Convex Hull Watchdog: Mitigation of Malicious Nodes in Tree-based P2P Monitoring Systems

Andreas Disterhöft
Technology of Social Networks, University of Düsseldorf
Universitätsstr. 1, 40225 Düsseldorf, Germany
Email: disterhoeft@cs.uni-duesseldorf.de

Kalman Graffi
Technology of Social Networks, University of Düsseldorf
Universitätsstr. 1, 40225 Düsseldorf, Germany
Email: graffi@hhu.de

***Abstract*—Monitoring the global state in peer-to-peer networks through decentralized mechanisms allows targeted optimization and improvement of the peer-to-peer network. However, malicious nodes could aim to distort the process of gathering the global state through monitoring. In this paper, we propose DOMiNo, a security solution for tree-based peer-to-peer monitoring mechanisms. It passively listens to incoming events, e.g. data, and rates its suspiciousness based on outlier detection, structural verification and sanity check mechanisms. For our main objective, which is to limit the monitoring error of the desired global view, we performed an extensive evaluation. Evaluation shows tolerance with normal fluctuations but effective filtering of outliers, that severely influence the global view. As our watchdog solution operates passively, we do not add any costs nor create new surface for attacks to the monitoring system.**

I. Introduction

Peer-to-peer (P2P) networks offer many advantages compared to client-server network models. They are able to build networks using only the capacities of every single peer participating in the network and build up applications with a minimum of operational costs. However, as no central instance is present it is hard to manage these networks to offer a certain Quality of Service. Monitoring systems present a solution to gain information about the status of the network so peers may adapt several parameters to improve the overall performance (e.g. reducing the output during high package loss or increasing their routing tables' size in unstable networks). In particular, structured monitoring systems provide highly precise monitoring results by building a topology to effectively aggregate and distribute data. Since every peer gathers its local view of the network (e.g. local hop count, response time etc.) and has to contribute its known information to the monitoring system to gain a profound view on the underlying P2P overlay, new attack possibilities arise. Even more as no trustworthy instance, i.e. a server, is present.

P2P overlays with controlled quality of service might play a role in future, such as for P2P-based online social networks (LibreSocial [9]), WebP2P-based communication platforms [6] or future communication platforms for decentralized energy sources. As an example the Department of Computer Science from the University of Oldenburg [26] uses a constructed communication overlay to exchange data between several power plants which monitor each other. If one of those power plants gets compromised the results could be fatal. Not just in this cases, but in general, more information on the P2P network, in specific global information as obtained from monitoring mechanisms, are beneficial for the operation of the network. Bottlenecks can be identified, parameters configured ideally and an autonomous control loop could be implemented on the long run, as suggested in [12]. As information is crucial in the correct operation of a P2P overlay, malicious nodes could aim to distort the monitoring mechanism, thus aiming at indirect damage through the processing of false observations.

In our paper, we propose a security solution, DOMiNo, for existing structured tree-based monitoring systems. Our approach is applicable for tree-based monitoring systems which aggregate gathered system-specific information in a push-based manner. Examples for suitable systems are CONE [3], SDIMS [28], SOMO [30] and SkyEye.KOM [10]. Appropriate systems require a dissemination process, where each participating node is supplied with the monitoring mechanism's global view. The construction of the monitoring tree must be deterministic. Thus a parent node in the tree must be able to identify a certain ID space where its child nodes must be located. This requirement is interchangeable with any other criteria where a node's parent is able to verify its child nodes. Please note, if this requirement is not met by the tree-based monitoring system, malicious nodes can inject their (manipulated) data at arbitrary and even multiple points in the monitoring tree, which may diminish our system's efficiency. Also, a node can only be considered for the monitoring process if it is actively participating in the underneath p2p overlay, thus nodes participating in the monitoring process can check each node's activity. With these assumptions, our solution provides following properties:

- Passive Rating: No overhead is added to the existing monitoring approach. The detection does not add any overhead in terms of exchanging information, i.e. sending messages, nor does it perform any additional non-local operations like lookups. Thus, DOMiNo is a passive rating mechanism which judges by listening to inbound data messages only. By doing so, we do not introduce new possibilities for attackers to disturb or influence the monitoring system in a bad way.
- Precision: The main objective is to minimize the relative

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

DOI 10.1109/LCN.2016.16

error from the measured global view in a malicious environment to the measured global view in a non-malicious environment respectively desired global view. Thus, we prioritize a high precision over a minimization of the false positive rate.
- Performance: The detection mechanism should need as little effort in terms of computation time as possible.

First, in Section II we introduce briefly in the literature in the field of monitoring systems in general and security of them in particular. In order to be able to evaluate our solution, we focus on the motivation of attackers and concretize a fitting attacker model in Section III. Our main contribution, called DOMiNo, which makes use of outlier detection mechanisms based on Z-Scores, structural and other plausibility checks, is presented in Section IV. The quality of our proposed system DOMiNo is evaluated in Section V through simulations. We show that the influence of the malicious node's attempt on the global view are effectively limited. This paper closes with a conclusion and comments on future work in Section VI.

II. Related work

In the research field of monitoring in decentralized networks there exist several approaches but none of them handle malicious behavior. On one hand, the unstructured gossip-based approaches like [16], [18], [23] periodically exchange messages with their neighbors and average the obtained values. Over time, the system wide average is obtained, which can be used also for sum or count calculations. Non of these solutions propose a mechanism to protect the system against misbehaving nodes. On the other hand, there are the structured monitoring approaches, which are mostly tree-based. GAP [5] and its follow ups Adaptive GAP [22] and TCA-GAP [27] build up a tree to aggregate monitoring information and provide a threshold crossing alert but they do not support malicious behavior analysis which is also mentioned in TCA-GAP. Other approaches like CONE [3], SDIMS [28], SOMO [30] and SkyEye.KOM [7] also do not provide such mechanisms. In this paper, we propose an add-on which can be used by the before mentioned approaches.

In the field of sensor networks there exist some approaches like SDAP [29] or from Chan et al. [4]. In the first approach, SDAP, the authors propose a grouping technique for aggregates to determine the suspiciousness for those groups. In their approach they build up new communication paths for their group-ups, in contrast, we passively listen to incoming data and rate it. Also, we reserve the right to drop suspicious data. The second approach also builds up its own structure which violates our requirement of being passive. Our solution is a passive module which can be integrated to the already existing tree-based aggregation structure.

