum number of children without exceeding the upper bound [Wang et al., 2010].

Performance Metrics

Assume an overlay network of n nodes that forms a tree topology. Each node i is ranked with a unique weight w_i in the range $[0, 1)$ that represents one or more application criteria. Furthermore, each node i has a node degree d_i that refers to the maximum permitted number of established links. The maximum number of children that a node i has is $k_i \leq d_i - 1$. Each node i has a *tree view* $\mathbf{v}_i(\text{tree})$ that is a list of other ranked nodes. The first element of a tree view corresponds to the parent and the rest of the elements to the children of node i .

A degree-bounded, ordered, balanced and complete tree can be optimally built according to Algorithm 1 in Appendix A. For illustration purposes, this organizational goal is explained as a centralized algorithm. However, in a decentral-

ized self-organization, nodes have limited information about other nodes and therefore, they need a certain execution time to discover, structure and coordinate with other nodes with which they form parent-child links. Performance metrics are required to quantify the degree to which the illustrated graph properties are met during the self-organization runtime. This paper introduces four performance metrics for this purpose: (i) connectedness β , (ii) connectivity γ , (iii) fitness ρ and (iv) instability $\delta()$.

Connectedness β is the proportion of nodes that are connected to the largest component of the forest: the main tree. For a forest \mathbf{F} of n tree components $\{\mathbf{T}_0, \dots, \mathbf{T}_{n-1}\}$, connect- edness is defined as:

$$\beta = \frac{\max(|\mathbf{T}_0|, \dots, |\mathbf{T}_{n-1}|)}{n} \quad (1)$$

In contrast, the tree *connectivity* γ refers to the number of links established between the nodes relative to the maximum number of links $n - 1$ that can be established in a tree topology [Gross and Yellen, 2005]:

$$\gamma = \frac{1}{2(n-1)} \sum_{i=0}^{n-1} |\mathbf{v}_i(\text{tree})| \quad (2)$$

The *fitness* ρ expresses the quality of the ordering process based on the selection of the parent and children for each node. Fitness is associated with the ranking of nodes that form parent-child links. This paper correlates the fitness ρ with the relative ranking distance between nodes and their parents. This is the distance $w_0 \in \mathbf{v}_i(\text{tree}) - w_i$ between a ranked node w_i and its parent $w_0 \in \mathbf{v}_i(\text{tree})$ in relation to the maximum distance that they can have if the node i has for parent the highest ranked node. For example, assuming nodes with uniformly distributed weights in $[0, 1)$, the maximum possible ranking distance is approximately $1 - w_i$. Therefore, fitness ρ is defined as:

$$\rho \approx \frac{1}{n\rho'} \sum_{i=0}^{n-1} \frac{w_0 \in \mathbf{v}_i(\text{tree}) - w_i}{1 - w_i} \quad (3)$$

where ρ' is the optimum fitness computed using Algorithm 1. The optimum fitness ρ' depends on the topology built and therefore it is related to the other tree properties such as the degree-bounding¹.

Finally, the *instability* $\delta()$ of a performance metric describes the variation of this metric during a self-organization runtime period. Instability is calculated using the relative standard deviation. For example, instability $\delta(\rho)$ of fitness ρ during a period of time \hat{t} is defined as follows:

$$\delta(\rho) = \frac{1}{\hat{t}} \sqrt{\frac{1}{\hat{t}-1} \sum_{t=0}^{\hat{t}-1} [\rho(t) - \bar{\rho}]^2} \quad (4)$$

Note that, minimizing instability of a performance metric is a factor to detect the *convergence* of self-organization according to this performance metric.

System Overview

Figure 1 illustrates the three-level architecture of AETOS. Each level of AETOS is managed by a software agent. The bottom level in AETOS provides a gossip-based dissemination and collection of random agent samples. These samples contain agent resources such as their network address for communication and weights with which their nodes are ranked. The gossiping criteria are actual gossiping configurations that parameterize the bottom level and inject the weights of the nodes that participate in the tree self-organization. These resources together with the network addresses of the agents are continuously exchanged in a gossip fashion. Agent samples are provided to the middle level that performs clustering based on the proximity of their weights defined by an adopted adaptation strategy. The structuring criteria contain an actual adaptation feedback and other clustering configurations that parameterize the adopted strategy. Clustering provides a search space from which candidate parents and children are selected and provided to the top level responsible for coordinating the building and maintenance of the tree topology. The agents corresponding to the candidate parents and children are contacted for the establishment of parent-child links. If this negotiation is successful, the tree view is filled and is provided to applications. Finally, the tree criteria are tree configurations containing information about the graph properties of the built tree.

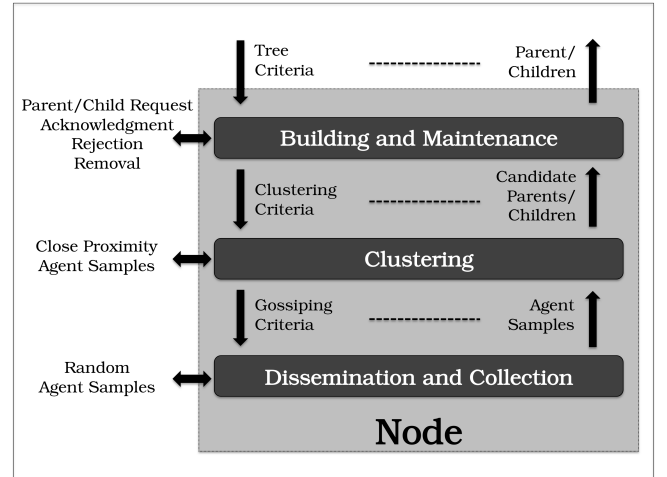


Figure 1. The AETOS middleware architecture.

¹The maximum optimum fitness value corresponds to a tree organized in a star topology. In this case, every node has the highest ranked node as a parent. In contrast, the minimum optimum fitness value corresponds to a tree organized in an ordered list.

Figure 2 illustrates the main concept of how AETOS performs self-organization of tree topologies. Each level of AETOS contains a view containing a limited number of ranked agent samples. Each view is filled incrementally by the view at the level below. Each node of a network has three views that correspond to layered overlay networks. The overlay network of the bottom level is defined by random samples of agents collected by the bottom level. These random samples become the input at the middle level in which they are clustered to an overlay network of candidate parents and children based on the proximity of their weights. For example, the ranked node 0.58 has in closest proximity the ranked nodes 0.67 and 0.53. Finally, based on a selection scheme defined by an adaptation strategy, such as the selection of the highest ranked agents, candidate parents (0.75) and children (0.53 and 0.45) potentially fill the tree view.

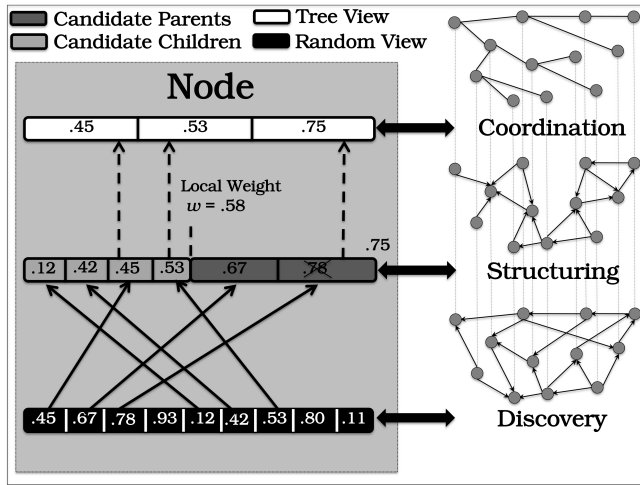


Figure 2. Managing the agent views in three levels.

A more detailed illustration of the functionality of each level follows in the next sections.

Dissemination and Collection

The bottom level of AETOS is realized by the peer sampling service, a gossiping protocol introduced by Jelasity et al. [2007]. It periodically updates the *random view* that is a list of limited size r containing the agent samples. These samples are exchanged between remote agents of the bottom level and concern the network identifiers of the respective nodes, e.g., the IP address and the port number. For each unique identifier of an agent, the weight w_i of its node i is included in the agent samples.

Clustering

This section illustrates the cluster-based structuring performed in the middle level of AETOS.

The input of the proximity view

Searching randomly for the parent and children of each node may not be a cost-effective approach for the self-organization of tree topologies. A metric based on a proximity distance may be more effective. The proximity distance between a ranked node i and its input node j provided by the bottom level is defined by their ranking distance $|w_i - w_j|$. This computation indicates the proximity of the nodes. Agent samples are structured in an ordered list of limited size, the *proximity view* $v_i(\text{proximity})$, based on their ranking distance. Each proximity view $v_i(\text{proximity})$ of a node i is virtually split in two parts containing candidate parents and candidate children. The weight w_i of a node i is the reference point for this split: For example, nodes ranked higher than w_i are candidate parents and nodes ranked lower are candidate children. The maximum size of the proximity view q_i of a node i is controlled by the *scaling factor* f and is related to the maximum number of children k_i as follows: $q_i = f * (k_i + 1)$. The scaling factor determines the number of times the proximity view is larger than the tree view.

The proximity view has random samples of ranked nodes provided by the bottom level of AETOS. This section outlines the T-MAN gossiping protocol [Jelasity et al., 2009] that performs remote clustering to improve performance of AETOS, i.e., higher convergence speed in the self-organization process. Nodes in T-MAN periodically exchange their proximity views with other agents in close proximity via horizontal interactions. T-MAN is relevant in the context of AETOS for three reasons: (i) It is a decentralized self-organization mechanism. (ii) It performs organization of nodes based on their ranking distance indicating their proximity. (iii) It requires samples of ranked nodes that the architecture of AETOS also defines at the bottom level. Although T-MAN is a generic topology management mechanism for building and maintaining various topologies based on the ranking distance of nodes, it is not generic enough to organize complex topologies such as the tree topologies illustrated in ‘Problem Description’. Furthermore, T-MAN cannot guarantee a cycle-free topology during self-organization.

