Continuous Gossip-based Aggregation Through Dynamic Information Aging

Vitaliy Rapp, Kalman Graffi
Technology of Social Networks Group, University of Düsseldorf, Germany
Email: graffi@cs.uni-duesseldorf.de

***Abstract*—Existing solutions for gossip-based aggregation in peer-to-peer networks use epochs to calculate a global estimation from an initial static set of local values. Once the estimation converges system-wide, a new epoch is started with fresh initial values. Long epochs result in precise estimations based on old measurements and short epochs result in imprecise aggregated estimations. In contrast to this approach, we present in this paper a continuous, epoch-less approach which considers fresh local values in every round of the gossip-based aggregation. By using an approach for dynamic information aging, inaccurate values and values from left peers fade from the aggregation memory. Evaluation shows that the presented approach for continuous information aggregation in peer-to-peer systems monitors the system performance precisely, adapts to changes and is lightweight to operate.**

I. Introduction

Overlay networks allow the operation of content-centric and communication-centric applications in the Internet. These networks may index content, nodes and users and make them searchable in order to decentrally connect providers and consumers. One important element of a professionally run overlay is the estimation of the performance of the running overlay in terms of quality of service. The system performance of the overlay is described in statistics on the number of users, available documents, lookup and search delays as well as statistics on the traffic and bandwidth consumption, among others. This information can be used to determine the current system status so that locally suitable behavioral strategies in the nodes can be derived in order to to be able to take actions in case of detected misleading quality trends. In perspective, the monitoring of system statistics may lead to a full control loop for large-scale self-organizing networks, as in [1] and [2], allowing the distributed system to adapt automatically to a desired performance goal.

While central monitoring approaches like the SNMP [3] or Intel's Active Management Technology (AMT) [4] support the monitoring of small- to mid-scale networks, e.g., the quality of service in the network and transport layer [5], there are several reasons why they are not suitable for self-organizing large-scale peer-to-peer networks that aim at full decentrality. Decentralized monitoring approaches (see Section II) are required, these are mainly either tree-based or gossip-based. The tree-based approaches require a structured overlay with key-based routing [6] to create a tree topology on top of the distributed hash table (DHT). The tree is then used for efficient gathering and aggregation of statistics from the nodes in all sub-trees and thus in the overall tree. While this approach is quick and efficient, its strict structure is cumbersome to maintain during strong dynamics in node participation, such as churn. During churn, several outliers occur.

Gossip-based approaches on the other hand work in any connected topology, as the existing overlay topology is used: nodes periodically exchange their estimation on the network statistic with randomly chosen neighbors. The views are merged (aggregated) within $O(logN)$ rounds, with N being the number of nodes and a a precise estimation of the global statistics is converging in all nodes. One main drawback of today's gossip-based approaches is their operation in epochs. Only at the begin of an epoch do the nodes consider their current local performance measurements. During the epoch, these initial measurements are aggregated over all nodes in the network and no fresh measurements are considered meanwhile. The epoch ends after a predefined number of inter-node exchange rounds. The more rounds that are used, meaning the longer the epoch, the more precise the aggregation of the initial (and thus old) measurements becomes. Thus, a dilemma occurs when choosing the right epoch length: Either, in the case of long epochs, the aggregation is precise but relies on old measurements or, in the case of short epochs, the measurements are fresh but the aggregation imprecise and less likely to converge.

In this paper, we present a gossip-based aggregation protocol which considers the local measurements of the nodes in every round. Thus, no initial measurement is needed as the aggregation and inter-node exchange rounds are continuous, i.e., endless. The challenge lies in considering fresh measurements as well as in the current network-wide estimation on the network statistics in the values to be exchanged between nodes. The main idea is to apply dynamic information aging on the network statistics and to introduce in exchange the current measurements in small ratios. Thus, over time the system-wide estimations on the network statistics converge to common values, which are biased to each individual node's local measurement with a configurable ratio.

In the following, the paper is structured as follows. In Section II, we review the state of the art in decentralized monitoring approaches and show the need for a continuous gossip-based approach. In Section III, we present our approach for continuous gossip-based aggregation and describe in detail

This work was partially supported by the German Research Foundation (DFG) within the Collaborative Research Centre 'On-The-Fly Computing' (SFB 901).

the concept of dynamic information aging. The evaluation presented in Section IV shows the performance and costs of the approach in a network with 5000 nodes. The aggregated estimations are precise and come at low costs. In Section V, we conclude that the presented approach for continuous gossip-based aggregation is suitable to overcome the limitations of today's gossip-based approaches working with epochs, while it preserves its good properties: quick converging, low costs and general applicability in any connected graph.

II. Related Work

In this section, we review the state-of-the-art monitoring approaches for large-scale networks, mainly peer-to-peer systems. We start with currently applied approaches and more over to more sophisticated monitoring approaches that we think should be applied in future peer-to-peer systems.