Another field of research are reputation systems in P2P networks, wherefore Marti and Garcia-Molina provide a survey [19]. Here, the most prominent solution is EigenTrust [17], where nodes actively distribute trust vectors whereas the trust system as a whole is transitive (if node A trusts B then A trusts all nodes B trusts). However, we focus on passive mechanisms, which do not add any message exchange to the current monitoring service and do work only locally.

The field of (distributed) intrusion detection systems, which work passively, are well suited for our purpose. But in contrast to traditional (distributed) intrusion detection systems like mentioned in the survey from Axelsson [2], our desire is to provide a mechanism with a decision-making component which i) operates only on the local view ii) works decent on normally distributed data iii) does not need a preceding training and iv) operates reliably on small data sets. In the field of Mobile Ad Hoc Networks, Watchdog mechanisms [20], [21] have been proposed that observed the forwarding behavior of neighboring nodes and mark them as potentially malicious. While in Mobile Ad Hoc Networks, communication is assumed to be observable, in P2P networks, communication paths are hidden. Nevertheless, the idea is appealing.

Therefore we have chosen to apply methodologies from the field of *outlier detection* [13]. Due to our restrictions we can not make use of Supervised and Semi-supervised outlier detections, thus Unsupervised methods, which are proximity-based, cluster-based, density-based, distance-based or using probabilistic / statistical models, are left. Most of the mentioned methods operate reliably on bigger data sets and / or build clusters, which do not fit our needs. Except statistical models, like the Extreme Value Analysis with Z-Scores, which is used by Iglewicz and Hoaglin [15], fit our needs. Thus using the divergence to the standard deviation of the data set we are able to locally rate the badness of incoming data. Additionally, the proposed method from Iglewicz and Hoaglin works best on a normally distributed data base.

Hence, in the theory, using an appropriate divergence factor, each node should be able to distinguish between data from honest nodes including some noise and bad data, if the number of honest data exceeds the number of malicious data. In Section IV we describe how we utilized this in our solution.

III. Attacker Model

Attackers in monitoring systems can have different intentions to behave not in accordance with the monitoring algorithm. We state briefly the motivation of attackers, classify attack patterns and extract an attacker model.

A. Motivation of Attackers

The literature offers a wide range of an attacker's motivation to act malicious as stated in the survey from Rounds and Pendraft [24]. However, irrespective of the reason for being malicious, virulent nodes participating in a monitoring system can be categorized into the following classes.

In the first class, participants may manipulate monitoring data to hinder / influence the QoS (= Quality of Service) mechanism of the system by changing the current system view or certain node's capacities. Applying nodes may aim to attack honest nodes' global views to, for example, let QoS mechanisms regulate accordingly which may lead to an overall worse P2P experience.

The second class deals with free riders which want to maximize their benefit with as little own effort as possible. So, these malicious nodes are only interested in the global view but ideally do not want to forward any messages.

The third and last class of malicious nodes want to purposefully disable monitoring by not following rules or even shutting down specific mechanisms or sealing off single nodes. In contrast to the first group, this group tries to break links in the monitoring system and to exclude respectively eclipse specific nodes or whole sub trees from participation. This implies an incomplete global view for nodes not eclipsed and no global view for eclipsed nodes which leads to unawareness of the P2P network's current state.

B. Attack Patterns

Next, we describe attacks that achieve the aims mentioned in the last section. For this, we want to stick to an example monitoring system that fulfills the requirements mentioned in Section I. SkyEye.KOM [10] is a tree-based monitoring system which aggregates statistics in a bottom-top manner and disseminates the global view top-bottom. For that, all nodes calculate initially their position in the tree as well as their parent's DHT ID based on the node's ID responsibility in the DHT. Periodically, each nodes captures its local statistics, aggregates it with those received from its child nodes, and sends the aggregated snapshot of its subtree to its parent node. The parent node replies with an ACK containing its currently known global view. Over time, all information is aggregated towards the root, where the global view is created and from there trickled down to all nodes using the ACKs.

Figure 1 shows an example topology in SkyEye.KOM with branching factor 2. Black lines denote relevant respectively *active* connections of an attack, where data is sent in the direction the arrow points. Dotted lines show connections which should be established following the monitoring algorithm, but will not be maintained as a result of an occurring attack. If marked with an arrow, a dotted line shows that one side tries to establish the connection unsuccessfully. Gray lines show existing connections according to the SkyEye.KOM algorithm which are not affected by either attack depicted in the Figure.

We will denote the different attack types via the schema $AT_{Abbreviation-for-Attack}$ for future references.

1) Manipulation of Monitoring Results: This group contains malicious nodes which send data to other nodes to change the current view of them. With this they still keep the monitoring procedure running, but can influence the behavior of the whole network, e.g. by sending parent and child nodes the information that only little bandwidth is available in the corresponding sub tree. Another possibility would be to declare high available bandwidth which may lead to the failure of single nodes due to congestion or similar. Next, we want to differentiate the following attacks in this group.

a) AT_{Parent} – Send modified Data Sets to the Parent Node: If a peer sends wrong data to its parent node it is able to modify the local view in the parent node and therefore such a peer has indirect influence on adjacent sub trees, as the parent node will send the influenced local view up towards the root. At some point the root will get the modified data and uses it to build the global estimation which will be sent down the tree. Furthermore peers may present themselves as weak nodes to maximize their benefit from the running P2P overlay without providing too much resources themselves as many overlays have mechanisms to balance load among the peers according to their capabilities.

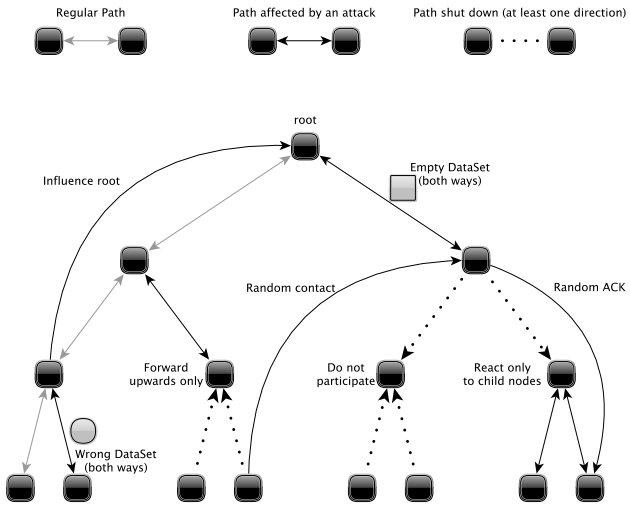

Fig. 1: Overview of attack types against SkyEye.KOM, a tree-based monitoring system fulfilling our requirements

This attack type is best suited for having influence on the monitoring results while being as unobtrusively as possible since the node behaves *correctly* regarding the message forwarding algorithm, i.e. it addresses the correct parent node.

b) AT_{Child} – Send modified Data Sets to the Child Node(s): Peers can influence all their child nodes or only a specific child node by forwarding incorrect global views downwards the tree with the ACKs. With this they are able reduce the profit child nodes get from the P2P overlay, as they may not be able to react on potential network changes.