Adaptations

The agent of the middle level in node i includes in its proximity view nodes ranked in the range $[0, w_i) \cup (w_i, 1)$ by default. Restricting this space results in a more targeted search for candidate parents and children given the limited size of the proximity view. Performing exclusions of a range of weights from the default range of weights $[0, w_i) \cup (w_i, 1)$ is the approach followed to limit this search space. Figure 3 illustrates the concept of such adaptations applied in the range of ranked nodes in proximity view. Four types of adaptations are applied:

- **Reset:** This adaptation defines the default range

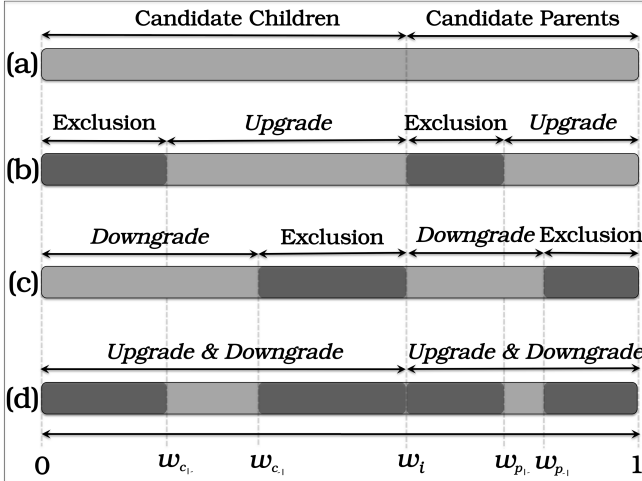


Figure 3. Adaptations in the range of ranked nodes in the proximity view. (a) Reset adaptation, (b) upgrade adaptation, (c) downgrade adaptation, (d) upgrade and downgrade adaptation.

$[0, w_i) \cup (w_i, 1)$. It is applied if the proximity view remains empty for a period of time as a result of multiple *upgrade* and *downgrade* adaptations.

- **Upgrade:** This adaptation excludes nodes ranked with low weights in favor of highly ranked nodes. The new range is defined as $(w_{c_+}, w_i) \cup (w_{p_+}, 1)$ with c_+ and p_+ indexing a child and parent respectively based on which this reconfiguration is defined. This adaptation results in potentially higher ranked candidate parents and children provided to the top level.

- **Downgrade:** This adaptation excludes nodes ranked with high weights in favor of low ranked nodes. The new range is defined as $[0, w_{c_-}) \cup (w_i, w_{p_-})$ with c_- and p_- indexing a child and parent respectively based on which this reconfiguration is defined. This adaptation results in potentially lower ranked candidate parents and children provided to the top level of AETOS.

- **Upgrade and downgrade:** This adaptation excludes low and highly ranked nodes and defines an in-between range from which candidate parents and children are selected. This new range is defined as $(w_{c_+}, w_{c_-}) \cup (w_{p_+}, w_{p_-})$.

The purpose of adaptations is to improve selections of candidate parents and children provided to the top level. The clustering criteria contain a *positive or negative feedback* about the earlier provided candidate parents and children. This feedback is associated with the applied adaptations. For example, positive feedback can be interpreted by the middle level as selecting higher ranked candidate parents and children than before and therefore, positive feedback is associated with an *upgrade* adaptation. A ranked node, such as the w_{p_+} , w_{p_-} , w_{c_+} and w_{c_-} in the above examples is a reference weight based on which adaptations are performed and is also

part of the clustering criteria in AETOS.

Building and Maintenance

The top level is responsible for building and maintaining the tree topology. Self-organization is performed by coordinating parent-child links that respect the graph properties of nodes, e.g., degree-bounding. The links of the organized tree topologies should be bidirectional. For this reason, the agents of the top level contact the respective agents of the candidate parents/children provided by the middle level to establish parent-child links that respect these properties. Links are established if and only if they are mutually accepted between communicating agents. Furthermore, a link can be removed by a connected node without mutual acceptance in favor of a new link with another node. Mutual acceptance of a link corresponds to positive feedback whereas rejection or removal of a link is negative feedback. This feedback is part of the clustering criteria provided to the middle level.

Furthermore, the agents of the top level are *competitive* and seek to connect with the highest ranked parent and children. This requirement is part of the tree criteria.

AETOS introduces peer-to-peer coordination based on four types of exchanged messages:

- **Request:** This message is sent to a candidate parent (*parent request*) or candidate child (*child request*). It contains agent samples of the sender agent such as the weight of its node and its network identifier. A *request* is sent if and only if the tree view is not filled or the candidate parents and children provided by the middle level are ranked higher than the parents and children in the tree view.

- **Acknowledgment:** This message is an acceptance of a *request*. Given a *parent request*, an *acknowledgment* is sent if the parent in the tree view is ranked lower than the sender node of the *parent request*. Similarly, given a *child request*, an *acknowledgment* is sent if the children in the tree view are not all ranked higher than the node of the sender of the *child request*. The receipt of an *acknowledgment* follows the filling of the tree with a new parent or child and positive feedback to the middle level. If these actions require the replacement of a parent or child from the tree view, a *removal* message is sent to this parent or child.

- **Rejection:** This message is denial of a *request*. For a *parent request*, a *rejection* is sent if the parent in the tree view is ranked higher than the sender node of the *parent request*. Similarly, in case of a *child request*, a *rejection* is sent if the children in the tree view are all ranked higher than the sender of the *child request*. The receipt of a *rejection* follows a negative feedback to the middle level.

- **Removal:** This message triggers deletion of a parent or child from the tree view. In case of a receipt of a *removal* message, negative feedback is provided to the middle level.

Appendix B illustrates the communication algorithm of the agents in detail.

Adaptation Strategies

The middle level of AETOS performs selection of ranked nodes within the adaptive range of the proximity view based on two concepts:

- Selecting candidate parents only, candidate children only, or both: three selection options
- Selecting the highest or lowest ranked candidate parents and children: two selection options

	0 Candidate Children		w_i Candidate Parents		1
Strategy	Low	High	Low	High	
PRESBYOPIC	✓				✓
MYOPIC			✓	✓	
HUMBLE	✓			✓	
GREEDY			✓		✓
TOP-DOWN			✓		
BOTTOM-UP				✓	
TOP					✓
BOTTOM	✓				

Figure 4. The selection schemes of the adaptation strategies.

These selection options create $2^3 = 8$ combinations that are the adaptation strategies of AETOS and they are illustrated in Figure 4. Strategies provide (i) different preference schemes of candidate parents and children and (ii) different schemes² for the applicability of adaptations in the range of ranked nodes in the proximity view as illustrated in Figure 5.

The adoption of an adaptation strategy is performed as follows:

- *Static adoption*: A strategy is adopted as a system parameter. In this case, a strategy is designed explicitly to serve the self-organization of one or more specific topologies.
- *Dynamic adoption*: The adopted strategy changes dynamically during system runtime. However, the decision for change is based on static criteria selected empirically or based on some monitored parameters. For example, the adopted strategy may change after some runtime period or after a topology change, e.g., insertion of new nodes in the network.

This paper studies these two adoption schemes.

Experimental Evaluation

AETOS is implemented and evaluated in Protopeer [Galuba et al., 2009], a prototyping toolkit for distributed systems. A concurrent implementation of AETOS in the AgentScape simulation framework [Oey et al., 2010] confirms the results of Protopeer. Table 1 summarizes

Strategy	Candidate Children Feedback		Candidate Parents Feedback	
	Negative	Positive	Negative	Positive
PRESBYOPIC	upgrade	downgrade	downgrade	upgrade
MYOPIC	downgrade	upgrade	upgrade	downgrade
HUMBLE	upgrade	downgrade	upgrade	downgrade
GREEDY	downgrade	upgrade	downgrade	upgrade
TOP-DOWN	downgrade	upgrade		
BOTTOM-UP			upgrade	downgrade
TOP	upgrade	downgrade		
BOTTOM			downgrade	upgrade

Figure 5. Adaptations applied for each feedback and adaptation strategy.

the selected experimental settings. Note that multiple values for a single parameter denote the tested variations of this parameter in some of the illustrated experiments. The values depicted with bold are the default ones.

A network of 1500 nodes is simulated running AETOS for $t(\text{AETOS}) = 400$ epochs. Every epoch lasts for $T(\text{AETOS}) = 1000$ ms. Protopeer initially bootstraps nodes in a ring topology. The bootstrapping time period is $t'(\text{AETOS}) = 6$ epochs and the size of the ring view $|v(\text{ring})| = 5$ for every node. Nodes are ranked with a random weight $w \in [0, 1)$.

The top level defines the number of children $k = 4$. Other topologies with $k = 3, 5$ and a random number k of 3, 4 or 5 children for each node of the tree topology are evaluated. Furthermore, the top level sends $z = 2$ number of *request* messages. If the adopted strategy includes both candidate parents and children in the proximity view, the requests are one for each candidate type. Finally, the *request* messages are sent periodically every $T(\text{top}) = 1000$ ms.

In the middle level, the size of the proximity view is $q = 30$ with the scaling factor³ chosen to be $f = 6$. The static and dynamic adoption schemes of adaptation strategies are evaluated. Adaptations and the T-MAN mechanism are enabled and disabled in some of the experiments. When adaptations are enabled, the resetting period is chosen to be $T(\text{reset}) = 3000$ ms. The alternative values of 2000, 5000, and the case of no resetting are examined. Moreover, when T-

²In contrast to our earlier work [Pournaras et al., 2010a], the adaptations applied by the strategies in this paper are aligned to the selections performed for higher or lower ranked candidate parents/children.

³In the case that the number of children for each node is $k = 4$, the proximity view contains 6 candidate parents and 24 candidate children as defined by $q_i = f * (k_i + 1)$.

Table 1

The experimental settings for the evaluation of AETOS. The bold values are the default ones in the performed experiments.

