Compliance Management for P2P Systems

Alexander Schneider
Institute for Computer Science
Univeristy Düsseldorf
Düsseldorf, Germany
Email: aschneider@cs.uni-duesseldorf.de

Martin Mauve
Institute for Computer Science
Univeristy Düsseldorf
Düsseldorf, Germany
Email: mauve@cs.uni-duesseldorf.de

***Abstract*—Compliance management for peer-to-peer networks describes a process ensuring that content inside the network is distributed and stored in a way that does not violate user defined preferences. Several use cases, ranging from filesharing networks to distributed computing and content delivery networks, can be enhanced with compliance management. To our knowledge there are no existing peer-to-peer architectures which allow for compliance management. In this paper we propose an architecture, which utilizes policy-based routing and storage as well as a categorization of content in order to provide compliance management. We implement a prototype and evaluate it through simulations to show that compliance management in peer-to-peer networks is actually feasible.**

I. Introduction

Peer-to-Peer (P2P) networks enable users to exchange a wide variety of information. Participation in those networks typically requires that each user is willing to forward and store arbitrary data. This may cause legal problems or, in a somewhat less severe case, raise ethical concerns. To solve this problem we introduce the idea of peer-to-peer networks with integrated compliance management. In those networks users are only involved in managing data that they explicitly agree to handle.

One example application is the distribution and storage of arbitrary data, such as music or video files, as it is common in many current peer-to-peer networks. With compliance management each participant would specify the content she is willing to forward and store. Hence she could participate and support the network without risk of breaking the law or handling unwanted content. Another example is a social network, managed and maintained by a single company which uses a peer-to-peer network for distributed storage. The nodes of the peer-to-peer network could be located in distinct countries each with their own jurisdiction and customs. The company then needs to make sure that each node in the peer-to-peer network stores and manages only data that is legal and acceptable in the country the peer is located in. This allows the social network to maintain content that is acceptable in any country it operates in, instead of restricting content to what all countries deem as acceptable.

In order to specify which content is acceptable for a given node, we propose that each chunk of data handled by the peer-to-peer network is assigned one or more tags describing its content. We acknowledge that assigning these tags in a reliable and trustworthy way is a significant challenge that we do not address at length in this paper. We will, however, provide reasoning why we believe this to be a solvable problem. Given a chunk of data that is described by a list of tags, we seek to answer the following question: is it possible to store and forward it in such a way, that none of the individual preferences of the users are violated? In this context the preferences of a user are given as a list of tags, that she is willing to forward and store, while the network organizes itself in a way that incorporates the participants preferences. In this paper we answer this question by adapting Kademlia to use compliance management. We show that compliance management in peer-to-peer networks is, in fact, feasible and we provide insights on the impact that the preferences of the participants have on the performance of the network.

The remainder of this paper is structured as follows. Section II briefly reviews related work. Following this, we introduce a modified version of Kademlia - Comademlia - that is able to provide compliance management in section III. In section IV we evaluate the performance of Comademlia for several parameters and enhancements. Lastly, section V concludes the paper and gives an outlook on future work.

II. Related Work

There are many peer-to-peer systems that can serve as distributed storage. Approaches such as Kadmelia [5], Chord [11] or Pastry [7] serve this purpose very well. However none of them allow for compliance management by the participating nodes. There are also filesystem-like destributed storage solutions like Oceanstore [4] and Ivy [6]. They, however, also do not provide compliance management.

Da Silva et al. [1] published work, that outlines policy based access in P2P grids. In contrast to our architecture it only regulates the access to data based on policies, but does not use policies to distribute and store the data. To our best knowledge there are currently no systems that use content policies to determine storage and forwarding rules for individual nodes inside a peer-to-peer network.

There exists work regarding peer-to-peer networks, which utilize content tags, e.g. [2], [3]. The existing systems however define the possibility to share data annotated with tags and the ability to calculate and maintain feature-vectors in the tagging environment, while missing out on a cryptographic link between content and tag. Other work by Smetters and Jacobson [9] introduces the idea to cryptographically link arbitrary names to content, which is discussed later in the paper.

III. Comademlia

Compliance management requires that a node can somehow judge the content contained in a chunk of data. We believe that this can be done, e.g. by annotating each chunk of data with tags. The process of assigning those tags is not part of the work we describe here. However, we briefly reason, why this is a solvable problem. The focus of our work, then, is the actual storage and routing, under the constraint that each node only participates in tasks that do not violate its local policy.

A. Tagging

We assume that all chunks of content are associated with tags, that characterize the content. Tags either describe the content, e.g. “violence” or “explicit speech”, or they provide meta information such as “legal for all audiences in Germany”. They are assigned by trusted parties, like the publisher of the content, or by means of collective decisions. The latter is very similar to what is regularly done in order to realize quality control at popular web-sites. Each node in the peer-to-peer network specifies its policy by maintaining a list of tags and announcing it to its neighbors. A node will not participate in routing and storing content, with tags that are not contained in its policy. Of course the link between a tag and some data has to be trustworthy, which can be achieved by e.g. cryptographic signatures. For example, it was showed in [9] that it is possible to establish cryptographic links between names and content.