Just like AT_{Parent}, malicious nodes using this approach behave correctly according to the monitoring procedure.

c) AT_{Random} – Send Data to a Random Contact: Sending data to random nodes corresponds to sending messages upwards the tree and lets the sender appear as a child node of the receiver. In addition, nodes applying this attack also manipulate the data to increase their impact. As depicted in Figure 1 this attack can affect arbitrary nodes at any position and may cause loops in the tree if forwarded to a node lower in the tree, which may lead to exorbitant global views. Furthermore, the number of child nodes may exceed a possible maximum, i.e. the branching factor, if all possible child nodes are present, which may lead to further actions depending on the monitoring system, e.g. reorganization respectively applying load balancing mechanisms.

d) $AT_{RandomACK}$ – Send `ACK` to a Random Contact: In SkyEye.KOM, application level ACK messages are sent

downwards the tree to disseminate the global view. In general, this kind of message is interchangeable, but in the following, ACKs are linked to the dissemination of the global view. Consequently, sending arbitrary ACK messages will influence or even override the global view in the receiving node.

Using this attack results in a more suspicious behavior as parent nodes are usually well known since child nodes initiate the communication between themselves and the parent.

2) Free Riders: max benefit, as little own effort as possible: Free riders want to maximize their benefit from a mechanism with little own effort. This can be achieved requesting the global view which is calculated, e.g. at the root node.

a) AT_{Root} – Send Data to the Root directly: Data sent to the root node directly by a node that is not a legitimate child is a special case of the attack AT_{Random}. As demanded in Section I, a certain ID space can be identified where child nodes live in but they cannot be fully verified. In contrast to AT_{Random}, this type of attack has a much higher influence (on the global view), as the root node aggregates the global view and starts the dissemination down the tree.

3) Shutting down of Monitoring Mechanisms: The following attack types aim to shut down specific mechanisms of the monitoring system or to seal off single nodes from it. These attacks are more aggressive towards the concerned parts of the system as they do not provide any information at all.

a) AT_{Empty} – Forward Empty Data Sets: Forwarding empty data sets results in the erasure of parts of the gathered statistics if the forwarding is upwards to the root or even in the erasure of the whole statistics if empty sets get forwarded downwards the tree. Attackers are able to shut down significant parts of the monitoring tree completely as the receivers have to work with sparse data or even no data if the attacker is the only connection to the other tree parts. In Figure 1, this would mean that if the root node uses this attack, the right sub tree would not be able to gain any global data according to the message forwarding algorithm.

b) $AT_{Upwards}$ / $AT_{Downwards}$ – Forward Data only upwards or downwards: On the one hand, if nodes forward data only upwards, i.e. without participating in the dissemination process, they take advantage of the monitoring system without contributing much to it except the message they send towards the root and the contribution to the global view. On the other hand, if nodes forward data only downwards, they can eclipse a whole sub tree without instantly being suspicious. Nodes in the sub tree will not be able to commit to the global view and may only receive information about the sub tree itself if the attacker decides to do so.

c) $AT_{DoNothing}$ – Do not participate: Missing participation in the monitoring system will result in broken links between multiple tree parts since the denying nodes are the proper parent respectively child nodes and thus need to be contacted as part of the monitoring algorithm. This behavior will both influence the monitoring system's results and shut off specific nodes from the system. It combines both attacks $AT_{Upwards}$ and $AT_{Downwards}$.

IV. Our Solution: DOMiNo

***D**etection **O**f **M**alicious **No**des* (DOMiNo) is our approach to identify inappropriate behavior in a tree-based structured monitoring system like SkyEye.KOM, which meets the requirements stated in Section I. Our approach is based on a rating system containing multiple rating mechanisms, each returning a numerical rating for an examined data set or peer, respectively. Ideally, once a peer behaves maliciously, it would trigger one or more active rating systems and be detected.

In the following, we discuss the detection criteria and identify which attacks these criteria are able to detect. We divide our rating mechanisms into *instant rating mechanisms* which instantly determine malicious behavior and *collecting rating mechanisms* which operate on a collected data base.

A. Instant Rating Mechanisms

Instant rating mechanisms check different properties of a received message, e.g. its origin or the sender's properties. Furthermore they review the containing data sets regarding several points, e.g. the completeness or the plausibility of data, using simple but effective rules.

1) Check Data's Origin: This mechanism checks each received data's origin whether its ID lies in the acceptable ID range where data is excepted to come from, as demanded in Section I. By doing so, false negatives cannot arise while false positives are still possible. For this, nodes should not be able to forge their IDs. To achieve this, several approaches can be applied like using a PKI, where the public key equals the node ID, thus encrypting data using the public key can only be decrypted by the node owning the corresponding private key (See Graffi et al. [11]). By doing this, IDs are coupled to an identity and its asymmetric key pair, i.e. to a user. IP/MAC spoofing for example is useless as incoming data can not be decrypted due to the missing private key. This mechanism can be exploited to detect AT_{Random} attacks.

2) Identifying correct Parent Nodes via Nonces: By introducing numbers-used-once (nonces) set by the requester and used by the responding node to identify a pair of request and corresponding response message, the monitoring mechanism can cope with the attack type $AT_{RandomACK}$. So, the requesting node is able to identify the response to a sent request and can ignore or filter out ACK messages with an unknown nonce.

3) Implausible Values: Based on the knowledge we have about the data we receive respectively expect we can perform several checks for implausible values. Depending on each data set's values, numerous sanity checks like $min <= max$, $min <= mean <= max$, $sumOfSquares <= max^2 * count$, etc. should hold. These criteria support the detection of attacks which send insufficient data, i.e. AT_{Empty}, but also every attack which carelessly manipulates data.

4) Awareness of Missing ACKs: As part of the monitoring mechanism, once contacted, each node can expect a reaction from the contacted node if it is active. A node ignoring this policy is acting highly suspicious. In order to not produce false positives, as a message can get lost or just delayed, an

active node is identified to be malicious if it repeatedly does not answer, e.g. after three unsuccessful tries.

