Mr.Tree: Multiple Realities in Tree-based Monitoring Overlays for Peer-to-Peer Networks

Andreas Disterhöft, Phillip Sandkühler, Andre Ippisch and Kalman Graffi
Technology of Social Networks, Heinrich Heine University of Düsseldorf
Universitätsstrasse 1, 40225 Düsseldorf, Germany
Email: {andreas.disterhoeft, phillip.sandkuehler, ippisch, graffi}@hhu.de

***Abstract*—Peer-to-peer systems are difficult to manage due to their decentralized nature and the lack of a central controlling instance. Thus, establishing a quality of service is a challenging task. In the last decade, decentralized monitoring approaches have been proposed, which are either precise or fault-tolerant. Tree-based approaches, which belong to the former category, are prone to churn as the structure needs to reorganize itself on node failures. In this paper, we propose the monitoring approach Mr.Tree, a protocol that enhances the robustness of existing tree-based approaches by introducing multiple realities, through building up multiple trees, and smartly replicating and maintaining aggregated data. Three modes are proposed to fit individual scenarios and needs. The evaluation shows a significant decrease of the error in completeness of monitoring data while keeping the costs relatively low. The error of the extended protocol under heavy churn is in the magnitude of the original protocol under low churn conditions, which is 8%, whereby the error of the original protocol under heavy churn raises to a value above 22%. The superior precision cancels out the increased costs already in scenarios with moderate churn.**

I. Introduction

Nowadays, it is common, widespread and especially good practice to monitor communication systems. Their operators are mainly interested in a certain quality of service (QoS) with the aim to maximize the user experience. Beside monitoring centralized networks, decentralized peer-to-peer (p2p) systems also benefit from monitoring approaches. Peers participating in p2p systems run software on their machine, that comes with preconfigured settings. These are optimized for certain network and peer states and may be inappropriate for other unanticipated states. Especially, these systems are operated by heterogeneous peers whose capabilities and reliability are not comparable with powerful server farms. For that reason, it is essential to monitor the network and to survey important metrics in order to adjust to changes and thereby to ensure a certain QoS. For example, by monitoring system state metrics like the current average hop count or replication status of data, we are able to optimize the network's parameters, such as time to live interval, or replication degree for distributed storage applications. Graffi et al. [9] propose such a controlling unit.

Therefore, in the last decade, monitoring approaches have been proposed whose structure can be divided into *gossip* and *tree* as well as a merge of them called *hybrid*. A detailed categorization can be looked up in the survey by Makhloufi et al. [14]. Gossip-based approaches like the push-sum algorithm [12] or T-Man [10] are robust against node failures, but they are not as precise as the ones using trees. Tree-based monitoring techniques have the potential to collect very precise information about the network's state. They rely on aggregated data from each participant of the network instead of estimating the overall capacities based on the average of data from immediate neighbors. Makhloufi et al. [14] have shown the downside of tree-based approaches. Those highly accurate techniques are prone to highly dynamic environments, where nodes join and leave the system alike: churn. Churn heavily impacts the stability of the tree structure of such monitoring techniques for they rely on an intact tree infrastructure. Depending on the actual approach, whole subtrees may be eclipsed which ultimately leads to loss of data and accuracy.

We propose Mr.Tree, a replication technique which tackles the aforesaid problem. In a nutshell, Mr.Tree smartly replicates monitoring data within logical hypernodes consisting of physical peers. Our proposed protocol can be implemented by state-of-the-art tree-based monitoring methods. We require the following attributes in order to operate our solution:

- The Monitoring overlay operates on top of any structured peer-to-peer network which is KBR-compliant [4].
- The Monitoring method follows the global aggregation scheme, thus all nodes of the p2p system are involved.
- In the aggregation process nodes proactively push data to their parents.

Candidates fulfilling these requirements are SDIMS [19], SkyEye.KOM [8] and GAP [5] with its extensions A-GAP [15] and TCA-GAP [18]. Our exact contributions are as follows:

- Proposal of Mr.Tree, a generic protocol to enhance the resilience of tree-based monitoring systems against churn.
- Prototype implementation on top of SkyEye.KOM [8].
- Detailed evaluation with churn models inducing increasingly node fluctuations and an in-depth comparison between our enhanced version and the vanilla protocol.

The remainder is structured as follows. Section II presents related work in the field of robust structured monitoring approaches. Section III presents our protocol Mr.Tree, which is evaluated in Section IV. We conclude our work in Section V.

II. Related work

The field of robust tree-based aggregation techniques is discussed by research fields like the aggregation by broadcasting

This work was supported by the DFG Grant GR4498 1-1 - OverlayMeter

and multicasting over DHTs, efficient data dissemination in sensor networks and the decentralized monitoring.