	Parameter	Value
AETOS	n	1500
	$t(\text{AETOS})$	400
	$T(\text{AETOS})$	1000
	$t'(\text{AETOS})$	6
	$w \in [0, 1)$	random
	$ v(\text{ring}) $	5
Top Level	k	3, 4 , 6, varied
	z	2
	$T(\text{top})$	1000
Middle Level	q	30
	f	6
	strategy adoption	static , dynamic
	$T(\text{reset})$	7, 15 , 30
	strategy adoption	static , dynamic
	$T(\text{reset})$	2000, 3000 , 5000, ∞
	T-MAN inclusion	enabled , disabled
	$t'(\text{middle})$	6
	$T(\text{middle})$	100, 200 , 250, 500, 1000
	r	50
Bottom Level	view selection policy	swapper
	view propagation policy	push-pull
	peer selection policy	random
	$T(\text{bottom})$	100, 200 , 250, 500, 1000

MAN is enabled, it runs with the parameters $m = 40$, $\psi = 20$, $l = 6$. The role of these parameters is out of the scope of this paper and is illustrated by Jelasity et al. [2009]. Finally, the middle level is bootstrapped within $t'(\text{middle}) = 6$ epochs and its execution period is selected at $T(\text{middle}) = 200$ ms. The alternative values of 100, 250, 500 and 1000 ms are also examined.

The bottom level is realized by the peer sampling service [Jelasity et al., 2007]. The size of the random view is $r = 50$ and the execution period is $T(\text{bottom}) = T(\text{AETOS})/5 = 200$ ms. Other values examined are 100, 250, 500 and 1000 ms. The values of the ‘view selection’, ‘view propagation’ and ‘peer selection’ policies are selected to maximize the randomness and dissemination speed of gossiping. More information about these parameters are provided by Jelasity et al. [2007].

The evaluation of AETOS is based on the four performance metrics illustrated in ‘Problem Description’. The optimum fitness of a tree for given graph properties is computed in a centralized fashion as illustrated in Algorithm 1. The system runtime is $t(\text{AETOS}) = 400$ epochs. The number of messages $\lambda(\text{level})$ generated by each level of AETOS is computed as a measure of the communication cost. Finally, qualitative comparisons are performed by visualizing the self-organized tree topologies using the JUNG visualization library of O’Madadhain et al. [2005]. The trees are vi-

sualized in a radial layout that depicts clearer the balancing and completeness graph properties.

Adaptation strategies

Figure 6 illustrates the convergence of connectedness for the eight adaptation strategies. For higher readability, the figures of this section that concern the adaptation strategies are split in four subgraphs, (a)-(d), each showing the results of two strategies. The order represents the performance of the strategies, starting from the strategy with the highest to lowest performance. MYOPIC, HUMBLE, BOTTOM-UP and TOP-DOWN converge to the maximum connectedness of 1 within the first 40 epochs as depicted in Figure 6a and 6b. The instability of connectedness in MYOPIC and HUMBLE is approximately 0.002. For BOTTOM-UP and TOP-DOWN, instability increases to 0.004 and 0.007 respectively. In contrast, GREEDY, PRESBYOPIC, TOP and BOTTOM follow in Figure 6c and 6d at a lower performance. More specifically, GREEDY and PRESBYOPIC converge approximately 20 epochs slower and reach an average connectedness of 0.82. Their instability is 0.029 and 0.026 respectively. TOP is even more inefficient with an average connectedness of 0.54 and an instability of 0.067. BOTTOM does not converge and results in multiple disconnected parent-child links.

Figure 7 illustrates the convergence of connectivity for the eight adaptation strategies. The performance comparisons of

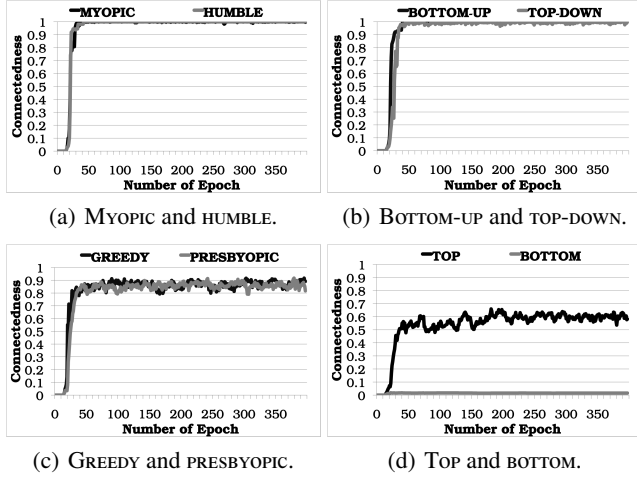


Figure 6. Convergence of connectedness for the adaptation strategies.

this performance metric are similar with the ones of connectedness. MYOPIC, HUMBLE, BOTTOM-UP and TOP-DOWN of Figure 7a and 7b converge to the maximum connectivity of 1 in fewer than 30 epochs. The instability of connectivity in these strategies is approximately 0. In contrast, PRESBYOPIC and GREEDY in Figure 7c converge to an average maximum connectivity of 0.95 in approximately 40 epochs with an instability of 0.012. Despite the relative low connectivity of BOTTOM shown in Figure 7d, it converges to an average maximum connectivity of 0.85 and an instability of 0 in approximately 40 epochs. This is because there is a significant number of parent-child links formed whose nodes remain disconnected between each other. In other words, these links remain decoupled. Finally, top, also depicted in Figure 7d, achieves the lowest connectivity. The average maximum connectivity is approximately 0.67 with an instability of 0.031.

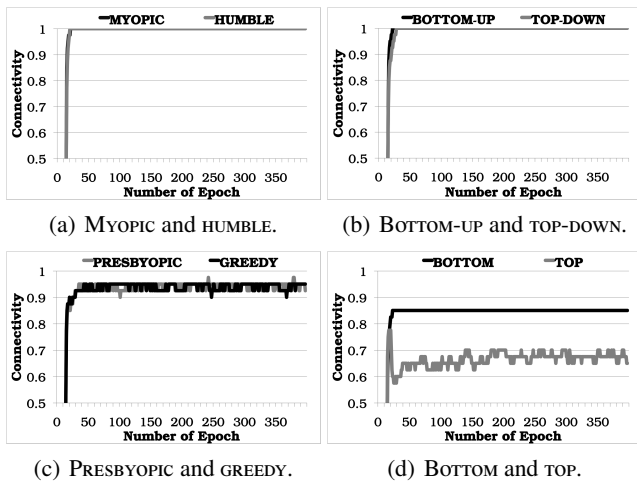


Figure 7. Convergence of connectivity for the adaptation strategies.

Note that MYOPIC, HUMBLE, BOTTOM-UP and TOP-DOWN that achieve the highest connectedness and connectivity do not define highly ranked candidate parents in the proximity view of agents. Furthermore, BOTTOM is ineffective as selection of the lowest candidate children is not consistent with the selection criteria of the top level that defined a preference for the highest possible parents and children.

Figure 8 illustrates the fitness of the tree topology built by each adaptation strategy. The performance comparisons show different results in this case. TOP-DOWN and MYOPIC converge to the average maximum fitness of 0.95 in more than 150 epochs as depicted in Figure 8a. The instability of their fitness is 0.007 and 0.008 respectively. Therefore, these two strategies provide the highest fitness and the slowest convergence, yet the fitness is higher than the fitness in the other strategies during this convergence period. HUMBLE and BOTTOM, illustrated in Figure 8b, follow with an average maximum fitness of 0.87 and 0.86 respectively and instability of 0. These two strategies converge in a shorter period of time, approaching their maximum fitness in approximately 60 epochs. Figure 8c shows the results of PRESBYOPIC and GREEDY. The average maximum fitness is 0.85 and 0.83 respectively, converging in approximately 40 epochs. Their instability is 0.011 and 0.018 respectively. Finally, TOP and BOTTOM-UP of Figure 8d result in the lowest fitness of 0.72 and 0.65 respectively. TOP converges in more than 100 epochs, whereas the BOTTOM-UP converges in 30 epochs. In contrast to BOTTOM-UP that has an instability of 0, TOP has the highest instability of 0.025.

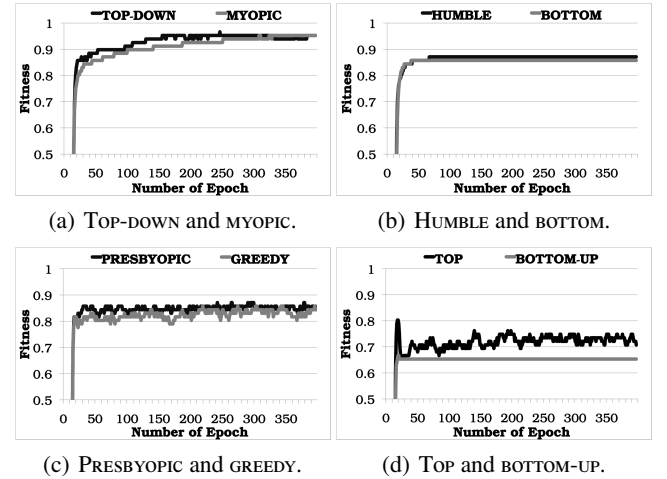


Figure 8. Convergence of fitness for the adaptation strategies.

Note that, BOTTOM achieves a relatively high fitness but low connectedness and connectivity. This is because the highly ranked nodes are preferred in the top level and therefore they connect with the lowest ranked nodes resulting in robust parent-child links.

The communication cost of the bottom and middle level

is constant and depends on the gossiping period of the peer sampling service and T-MAN respectively. During the runtime of these 2 gossiping protocols, agents exchange periodically 2 messages in each case as both protocols are ‘push-pull’. Furthermore, each gossiping protocol is executed 5 times during an epoch as $T(\text{bottom}) = T(\text{middle}) = T(\text{AETOS})/5 = 1000/5 = 200$ ms. Therefore, the total number of messages that an agent of the bottom and middle level sends during an epoch is $2 * 2 * 5 = 20$ messages or $\lambda(\text{bottom}) + \lambda(\text{middle}) = 20 * 1500 = 30000$ messages for the total number of these agents in the network. Results confirm these estimations.

