Applying Certificate-Based Routing to a Kademlia-Based Distributed Hash Table

Michael Kohnen, Jan Gerbecks, Erwin P. Rathgeb
University of Duisburg-Essen
Computer Networking Technology Group
Essen, Germany
{Michael.Kohnen, Erwin.Rathgeb}@iem.uni-due.de, Jan.Gerbecks@stud.uni-due.de

Abstract–Most Distributed Hash Table (DHT) algorithms have proven vulnerable against a multitude of attacks. Countermeasures using reputation systems to generate trust values have been developed and analyzed. These analyses mostly refer to unstructured peer-to-peer (P2P) networks. In this paper, we present our concept for applying trust values to the bootstrap, lookup, PUT and GET processes of structured P2P networks and evaluate it using the Kademlia DHT algorithm in a binary trust environment created by certificates.

Keywords–DHT; Security; Kademlia; Trust; Reputation; Certificates

I. INTRODUCTION

Research has proven that Distributed Hash Table (DHT) algorithms are vulnerable to different kinds of attacks [1]. These attacks include the Sybil Attack [2], eclipse attacks and attacks on the routing and storage mechanisms. As one possible solution against those threats, trust-based systems have been invented to improve security. A lot of the existing research about trust and reputation management in a peer-to-peer (P2P) environment focuses on unstructured networks [3] [4]. However, considering real-world implementations, the structured networks prevail: Popular P2P applications such as BitTorrent [5] and eMule [6], used by millions of users, implement the Kademlia algorithm.

We, therefore, aim to analyze the feasibility of trust and reputation mechanisms in structured P2P networks. To test the general functioning, we use a simplifying assumption of binary trust created by certificates. In the following section, we present the related work. In Section III, we explain our concept, followed by Section IV with its evaluation. Section V concludes the paper.

II. RELATED WORK

Marti and Garcia-Molino [7] categorize P2P reputation systems and divides them into the three functionalities “information gathering”, “scoring and rating” and “response”, each having several sub functions. Furthermore, the authors define factors influencing reputation systems and discusses them.

Gomez Marmol and Martinez Perez [8] offer an overview of the current state of P2P reputation systems. EigenTrust [3] is one of the popular ones. It uses a rating system similar to eBay’s: A node can receive either a positive or a negative rating after a transaction. The EigenTrust algorithm then defines how the ratings from different nodes can be combined and normalized.

EigenTrust and the other algorithms presented in [8] either have been tested using unstructured P2P networks or mention structured networks only for storing the trust information. They do not analyze the specific impact of using trust information for routing and storing in structured networks.

Therefore, we aim to analyze the consequences of using trust information in structured P2P networks during the joining, routing, storing and retrieving processes. According to [9], these processes will be referred to as bootstrap, lookup, PUT and GET process, respectively. In this paper, we present a basic concept of using trust values to enhance the security of a DHT algorithm.

In the following, we discuss the trust values’ consequences for bootstrapping and performing lookup, PUT and GET actions. Afterwards, we evaluate our concept using the Kademlia DHT algorithm [10].

III. OUR CONCEPT

We seek an approach that enables a node to determine the authenticity of a result on its own. A node shall be able to decide for itself whether it regards an action as successful. It shall also be able to abort or ignore an action if it does not trust the result.

To achieve this, trust values are used: We have each node assign trust values to each other node it encounters. Then, we define a minimum trust value another node needs to have so that a node uses it for its actions in the network [7]. As a consequence, a node is able to determine whether the result of an action (bootstrap, lookup, GET, PUT) is valid. If it does not find enough trustworthy nodes, it cancels its action in order to protect itself from invalid results.

A. Functioning

Once a node’s trust value is known, it is used to determine whether the node should be used. We propose to use a single minimum trust value threshold defining whether another node is used for outgoing requests of all kinds, as a node that does not answer GET requests correctly, for example, should not be used for other purposes.

Trust values are not assigned globally, but individually by each node. Possible reasons for this are, e.g., a result node possessing a certificate issued by another CA or the requesting node using a lower minimum trust value threshold than the responding node. This individual assignment of trust values leads to their local storage on each node and therefore an extension of the routing table. From this, it follows that…

- … incoming requests shall always be answered in a correct way, regardless of the trust status of the requesting node, and …
- … that the routing table of a node shall not only contain nodes it trusts, but also untrusted nodes.