A. Crawler-based Approaches

Popular peer-to-peer-based file-sharing applications with millions of users are eDonkey [7] and BitTorrent [8]. They both apply specific concepts of monitoring. In BitTorrent a tracker manages a swarm of peers interested in a file. By having each peer signaling his status to the tracker, the creation of statistics on the swarm is simple to do at the tracker, as in [9] for example. Information on other swarms are, however, not gathered or aggregated. In eDonkey, a small number of nodes act as superpeers and manages the statuses of the other normal peers. The node count can thus be exactly aggregated by all superpeers. Both BitTorrent and eDonkey support the usage of the Kademlia implementation KAD [10], which does not come with integrated statistic aggregation protocols.

In order to measure running overlays, like KAD, crawler-based approaches have been introduced. Crawlers assume that nodes in the network may be measured externally in terms of the properties or contacts. Thus, scanning and crawling the network by a few nodes allows to gather information on the nodes in the network as well as on further potential nodes to crawl. Crawlers grasp the status of a large set of nodes, but, the actuality of the obtained estimation vanishes quickly. While crawler-based approaches are useful to measure running networks, they are limited to gather only information exposed by other nodes. These are often only an estimation on the load or online time, but not on the actual performance or quality of service experienced by the nodes. For future overlays and peer-to-peer applications, such as peer-to-peer-based online social networks [11], [12] it is more advisable to integrate aggregation protocols in order to gain a far broader view of the quality of a running peer-to-peer system.

B. Aggregation of Network Statistics

Further and richer information on the system statistics can be obtained through aggregation protocols [13]. The assume that nodes actively measure local observations, exchange this information with specific nodes in the network, aggregate the obtained information and create a global view on the network with time. Aggregation protocols use aggregation functions [14], like sum, maximum, minimum or average, to generate statistics on the network. Three main classes of aggregation protocols are sampling, gossiping and tree-based approaches.

1) Sampling: Sampling is closest to crawling. In sampling-based approaches, nodes interested in the system statistics initiate a measurement on a selected small set of nodes. The result of these nodes is then interpolated to all nodes in the network. One approach for this is initiating a random walk of defined length, as in [15] for example. In [16], a consistent view on the network for all nodes is aimed at. Random walks, however, are biased towards nodes with high node degree. Approaches for a uniform node selection has been presented in [17]. While the estimation of average values might be precise with a large sample size, some aggregation functions like minimum, maximum and sum cannot be implemented through sampling approaches. Thus, aggregation protocols covering all nodes are required to fully estimate average, minimum, maximum and sum functions, as shown in [18].

2) Tree-based Approaches: Tree-based approaches establish besides the existing overlay topology further contacts to selected nodes in the network in order to create an optimized aggregation topology. The aggregation topology is typically a tree, as in GAP [19], CONE [20], SDIMS [21], SkyEye.KOM [22], [23] or SOMO [24]. These approaches assume a structured peer-to-peer overlay with DHT functionality. GAP [19] and its extension M-GAP [25] as main examples establish a tree topology and aggregate stepwise the measurements towards the root. In the root of the tree, the global system statistics are complete and from then distributed down the tree to all leaves.

In the tree-based approach, the aggregation time is $O(logN)$ for a single measurement with N being the number of nodes in the network, which is quite efficient. The statistic gather times can even be accelerated [26]. Each node has a state and a load of $O(k)$ with k being the tree degree. Fresh measurements at the nodes are considered through periodic submissions of the local and obtained estimations to the parent node in the tree. One main limitation of tree-based approaches is the need for structured overlays with ID-based routing. This is required to easily create and communicate the tree topology. Applications of monitoring trees are in [27] and [28]

3) Gossip-based Approaches: Gossip-based aggregation uses only the existing network topology and limits the nodes to interact only with existing neighbors, thus being widely applicable and supporting a broad application range [29]. Through the local communication, the nodes exchange their view on the network statistics and aggregate them, thus the estimations converge to the global statistics. In [30] the fundamental basis of the anti-entropy calculation is described. The functionality of the algorithm relies on pair-wise exchanging of calculated values and eliminating the entropy between them by calculating the common average or minimum/maximum. Examples for proactive gossip-based approaches are [31] and [32]. T-MAN [33] is a proactive overlay topology management protocol, which through gossiping can transform an unstructured overlay in a desired structure.

Gossip-based approaches work with epochs: the actual local measurements are only considered at the begin of an epoch, during the epoch aggregations of these initial values are created and the epoch ends after a fixed set of rounds, when the local estimations have converged in $O(logN)$ rounds, with N being the number of nodes. A long epoch leads to a more precisely aggregated global view on old measurements, while a short epoch leads to an imprecisely aggregated global view on fresh measurements. Our approach does not use epochs, it runs continuously and considers partially in every round a fresh measurement while creating the aggregated global view, thus keeping the strengths of the gossip-based approach while decreasing its limitations.