By doing so, we are able to detect those attacks where nodes do not or partly participate in the monitoring system, i.e. $AT_{Upwards}$ and $AT_{DoNothing}$ attacks. Please note, that leaf nodes running $AT_{DoNothing}$ cannot be detected with a passive approach such as DOMiNo. Depending on the used monitoring system, parents may be not aware of their children. In order to rectify this, new lookups would be needed producing a significant amount of new overhead to the overlay.

B. Collecting Rating Mechanisms

In contrast to instant rating mechanisms, collecting rating mechanisms need a certain data base in order to work. With the help of this data base, our proposed mechanisms determine the likeliness that the data was produced by the same system and try to differentiate manipulated data from correct data.

In the following, we propose our collecting rating component which uses Z-Scores.

Outlier detection: Z-Scores: Our outlier detection, which has its root in the Modified Z-Scores introduced by Iglewicz and Hoaglin [15], rates each incoming data set by using its local data base. This data base contains all received data sets which are not older than a certain maximum age, thus, the branching factor of the monitoring tree affects the data base used by our algorithm.

First, based on the Modified Z-Scores, the median absolute deviation (in short MAD) is calculated using the Equation 1, where x is the whole data base and x_i is the $i'th$ entry which gets rated. Please note, the use of the median is crucial in order to not get biased by extreme values. The Modified Z-Score is calculated using the Equation 2. The authors propose the threshold, for labeling a data set as a potential outlier, to be 3.5. This threshold identifies when noise turns into suspiciousness, which is defined by a convex hull around a multiple of the median absolute deviation as visualized in Figure 2. By increasing this threshold the algorithm gets increasingly tolerant to noise.

$$MAD = median(|x_i - median(x)|) \tag{1}$$

$$Score_i = \frac{|0.6745(x_i - median(x))|}{MAD} \tag{2}$$

In the rare case where at least 50 percent of the provided data sets report the same value, Equation 1 turns to 0, thus we would divide by 0 in Equation 2. In this case all points that do not equal the median could be marked as an outlier. In our context this is not a desired behavior as we require to operate on diverse data input with a possibly high deviation. Especially when using discrete data, e.g. the hop count, this case might happen. To be able to make a meaningful decision nevertheless, we use a solution used by IBM in their software SPSS Statistics [14]. They use the mean absolute deviation (MeanAD) in case the MAD equals 0. The corresponding calculations for the $MeanAD$ respectively $Score_i$ are denoted in Equations 3, 4. MeanAD can only be 0, if all data set values

Algorithm 1 Modified Z-Median Score Algorithm

```
1: Input: A set of collected data D, the data point to examine d_i
2: procedure CalculateModifiedZMedian
3:     dataSetMedian ← median(D)
4:     for all d_x in D do
5:         add |dataSetMedian − d_x| to a median deviation list MD
6:     end for
7:     MAD ← median(MD)
8:     if MAD == 0 then                ▷ If more than 50% of the value were equal
9:         MeanAD ← average(MD)
10:        if MeanAD == 0 then                          ▷ All values are equal
11:            Score_i ← NOTSUSPICIOUS
12:        else
13:            Score_i ← |(0.7979 ∗ (d_i − dataSetMedian))|/MeanAD
14:        end if
15:    else
16:        Score_i ← |(0.6745 ∗ (d_i − dataSetMedian))|/MAD
17:    end if
18:    return Score_i
19: end procedure
```

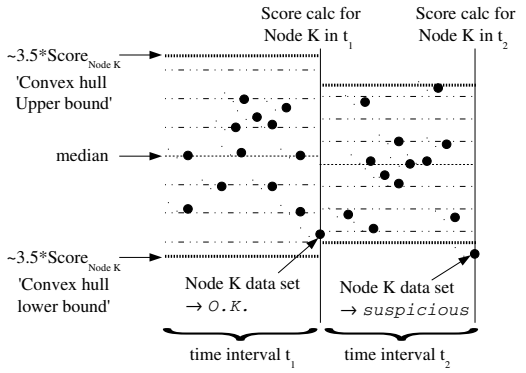

Fig. 2: Visualization of a node rating its incoming data sets.

are the same and therefore no outlier is present.

$$MeanAD = average(|x_i - median(x)|) \tag{3}$$

$$Score_i = \frac{|0.7979(x_i - median(x))|}{MeanAD} \tag{4}$$

Algorithm 1 presents our method to get a rating for a single data set. Due to the possibility of false positives, especially given the reduced data base, one should keep in mind to control a node's behavior over a certain time interval and not prejudge a node because of a single statistical abnormality. There is no guarantee that a node may not look suspicious for a short amount of time due to sudden changes in the network or other reasons. We therefore propose to evaluate a node's behavior several times before making a final judgment when using statistical approaches. This countermeasure is most suited for manipulation attacks sending data upwards the tree, i.e. AT_{Parent}, AT_{Random}, and AT_{Root}.

Additionally, all attacks mentioned in Section III may be applied over a long period of time which could increasingly influence the global view created by the monitoring mechanism. In order to minimize the impact of false values on the calculation of the convex hull, old monitoring data is cleaned periodically by using a sliding window approach, i.e. only a limited number of data sets, in particular the latest, is taken into account.

C. Blocking Data Sets

In order to reduce or even clear the effect of abnormal behaving nodes, we propose to drop malicious data. If a participant does not behave correctly, we do not want him to influence the monitoring mechanism.

We propose to discard suspicious data according to three policies implementing an increasing cautiousness against the impact of malicious data. Discard *none* is the incautious approach which does not drop data. The *discard_safe* policy only discards data sets which are certainly malicious. These are attacks against the monitoring structure and rated by our instant rating mechanism, thus false positives are impossible, as the participant's behavior is checked. Lastly, *discard_all* drops each suspicious data set. Please note, due to the use of an outlier detection system, we may drop false positive data.

Data Set Discard Theory: If aggregated data sets have to be dropped due to a discard policy, then the data of a complete subtree will be discarded. To have a rough estimation how many data sets are expected to be dropped we introduce a formula modeling the tree with following simplifications: i) all malicious data sets have been found ii) the tree is balanced and iii) complete. Here is the formula:

$$\phi = Round(log_{bF}(N * p * (bF - 1))) - 1 \tag{5}$$

$$\sum_{i=0}^{\phi}(\frac{bF-1}{bF^{i+1}} * p * N * \sum_{j=0}^{i}(bF^j - p * bF^j)) \tag{6}$$

Here, N is the total number of nodes, p the ratio of malicious nodes and bF the branching factor of the tree. The idea behind Equation 6 is to traverse the tree bottom-top and calculate on each level how many nodes respectively data sets should be dropped if this node has been detected from his parent. The inner sum calculates how many nodes this dropped sub tree has and subtracts already dropped nodes. The outer sum stops when the probability that a malicious node is located at that level is close to zero.