In contrast, the communication cost of the top level is influenced by the adopted strategy. Figure 9 illustrates the convergence of this communication cost. All strategies converge in fewer than 40 epochs. In Figure 9a, BOTTOM-UP minimizes the number of exchanged messages to 0 in 30 epochs. The communication cost of HUMBLE consumes 2100 messages per epoch in average after its convergence. The instability of the communication cost is approximately 0.019. MYOPIC and BOTTOM of Figure 9b converge to an average maximum communication cost of 2600 and 4600 messages respectively. The instability of the communication cost is 0.017 and 0.015 respectively. In Figure 9c, PRESBYOPIC and TOP-DOWN have a communication cost of 5300 and 5400 messages per epoch with an instability of 0.018. Finally, GREEDY and TOP have the highest average maximum communication cost of 5700 and 6300 messages and an instability of 0.017 and 0.02 respectively.

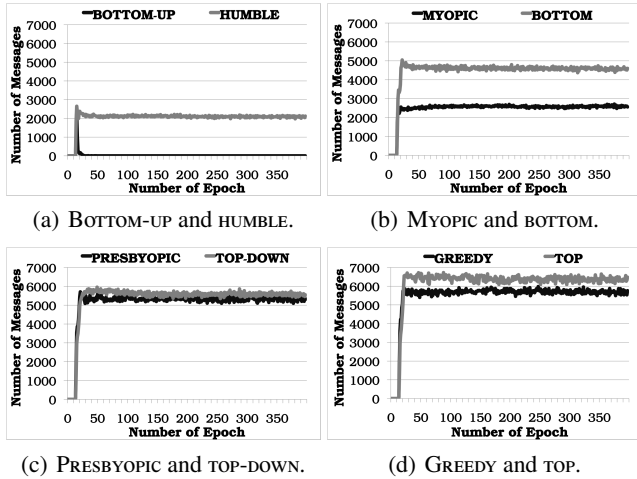


Figure 9. Convergence of messages exchanged at the top level for the adaptation strategies.

Figure 10 shows the optimally organized topology computed in a centralized fashion as shown in Algorithm 1. Table 2 provides a visualization of the eight adaptation strategies at four time points during convergence: 20th, 35th, 50th and 350th epoch.

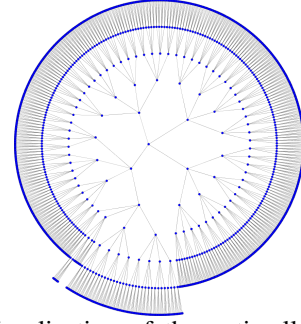


Figure 10. Visualization of the optimally organized tree topology of 1500 nodes.

Visually, GREEDY and PRESBYOPIC result in similar topologies: Well balanced branches close to the root that become less balanced and less complete close to the leaves of the tree. A few disconnected branches are present. TOP has the effect of a large well balanced and connected branch disconnected from a large number of smaller branches. MYOPIC, HUMBLE, and BOTTOM-UP result in significantly better connected topologies that have, however, imbalanced branches. From these three strategies, the imbalance of HUMBLE is the highest. Moreover, the topology of TOP-DOWN is organizationally the closest to the optimally organized tree topology of Figure 10. Most branches are balanced and there is a minimum number of disconnected nodes. Finally, BOTTOM results in a forest with a large number of disconnected trees without a distinction of the main tree (branch).

The effect of adaptations

The *reset*, *upgrade* and *downgrade* adaptations of the proximity view contribute significantly in the achieved performance of the strategies. Without adaptations, meaning only performing selections and ignoring the feedback provided by the top to middle level, the connectedness, connectivity, and fitness remain low with increased instability. However, communication cost decreases without adaptations as fewer connections are negotiated in the top level.

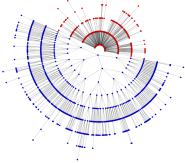
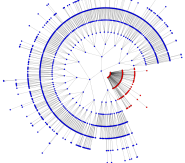
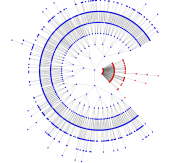
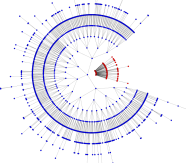
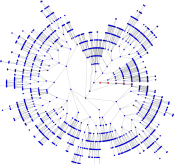
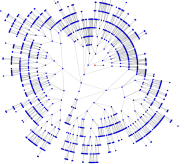
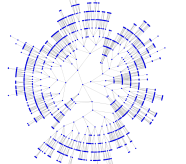
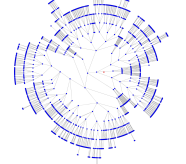
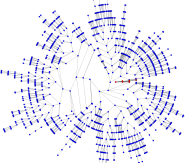
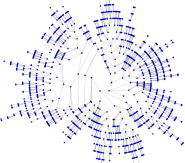
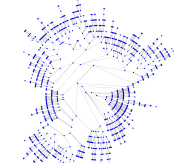
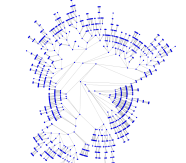
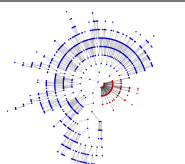
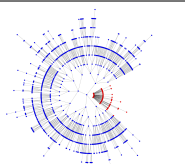
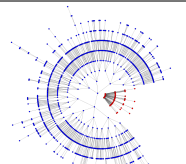
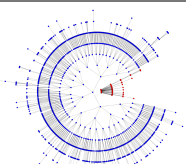
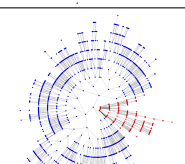
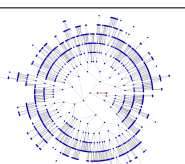
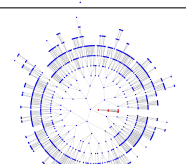
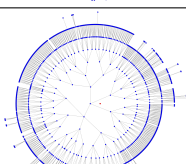
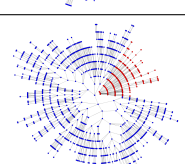
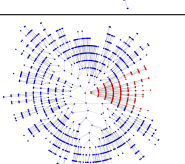
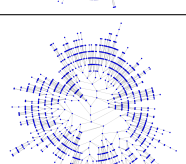
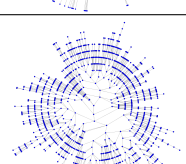
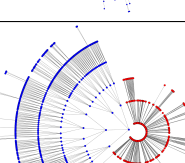
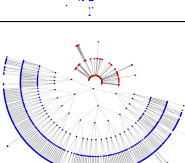
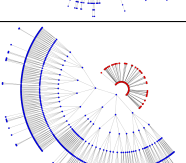
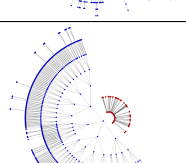
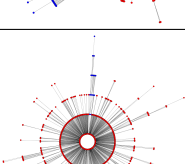
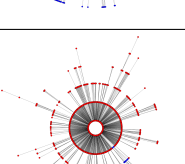
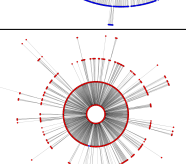
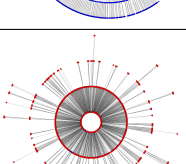
For example, the achieved connectedness without adaptations is approximately 20% lower for GREEDY and 10% lower for MYOPIC compared with the case in which adaptations are enabled in the system. Fitness is almost double for GREEDY and 20% higher for MYOPIC when using adaptations. In contrast, the communication cost with adaptations increases 30% for GREEDY and more than double for MYOPIC.

Dynamic adoption of the adaptation strategies

Results show that strategies result in performance trade-offs. Motivated by this observation, this section shows how strategies can be combined to collectively achieve higher performance than each one individually. This section describes

Table 2

Visualization of the adaptation strategies at four time points during the runtime of AETOS: (i) 20th, (ii) 35th, (iii) 50th and (iv) 350th epoch.

	20 th Epoch	35 th Epoch	50 th Epoch	350 th Epoch
PRESBYOPIC				
MYOPIC				
HUMBLE				
GREEDY				
TOP-DOWN				
BOTTOM-UP				
TOP				
BOTTOM				

a number of possibilities rather than an exhaustive illustration of how adaptation strategies can be adopted dynamically. Self-organization of a tree based on the synergy of two or more strategies is referred to as *hybrid* adaptation strategy.

Hybrid strategies introduced in this section are based on the following intuition. Certain adaptation strategies, such as TOP, GREEDY and PRESBYOPIC cannot meet the requirement of building a fully connected tree. They also consume a high number of messages without approaching the fitness of TOP-DOWN or MYOPIC. In contrast, BOTTOM-UP builds a fully connected tree at a minimum communication cost but with significantly lower fitness than the aforementioned strategies. Therefore, the possibility of combining BOTTOM-UP with one of the other strategies potentially results in a maximization of connectedness, fitness and a minimization of the communication cost. This section experimentally investigates this intuition.

Three hybrid strategies are examined: (i) $\text{TOP} \rightarrow \text{BOTTOM-UP} \Rightarrow \text{HYBRID-01}$, (ii) $\text{GREEDY} \rightarrow \text{BOTTOM-UP} \Rightarrow \text{HYBRID-02}$, and (iii) $\text{PRESBYOPIC} \rightarrow \text{BOTTOM-UP} \Rightarrow \text{HYBRID-03}$. AETOS is initiated with the former strategy in each case. After the 250th epoch, the nodes that do not have a parent, therefore they are disconnected from the main body of the tree, switch to BOTTOM-UP.

Figure 11 illustrates connectedness, connectivity, fitness and communication cost of the three hybrid strategies. Note that, all hybrid strategies inherit the property of a maximum connectedness and connectivity from BOTTOM-UP. After the 250th epoch, both metrics converge within a short time to the maximum of 1.0 as depicted in Figure 11a and 11b. Note also the significant reduction of the communication cost in Figure 11d. HYBRID-01 minimizes this cost to 0, whereas, HYBRID-02 and HYBRID-03 continue generating approximately 700 and 1800 messages per epoch respectively. Fitness has an increase of 10% and 8% for HYBRID-02 and HYBRID-03 respectively as shown in Figure 11c. However, this is not the case for HYBRID-01 in which adoption of BOTTOM-UP negatively influences the fitness by approximately 10%.