III. Continuous Gossip-based Aggregation

In this section we present our approach for continuous gossiping which aims at precisely aggregating the minimum, maximum, average and sum of a set of initial values in a large-scale (peer-to-peer) network. We assume that all peers in the network participate in the aggregation protocol and do not misbehave. All nodes act according to the same configuration and use same waiting, timeout and update times. Further, we assume that the network is connected.

Our approach follows a proactive aggregation scheme based on the protocol described in [30]. Each node i has a current local measure (v_i) on a relevant system performance detail. Please note that in the following we refer to a single value for clarity, while the protocol can be also applied to a set of values. The functions then need to be applied to each value in the set. Using our gossiping approach, each node i has a calculated metric value (x_i), which represents a current estimation of aggregated value in the system. Next, we present the protocol description, aggregation functions and the dynamic aging approach, which enables to consider fresh measurements during the aggregation of system-wide statistics.

A. Protocol Description

In order to obtain proper estimations for x_i, the node participates in our gossiping approach and applies an active process, which is periodically called, and a passive process, which reacts on incoming messages: In the first step, the node applies a function to consider its freshest measurement v_i in the estimated aggregation x_i. This function makes sure that the gossip-based aggregation can continue without interruption, as fresh measurements are periodically included. For simplicity, we first assume the simple case that an initial setting of values in the network is to be aggregated, which does not change over time. The more realistic and challenging case with changing values is discussed in Subsection III-C. There, the special purpose and the full specification of the function to consider fresh measurements is discussed in detail. In the simple case, when the node firstly participates in the aggregation protocol, its x_i is not yet set and thus initialized to v_i in this step.

In the second step, a node then selects a random neighbor as the exchange partner in the current cycle. Next, the node sends its calculated metric value list to the selected neighbor, e.g.,

```
1: while true do
2:    wait(getWaitingTime())
3:    x_i = considerMeasurement(v_i, x_i)
4:    n_j = selectRandomNeighbour()
5:    send x_i to n_j
6:    receive x_j = from n_j
7:    x_i = aggregate(x_i, x_j)
8: end while
```

Fig. 1. Active Process on Node n_i

```
1: receiveApproximation(x_i, n_i)
2: send x_j to n_i
3: x_j = aggregate(x_j, x_i)
```

Fig. 2. Passive Process on Node n_j

n_j. The message, containing the metrics x_i, is denoted as a metric-exchange message. After sending, the node waits for an acknowledge (ACK) message, which also contains the metric values x_j from selected neighbor n_j. When the metrics from the neighbor are received, the node calculates a new estimation for x_i using an aggregation function specific for all observed statistics (e.g., minimum, maximum, average, sum). The ACK messages finishes the exchange transaction. For consistency reasons an exchange transaction must be first completed to take actions on the aggregates. To avoid this behavior, we use a lock approach. When a peer sends an exchange message to its neighbor it sets a lock, which is released when the receiver replies (negative or positive) or a timeout occurs.

Besides the active process each node also executes a passive process. Within this process a node waits for a metric-exchange message. When a node n_j receives the message from a node n_i, it first sends own estimated values x_j to n_i in the form of an acknowledge message. Afterwards, it aggregates the received values x_i with its own estimated values x_j and updates its x_j. Apparently since both nodes use the same aggregation function, after a metric-exchange they will hold same estimation values. The set of aggregation functions applied in the algorithm is discussed next.

B. Aggregation Algorithms

All relevant aggregation functions may be expressed in a form of the combination of the following basic aggregation functions: minimum/maximum, average and count. The also relevant functions sum and standard deviation can be derived from this basic set of aggregation functions. All functions assume that the current node i's estimation of the aggregated value is x_i, while receiving an estimation x_j from node n_j. In the protocol, the nodes exchange their aggregated estimations and use following aggregation functions:

- $f_{minimum}(x_i, x_j) = min(x_i, x_j)$
- $f_{maximum}(x_i, x_j) = max(x_i, x_j)$
- $f_{average}(x_i, x_j) = \frac{1}{2}(x_i + x_j)$

The count and sum functions are discussed in Subsection III-D. First, we discuss the application of dynamic aging.

C. Dynamic Aging - Consideration of Fresh Local Values

While operating a gossip-based aggregation approach, fresh local values must be injected in the aggregate calculation. Related work in literature applies for this epochs, which

start with an initial fresh local value which is used until the end of the epoch. The end of the epoch is characterized by a convergence of the aggregate estimations system-wide. In contrast to that, our approach considers fresh local values in every aggregation round (see Fig. 1).

The key idea of the approach is to dynamically adjust the calculated values of peers, so that the calculated aggregation estimation, which each peer holds, converges to the metric values measured by the peer. If the aging rate is the same at every peer, then the deviations of aggregated values, caused by such adjustments, are corrected by the further protocol execution and the associated value distribution. If a change of a metric occurs, then the distributed sum of values, held by all peers, will converge to a new actual aggregated value.