V. Evaluation

To evaluate our developed rating system, we simulated several scenarios using our implementation of SkyEye.KOM, which meets our requirements to the monitoring system (Section I), in the PeerfactSim.KOM simulator [8]. As a preparation, we marked all (aggregated) statistics containing information from malicious nodes with an "evil bit" [1] which tells us the number of malicious nodes that influenced this piece of information. Ideally, all information marked with evil bits should be discarded by DOMiNo.

For the evaluation, we consider the following metrics: i) number of evil bits in the the global view, which equals the detection rate (in short #evilBits), ii) the number of discarded data sets, which gives together with #evilBit a hint about the false positives, iii) precision, where we compare the measured global view with the desired one and iv) at a certain point the costs respectively the fairness of the system's load distribution. Next, we describe the different used simulation parameters along with our scenarios.

Simulation Parameters and Scenarios

For our evaluation, we simulated beforehand a lot of simulation setups and extracted the most important simulation parameters which significantly influenced the outcome. As the configuration file of the simulator is very extensive we only describe the relevant parameters for our simulation results, which are summarized by the Table I. We simulated a Pastry [25] overlay network consisting of 5000 nodes over a total of 200 minutes without churn. Exemplarily, we let SkyEye.KOM create statistics about the average number of bytes sent per node on the network layer.

The actions performed during a simulation are the same in each scenario. The first 60 minutes are the join phase where all nodes join the overlay and start distributing data in the monitoring system from minute 90 on. In parallel, we had a file sharing application running, publishing data sets between minute 60 to 80 and starting lookups after them every 10 minutes to feed the monitoring system. From minute 100 on, each node starts the rating phase. Note that malicious nodes also use DOMiNo. After feeding DOMiNo with monitoring data for 10 minutes, we get first results at minute 110.

In the scope of this paper we performed an extensive evaluation with 228 parameter combinations, drafted by Table I, with at least 10 seeds each. Out of these, we present six scenarios. In Scenario A (Section V-A) we show the impact of a data manipulating attack with DOMiNo deactivated. The impact of DOMiNo on a non-malicious environment is shown in Scenario B (Section V-B). In Scenario C.1 to C.3 (Section V-C) we expose a parameter study when performing data manipulation attacks with DOMiNo activated, which gives a first impression on how well our proposed system works. Lastly, Scenario D (Section V-D) reveals the outcome when releasing all attacks with DOMiNo activated while discarding all malicious rated data and discarding only data which is marked as assured malicious, respectively. At the end, in Section V-E, we present a discussion on our proposed system's characteristics.

Table II presents the simulation outcome ordered by our scenarios. For visualization purposes for each metric we present the absolute and relative error of the simulations average from minute 110 till 200 ($\Delta_\emptyset$ and $\delta_\emptyset$). Additionally, for the data set discard count and global view metrics we exhibit the standard deviation, denoted as σ_X with X being the value we create the standard deviation for. To enhance the overview, for each scenario and metric we color the best error value with light gray respectively the worst with darker gray.

A. Scenario A: Manipulation attacks / DOMiNo deactivated

Scenario A demonstrates the impact of manipulation attacks on an unprotected monitoring system. The simulations' outcome can be taken from Table II. Here, the branching factor (= bF) is set to 96 for future comparisons and we vary the malicious ratio and malicious data manipulation factor. We encounter no thrown away data sets and $5000 * m\_ratio$ evil bits in the global view. The global view's mean attribute shows we can arbitrarily influence the global view, which scales with the number of malicious nodes and their manipulation factor.

TABLE I: Parameters chosen for our simulations

Parameter	Value(s)
Number of nodes	5000
Number of seeds	≥ 10
Churn	Inactive
Simulation time	200 minutes
Used P2P overlay	Pastry
Attack types, (see Section III-B)	No, only data manipulation, all
Branching factor	$8, 32, 96$
Z-Score threshold	$3.5, 4.5, 5.5$
Discard policy	none, discard_safe, discard_all
Data manipulation factor	$2, 10$
Malicious node ratio	$5\%, 10\%, 20\%$