Table 3 visualizes the three hybrid strategies at three different points during runtime: (i) On the 245th epoch that is 5 epochs before BOTTOM-UP is adopted, (ii) on the 255th epoch that is 5 epochs after the dynamic adoption of BOTTOM-UP and (iii) on the 380th epoch. This table shows the topological transitions of the strategies in time. The disconnected branches are connected to the leaves of the main tree forming a fully connected tree. The deterioration of fitness in HYBRID-01 is explained by the long ‘lists’ of nodes attached at the bottom of the tree. This attachment is more uniform for HYBRID-02 and HYBRID-03.

Note that, hybrid strategies achieve a highly comparable performance to TOP-DOWN and MYOPIC. Furthermore, they suggest additional performance trade-offs. For example, HYBRID-02 and HYBRID-03 achieve a significantly lower communica-

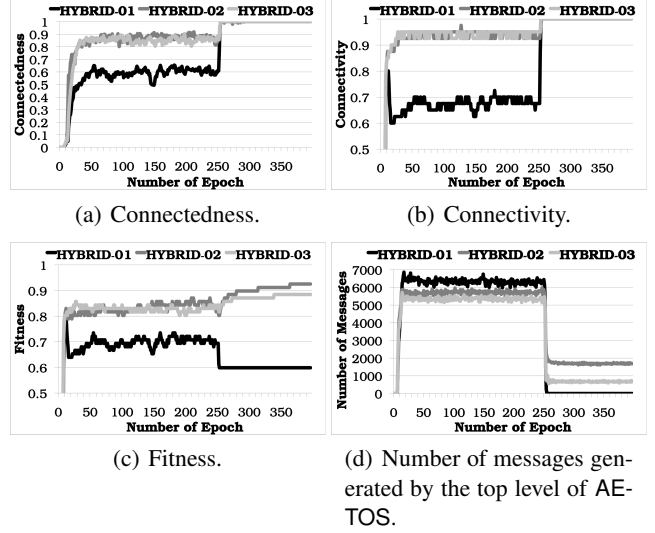


Figure 11. The performance of the hybrid adaptation strategies.

tion cost at a price of 3% – 5% lower fitness compared to TOP-DOWN and MYOPIC.

Resetting adaptations

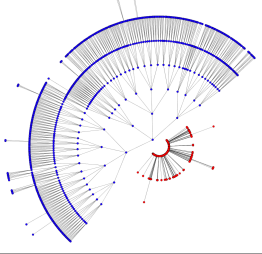
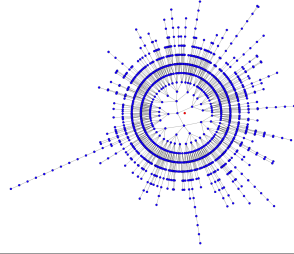
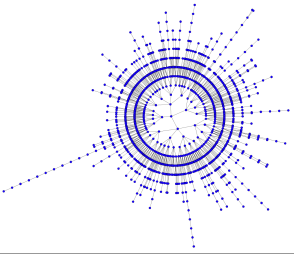
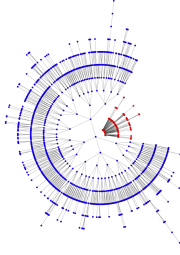
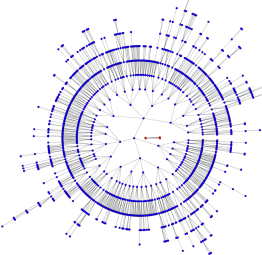
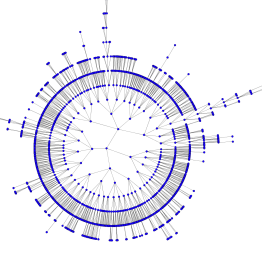
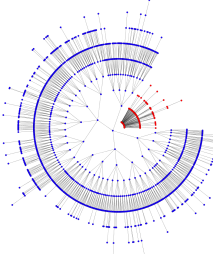
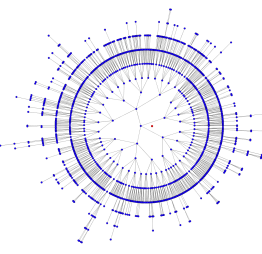
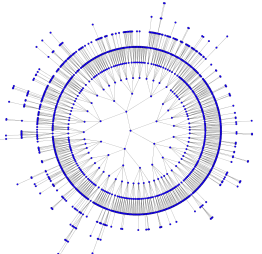
The *reset* adaptation is applied to the proximity view if it remains empty for a period of time. Multiple *upgrade* and *downgrade* adaptations applied may restrict the range of ranked candidate parents and children to such an extent that excludes all ranked nodes. This may be the case at the beginning of self-organization in which temporary links that are removed later on affect the proximity view of other nodes via the applied adaptations. In this case, the system is actually ‘trapped’ in a few locally optimum parent-child links, leaving a large number of nodes isolated with an empty proximity view. Conditional and periodic *reset* adaptation solves this problem.

Experimental evaluation confirms this intuition. Without *reset* adaptations, connectedness achieved with GREEDY remains lower than 0.45 for more than 160 epochs. Fitness remains high as few links formed are highly robust with nodes close to the root getting connected with the leaves of the tree in the respective optimum topology. However, the difference is insignificant to claim an improved fitness in the case of no adaptations in the proximity view. When $T(\text{reset})$ decreases to 5, 3 and 2 epochs, the performance metrics converge fast as shown in ‘Adaptation strategies’.

Note that, the experimental results also show that the lowest resetting period of $T(\text{reset}) = 2$ epochs results in the highest communication cost as the probability of sending parent and child requests to the same nodes increases when adaptations are reset more frequently. For GREEDY, the communication cost is 2% higher for $T(\text{reset}) = 2$ compared to

Table 3

Visualization of the tree topologies at three different epochs for three hybrid strategies on the (i) 245th, (ii) 255th and (iii) 380th epoch.

	245 th Epoch	255 th Epoch	380 th Epoch
HYBRID-01			
HYBRID-02			
HYBRID-03			

$T(reset) = 5$.

Tree topologies with different graph properties

Three different tree topologies are evaluated: (i) A 3-ary tree, (ii) a 5-ary tree and (iii) a tree with randomly selecting 3, 4 or 5 children per node. These tree topologies represent different distribution of links among their nodes. An application that uses a 5-ary tree may require higher computational and network resources than a 3-ary tree, e.g., bandwidth [Tan et al., 2005a]. Results show a clear trend for most of the strategies. More levels result in a higher complexity meaning that the performance of AETOS degrades as the size of the tree view decreases. Therefore, a 3-ary tree performs worse compared to a 5-ary tree that has a fewer number of levels.

Table 4 illustrates performance results of the eight adaptation strategies for each tree topology. Fitness, connectedness and connectivity of a 3-ary tree topology is lower and their convergence slower for PRESBYOPIC, MYOPIC, HUMBLE, GREEDY, TOP-DOWN, and TOP. However, the communication cost is higher for the 5-ary tree in these strategies. For example the fitness for GREEDY is $\rho = 0.77$ on the 380th epoch for a 3-ary

tree whereas for the 5-ary tree is $\rho = 0.87$. The respective values for the rest of the strategies are $\rho = 0.83$ and $\rho = 0.85$ for PRESBYOPIC, $\rho = 0.85$ and $\rho = 0.88$ for MYOPIC, $\rho = 0.89$ and $\rho = 0.98$ for TOP-DOWN, $\rho = 0.7$ and $\rho = 0.74$ for TOP. Similar trends appear for the connectedness, connectivity and communication cost.

Finally note that the topology with random selections of 3, 4 or 5 children per node performs comparable to the topology with a fixed number of 4 children per node concluding that the effect of a random number of children per node is averaged.

Adjusting the periodical executions

Relative execution periods of the three levels in AETOS are crucial for the convergence speed and performance of self-organization. For example, an adaptation applied in the proximity view of the middle level requires a period of time for updating the candidate parents and children. This is because the three levels are decentralized and interdependent within AETOS. The update of the proximity view requires some gossip exchanges at the bottom level to collect new

Table 4

Performance results of adaptation strategies for different tree topologies: (i) $k = 3$, (ii) $k = 5$ and (iii) $k = \text{rand}(3, 4, 5)$. The values concern the (i) connectedness β , (ii) connectivity γ , (iii) fitness ρ and (iv) the number of messages $\lambda(\text{top})$. Each pair of values refers to the 30th and 380th epoch of runtime.

	PRESBYOPIC	MYOPIC	HUMBLE	GREEDY
$k = 3$				
β	0.68, 0.81	0.99, 1.0	1.0, 1.0	0.67, 0.85
γ	0.92, 0.94	1.0, 1.0	1.0, 1.0	0.9, 0.92
ρ	0.83, 0.83	0.82, 0.92	0.83, 0.85	0.76, 0.77
$\lambda(\text{top})$	5116, 5101	2478, 2592	1968, 1920	5737, 5777
$k = 5$				
β	0.82, 0.91	1.0, 1.0	1.0, 1.0	0.84, 0.88
γ	0.93, 0.96	1.0, 1.0	1.0, 1.0	0.93, 0.96
ρ	0.84, 0.85	0.85, 0.95	0.84, 0.88	0.83, 0.87
$\lambda(\text{top})$	5587, 5405	2430, 2598	2327, 2247	5575, 5683
$k = \text{rand}(3, 4, 5)$				
β	0.78, 0.92	1.0, 1.0	1.0, 1.0	0.83, 0.88
γ	0.95, 0.98	1.0, 1.0	1.0, 1.0	0.95, 0.95
ρ	0.84, 0.87	0.83, 0.92	0.84, 0.87	0.81, 0.83
$\lambda(\text{top})$	5313, 5296	2557, 2605	2212, 2145	5643, 5665

	TOP-DOWN	BOTTOM-UP	TOP	BOTTOM
$k = 3$				
β	0.31, 0.99	0.91, 1.0	0.33, 0.56	0.01, 0.02
γ	1.0, 1.0	1.0, 1.0	0.58, 0.66	0.84, 0.84
ρ	0.8, 0.89	0.65, 0.65	0.64, 0.7	0.86, 0.86
$\lambda(\text{top})$	5830, 5561	2, 0	6751, 6699	4329, 4234
$k = 5$				
β	0.97, 1.0	0.5, 1.0	0.59, 0.68	0.02, 0.02
γ	1.0, 1.0	1.0, 1.0	0.66, 0.72	0.87, 0.87
ρ	0.92, 0.98	0.61, 0.61	0.69, 0.74	0.81, 0.84
$\lambda(\text{top})$	5708, 5551	0, 0	6411, 6232	4881, 4780
$k = \text{rand}(3, 4, 5)$				
β	0.86, 0.97	0.83, 1.0	0.46, 0.6	0.02, 0.02
γ	1.0, 1.0	1.0, 1.0	0.63, 0.68	0.88, 0.88
ρ	0.88, 0.94	0.65, 0.64	0.68, 0.72	0.84, 0.86
$\lambda(\text{top})$	5793, 5612	4, 0	6539, 6360	4697, 4582

agent samples. For this reason, the top level should not synchronize with its bottom levels.