The idea to build up multiple trees to enhance the robustness of mechanisms, that utilize these, is not new. Li et al. [13] propose an aggregation and broadcast building block, that operates on DHTs and aims to efficiently disseminate global information. The aggregation tree is built up bottom-up using a parent function which maps nodes to their parent. In order to overcome single point of failures the authors propose to build up parallel trees. After a node has decided to do such, it registers itself as the root of this tree and makes itself known to roots of existing trees by announcing the appropriate parent function. The just announced tree gets propagated by that tree and nodes eventually join the new tree. By choosing parent functions smartly, parallel trees with interior-distinct nodes can be built up to maximize robustness of such. Castro et al. [2] propose SplitStream, a robust mechanism for multicasting in p2p systems. The robustness is achieved by building forests of multicast trees with interior-distinct nodes. Wang et al. propose mTreebone [17], a hybrid overlay operating a multicast tree used to efficiently stream live content. The structure consists of a tree-based backbone, maintained by stable peers, and mesh-based outskirts, which are attached to the backbone. However, the main difference to Mr.Tree is that no data maintenance is considered, whereas distributed monitoring approaches require an infrastructure to collect, hold and maintain aggregated data. Mr.Tree supplies a reliable intra-tree communication and replication of monitoring data between homogeneous tree nodes at the same virtual tree positions. Additional trees are dynamically constructed and pruned if no longer required.

In the research field of sensor networks, sensors collect and route data to a common sink. To save energy during the routing process and enhance the robustness of such, it is of high interest to minimize data to be routed and introduce multipath routes. Ideally, this can be achieved efficiently by aggregating and sending data along multiple routing paths. Unfortunately, the multipath routing and aggregation of data contradict as duplicates of data sets arise. This ultimately leads to precision issues. Therefore, Considine et al. [3] propose Sketches which efficiently mitigate the negative effect duplicate data have on the precision. Unlike sensor networks in DHT-based p2p networks, a structure can be used to overcome the problem of duplicated data in multipath aggregation algorithms. Mr.Tree provides a multi tree approach to aggregate data along logical distinct paths, thus, duplication of data can not arise.

Jesus et al. [11] divide the field of distributed data aggregation into structured, unstructured and hybrid. The structured ones are either hierarchical, which is subdivided into tree-, cluster-, or multipath-based, or they form a ring. Qualitative evaluations show on one hand, that the multipath hierarchical approach of Sketches is fast and fault-tolerant but sacrifices accuracy. On the other hand, tree-based hierarchical approaches are highly accurate, as aggregated data is forwarded using a spanning tree. But they are prone to node failures and single point of failures, because the result is calculated at a single node. Makhloufi et al. [14] propose to refine the classification

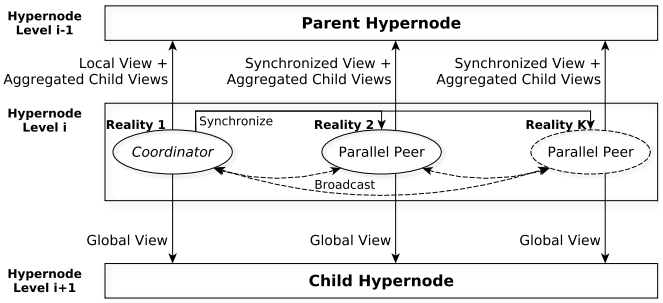


Fig. 1. Mr.Tree: Hypernode Overview

of decentralized monitoring of aggregated information by adding two criteria: neighborhood information and network view. The quantitative evaluation supports the qualitative analysis of Jesus. It shows, that their implemented versions of GAP [5] and SDIMS [19], which both enable proactive dissemination of aggregation bottom-up and spreading global view top-bottom, performs poor under dynamic network conditions where nodes fail. This structural weakness also applies to A-GAP [15], TCA-GAP [18] and SkyEye.KOM [8] as all of them are classified in the category of tree-based, proactive and global aggregation systems. In this paper, the structural weakness of SkyEye.KOM is verified and a mitigation of such systems is proposed: Mr.Tree, which applies a hypernode structure with multiple trees. Nodes at equal positions, in terms of virtual ids, share aggregated data for replication purposes and duplicate data is therefore eliminated.

III. Mr.Tree - Our Approach

In this section, the proposed protocol, which enhances tree-based monitoring services meeting the requirements stated in Section I, is described. First, we establish the structure of our replication mechanism and the terminology used to describe the functionality of the solution.

A. Overview

The vulnerability of tree-based structures when facing churn is caused by the truncation of potentially large subtrees when a node on a high level in the hierarchy suddenly leaves the network. To counteract this, we propose the introduction of *hypernodes*. Instead of a single peer taking over responsibility for a node of the tree, we rather identify the node with a group of peers as depicted in Figure 1. Because each node now contains several peers, the failure of a single peer can be compensated by the remaining peers inside the node. This compensation is achieved by replicating, maintaining and providing the aggregation data base inside such a hypernode. Thus, a leaving peer, responsible for an interior tree node, does not automatically lead to data loss.