We define the aging function, which all peers use, as: $f_{considerMeasurement} : (v_i, x_i) \rightarrow (1-\alpha) \cdot x_i + \alpha \cdot v_i$, where α is the aging rate. The value can be chosen between 0 and 1, and is the same for all peers. The lower the value, the slower the system reacts on local metric changes and churn, but also the accuracy of possible aggregate estimations is higher. In our simulation we used 0.01 for the aging rate, which is a good trade-off between accuracy and reaction speed.

D. Calculation of the Count and Sum

The sum of a particular metric can be expressed as the aggregated average value multiplied by the number of peers in the system. While the calculation of the average value has been presented, the count function is special and needs further attention. The calculation of the node count is motivated by [30]. In a network with n nodes, if $n-1$ peers are holding the value 0 and only one peer holds the value 1, then the calculated average of that value is equal to $\frac{1}{n}$. By inverting this value, every peer is able to estimate the network size.

The key requirement of the algorithm is to assure that only one peer holds the value of 1. We also refer to such a peer as a master peer. Metric and node count data are packed into the same message during the exchange. Such exchange message contains two information parts: a metric aggregation part and a network size estimation part. The metric estimation part contains the estimated values of all metrics, while the size estimation part contains following values held by a single peer, except of the original value:

- v_i^{count}: initial value of the size value, i.e., only 0 and 1
- c_i^{count}: estimated count aggregate
- max^{count}: estimated maximum of the initial values
- Current version of the algorithm

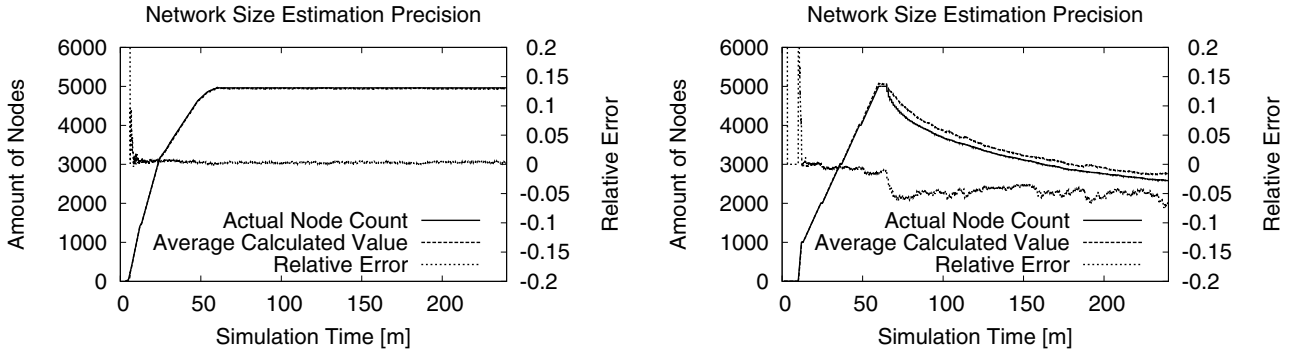
When a peer joins the network it initializes its own aggregation values (c_i^{count} and v_i^{count}) with 0. In this case the network size is estimated by 1. The maximum known value is set to 0 as well as the current version. Before a peer starts to exchange the size metrics it waits for a predefined period of time until it gets an incoming exchange request. An exchange request contains all the data related to the node counting from the other peer including the version of the algorithm. When the current version is received the peer sets its own version to the received one and continues with aggregation exchange transactions. In the case when a peer does not get an exchange request within the waiting time it initializes its own values with the value of 1 and the current version with a random real number, and afterward continues to behave normally. This approach assures that a peer either gets the current algorithm version or creates a new one and becomes the master peer. The estimated size and the maximum values are calculated and aged according to the described approach. The maximum value is used to indicate when a master peer goes offline. Following the dynamic aging approach, every peer in the system will converge its local estimation of the maximum known value to the own value, which is 0 except for the master node. Without a master peer, the estimated maximum known value converges to 0. When a maximum value falls under a certain threshold value a peer can decide to restart the algorithm and become the new master peer. For that it just selects a new random version number, which should be greater than the old one and initialize its own values with 1. Afterward it behaves as usual. To reduce the amount of conflicting versions, the chance that a peer decides to become a new master peer may be restricted by a small probability value (e.g., $\frac{1}{n}$), based on the last "good" network size estimation. By receiving an exchange message, first the algorithm version is checked. A greater version indicates that one of the peers did make a restart. Thus, the own version is replaced by the received one and the values are initialized with 0, before the incoming message is processed. If both versions are equal the message is processed as usual. Finally, if the received version is lesser than the current one, then the incoming message is discarded. Considering the aging rate, calculating the $count$ function presents one of the worst cases, since the system has the highest possible entropy at the start.