If nodes would only answer request coming from trusted nodes, a network partitioning could result. The same applies to the routing table: If a node can only answer with next hops that it trusts, the requesting node may not be informed about existing nodes itself may trust, but the responding node does not.

When a node joins the network, it needs to perform a bootstrap procedure to obtain information about other nodes to fill its routing table. A node must send its bootstrap requests only to trusted nodes. As mentioned before, every node has its own view of trust, so the responses may contain all kinds of nodes.

When a node performs a node lookup for a GET or PUT action, it must only query nodes it trusts. During a lookup, newly encountered nodes must be evaluated and the node must determine if they are trustworthy before they are used. At the end of the lookup, the candidate nodes for the GET or PUT action must also be evaluated (if not done so already) so that the action is performed using trusted nodes only.

B. Consequences

Our concept enables a node to determine the trustworthiness of e.g. the retrieved results of a lookup. Every node can individually choose a minimum trust value threshold and decide on its own whether it regards an action or a result as valid. The nodes do not need to refer to general assumptions such as the amount of malicious nodes in the network to evaluate the correctness of actions and results.

As a drawback, our concept decreases the number of nodes that can be used for a node's actions. A certain minimum amount of trusted nodes is therefore essential. Furthermore, the availability of content items cannot be as easily guaranteed as in normal DHT networks: The assignment of content items to nodes is not unique any longer, but it differs due to the different trust values nodes assign to other nodes. It is therefore possible that content inside the network cannot be found by a node despite its existence. Possible reasons are the following:

- The content is only stored on nodes the requesting node does not trust.
- There are not enough trusted nodes on the route to the (trusted) nodes storing the content so that the lookup terminates prematurely.

IV. EVALUATION

In order to analyze to which extent these effects influence the ability of the nodes to use the network, we evaluate our concept in a Kademlia-based DHT using a simulated network of 1,000 nodes.

A. Assumptions

This paper is intended as a proof of concept to show that trust values can improve secure bootstrapping, lookup, PUT and GET in structured P2P networks. For this proof of concept, we assume the following:

- Nodes are either fully trusted or untrusted: We assume the existence of a certification authority. Nodes that possess a certificate are fully trusted, other nodes are untrusted.
- Nodes possessing a certificate are never malicious: We assume that the algorithm generating the trust values (here: certification) determines the trustworthiness correctly.

The first assumption requires a central entity which does not follow the peer-to-peer principle. The second assumption does not necessarily hold for real networks, as also nodes that are regarded as trustworthy may act maliciously. However, if the DHT would not work under these "perfect" conditions, it would not work in reality either.

The simulation scenario differentiates between nodes with and without a certificate: Nodes without a certificate ("No Cert" nodes) use all other nodes, whereas nodes with a certificate ("Cert" nodes) use only other nodes with a certificate for their actions. In this "binary trust" environment, we are able to demonstrate the worst case for the application of our concept. For small fractions of trusted nodes, the absolute number of them is below 100, which is rather low. However, we will demonstrate that even this small number of nodes is able to operate.

B. Choice of Kademlia

We choose the Kademlia algorithm because its routing process does not set hard restrictions on the next hop choice: Kademlia uses the XOR operation to calculate the distance between two IDs. During the routing process, a node uses a list of potential next hop nodes that is ordered by XOR distance with the closest nodes at the top of the list. When the first k nodes on the list do no longer change and have been queried for closer nodes, the lookup terminates and the action is performed on those nodes. This action can either be a PUT action or a GET action. During a PUT action, a node stores a content item on the configured amount of nodes. Using a GET action, a node tries to retrieve a content item.