In order to make this work, some changes are introduced in the original structure of the tree-based monitoring service. The peers in a hypernode must maintain a consistent data replication which is achieved by employing an agreement protocol. In our approach, we deal with this issue by specifying two roles each peer inside a node performs. As shown in Figure 1, there is one peer that adopts the role of the *Coordinator*, whereas the

remaining peers are identified as *Parallel Peers*. A Coordinator is meant to be the peer which was responsible for that tree node in the original protocol. The Parallel Peers are synchronized by the Coordinator with a data set that contains the Coordinator's local view on the network. They accept the received data as the new view until they receive a new synchronization message which then overrides the existing view. Because they do not inject any data, this mechanism keeps the original aggregation protocol unimpacted.

Another required alteration concerns the communication between the tree nodes. Hence, we introduce the term of *Multiple Realities*. Instead of establishing a single tree structure, our approach uses a modification of the used algorithm to create several trees according to the desired degree of replication. We call each tree a *Reality*. The original tree becomes the first Reality and each additional tree is denoted as a *Parallel Reality*. Note that each tree is shifted by an offset relative to the other trees. The spacing between the root nodes of the trees depends on the specified number of Realities. As depicted in Figure 1, we group all peers of equal positions together in different Realities to form hypernodes. For instance, the peers, that represent the root node in different Realities, form the root hypernode of the hypernode tree and the peer from the first Reality becomes the Coordinator. This approach now yields multiple tree structures in which communication follows the links of the tree nodes, except that we also introduce links between the different trees by interconnecting peers inside a hypernode. As a result, links between two hypernodes comply with the number of used Realities. In contrast to single links or each-to-each connections, this provides a middle ground between redundancy and resource efficiency. Because we use the existing functions for setting up the slightly modified tree structure and duplicate the already existing communication in the tree-based base service, our approach only requires minimal changes in the original communication protocol. In the detailed description of our general protocol in Section III-B3, we describe a technique to reduce the overhead even further during the more stable phases of the network.

B. Protocol

1) Virtual IDs: Because of churn characteristics and changing responsibilities, messages must be sent to a certain position in the hypernode tree structure rather than a physical peer. Hence, each message holds two additional header entries: information on the receiving position and the position from where the message is sent. For the identification of the positions in the hypernode tree we define *Virtual IDs*. Virtual IDs consist of two aspects, the tree node ID, where the message would be routed to in the first Reality, and the target Reality, which denotes the required offset in the ID space. The service uses the base position and the required offset to calculate the specific key y in the overlay ID space.

Position Determination is a crucial part of the hypernode structure. For a reliably working data replication system, we need to ensure that each Reality inside one hypernode is occupied by distinct physical peers. This decreases the chance of all replicating participants failing during the same data distribution phase, that is, a push interval. Depending on the overlay's metrics, a certain tree node might fall into a single peer's responsibility in different Realities because a peer in the KBR environment is responsible for all keys within its responsibility domain. Also, the number of peers needed inside a hypernode can vary depending on the expected severeness of churn in different environments. To achieve this, a maximum number of Realities is set and the offset for each reality is calculated by dividing the ID space by the maximum number of Realities. By this means, the ID of the peers for one hypernode is distributed evenly across the ID space.

While this does not pose a problem for a setup where a fixed number of Realities is used at all times, it is not suited for networks that may change the number of employed Realities adaptively according to the current level of churn to better utilize their resources.

To allow each peer to individually measure the level of churn from a local perspective and adjust the number of employed Realities dynamically according to the perceived level of churn, we propose a pattern that defines the ID offset for a given Reality without the need to readjust offsets. Yet it provides a good distribution of IDs over the ID space. When assuming a unified ID space $S_M \subseteq \mathbb{R} \mid \{0 \leq S_M < 1\}$ for the monitoring layer, the first Reality maintains the original ID as defined by the p2p overlay. However, the IDs in the first Reality are shifted by 0.5 to the right, wrapping around if necessary, which allows a maximum distance between the first and the second Reality. The next two Realities fill in the *gaps* between the first two Realities by shifting the ID by 0.25 and 0.75, respectively, to the right based on the original ID. The following four Realities again fill in the gaps between the last four Realities. This specific distribution resembles the structure of a binary tree that is being traversed breadth first and allows for unbounded and varying numbers of Realities.

2) Synchronization and Recruiting: Our solution needs to synchronize the peers of each hypernode to maintain a consistent view on the data. In order to achieve this, the Coordinator of each hypernode has the task to periodically send its aggregated data to the Parallel Peers of its hypernode. Another approach would be to synchronize the data whenever a change occurs. However, this potentially induces a huge traffic overhead, thus the periodic update is a trade-off between the data's freshness and the consumed bandwidth. The length of one push interval is a configurable parameter and impacts the service's performance significantly. While a short value means fresher data, due to the frequent updating of the consistent view, it also requires more messages during one frame of time. In addition to establishing a consistent view, the synchronizing process also enables the Coordinator to change the size of the hypernode by sending synchronize messages to more or to less Realities. This *recruiting* allows for dynamical changes of the hypernode size and a recruited peer can use the information from the target Virtual ID to determine its new responsibility in the hypernode.