E. Discussion

In comparison to other gossip-based approaches, the aging approach brings an advantage - it does not require the restart of the aggregation (e.g., in epochs), so the final result of the aggregate estimation is not discarded. The protocol is also easy to implement and does not require any synchronization of peers. It can easily be adopted for different churn rates by varying the aging-rate parameter α. These benefits come with limitations: the accuracy of the provided estimation and the distributing speed of aggregates are limited by the aging rate. Since each peer ages the value in each round, the estimation error of $\alpha \cdot (v_i - x_i)$ is always present. On the other hand the reaction speed of the protocol on metric changes is described with the following formula: $((x_i) \cdot (1-\alpha) + \alpha \cdot (v_i))$. The aging rate also makes a trade-off between the accurancy of the estimation and update speed. The aging rate allows exchanging accuracy and update speed for bandwidth usage. Setting α to zero is also equivalent to turning the dynamic aging off.

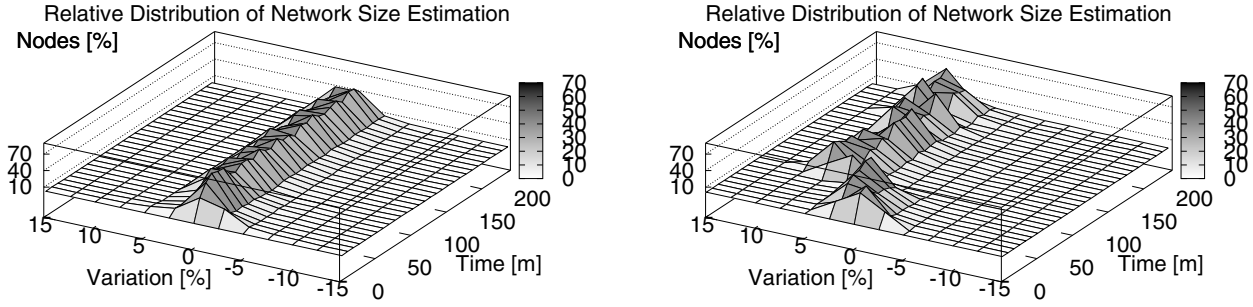
IV. Evaluation

We evaluate the presented continuous gossip-based aggregation protocol with the discrete and event-based peer-to-peer simulator *PeerfactSim.KOM* [34], [35]. The simulator is optimized to run several peer-to-peer layers like a monitoring



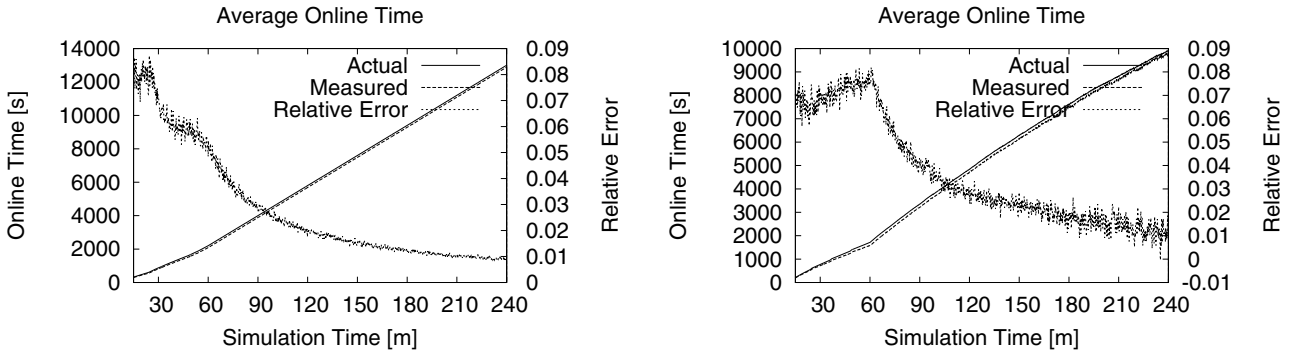
(a) No Churn (b) Churn

Fig. 3. Results on Node Counting



(a) No Churn (b) Churn

Fig. 4. Distribution of Monitored Node Counts



(a) No Churn (b) Churn

Fig. 5. Results on Online Time Monitoring

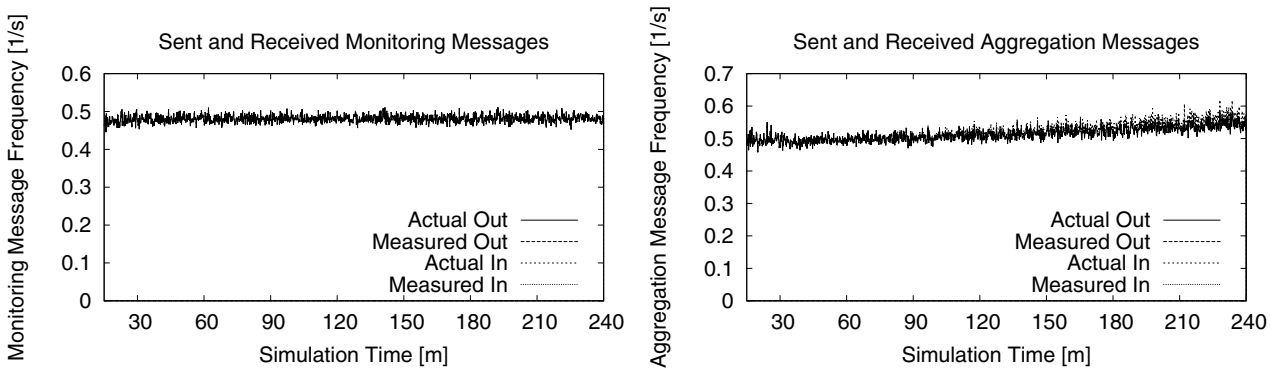
layer, overlay layer and a validated network layer. It can be used to evaluate or benchmark peer-to-peer algorithms [36].