The experimental findings confirm this. If all levels are synchronized at 1000 ms, the maximum average connectedness converges at 350th epoch for GREEDY. However, the convergence speed increases proportionally to the maximum performance for this strategy as the periodical execution of the bottom and top level decreases to 500, 250, 200 and 100 ms at a cost of a proportional increase in the number of messages at these two levels. The number of messages exchanged by the top level also increases by approximately 600 messages in this range of periodical executions. This is because the proximity view is updated and therefore, there are available candidate parents and children that are contacted for the establishment of parent-child links. This trend is observed in the evaluation of all of the strategies.

The effect of remote clustering

T-MAN enhances clustering at the middle level and therefore plays a crucial role for the convergence speed of the performance metrics. For example, the connectedness achieved by GREEDY without T-MAN is 10% less and its convergence 200 epochs slower. Similarly, the fitness achieved with GREEDY by excluding T-MAN is 2% lower and 150 epochs slower. Because of the slower update of the proximity view, the communication cost in the top level is lower by 500 messages in every epoch.

Comparison with Related Work

In contrast to the application-level multicasting methodologies of Banerjee et al. [2003] and Wang et al. [2010] that

are based on central components for managing or bootstrapping a tree overlay, AETOS is designed as an application-independent middleware. Table 5 summarizes the related decentralized mechanisms discussed in this section. Note that the comparison is mainly qualitative. AETOS contributes a more generic, modular and reconfigurable self-organization of trees instead of performance enhancements.

Self-organization and tree optimization in related approaches is designed based on the requirements of a specific application type or domain. Tan et al. [2005a] introduce a combination of a bandwidth-ordered and time-ordered tree overlay for streaming applications. The position of each node is continuously evaluated and improved by calculating the 'service capability contribution' (SCC). This metric is the product of the outbound bandwidth of a node and uptime in the system. Based on the SCC metric, nodes are sorted by shifting parent-children positions. In contrast, the nodes of AETOS are ranked with an abstract weight related to applications.

The approaches of Leita et al. [2007], Yuan and Wei Tsang [2004], Choe et al. [2004] and Tan et al. [2005b] are limited to building a minimum spanning tree known to minimize latency in multimedia multicasting applications. These approaches do not adopt any ordering process of nodes that could express the heterogeneity of their performance. AETOS does not explicitly optimize trees for a given underlying network infrastructure. Nevertheless, it can provide such optimization if it is instructed appropriately. Similar issues concern tree overlays designed for optimizing distributed database queries, e.g., range queries. In this case, trees act as indexing structures. Load-balancing for minimizing the computational load of nodes is the optimization applied in this application type by Jagadish et al. [2005, 2006b] and Sapiecha and Grzegorz [2013].

AETOS combines proactive and reactive features in its design. Proactiveness is related to the (i) ordering of the tree to minimize, for example, the impact of node failures and (ii) the dynamic maintenance of the random and proximity view. In contrast, proactive mechanisms may cause a high communication and computational cost. Fei and Yang [2007] introduce a highly proactive method for maintaining a minimum spanning tree for multicasting. Alternative back-up parents are negotiated before departures of nodes occur. Therefore, reconnections are rapid and the tree becomes more resilient. However, the benefit of this high responsiveness comes with a high bandwidth consumption. This proactive method pays back the high bandwidth cost in case of frequent node departures. If node departures are rare, the system experiences a significant bandwidth overhead without an actual benefit in the resilience of the tree. Alternative proactive methodologies for tree resilience are evaluated by Birrer and Bustamante [2007]. These methodologies are based on various schemes of redundancy: (i) cross-link, (ii) in-tree and (iii)

multiple-tree redundancy, however, communication cost may increase significantly. Proactive and reactive components of the epidemic broadcast trees introduced by Leita et al. [2007] work in synergy to provide a fast tree construction and repair. However and in contrast to AETOS, the proximity of nodes is not considered and the formed trees are not ordered to express the heterogeneity of nodes.

Most other related approaches [Tan et al., 2005a; Costa and Frey, 2005; Hoai Son Nguyen and Lan, 2013; Walters et al., 2008; Jagadish et al., 2006b] are based on adaptation without the use of proactive back-end mechanisms. Frey and Murphy [2008] propose a number of transformation and rewiring strategies for reconnecting disconnected branches. The configuration of tree connections is similar to the configuration which the top level of AETOS performs. However, the adaptations of Frey and Murphy [2008] do not sort a tree or control its graph properties required for an application.

AETOS is highly customizable as it is based on an architecture composed of three continuously reconfigurable levels and several adaptation strategies. The application-dependence of most of the related approaches [England et al., 2007; Jagadish et al., 2006b; Leita et al., 2007; Tang et al., 2005; Tan et al., 2005a] unavoidably results in a low customization and optimization options. Some of these approaches introduce additional costly protocols and management mechanisms to improve customization of self-organization in dynamic scenarios, e.g., custom tree repairing and load-balancing.

Exploration of multiple strategies for self-organization of trees is significantly limited in related work. Frey and Murphy [2008] illustrate six repair strategies for trees. These strategies are related to the satisfaction of application requirements, e.g., node degrees, during parent-child reconnections in the tree. These strategies appear to influence the tree topology in a similar way to the strategies of AETOS, for example, forming lower or higher branches. However, compared to AETOS, there are two fundamental differences: (i) The AETOS agents always respect the tree criteria, such as the maximum number of children. (ii) The adaptation strategies do not only concern the tree repairs, but both the building and maintenance of trees. Therefore, the adaptation strategies are integrated to a higher degree in the core self-organization of AETOS.

Conclusions

This paper concludes that the adaptation strategies of AETOS are able to build and maintain complex self-organizing tree topologies defined by several graph properties related to a wide range of application requirements. Our experimental findings confirm the cost-effectiveness of AETOS by introducing and measuring a wide range of generic performance metrics applicable for various applications. Self-organization is designed as a service in contrast to other re-

Table 5

An overview of related decentralized mechanisms to AETOS.

	Self-organization	Optimization	Reconfigurability	Applications
AETOS	gossiping, clustering, negotiation	degree-bounding, ordering, balancing and completeness	three-level modularity and adaptation strategies	generic
Choe et al. [2004]	edge changes	degree-bounding and diameter minimization	-	multicasting
Costa and Frey [2005]	DHT mapping and breadth-first traversing	degree-bounding	-	content-based publish-subscribe systems
England et al. [2007]	minimum spanning tree algorithms, e.g., Bellman-Ford algorithm	hop counting, path weighting	trade-offs by weighted optimization metrics	sensor networks and load scheduling
Fei and Yang [2007]	minimum spanning tree algorithms, e.g., Prim's algorithm	degree-bounding, proactive recovery	NAT ^a /firewalls, grandfathers-siblings and uncles-granduncles candidates	multicasting
Frey and Murphy [2008]	negotiation protocol and cycle prevention	degree-bounding	repair strategies	generic
Jagadish et al. [2005, 2006b]	join at first available bottom node	balancing	network restructuring and load-balancing	multi-dimensional data indexing, exact and range queries
Leitao et al. [2007]	pull and lazy push gossiping strategies	hop counting	threshold-based optimization and gossiping parameters	broadcasting
Tan et al. [2005a]	leaf joins and sift-up operations	bandwidth and time based ordering	frequency of sift-up operations	live streaming
Tang et al. [2005]	DVMRP ^b mechanism over random/proximity-based gossiping	degree-bounding, latency	dynamic periodical executions	group communication
Yuan and Wei Tsang [2004]	mesh-first protocol and top-down construction	minimum spanning and shortest path tree	timer for parent selection	video streaming

^a Network Address Translation. ^b Distance Vector Multicast Routing Protocol.

lated methodologies that (i) provide a dedicated mechanism for an application domain and (ii) study a subset of the graph properties introduced in this paper. The separation of organizational complexity in different levels proves to be a modular and customizable design choice for this service. The adaptation strategies result in different tree topologies by exploring different performance trade-offs. The adaptation strategies can also be combined to compose hybrid strategies with collective properties suggesting a new meta-level of abstraction and reasoning in future work.