3) Data Distribution: After the data is synchronized with the Parallel Peers, the hypernodes eventually exchange data between each other. This is accomplished by each peer of the hypernode sending the exact same messages via the inter-hypernode links to the peers higher or lower in the hierarchy, but in the same Reality (see Figure 1). Data duplicates are completely circumvented as each peer maintains one data base per Virtual ID and incoming data from identical Virtual IDs is overwritten by such, thus, each data set is never aggregated twice.

To minimize the chance of losing not yet replicated data due to a peer leaving the network, each peer inside a hypernode immediately broadcasts any received data. With regard to the consumed bandwidth, it would be more efficient to collect the information received during one round and to only exchange an aggregate at the end of the round or during the synchronization phase. But this requires a more complex synchronization protocol and holds the risk of losing one or several aggregates due to churn preliminary to the synchronization. Considering changing responsibilities, a peer Y that receives a message with the target key y, but is not yet aware of its responsibility for this position, broadcasts this information through the usage of Virtual IDs. Such an event indicates that the responsibility for this position has shifted recently and the peer will eventually become recruited through the next synchronization phase. As the peer received the message that was addressed to the specific Virtual ID, it will also receive the next synchronization message aimed for this position.

C. Approaches to the Hypernode Protocol

We propose three variants, the *Fixed Approach* with a focus on maximizing the reliability of the technique with reasonable usage of resources and the *Adaptive Approach* with a focus on minimizing the usage of bandwidth while maintaining a good precision of the monitoring technique through variation of employed Realities. Finally, the *Slotted Adaptive Approach* aims to reduce the bandwidth consumption even further by reducing the number of redundant transmissions.

1) Fixed Approach: This approach utilizes a predefined fixed number of Realities at any time, sacrificing bandwidth for a higher precision. Further, in this approach Parallel Peers only transmit their data to their respective parents if they have received child aggregates recently. Thus, leaves do not send any additional data. This is because leaf nodes, that depart from the network, cannot eclipse subtrees from the monitoring tree and therefore their data does not need to be replicated.

2) Adaptive Approach: The first technique does a good job of masking the failure of single nodes, but it utilizes a lot of resources during times when peers only occasionally leave the network. The Adaptive Mode takes this into account. It attempts to reduce the number of transmissions when the churn level is low while retaining the ability to increase the level of replication as the need arises. However, this requires a technique to measure the network's stability from a local perspective for each individual peer. We use a backoff algorithm, but will not go into detail, because this is out of scope for this paper and other techniques might be equally viable. In a nutshell, the algorithm exponentially punishes missing incoming data, which indicates ongoing churn, and linearly decrements on incoming expected data. In contrast to the Fixed Approach, the Adaptive Approach specifies a number of Realities (*Min Realities*) that must be used at any time to sustain a certain level of replication. But also offers a higher number (*Max Realities*) and allows each Coordinator to synchronize between *Min Realities* and *Max Realities* Parallel Peers for its hypernode, depending on the locally perceived level of churn. The Coordinator may adjust the number of Realities used dynamically during each synchronization phase. The parameter NetStability sets the tradeoff between fast recruiting new nodes to increase the robustness and relieve nodes to save bandwidth and free their resources.

3) Slotted Adaptive Approach: This approach extends the Adaptive approach to reduce the bandwidth consumption even further during phases with low churn. Instead of all peers of a hypernode sending the messages simultaneously after synchronization, only the Coordinator transmits its data immediately while the Parallel Peers schedule a timed transmission event. The length of a timeslot is derived through $t_{timeslot} = \frac{pushInterval}{Realities}$. Note, that every Parallel Peer is aware of the reality it represents and the number of Realities currently available in the hypernode through the information received during the synchronization phase. Each Parallel Peer delays its transmission by the length of a timeslot, multiplied by the peer's Reality. However, when a peer in a lower Reality, e.g. the Coordinator, successfully transmits its data and receives an acknowledgment, it sends a broadcast with the received information to the remaining peers of the hypernode to replicate the response. Peers, that receive such a broadcast, should cancel the scheduled event, because there is no achievement in re-sending a message that has been successfully transmitted already. In addition, whenever a new Coordinator takes over responsibility for a position, Parallel Peers, that are already recruited for the hypernode's position, provide the current data replicated in the hypernode to the new Coordinator as a response to its first synchronization message.

IV. Evaluation

For the evaluation we integrated the protocol stated in Section III into the push and tree-based global aggregation monitoring system SkyEye.KOM [8], which satisfies the requirements stated in Section I. Both, the vanilla implementation and our enhanced version using the Mr.Tree protocol, were implemented in the large-scale peer-to-peer simulator PeerfactSim.KOM [6].