Nodes were simulated to be located all over the world with Internet connection characteristics being distributed as determined by the OECD Bandwidth Report from 2007 [37]. The delay between the peers was determined by GNP [38]. As an overlay to operate our aggregation protocol in, we chose Chord, which was implemented according to [39]. Please note that we look at Chord as a random graph. Only the topology is used, but no lookups are initiated.

Our simulation setup is as follows. We simulate 5000 nodes, which establish a Chord network and are actively using our aggregation algorithm. The dynamic aging parameter α is set to 0.01. The length of a gossip round is set to $10\ seconds$, thus every node exchanges its statistic estimation every 10 seconds. These update actions are unsynchronized. We inspect the behavior of our aggregation protocol in two scenarios: with and without churn. In both cases, initially, all 5000 nodes join the Chord network within the first 60 minutes of simulation time. In the scenario with churn, we apply churn to the network from minute 65 on, using the KAD churn model presented in [10]. In this, the number of nodes drop to half within the following 180 minutes.

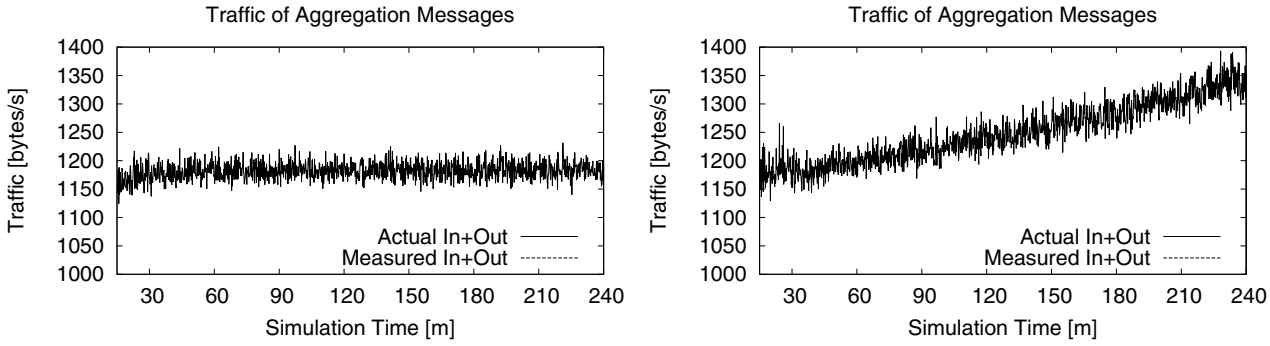
In the evaluation we inspect the aggregation precision by looking at the node count (see Fig. 3 and Fig. 4) and statistics on the online time of the nodes (see Fig. 5). The overhead cost of running the aggregation protocol is depicted in terms of messages (see Fig. 6) and traffic (see Fig.7).

Looking at the the aggregation of the node count in Fig. 3, we see that in the scenario without churn the actual node count



(a) No Churn (b) Churn

Fig. 6. Protocol Costs in Terms of Message Overhead



(a) No Churn (b) Churn

Fig. 7. Protocol Costs in Terms of Traffic Overhead

is almost precisely measured. The relative error stays slightly below 0.01. Through the dynamic aging of the aggregated values, the expected relative error is α, which is 0.01 in this setup. In the scenario with churn, the relative error varies around 0.05. This is due to the strong dynamics in the network, the large number of failing and joining nodes show effect in the measurements. Often the master node of the count aggregation leaves the network in this scenario and thus a new count process needs to initiated. This takes time and leads to a less precise monitoring. However, a relative error of 0.05 seems acceptable for many applications.

In Fig. 4, we depict how the relative measurement error distributed over the nodes evolves over time. In the scenario without churn the relative error stays close to 0.01 and does not vary over time. In the scenario with churn, we see a worsening of the relative error for all nodes as churn begins. At this point the distribution of the relative error over all nodes is shifted from around 0.01 to 0.05 and slightly beyond. The graphs show that the relative error spread over all nodes is 0.07 in case of no churn and 0.1 in the case of churn. Thus, all nodes have a similar, consistent node count estimation. Please note that the node count is the most challenging metric to measure with a gossip-based approach and it defines a worst-case metric. The aggregation of the online time, as well as the aggregation of overhead in terms of messages and traffic, is using an averaging function and is thus more precise. Average functions are quickly and precisely calculated.