C. Simulation Environment

Our simulation scenario consists of one large network in which only a subset of the nodes uses trust-based routing. This way, we can compare the performance of nodes applying and not applying our concept. In our simulation, malicious nodes perform a storage attack called "invalid data attack", which means they deliver randomly altered data if asked for a content item. We choose this attack type, because the Kademlia algorithm is rather robust against routing attacks: It has few restrictions regarding the choice of the next hop, so if a node propagates faulty routing information, the information might well be overridden by the responses of other nodes, only

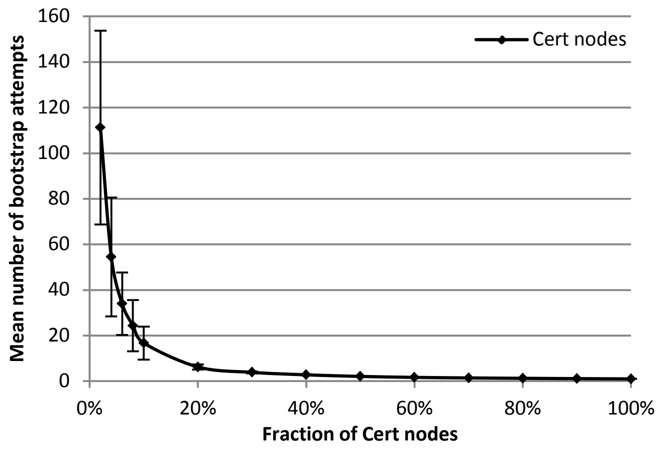

Figure 1. Mean number of Cert nodes' bootstrap attempts

causing higher delays. So storage attacks generally pose a greater threat and therefore show the influence of our concept more clearly.

For simulation, we use the OverSim [11] framework, which bases on OMNeT++ [12]. We vary the fraction of nodes possessing a certificate from 0% to 100% in steps of 10% and conduct additional simulations for fractions of 2% to 8% in steps of 2% for a detailed analysis. For each parameter combination, we perform 30 simulation runs using different seeds for the random number generator.

D. Results

All figures below show the arithmetic mean and the standard deviation of the results.

1) Bootstrap

At the beginning of each simulation run, the nodes perform bootstrap procedures. One node per second is inserted into the network and tries to bootstrap using a randomly chosen existing node. The first node is always a Cert node. As No Cert nodes use every kind of node to bootstrap, their first attempt to bootstrap will always succeed. Cert nodes use only other Cert nodes to bootstrap; their attempts to bootstrap may therefore fail if the randomly selected target node of the bootstrap request is not a Cert node. If the attempt fails, they pause for ten seconds and try to bootstrap again until they succeed.

Figure 1 shows that for very low fractions of Cert nodes, the mean number of bootstrap attempts of the Cert nodes is rather high. However, this value quickly decreases when the fraction of Cert nodes increases.

2) PUT and GET requests and malicious nodes

In order to demonstrate our concept's resilience against attacks, we introduce malicious nodes into the simulation network: We vary the fraction of No Cert nodes which are malicious from 0% to 100% in steps of 10%. As we assume that the trust values are correct, nodes with certificate cannot be malicious.

The measurement phase begins when all nodes have attempted to bootstrap at least once (in our case after 1,000 seconds). During the simulation, the nodes publish content with random identifiers and try to retrieve it. The Kademlia algorithm makes use of replication per definition. Baumgart and Mies argue in their S/Kademlia paper [13] that smaller replication factors than the original Kademlia's 20 are sufficient. So, in our simulation, we use S/Kademlia's default values: Content is stored on 4 nodes during a PUT action and a GET action tries to obtain the content from 4 nodes as well. The GET action is regarded as successful if at least 50% of the responses are identical. The nodes only try to retrieve content IDs that have been published previously.

Content can be published either by Cert or by No Cert nodes. The same also applies to retrieving content, so there are four possible combinations of PUT and GET actions: Cert PUT & Cert GET, Cert PUT & No Cert GET, No Cert PUT & Cert GET and No Cert PUT & No Cert GET.

The results show that our concept allows the Cert nodes to retrieve content published by other Cert nodes regardless of the fraction of malicious nodes. Figure 2 shows the success ratio of GET requests of Cert and No Cert nodes for content published by their respective kinds in dependence of the fraction of malicious nodes. In the absence of malicious nodes, the success ratio for retrieving content by No Cert nodes that was also published by No Cert nodes was the same as for the Cert/Cert