For stating the performance of our techniques, we analyze the outcome according to the methodology stated in Section IV-A. In the scope of this paper, we set our main focus on the following metrics: i) the precision and ii) the costs.

It is important to note that the precision is at the maximum for any type of attribute being monitored, if the monitoring approach was able to gather data from all peers. Thus, the number of lost data sets must tend towards zero. By defining

the precision independently of the attribute being monitored, it is equivalent to considering the totality of the gathered data. Hence, we compare the precision of the vanilla and our implemented techniques by comparing the completeness of the global view gathered by the monitoring.

The costs measure the overhead generated by our techniques. For this, every member of a hypernode (Coordinator and Parallel Peer) reports the throughput for internal and external traffic to measure the technique's caused traffic overhead.

A. Setup and Parameters

All simulations were performed with the PeerfactSim.KOM [1] simulation framework using the same basic setup, which is shown in Table I. The number of hosts was set to 5000, as earlier simulations had shown that larger numbers of hosts significantly prolong the simulation process while marginally influencing the results. To prevent runaway values from distorting the results, we performed 10 simulations with different seeds for each scenario.

The simulation starts with a 60 minute joining phase in which all peers join the overlay. Next there is a stabilization phase with a duration of 30 minutes to give the overlay time to stabilize itself. Following, at $t = 90m$, the data distribution process of the monitoring service starts while the network is in a stable condition. The churn generator starts at $t = 120m$ and remains active until the simulation finishes at $t = 180m$.

The last hour of simulation is the most interesting for the evaluation of precision and cost under churn. During the monitoring phase, the vanilla and our enhanced version's implementation of SkyEye.KOM aggregate the monitoring data in a bottom-up manner, create the global view at the root node, which equals the normal aggregation process, and back-propagates the global view top-bottom by piggybacking it with acknowledgment messages. Among other metrics, the number of aggregated data sets is reported to the analyzing mechanism by the root node along with the desired value calculated by the simulator. The absolute error between the measured and the desired values is used to calculate the number of missing data sets in the global view.

The averaged error per simulation scenario throughout all seeds is our first precision metric which is mentioned in Table II. Here, we raise the absolute (abs.) and relative (rel.) error together with their averaged standard deviation (SD). While the abs. and rel. errors of the precision metric are an indicator of the average performance, the averaged SD keeps track of its deviation. The costs of a scenario are defined as the averaged outgoing traffic per peer. Additionally, we investigated the overhead share, which is defined as the traffic this approach causes in addition to the normal monitoring traffic in relation to the overall traffic. Thus, the vanilla approach's overhead share is zero. In order to determine whether an approach is cost-effective, we introduce the error/cost (e/c) and SD/cost (sd/c) coefficients. These are defined by the product of the precision times cost and standard deviation times cost, respectively, whereas all multipliers are understood to be relative to the reference scenario. If the coefficient is above 1.0, the scenario is less cost-effective than the vanilla approach and, otherwise, a coefficient below 1.0 implies that the precision enhancement is cost-effective.

TABLE I
SIMULATION SETUP AND PARAMETER

Fixed General Simulation Settings	
Details	5000 simulated hosts, 10 seeds per scenario
push interval / bF	30s / 4
P2P Overlay	Pastry (KBR-compliance [4])
NetStability	0.1
Parameter Variations	
Churn Model	KAD, Exponential (30m, 20m, 10m, 5m)
# Realities	Fixed: 3, 5; Adaptive: 3-5

Other important parameters are the push interval, which specifies the periodicity peers send their aggregate up the tree and the branching factor (bF) of the monitoring tree. These parameters are chosen by the authors of SkyEye.KOM.

In the following, we compare four different techniques with the vanilla implementation. These techniques are the *fixed* mode with three and five realities, denoted as $F3$ and $F5$, as well as both adaptive modes *adaptive* and *slotted adaptive*, denoted as *A3-5* and *A3-5*$_{slot}$, respectively, ranging between three and five realities. The choice of the realities is motivated by the work from Graffi et al. [7], who showed that it needs 3 and 5 nodes to reach a probability of over 95% and 100%, respectively, and that at least one of these nodes stays online under KAD churn. Hence, we have chosen the bounds of the modes accordingly. Preliminary simulations have shown that less realities are underperforming while more realities dramatically increase the costs. Also, setting the NetStability parameter to a value of 0.1 provides a good tradeoff.

These five scenarios, including vanilla, are simulated in five churn scenarios. The first scenario is the KAD churn model proposed by Moritz Steiner et al. [16]. It models the behavior of users in the file sharing application KAD. The other churn scenarios use the Exponential churn model of the PeerfactSim.KOM simulator. This is a synthetic model which divides all participants into several groups with the purpose of representing user behavior in environments other than file sharing applications. The required mean time of this model specifies the mean time of the transient group of nodes, whereas the mean time is multiplied by a factor of ten for the normal and long lasting group of nodes. We chose 30, 20, 10 and 5 minutes mean time for our evaluation. Especially the 5 minute mean time scenario is very restrictive.