Looking at the aggregation of the online time in Fig. 5, we see the precision of the measurement. In the begin the relative error is around 0.07 due to the strong dynamics induced by the join of all nodes. Then, it quickly drops in both scenarios to the expected relative error of $0.01(=\alpha)$. The effect of failing nodes due to KAD churn is very low. The same counts when observing the aggregation precision in the measurements of the message overhead (see Fig. 6) and traffic overhead (see Fig. 7). Here the actual values are also precisely estimated by the aggregation protocol. The relative error converges to 0.01.

Regarding the costs for operating the protocol, Fig. 6 shows the message frequency for the protocol. On average a message is sent and received every two seconds. The traffic overhead is 1150 to 1350 bytes per second (see Fig. 7). Thus the load on the peer is very small. By adapting the configuration of the round frequency, i.e., to initiate an inter-node exchange more frequently than every ten seconds, one could improve the precision and convergence time while increasing the node load to a level which is still acceptable. The current load on the nodes is small and allows more frequent gossip rounds.

In summary, the presented approach precisely monitors the network status. It converges quickly to a relative error of α, the factor for dynamic aging and for considering fresh local measurements at the nodes. The costs for the presented continuous gossip-based aggregation approach is very small, with a message frequency of around 0.5 per second and a bandwidth consumption of around 1.25 Kb/s.

V. Conclusions

In this paper, we addressed the issue resulting from epochs in gossip-based aggregation protocols. In these epochs, a fresh measurement is only captured in the beginning and these initial values are aggregated during the epoch network-wide. At the end of the epoch, the aggregated estimations are consistent, but based on old measurements. We eliminate the limitation of epoch-based gossiping protocols by proposing a continuous gossip-based aggregation protocol without epochs. In our approach, in every gossip round a fraction (α) of current measurements is added to the aggregation, while dynamically forgetting a part (α) of the present aggregation. By this, the gossip-approach is continuous and converges quickly with a relative measurement error of α. The evaluation shows that the approach is quickly converging and comes at very low costs of a message frequency of 0.5 and a bandwidth consumption of 1.25 Kb/s with an update interval of 10 seconds. The presented approach is suitable to be applied in a wide range of connected networks, due to its simple assumptions. In the future, we aim at extending the algorithm to cope with malicious nodes that might disturb and corrupt the aggregation protocol.

References

[1] K. Graffi *et al.*, "From Cells to Organisms: Long-Term Guarantees on Service Provisioning in Peer-to-Peer Networks," in *ACM SIGAPP NOTERE '08: Proc. of the Int. Conf. on New Technologies of Distributed Systems*, 2008.

[2] K. Graffi *et al.*, "Monitoring and Management of Structured Peer-to-Peer Systems," in *IEEE P2P '09: Proc. of the Int. Conf. on Peer-to-Peer Computing*, 2009.

[3] J. Case *et al.*, "RFC 1157: Simple Network Management Protocol (SNMP)," http://www.ietf.org/rfc/rfc1157.txt, Internet Engineering Task Force, 1990.

[4] L. Durham *et al.*, "Platform Support of Autonomic Computing: an Evolution of Manageability Architecture," *Intel Technology Journal*, vol. 10, 2006.

[5] M.-S. Kim, M.-J. Choi, and J. W.-K. Hong, "A Load Cluster Management System using SNMP and Web," *Int. Journal on Network Management*, vol. 12, no. 6, 2002.

[6] F. Dabek *et al.*, "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.

[7] Y. Kulbak and D. Bickson, "The eMule Protocol Specification," School of Computer Science and Engineering, The Hebrew University of Jerusalem, Tech. Rep., 2005.

[8] B. Cohen, "Incentives Build Robustness in BitTorrent," in *P2PECON '03: Proc. of the Workshop on Economics of Peer-to-Peer Systems*, 2003.

[9] K. Graffi *et al.*, "Load Balancing for Multimedia Streaming in Heterogeneous Peer-to-Peer Systems," in *ACM NOSSDAV '08: Proc. of the Int. Workshop on Network and Operating System Support for Digital Audio and Video*, 2008.

[10] M. Steiner, T. En-Najjary, and E. Biersack, "Long Term Study of Peer Behavior in the KAD DHT," *IEEE/ACM Transactions on Networking*, vol. 17, 2009.

[11] K. Graffi *et al.*, "A Distributed Platform for Multimedia Communities," in *IEEE ISM '08: Proc. of the Int. Symposium on Multimedia*, 2008.

[12] K. Graffi *et al.*, "LifeSocial.KOM: A Secure and P2P-based Solution for Online Social Networks," in *IEEE CCNC '11: Proc. of the IEEE Consumer Communications and Networking Conf.*, 2011.

[13] R. van Renesse, K. P. Birman, and W. Vogels, "Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining," *ACM Transactions on Computer Systems*, vol. 21, no. 2, 2003.

[14] R. Sarkar, X. Zhu, and J. Gao, "Hierarchical Spatial Gossip for Multi-Resolution Representations in Sensor Networks," in *IEEE IPSN'07: Proc. of the Int. Conf. on Information Processing in Sensor Networks*, 2007.