References

- Baldwin, D. and Scragg, G. (2004). *Algorithms and Data Structures: The Science of Computing*. Charles River Media, Inc., Rockland, MA, USA.
- Banerjee, S., Kommareddy, C., Kar, K., Bhattacharjee, B., and Khuller, S. (2003). Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2003*, pages 1521–1531, Los Alamitos, CA, USA. IEEE.
- Bhagwan, R., Savage, S., and Voelker, G. (2003). Understanding Availability. In *Proceedings of Peer-to-Peer Systems II, Second International Workshop, IPTPS 2003*, volume 2735 of *Lecture Notes in Computer Science*, pages 256–267, Heidelberg. Springer-Verlag Berlin.
- Birrer, S. and Bustamante, F. (2007). A Comparison of Resilient Overlay Multicast Approaches. *IEEE Journal on Selected Areas in Communications*, 25(9):1695–1705.
- Chakravarti, A. J., Baumgartner, G., and Lauria, M. (2006). Self-Organizing Scheduling on the Organic Grid. *Internation*

- tional Journal of High Performance Computing Applications*, 20(1):115–130.
- Choe, H., Cho, S., and Kim, C.-k. (2004). A Dynamic Mechanism for Distributed Optimization of Overlay Multicast Tree. In *Proceedings of the International Conference on Information Networking, ICOIN 2004*, Heidelberg. Springer-Verlag Berlin.
- Costa, P. and Frey, D. (2005). Publish-Subscribe Tree Maintenance over a DHT. In *Proceedings of the 4th International Workshop on Distributed Event-Based Systems, DEBS 2005*, pages 414–420, Washington, DC, USA. IEEE Computer Society.
- England, D., Veeravalli, B., and Weissman, J. B. (2007). A Robust Spanning Tree Topology for Data Collection and Dissemination in Distributed Environments. *IEEE Transactions on Parallel and Distributed Systems*, 18(5):608–620.
- Fei, Z. and Yang, M. (2007). A Proactive Tree Recovery Mechanism for Resilient Overlay Multicast. *IEEE/ACM Transactions on Networking*, 15(1):173–186.
- Frey, D. and Murphy, A. L. (2008). Failure-Tolerant Overlay Trees for Large-Scale Dynamic Networks. In *Proceedings of the International Conference on Peer-to-Peer Computing*, pages 351–361, Los Alamitos, CA, USA. IEEE.
- Galuba, W., Aberer, K., Despotovic, Z., and Kellerer, W. (2009). ProtoPeer: A P2P Toolkit Bridging the Gap Between Simulation and Live Deployment. In *Proceedings of the Second International Conference on Simulation Tools and Techniques, ICST 2009*, pages 1–9, Gent, Belgium. ACM.
- González-Beltrán, A., Milligan, P., and Sage, P. (2008). Range queries over skip tree graphs. *Computer Communications*, 31(2):358–374.
- Gross, J. and Yellen, J. (2005). *Graph Theory and Its Applications*. CRC Press, Boca Raton, FL, USA, 2nd edition.
- Hoai Son Nguyen, N. A. N. and Lan, H. B. T. (2013). Bamchord: Dht-based bandwidth adaptive multicast system. *International Journal of Distributed Systems and Technologies (IJDST)*, 4(1).
- Jagadish, H. V., Ooi, B. C., Tan, K.-L., Vu, Q. H., and Zhang, R. (2006a). Speeding up Search in Peer-to-Peer Networks with A Multi-way Tree Structure. In *Proceedings of the International Conference on Management of Data, SIGMOD 2006*, pages 1–12, New York, USA. ACM Press.
- Jagadish, H. V., Ooi, B. C., and Vu, Q. H. (2005). BATON: a balanced tree structure for peer-to-peer networks. In *Proceedings of the 31st International Conference on Very Large Databases, VLDB 2005*, pages 661–672, Trondheim, Norway. ACM.
- Jagadish, H. V., Ooi, B. C., Vu, Q. H., Zhang, R., and Aoying, Z. (2006b). VBI-Tree: A Peer-to-Peer Framework for Supporting Multi-Dimensional Indexing Schemes. In *Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006*, pages 34–34, Washington, DC, USA. IEEE Computer Society.
- Jelasity, M., Montresor, A., and Babaoglu, O. (2009). T-Man: Gossip-based fast overlay topology construction. *Computer Networks*, 53(13):2321–2339.
- Jelasity, M., Voulgaris, S., Guerraoui, R., Kermarrec, A.-M., and van Steen, M. (2007). Gossip-based Peer Sampling. *ACM Transactions on Computer Systems*, 25(3).
- Leitao, J., Pereira, J., and Rodrigues, L. (2007). Epidemic Broadcast Trees. In *Proceedings of the 26th International Symposium on Reliable Distributed Systems, SRDS 2007*, pages 301–310, Los Alamitos, CA, USA. IEEE.
- Manouselis, N. and Costopoulou, C. (2007). Analysis and Classification of Multi-Criteria Recommender Systems. *World Wide Web*, 10(4):415–441.
- Oey, M., van Splunter, S., Ogston, E., Warnier, M., and Brazier, F. M. T. (2010). A Framework for Developing Agent-Based Distributed Applications. In *Proceedings of the International Conference on Web Intelligence and Intelligent Agent Technology, WI-IAT 2010*, pages 470–474, Los Alamitos, CA, USA. IEEE.
- O'Madadhain, J., Fisher, D., Smyth, P., White, S., and Boey, Y.-B. (2005). Analysis and Visualization of Network Data using JUNG. *Journal of Statistical Software*, 10(2):1–35.
- Pournaras, E. (2013). *Multi-level Reconfigurable Self-organization in Overlay Services*. PhD thesis, Delft University of Technology.
- Pournaras, E., Warnier, M., and Brazier, F. M. T. (2010a). Adaptation Strategies for Self-management of Tree Overlay Networks. In *Proceedings of the 11th IEEE/ACM International Conference on Grid Computing, Grid 2010*, pages 401–409, Los Alamitos, CA, USA. IEEE.
- Pournaras, E., Warnier, M., and Brazier, F. M. T. (2010b). Self-Optimised Tree Overlays Using Proximity-Driven Self-Organised Agents. In *Complex Intelligent Systems and their Applications*, volume 41 of *Springer Optimization and its Applications*, chapter 7, pages 137–161. Springer New York, New York, NY.

- Sapiecha, K. and Grzegorz, L. (2013). Scalable distributed two-layer data structures (sd2ds). *International Journal of Distributed Systems and Technologies (IJDST)*, 4(2).
- Tan, G., Jarvis, S. A., Chen, X., Spooner, D. P., and Nudd, G. R. (2005a). Performance Analysis and Improvement of Overlay Construction for Peer-to-Peer Live Media Streaming. *Simulation*, 82(2):169–178.
- Tan, S.-W., Waters, G., and Crawford, J. (2005b). MeshTree: Reliable Low Delay Degree-bounded Multicast Overlays. In *Proceedings of the 11th International Conference on Parallel and Distributed Systems, ICPADS 2005*, volume 2, pages 565–569, Los Alamitos, CA, USA. IEEE.
- Tang, C., Chang, R., and Ward, C. (2005). GoCast: Gossip-Enhanced Overlay Multicast for Fast and Dependable Group Communication. In *Proceedings of the International Conference on Dependable Systems and Networks, DSN 2005*, pages 140–149, Los Alamitos, CA, USA. IEEE.
- Walters, A., Zage, D., and Rotaru, C. N. (2008). A Framework for Mitigating Attacks Against Measurement-Based Adaptation Mechanisms in Unstructured Multicast Overlay Networks. *IEEE/ACM Transactions on Networking (TON)*, 16(6):1434–1446.
- Wang, F., Xiong, Y., and Liu, J. (2010). mTreebone: A Collaborative Tree-Mesh Overlay Network for Multicast Video Streaming. *IEEE Transactions on Parallel and Distributed Systems*, 21(3):379–392.
- Yuan, L. and Wei Tsang, O. (2004). Distributed Construction of Resource-efficient Overlay Tree by Approximating MST. In *Proceedings of the International Conference on Multimedia and Expo, ICME 2004*, pages 1507–1510, Los Alamitos, CA, USA. IEEE.
- Zhuge, H. and Feng, L. (2008). Distributed Suffix Tree Overlay for Peer-to-Peer Search. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):276–285.

Appendix A Organizational Goal

Assume an ordered list of ranked nodes $\mathbf{w} = \{w_0, \dots, w_{n-1}\}$ under the ‘>’ relation with the respective number of children $\mathbf{k} = \{k_0, \dots, k_{n-1}\}$. Each size $k_i \in \mathbf{k}$ corresponds to the ranked node $w_i \in \mathbf{w}$. An ordered tree $\mathbf{T} = \{\mathbf{w}_0, \dots, \mathbf{w}_{h-1}\}$ can be built by splitting the ordered list \mathbf{w} in h number of levels $\mathbf{w}_0, \dots, \mathbf{w}_{h-1}$. An indexed level $u - 1$ of parents $\mathbf{w}_{u-1} = \{w_{p_r}, \dots, w_{p_l}\}$ is defined by the list of the lowest ranked node $w_{p_r} \in \mathbf{w}$ and the highest ranked node $w_{p_l} \in \mathbf{w}$ in this level respectively. Similarly, the lowest ranked child $w_{c_r} \in \mathbf{w}$ and the highest ranked child $w_{c_l} \in \mathbf{w}$ form the list of

ranked children $\mathbf{w}_u = \{w_{c_r}, \dots, w_{c_l}\}$ that builds the next level with size $|\mathbf{w}_u| = g_u$.

A degree-bounded, ordered, balanced and complete tree can be built in a centralized fashion according to Algorithm 1. The tree is built level-by-level using a buffer of parents p_r, \dots, p_l and children c_r, \dots, c_l indexing nodes in the ordered list \mathbf{w} . The number of children that the next level includes is computed at the beginning (line 4-8 of Algorithm 1). Next, every indexed node from the level of parents is matched with the indexed nodes from the level of children (line 9-19 of Algorithm 1). Therefore, the tree views can be filled using these indices. This process repeats by incrementing the level of parents to level of children and recomputing the next level of children at the next iteration (line 21 of Algorithm 1). Figure A1 visualizes this incremental process.

Algorithm 1 A centralized computation of an optimally organized tree topology with the properties introduced in ‘Graph properties and applications’.