In the following, we discuss the outcome of the different techniques under different churn scenarios and compare it to the vanilla implementation, which is meant to be the reference. The absolute numbers of the simulator's outcome can be used to compare different approaches with each other. Because of the abstraction of the simulator, the outcome may be multiplied by a certain factor for real world scenario comparison.

B. Outcome

The outcome is presented in Table II, which is divided into six columns for the precision, three columns for the costs and two columns for the e/c and sd/c coefficients. The evaluation

TABLE II
SIMULATION OUTCOME CATEGORIZED BY CHURN MODEL. SCENARIOS ORDERED BY ABSOLUTE AVERAGED ERROR.

Scenario	Averaged error of count						Averaged costs per peer			Coefficient (cf. vanilla)	
	abs. ▲	abs. SD	rel.	rel. (cf. vanilla)	rel. SD	rel. SD (cf. vanilla)	abs. [byte/s]	overhead share	rel. (cf. vanilla)	error	SD
KAD churn model											
F5	55.7	21.9	0.014	0.155	0.006	0.056	1028.7	0.586	25.294	3.921	1.416
A3-5	58.6	45.3	0.014	0.164	0.012	0.116	483.7	0.587	11.893	1.950	1.380
F3	61.6	38.3	0.015	0.173	0.010	0.096	410.6	0.586	10.096	1.747	0.969
A3-5_{slot}	77.0	89.9	0.019	0.216	0.022	0.223	246.6	0.564	6.063	1.310	1.352
vanilla	334.3	378.7	0.088	1.000	0.100	1.000	40.7	0.000	1.000	1.000	1.000
Exponential churn model, 30m mean time											
F5	56.3	80.3	0.012	0.178	0.016	0.222	1065.2	0.556	13.703	2.439	3.042
A3-5	60.5	101.4	0.012	0.192	0.021	0.282	526.0	0.536	6.766	1.299	1.908
F3	69.2	114.4	0.014	0.219	0.023	0.316	451.1	0.528	5.803	1.271	1.834
A3-5_{slot}	117.3	167.3	0.024	0.371	0.034	0.464	293.2	0.480	3.772	1.399	1.750
vanilla	314.5	359.7	0.065	1.000	0.074	1.000	77.7	0.000	1.000	1.000	1.000
Exponential churn model, 20m mean time											
F5	65.0	82.0	0.013	0.158	0.017	0.196	1097.1	0.538	10.543	1.666	2.066
A3-5	80.0	131.9	0.017	0.195	0.027	0.317	577.3	0.508	5.548	1.082	1.759
F3	110.6	206.5	0.023	0.269	0.043	0.495	480.8	0.496	4.620	1.243	2.287
A3-5_{slot}	188.1	281.6	0.039	0.457	0.058	0.674	331.4	0.440	3.185	1.456	2.147
vanilla	408.7	415.3	0.085	1.000	0.086	1.000	104.1	0.000	1.000	1.000	1.000
Exponential churn model, 10m mean time											
F5	101.8	131.1	0.021	0.166	0.028	0.233	1188.7	0.493	6.458	1.072	1.505
A3-5	105.4	150.1	0.022	0.173	0.032	0.268	709.5	0.445	3.855	0.667	1.033
F3	110.3	174.2	0.023	0.181	0.037	0.311	567.7	0.421	3.085	0.558	0.959
A3-5_{slot}	211.6	236.9	0.045	0.347	0.050	0.423	440.5	0.360	2.393	0.830	1.012
vanilla	604.4	557.2	0.128	1.000	0.118	1.000	184.1	0.000	1.000	1.000	1.000
Exponential churn model, 5m mean time											
F5	157.0	210.3	0.034	0.148	0.045	0.291	1364.5	0.423	3.978	0.589	1.158
A3-5	168.2	267.1	0.036	0.159	0.057	0.371	927.4	0.364	2.704	0.430	1.003
F3	203.8	314.5	0.044	0.194	0.068	0.438	744.3	0.321	2.170	0.421	0.950
A3-5_{slot}	332.4	398.2	0.072	0.316	0.086	0.555	649.6	0.272	1.894	0.599	1.051
vanilla	1030.5	704.2	0.227	1.000	0.155	1.000	343.0	0.000	1.000	1.000	1.000

data consist of the six churn models ordered by their churn strength and node fluctuation, respectively, each covering the five approaches. Per churn model the scenarios are sorted by the abs. averaged error of count (2nd column) throughout all seeds of the simulation. We use a gray scale cell background to indicate the ranking per churn model, with light gray being the best value and dark gray the worst.