[15] M. Bawa *et al.*, "Estimating Aggregates on a Peer-to-Peer Network," Stanford InfoLab, Technical Report 2003-24, 2003.

[16] J. C. Nobre and L. Z. Granville, "Consistency of States of Management Data in P2P-Based Autonomic Network Management," in *IEEE DSOM '09: Proc. of the Int. Workshop on Distributed Systems: Operations and Management*, ser. LNCS, vol. 5841. Springer, 2009.

[17] C. P. Hall and A. Carzaniga, "Uniform Sampling for Directed P2P Networks," in *Springer Euro-Par '09: Proc. of Int. Euro-Par Conf.*, 2009.

[18] T. Bullot *et al.*, "A Situatedness-based Knowledge Plane for Autonomic Networking," *Int. Journal of Network Management*, vol. 18, no. 2, 2008.

[19] M. Dam and R. Stadler, "A Generic Protocol for Network State Aggregation," in *RVK'05: Proc. of the Radiovetenskap och Kommunikation*, 2005.

[20] R. Bhagwan, G. Varghese, and G. Voelker, "CONE: Augmenting DHTs to Support Distributed Resource Discovery," Technical Report CS2003-0755, University of California, San Diego, 2003.

[21] P. Yalagandula and M. Dahlin, "Research Challenges for a Scalable Distributed Information Management System," The University of Texas at Austin, Department of Computer Sciences, Tech. Rep. CS-TR-04-48, 2004.

[22] K. Graffi *et al.*, "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.

[23] K. Graffi *et al.*, "Towards a P2P Cloud: Reliable Resource Reservations in Unreliable P2P Systems," in *IEEE ICPADS '10: Proc. of the Int. Conf. on Parallel and Distributed Systems*, 2010.

[24] Z. Zhang, S. Shi, and J. Zhu, "SOMO: Self-Organized Metadata Overlay for Resource Management in P2P DHT," in *IPTPS '03: Proc. of the Int. Workshop on Peer-to-Peer Systems*, 2003.

[25] F. Wuhib, M. Dam, and R. Stadler, "Decentralized Detection of Global Threshold Crossings using Aggregation Trees," *Computer Networks*, vol. 52, no. 9, 2008.

[26] K. Graffi *et al.*, "Overlay Bandwidth Management: Scheduling and Active Queue Management of Overlay Flows," in *IEEE LCN '07: Proc. of the Annual Conf. on Local Computer Networks*, 2007.

[27] K. Graffi *et al.*, "Practical Security in P2P-based Social Networks," in *IEEE LCN '09: Proc. of the Int. Conf. on Local Computer Networks*, 2009.

[28] K. Graffi *et al.*, "LifeSocial.KOM: A P2P-based Platform for Secure Online Social Networks," in *IEEE P2P '10: Proc. of the Int. Conf. on Peer-to-Peer Computing*, 2010.

[29] N. Liebau *et al.*, "The Impact Of The P2P Paradigm," in *AMCIS '07: Proc. of Americas Conf. on Information Systems*, 2007.

[30] M. Jelasity and A. Montresor, "Epidemic-style proactive aggregation in large overlay networks," in *IEEE ICDCS '04: Proc. of the Int. Conf. on Distributed Computing Systems*, 2004.

[31] 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.

[32] M. Jelasity, A. Montresor, and Ö. Babaoglu, "Gossip-based Aggregation in Large Dynamic Networks," *ACM Transactions on Computer Systems*, vol. 23, no. 3, 2005.

[33] M. Jelasity, A. Montresor, and Ö. Babaoglu, "T-Man: Gossip-based Fast Overlay Topology Construction," *Computer Networks*, vol. 53, no. 13, 2009.

[34] A. Kovacevic *et al.*, "PeerfactSim.KOM - A Simulator for Large-Scale Peer-to-Peer Networks," Technische Universität Darmstadt, Germany, Tech. Rep. Tr-2006-06, 2006.

[35] K. Graffi, "PeerfactSim.KOM: A P2P System Simulator Experiences and Lessons Learned," in *IEEE P2P '11: Proc. of the Int. Conf. on Peer-to-Peer Computing*, 2011.

[36] A. Kovacevic *et al.*, "Towards Benchmarking of Structured Peer-to-Peer Overlays for Network Virtual Environments," in *IEEE ICPADS '08: Proc. of the Int. Conf. on Parallel and Distributed Systems*, 2008.

[37] OECD, "OECD Bandwidth Report," 2007. [Online]. Available: http://www.oecd.org/sti/ict/broadband

[38] T. S. E. Ng and H. Zhang, "Global Network Positioning: A New Approach to Network Distance Prediction," *ACM SIGCOMM Computer Communications Review*, vol. 32, no. 1, 2002.

[39] I. Stoica *et al.*, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," *IEEE/ACM Transactions of Networking*, vol. 11, no. 1, 2003.