TABLE II: Evaluation outcome for all scenarios

Simulation Configuration						Metric: data set discard count				Metric: global view (mean attribute)				Metric: evil bits in global view	
attacks: [n]o - [m]an. - [a]ll	malicious ratio	branching factor	Z-score threshold	discard policy: [s]afe - [a]ll	manipulation factor	Absolute error $\Delta_\varnothing$	Relative error $\delta_\varnothing$	Abs. error, standard dev. $\sigma_{\Delta_\varnothing}$	Rel. error, standard dev. $\sigma_{\delta_\varnothing}$	Absolute error $\Delta_\varnothing$	Relative error $\delta_\varnothing$	Abs. error, standard dev. $\sigma_{\Delta_\varnothing}$	Rel. error, standard dev. $\sigma_{\delta_\varnothing}$	Absolute error $\Delta_\varnothing$	Abs. error, standard dev. $\sigma_{\Delta_\varnothing}$
Reference Scenario: No attacks with DOMiNo deactivated															
n	0	8	-	-	-	0	0	0	0	5	0.008	3	0.003	0	0
Scenario A: Impact of manipulation attacks with DOMiNo deactivated															
m	0.05	96	3.5	s	10	0	0	0	0	805	1.186	232	0.308	248	11
m	0.05	96	3.5	s	2	0	0	0	0	78	0.114	25	0.032	251	13
m	0.1	96	3.5	s	10	0	0	0	0	2272	3.34	603	0.773	495	16
m	0.1	96	3.5	s	2	0	0	0	0	162	0.238	38	0.046	508	18
m	0.2	96	3.5	s	10	0	0	0	0	6771	9.975	1459	1.764	995	33
m	0.2	96	3.5	s	2	0	0	0	0	356	0.525	57	0.054	984	25
Scenario B: Evaluation of false positives with DOMiNo activated and no attacks															
n	0	8	3.5	a	-	-605	-0.121	234	0.047	-17	-0.027	4	0.007	0	0
n	0	8	4.5	a	-	-366	-0.073	178	0.036	-11	-0.018	3	0.005	0	0
n	0	8	5.5	a	-	-259	-0.052	125	0.025	-6	-0.011	2	0.005	0	0
n	0	32	3.5	a	-	-319	-0.064	54	0.011	-22	-0.034	4	0.004	0	0
n	0	32	4.5	a	-	-204	-0.041	32	0.006	-13	-0.021	2	0.002	0	0
n	0	32	5.5	a	-	-155	-0.031	12	0.002	-10	-0.015	2	0.003	0	0
n	0	96	3.5	a	-	-122	-0.024	43	0.009	-17	-0.024	4	0.004	0	0
n	0	96	4.5	a	-	-48	-0.01	9	0.002	-7	-0.01	2	0.002	0	0
n	0	96	5.5	a	-	-21	-0.004	8	0.002	-1	-0.001	2	0.002	0	0
Scenario C.1: Effect of manipulation attacks (factor 10) with DOMiNo activated															
m	0.1	8	3.5	a	10	-2465	-0.493	376	0.075	4	0.005	13	0.02	16	6
m	0.1	8	4.5	a	10	-2240	-0.448	308	0.062	34	0.053	25	0.033	30	10
m	0.1	8	5.5	a	10	-2024	-0.405	347	0.069	58	0.091	31	0.04	42	14
m	0.1	32	3.5	a	10	-1586	-0.317	198	0.04	-7	-0.012	8	0.012	15	7
m	0.1	32	4.5	a	10	-1435	-0.287	199	0.04	8	0.011	12	0.016	28	11
m	0.1	32	5.5	a	10	-1394	-0.279	189	0.038	26	0.039	16	0.019	39	11
m	0.1	96	3.5	a	10	-1116	-0.223	142	0.028	6	0.008	6	0.008	16	7
m	0.1	96	4.5	a	10	-1141	-0.228	164	0.033	21	0.03	9	0.009	27	8
m	0.1	96	5.5	a	10	-1101	-0.22	123	0.025	40	0.058	16	0.019	43	13
Scenario C.2: Effect of manipulation attacks (factor 2) with DOMiNo activated															
m	0.1	8	3.5	a	2	-1681	-0.336	384	0.077	77	0.123	25	0.027	244	32
m	0.1	8	4.5	a	2	-1313	-0.262	364	0.073	103	0.166	29	0.035	300	31
m	0.1	8	5.5	a	2	-1243	-0.249	389	0.078	127	0.205	33	0.039	315	30
m	0.1	32	3.5	a	2	-922	-0.184	206	0.041	57	0.088	11	0.01	326	20
m	0.1	32	4.5	a	2	-785	-0.157	389	0.078	73	0.112	14	0.011	345	32
m	0.1	32	5.5	a	2	-767	-0.153	260	0.052	91	0.141	16	0.013	366	27
m	0.1	96	3.5	a	2	-731	-0.146	85	0.017	42	0.062	8	0.009	345	19
m	0.1	96	4.5	a	2	-600	-0.12	122	0.024	61	0.089	10	0.008	383	14
m	0.1	96	5.5	a	2	-527	-0.105	121	0.024	74	0.109	12	0.009	418	20
Scenario C.3: Effect of manipulation attacks (factor / malicious share) with DOMiNo activated															
m	0.05	96	3.5	a	2	-324	-0.065	87	0.017	11	0.016	2	0.003	174	6
m	0.1	96	3.5	a	2	-731	-0.146	85	0.017	42	0.062	8	0.009	345	19
m	0.2	96	3.5	a	2	-1046	-0.209	145	0.029	119	0.175	19	0.014	685	35
m	0.05	96	3.5	a	10	-684	-0.137	122	0.024	-6	-0.009	3	0.004	7	2
m	0.1	96	3.5	a	10	-1116	-0.223	142	0.028	6	0.008	6	0.008	16	7
m	0.2	96	3.5	a	10	-1987	-0.397	140	0.028	65	0.094	30	0.034	60	20
Scenario D: Effect of all attacks with DOMiNo activated															
a	0.05	96	3.5	s	2	-246	-0.049	56	0.011	14	0.02	9	0.011	141	14
a	0.1	96	3.5	s	2	-445	-0.089	70	0.014	28	0.041	14	0.018	273	18
a	0.2	96	3.5	s	2	-884	-0.177	109	0.022	58	0.084	31	0.039	511	18
a	0.05	96	3.5	a	2	-420	-0.084	58	0.012	-20	-0.029	13	0.021	102	11
a	0.1	96	3.5	a	2	-687	-0.137	89	0.018	24	0.035	167	0.247	213	14
a	0.2	96	3.5	a	2	-1053	-0.211	144	0.029	10	0.013	23	0.03	456	17
a	0.05	96	3.5	s	10	-191	-0.038	85	0.017	153	0.225	77	0.108	139	9
a	0.1	96	3.5	s	10	-491	-0.098	114	0.023	285	0.418	86	0.11	268	19
a	0.2	96	3.5	s	10	-907	-0.181	199	0.04	inf	inf	inf	inf	504	23
a	0.05	96	3.5	a	10	-445	-0.089	112	0.022	-22	-0.032	4	0.006	81	8
a	0.1	96	3.5	a	10	-730	-0.146	95	0.019	-25	-0.038	3	0.007	154	9
a	0.2	96	3.5	a	10	-1075	-0.215	71	0.014	430	0.632	142	0.187	419	16

For example 20% malicious nodes with a manipulation factor of 10, i.e. multiplying monitoring data with 10, result in a relative error of 9.975 times the global view. I.e. the measured value in the network is nearly 10 times higher than the actual values.

B. Scenario B: No attacks / DOMiNo activated

In Scenario B, we stage the impact of DOMiNo on a network without any attackers, in other words, we want to evaluate the false positives and their influence on the global view. Here, we vary the bF and the Z-Score threshold, i.e. the range of accepted values. Table II shows the outcome and the reference, where no malicious nodes are included with deactivated DOMiNo. As expected, we do not detect evil bits in the global view.

By fixing the bF and increasing the threshold, the convex hull gets wider, so the system is getting more tolerant of outliers, thus the number of discarded data sets and the global view's error drops. But we still face a very small relative error from 0.1% to 3.4% compared to the reference with 0.8%. Increasing the bF has a positive effect on the number of dropped data sets as each node gets more data for its rating data base. The improving effect on the global view's mean is only reasonable for big bFs. One may argue that a bigger bF implies an increasingly disbalance of the node's load. Figure 3 shows the sent bytes per peer sorted by their amount at the end of the simulation. A tendency for a disbalanced fairness is noticeable as around 300 nodes have more load throughout the simulation but we increased the bF by a twelve-fold and the worst-case (Peer 1) only encounters an increase of 1.5.

So we outline: This scenario works best with a high bF and threshold, i.e. bF=96 and threshold=5.5. The overhead for an increased bF is manageable.