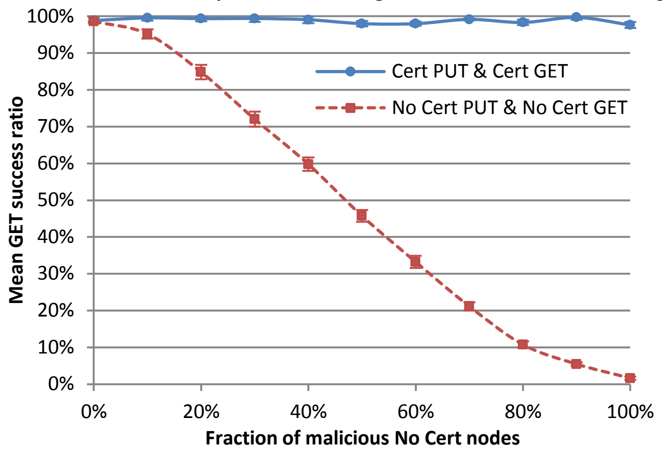

Figure 2. Success ratios of GET requests (10% Cert nodes)

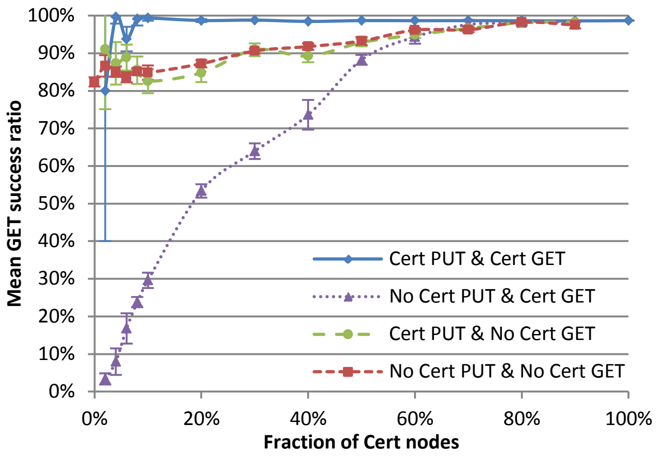

Figure 3. Success ratios of GET requests (20% malicious No Cert nodes)

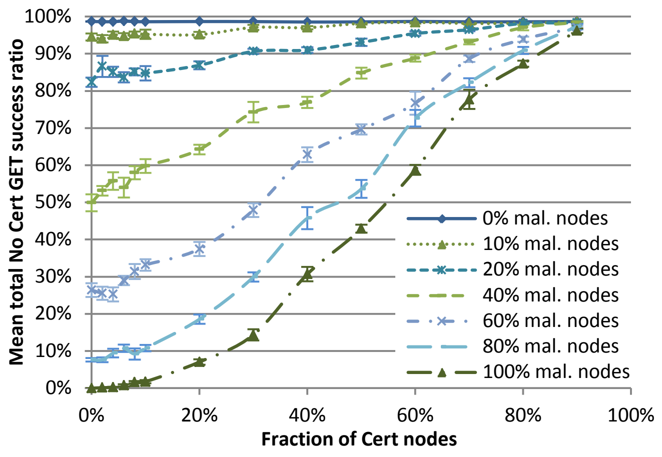

Figure 4. Total No Cert GET success ratio

case. The No Cert/No Cert success ratio decreases if the fraction of malicious nodes increases, whereas the Cert/Cert success ratio stays close to 100%. This demonstrates that the Cert nodes obtain a better performance than the other nodes: They are able to retrieve the desired content successfully despite the presence of malicious nodes. Furthermore, this result is independent of the fraction of malicious nodes.

Figure 3 shows the GET success ratios for all four possible combinations of PUT and GET requests. As an example, the fraction of malicious No Cert nodes in Figure 3 is 20%. The results show that even for small fractions of Cert nodes (2% to 4%, resulting in only 20 respectively 40 nodes), these are able to retrieve content published by other Cert nodes. Content stored by No Cert nodes can be retrieved increasingly successful by Cert nodes if the fraction of Cert nodes increases. However, the success ratio for this case is always higher than the fraction of Cert nodes: There are four replicas of each content item. If only one of them is stored on a Cert node, it can be found by another Cert node. The probability that at least one of the four nodes that store the item is a Cert node can be computed by using the following hypergeometric probability distribution formula:

$$1-\frac{\binom{\mathit{Total\ number\ of\ nodes} - \mathit{Number\ of\ Cert\ nodes}}{4}}{\binom{\mathit{Total\ number\ of\ nodes}}{4}}$$

The correlation coefficient of the simulation results and the theoretical values is 0.995, which shows that the results are close to the theory.