1) KAD churn model: Approaches applying Mr.Tree face 4.6x - 6.5x less rel. errors and 4.5x - 17.9x smaller SD than Vanilla. The ranking with its rel. error values is as follows: F5 (1.4%) $\geq$ A3-5 (1.4%) $>$ F3 (1.5%) $>$ A3-5_{slot} (1.9%) $>$ Vanilla (8.8%).

While the error decreases, the costs increase by factors between 6 - 25.3. Nevertheless, the traffic in absolute numbers for the scenario with the highest redundancy (F5) still stays low. Here, we encounter approximately $1\frac{kB}{s}$ outgoing traffic for the maintenance of the overlay and monitoring layers. The latter is also the reason for the high overhead share (56 - 59%). For the overlay layer, Pastry induces just small amounts of maintenance traffic. Due to the fact that KAD churn does not stress the overlay too much, Vanilla's costs are quite low, hence, none of the scenarios are e/c cost-efficient. Nevertheless, depending on the use-case, application operators may prioritize precision over costs, so that it is advisable to choose scenarios with less redundancy as their efficiency is acceptable. A3-5_{slot} shows the best e/c coefficient with 1.31, right behind the reference. The best sd/c coefficient is achieved by F3 being the only sd/c cost-efficient scenario with 0.969.

2) Exponential churn model (E30m): The Exponential churn model with 30m mean time performs differently than KAD having smaller (inter) session times (no figure). Still, the abs. error is on the level of KAD, while the rel. error goes down slightly. This is due to the fact that in total more nodes are participating at the same time than in KAD, which has a positive impact on Vanilla's performance. The rel. error of our solution compared to it degrades, resulting in 2.7x-5.6x less errors. A3-5_{slot} suffers the most loss from 4.6x in KAD to now 2.7x less errors. The ranking of the rel. error and costs equal the ranking observed under KAD churn. The SD degrades for our scenarios as the lowered inter session and session times lead to more node fluctuations. For the costs, approaches applying Mr.Tree face a slight increase but Vanilla's traffic is almost doubled, which leads to almost halved rel. costs. An explanation is that Pastry needs to fix its routing tables more often in scenarios with a higher node fluctuation. Scenarios applying Mr.Tree already start lookups more often than Vanilla does, thus, tables are actively fixed. The influence on the error and costs leads to e/c coefficients near 1.0, but the worsening of the SD leads to actual higher sd/c coefficients. The scenarios F3 and A3-5 show the best efficiency.

3) Exponential churn model (E20m, E10m, E5m): Following, we discuss the Exponential churn models with 20m, 10m

and 5m mean time. The rel. error and costs ranking is just like the rankings for 30m, but the costs gap between the Mr.Tree scenarios and Vanilla degrades. Interestingly, the abs. error of A3-5_{slot} (332.4) under strong churn (5m) is in the magnitude of the abs. error of Vanilla under the KAD churn model (334.2) utilizing only 649.6 $\frac{byte}{s}$ in average per peer. Similarly, the rel. error of F3 under strong E5m churn is half of the Vanilla version under the much weaker KAD churn model. Whereby, the rel. error of Vanilla under increasing churn is rising from 8.5% (E20m) and 12.8% (E10m) to 22.7% (E5m).

Starting from KAD and ending at E5m, the abs. traffic increases by approximately 300-400 $\frac{byte}{s}$ for all scenarios, thus the rel. comparison to Vanilla increasingly shrinks from 6 - 25.3 to 1.9 - 4 fold. The overhead share also shrinks from 56.4% - 58.6% to 27.2% - 42.3%. This is mainly due to increased overlay fixing costs.

We observe that the e/c efficiency is improving with strengthen churn. The *breakthrough point* is defined as the point where a scenario falls below the coefficient value of 1.0, thus the precision gain surpasses the cost overhead. For the e/c coefficient, all scenarios except F5 surpass this point for churn models equal to E10m. F5 surpasses this point for churn models like E5m. The best observed e/c coefficient is reached by F3 and A3-5 for strong churn like E5m. For the sd/c coefficient, all scenarios get closer to the breakthrough point with increasing churn level, thus, the reduced deviation counteracts the increased costs, which is a superior performance indicator. Only F3 falls below this point for KAD, E10m and E5m. A3-5 and A3-5_{slot} closely reach this point for E10m (1.033 and 1.012) and E5m (1.003 and 1.051).

4) Summary: A3-5_{slot} can be recommended as a cost saving approach in all churn scenarios as the e/c coefficient is ranging between 1.456 and 0.599, surpassing the breakthrough point three times for higher churn scenarios. F3 provides better accuracy but increased costs. A3-5 is declared to be the best tradeoff between good accuracy and costs, whereas F5 should be chosen when facing high churn.