C. Scenario C: Manipulation attacks / DOMiNo activated

In this scenario, we enable data manipulation attacks and activate DOMiNo in three Subscenarios C.1, C.2 and C.3.

First, in Scenario C.1, we repeat the parameter variation of Scenario B with 10 % malicious nodes and a reasonable manipulation factor of 10. With an increasing threshold, i.e. a wider convex hull, the system increasingly tolerates noise, thus the number of discarded data sets is decreasing, the number of evil bits is raising and so does the global view mean value. Meanwhile an increasing bF counteracts the number of dropped data sets, which is obviously true as the depth of the tree is shrinking and dropped aggregated data contains increasingly more data sets. This can be verified with our discard Equation 6. Following the formula we get 1736 (bF 8), 1335 (bF 32) and 895 (bF 96). An increasing bF also results in a decreasing standard deviation of discarded data sets but it does not significantly influence the other two metrics.

Then, in Scenario C.2, we perform the same parameter variation as in Scenario C.1, but we lower the manipulation factor from 10 to 2. With a decreasing manipulation factor the number of evil bits raises, thus less data sets are discarded as the factor 2 is mostly not enough to exceed the limits of the convex hull, a malicious data set is interpreted as noise in an acceptable zone. The decreasing manipulation factor also results in a greater error in the desired global view. Before

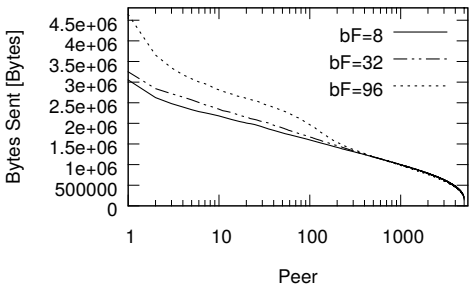

Fig. 3: Per peer sent bytes on net layer

referring to Scenario C.3 we conclude that the parameter setting bF=96 and Z-score threshold=3.5 works best in terms of precision and false positive discards.

Lastly, in Scenario C.3, we apply optimal parameters for bF and Z-score threshold (96/3.5) and perform a parameter variation on the malicious ratio and manipulation factor. Throughout all malicious ratios, we can observe similar effects like in Scenario C.2. For the small multiplication factor only 30% of all evil data sets have been discarded but the precision is quite decent compared to Scenario A. For malicious ratios of 5%, 10% respectively 20% we encounter the following relative errors: 1.6% / 6.2% / 17.5% with DOMiNo enabled and 11.4% / 23.8% / 52.5% with DOMiNo disabled. With a higher manipulation factor the full potential is unfolded and we hit a relative error of -0.9%, 0.8% and 9.4% in Scenario C.3 with DOMiNo enabled compared to 118.6% / 334% / 997.5% in Scenario A without protection. As the number of evil bits is close to zero, the discard Equation 6 can be applied, which supplies 473 for 5%, 895 for 10% and 2392 for 20%.

We conclude that the influence of malicious manipulation attacks is limited to the bounds of the convex hull around the median. Thus, small changes can influence the precision, but the degree of influence is limited. Furthermore greater manipulation factors are filtered out efficiently by DOMiNo.

D. Scenario D: All attacks / DOMiNo activated

In our last scenario, we apply the parameters found till Scenario C.3 (bF=96 / Z-score threshold=3.5) and see how the system is performing with all attacks enabled. Here, we want to figure out how the malicious node ratio, the discard policy and the manipulation factor influence DOMiNo's efficiency. Preliminary we have to state, if $p\%$ of all nodes behave maliciously then these nodes are equally distributed on the active attacks, which are nine (see Section III-B).

When applying the discard_all policy, DOMiNo clears more evil bits as it significantly discards more data sets. Furthermore, the relative error for the global view is smaller or equals the discard_safe policy's one. Increasing the malicious node ratio parameter while fixing the manipulation factor to two seems to have not a great impact for the discard_all policy. In numbers we got -2.9%, 3.5% and 1.3% for discard_all and 2%, 4.1% and 8.4% for discard_safe. For the manipulation factor of ten we encounter a peak when simulating with 20% of malicious nodes with a relative error of 63.2% for discard_all and inf (=infinity) for discard_safe while the other ratios perform much better. Here we encounter -3.2% vs. 22.5% for 5% and -3.8% vs. 41.8% for 10% of malicious nodes for discard_all respectively discard_safe. The infinite value is most probably due to the AT_{Root} and AT_{Random} attacks. In the first attack, $\frac{5000*0.2}{9} = 111$ nodes send manipulated data to the root node which has bF=96 honest child nodes, which supply data. This imbalance causes the system to crash if there also exist a loop in the tree due to the second attack. Owed to our requirement, that children must be located in a certain ID space, we are not able to detect all structural attacks. Keep in mind that loops are still possible due to the goal of having a passive rating, which does not allow to verify the potential child node which pushes monitoring data. Lastly, a comparison to an unprotected system is not useful as the relative error permanently increases, due to no mechanisms to detect loops in the monitoring tree.

E. Discussion

In this section we want to discuss and emphasize a few characteristics of DOMiNo.

In our proposed system attackers can only inject data which fit into their parents' continuously calculated bounds, as anything beyond these bounds is dropped by default. Please note that malicious nodes' data can be also within the convex hull and vice versa, deviated honest nodes' data may be considered as outliers and may be emitted. But as long as the majority of the nodes respectively each parents' nodes report similar data, our convex hull ensures that outlier will be filtered out.

Also, it is quite difficult to manipulate data which is still plausible and accepted by the parent, as nodes can not obtain their parent's data base to calculate the current convex hull. So single attackers must guess their parent's data base or control respectively cooperate with all other children of a particular parent. Though, assuming attackers are able to calculate their parent's convex hull, the influence on the global view greatly depends on the number of nodes performing this kind of attack. So it is possible to inject plausible but false data, and by doing this shifting the global data to the attacker's desire, if attackers 1) are able to calculate their parents' convex hull and 2) are cooperating with other attackers evenly distributed throughout the monitoring tree. Please note, that especially the first assumption is hard to achieve.

In conclusion, our proposed system DOMiNo performs efficiently, if the majority ($> 50\%$) of the participants are honest, parent nodes can fully verify their children nodes and children are not able to calculate their parents' convex hull bounds. If parent nodes are not able to verify their children nodes, DOMiNo's reliability decreases when the number of cooperating malicious nodes exceeds the branching factor of the monitoring tree, i.e. the number of potentially benign children of a parent node.