Require: $\mathbf{w} = \{w_0, \dots, w_{n-1}\}$, $\mathbf{k} = \{k_0, \dots, k_{n-1}\}$

```

1: // Initialization
2:  $p_r = p_l = 0$ ,  $c_r = c_l = 1$ ,  $u = 1$ ,  $g_0 = 0$ 
3: while true do
4:   // Computation of the next level of children nodes
5:   for  $i = p_r$  to  $p_l$  do
6:      $g_u = g_u + k_i$ 
7:   end for
8:    $c_r = p_r + 1 = c$ ,  $c_l = p_l + g_u - 1$ 
9:   // Filling the tree views of the nodes
10:  for  $i = p_r$  to  $p_l$  do
11:    for  $j = c$  to  $c + k_i - 1$  do
12:      if  $j > n - 1$  then
13:        return
14:      end if
15:       $\mathbf{v}_i(\text{tree}) \cup \{w_j\}$  // Adding a child
16:       $\mathbf{v}_j(\text{tree}) = \{w_i\}$  // Adding a parent
17:    end for
18:     $c = c + k_i$ 
19:  end for
20:  // Incrementing the level of parents to level of children
21:   $p_r = c_r$ ,  $p_l = c_l$  and  $u = u + 1$ 
22: end while
Ensure:  $\mathbf{v}_i(\text{tree}) \forall i \in \{0, \dots, n - 1\}$ 

```

Appendix B Top Level Interactions

Assume a ranked node w_i that has in its tree view $\mathbf{v}_i(\text{tree})$ the parent w_p and the ordered (>) set of children $\{w_{c_r}, \dots, w_{c_l}\}$. This node performs coordination with a ranked node w_j using horizontal interactions between the levels of the AETOS architecture. Algorithm 2 and 3 illustrate the peer-to-peer coordination between the ranked nodes w_i and w_j .

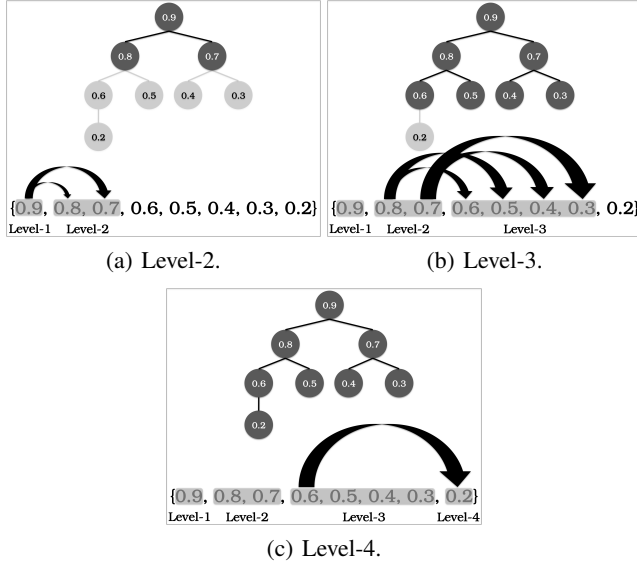


Figure A1. Centralized building of a binary, ordered, balanced, and complete tree by defining the tree levels incrementally from an ordered list of ranked nodes.

A *request* is potentially sent to a candidate parent or child w_j provided by the middle level. This *request* is actually sent if and only if this candidate parent/child either fills the tree view or updates it by replacing a lower ranked node (line 2 and 9 of Algorithm 2).

Algorithm 2 Initiation of the tree coordination.

Require: A candidate parent or child w_j

```

1: if  $w_j$  is candidate parent then
2:   if  $w_p = \emptyset$  or  $w_j > w_p$  then
3:     request (parent request,  $w_i$ ,  $w_j$ )
4:     return request
5:   else
6:     return null
7:   end if
8: else //  $w_j$  is candidate child
9:   if  $|v_i(\text{tree})| - 1 < k_i$  or  $w_j > w_{c_i}$  then
10:    request (child request,  $w_i$ ,  $w_j$ )
11:    return request
12:   else
13:     return null
14:   end if
15: end if

```

Ensure: *request*

The response of a node to a *request* depends on the feedback that the generated clustering criteria define. Positive feedback corresponds to an *acknowledgment* message and negative feedback corresponds to a *rejection* message.

Coordination logic includes the reactions to the re-

Algorithm 3 Completion of the tree coordination.

Require: Feedback and the ranked node w_j

```

1: if feedback='positive' then
2:   acknowledgment ( $w_i$ ,  $w_j$ )
3:   return acknowledgment
4: else // feedback='negative'
5:   rejection ( $w_i$ ,  $w_j$ )
6:   return rejection
7: end if

```

Ensure: *acknowledgment* or *rejection*

ceived messages. These reactions are shown in Algorithm 4. The rest of the algorithms illustrate the reactions to each of the received messages.

Algorithm 4 The reactions to received messages.

Require: A message from a ranked node w_j

```

1: if message=parent request then
2:   execute Algorithm 5
3: else if message=child request then
4:   execute Algorithm 6
5: else if message=acknowledgment then
6:   execute Algorithm 7
7: else if message=rejection then
8:   execute Algorithm 8
9: else if message=removal then
10:  execute Algorithm 9
11: else
12:  return null
13: end if

```

Algorithm 5 outlines the reaction to a received *parent request*. There are two conditions under which a parent-child link is established. In the first condition (line 1 of Algorithm 5), the tree view of the node w_i is not full. In the second condition (line 2 of Algorithm 5), the lowest ranked child c_i is replaced with the ranked node w_j (lines 3-9 of Algorithm 5) if it is ranked lower than w_j . The link establishment follows an *acknowledgment* message (lines 7-8 of Algorithm 5), whereas in case none of these conditions are met, a *rejection* message (lines 14-15 of Algorithm 5). Note that in case of an *acknowledgment*, positive feedback and the weight w_j is contained in the message (lines 10-12 of Algorithm 5).

Algorithm 6 illustrates the reactions to a received *child request* message that follow the same concept to the reactions of a received *parent request*.

Algorithm 7 illustrates the reactions to a received *acknowledgment* message that results in the establishment of a new parent-child link. The sender node w_j , as a new parent or child, is added in the tree view $v_i(\text{tree})$ of node w_i (line 7 and 17 of Algorithm 7). A potential *removal* message is sent if a

Algorithm 5 The reactions to a *parent request* message.**Require:** *parent request* message from a ranked node w_j

```

1: if  $|v_i(\text{tree})| - 1 < k_i$  or  $w_j > w_{c_i}$  then
2:   if  $|v_i(\text{tree})| - 1 = k_i$  and  $w_j > w_{c_i}$  then
3:     removal ( $w_i, w_{c_i}$ )
4:     send removal
5:      $\{w_{c_i}, \dots, w_{c_i}\} \setminus w_{c_i}$ 
6:   end if
7:   acknowledgment ( $w_i, w_j$ )
8:   send acknowledgment
9:    $\{w_{c_i}, \dots, w_{c_i}\} \cup w_j$ 
10:  feedback='positive'
11:  clustering criteria (feedback,  $w_{c_i}$ )
12:  return clustering criteria
13: else
14:   rejection ( $w_i, w_j$ )
15:   send rejection
16:  return null
17: end if

```

Ensure: *clustering criteria***Algorithm 6** The reactions to a *child request* message.**Require:** *child request* message from a ranked node w_j

```

1: if  $w_p = \emptyset$  or  $w_j > w_p$  then
2:   if  $w_p \neq \emptyset$  and  $w_j > w_p$  then
3:     removal ( $w_i, w_p$ )
4:     send removal
5:      $w_p = \emptyset$ 
6:   end if
7:   acknowledgment ( $w_i, w_j$ )
8:   send acknowledgment
9:    $w_p = w_j$ 
10:  feedback='positive'
11:  clustering criteria (feedback,  $w_j$ )
12:  return clustering criteria
13: else
14:   rejection ( $w_i, w_j$ )
15:   send rejection
16:  return null
17: end if

```

Ensure: *clustering criteria*

node is replaced by w_j (lines 3-4 and 13-14 of Algorithm 7). A positive feedback and the new ranked node added in the tree view form the clustering criteria for adapting the middle level of AETOS (lines 8-9 and 18-19 of Algorithm 7).

The reactions to a received *rejection* message do not introduce any changes in the tree view as illustrated in Algorithm 8. However, a *rejection* message triggers clustering criteria containing a negative feedback and the weight of the sender node w_j .

Algorithm 7 The reactions to an *acknowledgment* message.**Require:** *acknowledgment* message from a ranked node w_j

```

1: if  $w_j > w_i$  then
2:   if  $w_p \neq \emptyset$  then
3:     removal ( $w_i, w_p$ )
4:     send removal
5:      $w_p = \emptyset$ 
6:   end if
7:    $w_p = w_j$ 
8:   feedback='positive'
9:   clustering criteria (feedback,  $w_j$ )
10:  return clustering criteria
11: else //  $w_j < w_i$ 
12:   if  $|v_i(\text{tree})| - 1 = k_i$  then
13:     removal ( $w_i, w_{c_i}$ )
14:     send removal
15:      $\{w_{c_i}, \dots, w_{c_i}\} \setminus w_{c_i}$ 
16:   end if
17:    $\{w_{c_i}, \dots, w_{c_i}\} \cup w_j$ 
18:   feedback='positive'
19:   clustering criteria (feedback,  $w_j$ )
20:  return clustering criteria
21: end if

```

Ensure: *clustering criteria***Algorithm 8** The reactions to a *rejection* message received in the top level of AETOS.**Require:** *rejection* message from a ranked node w_j

```

1: feedback='negative'
2: clustering criteria (feedback,  $w_j$ )
3: return clustering criteria

```

Ensure: *clustering criteria*

Finally, the Algorithm 9 shows that a ranked node w_j is removed from the tree view if a *removal* message is received. A removal of a node from the tree view is always accompanied with clustering criteria containing a negative feedback.

Algorithm 9 The reactions to a *removal* message.**Require:** *removal* message from a ranked node w_j

```

1: if  $w_j > w_i$  then
2:    $w_p = \emptyset$ 
3: else //  $w_j < w_i$ 
4:    $\{w_{c_i}, \dots, w_{c_i}\} \setminus w_j$ 
5: end if
6: feedback='negative'
7: clustering criteria (feedback,  $w_j$ )
8: return clustering criteria

```

Ensure: *clustering criteria*

The condition of delivering the tree view is controlled by the tree criteria provided by the application. Examples of criteria that can be engaged are a periodic delivery or a minimum rate of changes occurring in the tree view.

Finally, note that the coordination performed in the top level assumes a synchronous communication for the matters of simplicity in the illustration of the algo-

rithms. Nonetheless, a concurrent version of the top level in AETOS is implemented in the simulation framework of AgentScape [Oey et al., 2010]. In contrast to the top level, the gossiping realizations of the bottom and middle level tolerate the lack of concurrency with various solutions discussed by Jelasity et al. [2007].