In Figure 3, it is also visible that the No Cert nodes benefit from the presence of the Cert nodes, as the success ratio of No Cert GET requests increases when the fraction of Cert nodes increases.

Figure 4 reveals this tendency more clearly: It shows the total GET success ratio of No Cert nodes (for content that is stored on both types of nodes) for different fractions of malicious No Cert nodes. The No Cert nodes benefit from the presence of Cert nodes: The higher the fraction of Cert nodes is, the better the success ratios of the No Cert nodes are as well. This can also be seen in the results in Figure 2: Even with 100% malicious nodes, the success ratio of the No Cert/No Cert case is not 0%: No Cert nodes can still retrieve content items correctly if at least 50% of the responses originate from Cert nodes.

V. Conclusion and Outlook

We have presented a concept that uses trust values to enhance the security of lookups and PUT and GET actions in a structured P2P network. Nodes shall react to incoming requests as usual and use only trusted nodes when performing their own actions. We have shown that this concept works as intended using a Kademlia-based DHT: Despite the presence of malicious nodes, nodes applying our concept are able to continue operation normally as if no malicious nodes were present.

Further research is required to investigate the application of our concept to other DHT algorithms: Other algorithms have stricter requirements regarding the placement of other nodes into a routing table, for example. This may require an extension of our concept.

Our simplifying assumptions regarding certification and maliciousness do not hold in reality, so further research is required to analyze the effects of using a reputation system to generate the trust values. These values are typically not binary, so research regarding the minimum trust value threshold is also required.

Our simulations did not include message exchanges for the determination of the trust values and did not account for additional time required to validate the certificate. Further analyses of performance issues are therefore required.

VI. References

[1] Guido Urdaneta, Guillaume Pierre, and Maarten van Steen, "A Survey of DHT Security Techniques," *ACM Computing Surveys*, vol. 43, no. 2, pp. 8:1-8:49, Jan. 2011.

[2] John R. Douceur, "The Sybil Attack," in *IPTPS '02: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, Cambridge, MA, USA, 2002, pp. 251-260.

[3] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina, "The Eigentrust Algorithm for Reputation Management in P2P Networks," in *Proceedings of the 12th International Conference on World Wide Web*, Budapest, Hungary, 2003, pp. 640-651.

[4] Loubna Mekouar, *Reputation-based Trust Management in Peer-to-Peer File Sharing Systems*. Waterloo, Ontario, Canada: University of Waterloo, 2010.

[5] (2011, Sep.) BitTorrent. [Online]. http://www.bittorrent.com/

[6] (2011, Sep.) eMule Project. [Online]. http://www.emule-project.net/

[7] Sergio Marti and Hector Garcia-Molino, "Taxonomy of Trust: Categorizing P2P Reputation Systems," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 50, no. 4, pp. 472-484, Mar. 2006.

[8] Felix Gomez Marmol and Gregorio Martinez Perez, "State of the Art in Trust and Reputation Models in P2P Networks," in *Handbook of Peer-to-Peer Networking*, Xuemin Shen et al., Eds.: Springer, 2010, pp. 761-784.

[9] Frank Dabek, Ben Zhao, Peter Druschel, John Kubiatowicz, and Ion Stoica, "Towards a Common API for Structured Peer-to-Peer Overlays," in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, Berkeley, CA, USA, 2003, pp. 33-44.

[10] Petar Maymounkov and David Mazières, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," in *IPTPS: Revised Papers from the First International Workshop on Peer-to-Peer Systems*,

Cambridge, MA, USA, 2002, pp. 53-65.

[11] Ingmar Baumgart, Bernhard Heep, and Stephan Krause, "OverSim: A Flexible Overlay Network Simulation Framework," in *Proceedings of 10th IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007*, Anchorage, AK, USA, 2007, pp. 79-84.

[12] (2011, Sep.) OMNeT++. [Online]. http://www.omnetpp.org/

[13] Ingmar Baumgart and Sebastian Mies, "S/Kademlia: A practicable approach towards secure key-based routing," in *Proceedings of the 13th International Conference on Parallel and Distributed Systems*, Hsinchu, Taiwan, 2007, pp. 1-8.