V. Conclusion and Outlook

Mr.Tree is proposed in this paper, a protocol to increase the robustness of existing tree-based monitoring overlays in highly dynamic p2p networks, which, due to their structure, suffer great loss of precision when facing churn. Three modes in total are presented to fit individual scenarios. Evaluations show a significant increase of the proposed modes' precision under heavy churn compared to the original implementation. The use of the adaptive mode is advised in scenarios where the churn rate might fluctuate, for instance, caused by day-scale or even world-scale patterns. Furthermore, this mode accomplishes best error/cost tradeoffs. The slotted adaptive mode is declared to be the best cost saving approach which still achieves significantly more precise information than the original protocol.

For future work, the traffic induced by our replication structure should be lowered by investigating more complex synchronization protocols. Additionally, our monitoring approach does not consider *malicious* nodes which might try to influence the aggregated monitoring data (forge attacks). It is of interest how the large data base which hypernodes maintain can be utilized to detect and to prevent malicious behavior.

Acknowledgment

Computational support and infrastructure was provided by the Centre for Information and Media Technology (ZIM) at the University of Düsseldorf (Germany).

References

[1] PeerfactSim.KOM Simulator. [Online]. Available: http://peerfact.com
[2] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-bandwidth Multicast in Cooperative Environments," in *Proc. of the 90th ACM Symposium on Operating Systems Principles*, ser. SOSP '03. New York, NY, USA: ACM, 2003, pp. 298–313.
[3] J. Considine, F. Li, G. Kollios, and J. Byers, "Approximate aggregation techniques for sensor databases," in *Proc. of the 20th International Conference on Data Engineering*, March 2004, pp. 449–460.
[4] F. Dabek, B. Y. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica, "Towards a Common API for Structured Peer-to-Peer Overlays," in *IPTPS '03: Proc. of the Int. Workshop on Peer-to-Peer Systems*, ser. LNCS, vol. 2735. Springer, 2003.
[5] M. Dam and R. Stadler, "A Generic Protocol for Network State Aggregation," in *Radiovetenskap och Kommunikation*, 2005, qC 20120125.
[6] K. Graffi, "PeerfactSim.KOM: A P2P System Simulator Experiences and Lessons Learned," in *Proc. of IEEE P2P '11*, 2011.
[7] K. Graffi, C. Groß, D. Stingl, H. Nguyen, A. Kovacevic, and R. Steinmetz, "Towards a P2P Cloud: Reliable Resource Reservations in Unreliable P2P Systems," in *IEEE ICPADS '10: Proc. of the International Conference on Parallel and Distributed Systems*, 2010.
[8] K. Graffi, A. Kovacevic, S. Xiao, and R. Steinmetz, "SkyEye.KOM: An Information Management Over-Overlay for Getting the Oracle View on Structured P2P Systems," in *IEEE ICPADS '08: Proc. of the Int. Conf. on Parallel and Distributed Systems*. IEEE, 2008.
[9] K. Graffi, D. Stingl, J. Rückert, and A. Kovacevic, "Monitoring and Management of Structured Peer-to-Peer Systems," in *IEEE P2P '09: Proc. of the Int. Conference on Peer-to-Peer Computing*, 2009.
[10] M. Jelasity, A. Montresor, and Ö. Babaoglu, "T-Man: Gossip-based Fast Overlay Topology Construction," *Computer Networks*, vol. 53, no. 13, pp. 2321–2339, 2009.
[11] P. Jesus, C. Baquero, and P. S. Almeida, "A Survey of Distributed Data Aggregation Algorithms," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 381–404, Firstquarter 2015.
[12] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-Based Computation of Aggregate Information," in *IEEE FOCS '03: Proc. of the Symposium on Foundations of Computer Science*, 2003.
[13] J. Li, K. Sollins, and D.-Y. Lim, "Implementing Aggregation and Broadcast over Distributed Hash Tables," *SIGCOMM Computer communication Review*, vol. 35, no. 1, pp. 81–92, Jan. 2005.
[14] R. Makhloufi, G. Doyen, G. Bonnet, and D. Gaïti, "A survey and performance evaluation of decentralized aggregation schemes for autonomic management," *International Journal of Network Management*, vol. 24, no. 6, pp. 469–498, 2014.
[15] A. G. Prieto and R. Stadler, "A-GAP: An Adaptive Protocol for Continuous Network Monitoring with Accuracy Objectives," *IEEE Transactions on Network and Service Management*, vol. 4, no. 1, pp. 2–12, June 2007.
[16] M. Steiner, T. En-Najjary, and E. W. Biersack, "Analyzing peer behavior in KAD," *Institut Eurecom, France, Tech*, 2007.
[17] F. Wang, Y. Xiong, and J. Liu, "mTreebone: A Collaborative Tree-Mesh Overlay Network for Multicast Video Streaming," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 3, pp. 379–392, 2010.
[18] F. Wuhib, M. Dam, and R. Stadler, "Decentralized detection of global threshold crossings using aggregation trees," *Computer Networks*, vol. 52, no. 9, pp. 1745 – 1761, 2008.
[19] P. Yalagandula and M. Dahlin, "A Scalable Distributed Information Management System," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 379–390, Aug. 2004.