VI. Conclusion and Future Work

In this paper, we presented our security system for tree-based monitoring systems with certain requirements to their architecture. The system's main objectives are to serve the service by only passively listening to incoming events, i.e. monitoring data sets, and to minimize the relative error to the global view by keeping the costs low. We developed an attacker model including manipulation attacks, structural attacks and free-riding patterns and proposed our rating system, called DOMiNo, which includes an outlier detection based on Z-Scores, structural and other plausibility checks. We implemented it in the P2P simulator PeerfactSim.KOM [8].

We evaluated our system with 228 configurations with at least 10 seeds each in a Pastry overlay with 5000 nodes. For the attackers the configurations cover variations of the malicious node ratio, their used attack types and, for manipulation attacks, the manipulation factor. On the detecting side we alter the outlier detection's Z-Score threshold, the discard policy after positively rating a node's data set and, for the monitoring structure, the branching factor. The outcome shows that we are able to limit the manipulation attackers' influence on the global view by defining a convex hull for acceptable values. Thus, normal fluctuations are tolerated but outlier, that harm the global view, can be filtered out effectively. For small manipulation factor attacks we face a relative error between 11% to 53% in an unprotected monitoring environment versus 2% to 18% with DOMiNo enabled. For higher manipulation factor attacks the unprotected system's relative error raises to 998% while we can correct it to 9%. Enabling the whole attacker model's repertoire we are able to decrease the relative error to values between 1% to 4% from potentially infinity, caused by loops through structural attacks. If we encounter more than the tree's branching factor cooperating malicious nodes, they are potentially able to break the system. To circumvent this, we propose stricter tree structural properties, where parent nodes can verify their children nodes' positions.

In the future we will work on an approach to loosen the passive rating requirement and compare it with DOMiNo. By doing this, we also plan to investigate systems with machine learning components.

Acknowledgment

We would like to thank Sebastian Brink for his great work.

References

[1] "The Security Flag in the IPv4 Header," RFC 3514, Jun. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc3514.txt

[2] S. Axelsson, "Intrusion Detection Systems : A Survey and Taxonomy," *Computer Engineering*, 2000.

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

[4] H. Chan, A. Perrig, and D. X. Song, "Secure hierarchical in-network aggregation in sensor networks," in *Proc. of the Conference on Computer and Communications Security, Alexandria, VA, USA*, 2006.

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

[6] A. Disterhoft and K. Graffi, "Protected chords in the web: Secure p2p framework for decentralized online social networks," in *Proc. of the IEEE Int. Conference on Peer-to-Peer Computing*, 2015.

[7] K. Graffi, "Monitoring and Management of Peer-to-Peer Systems," Ph.D. dissertation, Technische Universität, Darmstadt, August 2010. [Online]. Available: http://tuprints.ulb.tu-darmstadt.de/2248/

[8] ——, "PeerfactSim.KOM: A P2P System Simulator - Experiences and Lessons Learned," in *IEEE P2P '11: Proceedings of the International Conference on Peer-to-Peer Computing*, 2011.

[9] K. Graffi, C. Groß, D. Stingl, D. Hartung, A. Kovacevic, and R. Steinmetz, "LifeSocial.KOM: A Secure and P2P-based Solution for Online Social Networks," in *Proc. of the IEEE Consumer Communications and Networking Conferences*, 2011.

[10] 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 *Proc. of the Int. Conf. on Parallel and Distributed Systems*. IEEE, 2008.

[11] K. Graffi, P. Mukherjee, B. Menges, D. Hartung, A. Kovacevic, and R. Steinmetz, "Practical security in p2p-based social networks," in *Proc. of the IEEE Conference on Local Computer Networks, LCN*, 2009.

[12] 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.

[13] C. Hayward and A. Madill, "A Survey of Outlier Detection Methodologies," *Social Science and Medicine*, 2003.

[14] IBM, "IBM Knowledge Center," http://www.ibm.com/support/knowledgecenter/SSWLVY_1.0.0/com.ibm.spss.analyticcatalyst.help/analytic_catalyst/modified_z.html, accessed: 2016-06-10.

[15] B. Iglewicz and D. Hoaglin, *How to Detect and Handle Outliers*, ser. ASQC basic references in quality control. ASQC Quality Press, 1993.

[16] M. Jelasity, A. Montresor, and Ö. Babaoglu, "Gossip-based aggregation in large dynamic networks," *ACM Trans. Comput. Syst.*, 2005.

[17] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The Eigentrust algorithm for reputation management in P2P networks," in *Proceedings of the Twelfth International World Wide Web Conference, WWW 2003, Budapest, Hungary, May 20-24, 2003*, 2003, pp. 640–651.

[18] 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.

[19] S. Marti and H. Garcia-Molina, "Taxonomy of trust: Categorizing P2P reputation systems," *Computer Networks*, vol. 50, no. 4, pp. 472–484, 2006.

[20] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Proc. of the Annual Int. Conference on Mobile Computing and Networking*, 2000.

[21] P. S. Mogre, K. Graffi, M. Hollick, and R. Steinmetz, "AntSec, WatchAnt, and AntRep: Innovative Security Mechanisms for Wireless Mesh Networks," in *Proc. of Int. Conf. on Local Comp. Networks*, 2007.

[22] A. G. Prieto and R. Stadler, "Distributed Real-Time Monitoring with Accuracy Objectives," in *Proc. of the Int. Net.g Conf. 2006 - Netw. Techn., Services, and Protocols; Performance of Comp. and Comm. Net.; Mobile and Wireless Communications Systems*, 2006.

[23] V. Rapp and K. Graffi, "Continuous Gossip-based Aggregation through Dynamic Information Aging," in *Proc. of the Int. Conference on Computer Communications and Networks*, 2013.

[24] M. Rounds and N. Pendgraft, "Diversity in network attacker motivation: A literature review," *Proc. of the IEEE Int. Conference on Computational Science and Engineering*, 2009.

[25] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," in *IFIP/ACM Middleware '01: Proc. of the Int. Conference on Distributed Systems Platforms*, 2001.

[26] University of Oldenburg, "Cohda project," https://www.uni-oldenburg.de/en/computingscience/ui/research/topics/cohda, accessed: 2016-06-10.

[27] F. Wuhib, M. Dam, and R. Stadler, "Decentralized Detection of Global Threshold Crossings using Aggregation Trees," *Comp. Networks*, 2008.

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

[29] Y. Yang, X. Wang, S. Zhu, and G. Cao, "SDAP: A secure hop-by-hop data aggregation protocol for sensor networks," *ACM Trans. Inf. Syst. Secur.*, 2008.

[30] 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.