A real-world implementation needs mechanisms, to report and remove tags, since data can be falsely tagged, either by accident or by malice. In our prototype we did not include such a mechanism, but would like to sketch a possible solution. The general idea is to allow reporting of assumed false tags only if the reporting node is trustworthy. To find out whether a node is trustworthy, we could asses how cooperative the node is with the desired network operations. The more a node forwards and stores content, the more it is contributing to the health of the network, so either the nodes is an honest participant or it contributes for the purpose of being able to issue false reports. Ultimately if a node wants to report tags as incorrect, it has to “expend” some of the accumulated cooperation. The idea is to use cooperative actions as a resource in a proof-of-work-like system. We assume hereby that the threshold for a report can be adjusted to a level, where nodes with malicious and false reports add a significant contribution to the network, that balances out their wrongdoing. Individual nodes can then configure the amount of valid reports needed until they stop trusting the tags of a certain content publisher. The reports can be organized in different systems, e.g. in an overlay or using a blockchain. Developing such a system is out of scope for this paper, but constitutes our main focus for future work.

A good and efficient system for the assignment of tags - in particular in form of collective decisions - is certainly an interesting research challenge. However, given that content classification is regularly done in other contexts both in a centralized fashion and as collective decisions, leads us to the assumption that developing such a system is generally feasible. In the remainder of the paper we therefore focus on the networking aspects of a peer-to-peer system that provides compliance with the preferences of the individual users.

B. Distribution and Storage

Our goal is to design a peer-to-peer-based content storage network prototype that provides permanence, high availability of data and, most importantly, enforces compliance with the policies of each individual participant. As a starting point we used Kademlia and modified it to include compliance management.

1) Kademlia: In a Kademlia [5] network nodes are assigned a random identifier inside an ID space. Data is also assigned a value from the same ID space, by hashing the data. Inside this ID space XOR is used as a metric to determine the distance between two IDs, which is important for routing decisions.

Every Kademlia node uses a tiered routing table, with ”buckets”. A bucket contains a limited number of nodes which share a certain ID prefix. All buckets together cover the entire ID space without overlap or gaps. The buckets are organized in a way, that facilitates more complete knowledge of nodes in the immediate vicinity. Vicinity in this case is defined through a low XOR-distance.

Data is always redundantly stored at a set of nodes whose IDs are closest to the ID of the data. Thus, storing and retrieving data is about finding one or multiple nodes nearest to a certain ID. To find nodes, Kademlia uses an iterative lookup process. A node starts the lookup by querying the nearest known nodes to the desired ID. Those nodes then return their nearest known nodes, in respect to the target ID, to the requester. The process is repeated until no new nodes are returned and a sufficient subset of nodes has been queried or the desired target, or data, has been found. Since every node has extensive knowledge of its surroundings the process always converges with time.

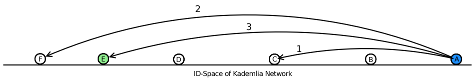

Fig. 1. The Kademlia lookup process illustrated.

An example for a lookup can be found in figure 1. Node A is searching for content stored at node E. A queries the nearest known nodes to the desired contents ID, which include C. Node C has not stored the desired data and answers in turn with a list of the nearest known nodes, including F, which gets queried next and answers with its nearest known nodes, which include E. Finally A queries E, which returns the desired data. This process is parallelized actually; usually several nodes are queried at once. A more detailed description of Kademlia can be found in the paper by Maymounkov and Mazieres [5].

2) Comademlia: To incorporate compliance management and to uphold the nodes policies at all times several changes had to be made to the routing table and the lookup procedure. One key problem when integrating compliance management into Kademlia routing is the possibility to "eclipse" a node from content tagged with certain tags. For example, if a network has three tags α, β, δ it is possible that the routing table fills, randomly, with nodes that support only α and β. This node now can not find any content tagged with δ, because it has no knowledge of contacts that can handle its desired data. To prevent such an eclipse, Comademlia uses multiple routing tables, where each node maintains one dedicated routing table per tag in its policy. Newly encountered nodes are placed in all routing tables according to the tags in their policies. Furthermore, a node does not maintain a routing table for tags that are not in its policy. This way each routing table represents a unique network view, containing only nodes that are willing to participate for a certain tag. The total size of all routing tables is not a problem for two reasons. Firstly, each table only holds a small subsets of nodes. Secondly, the tables only contain node references that are shared between multiple routing tables of the same node. Furthermore, because a node is most likely present in multiple routing tables, they can be compressed quite efficiently, if needed.

If data is only categorized by a single tag, the lookup works exactly the same as in Kademlia, using the corresponding routing table. However, consider a case where a node has routing tables for tags α containing nodes $\{A, D, F, G\}$ and β containing $\{A, B, D, E\}$ and wants to start a lookup for data that is tagged with both α and β. Respecting the policies of all nodes in the routing tables requires that only $\{A, D\}$ can be used for requesting this content.

More formally, the routing works as follows: let M be the node searching for data, C be the desired data, and C_{tags} a set of tags, which describe the data. Furthermore, let R^M be the set of all routing tables of node M and R_t^M be the routing table for tag t in M. M now first calculates an intersection of routing tables such that:

$$R_{intersection} = \cap_{e \in C_{tags}} R_e^M$$

$R_{intersection}$ can then be used to continue with the Kademlia-lookup, since $R_{intersection}$ only contains nodes that will accept the query regarding this content. The nodes queried in the process then construct the same intersection on their routing tables to determine their list of nearest known nodes in relation to the desired data.

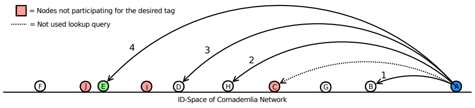
= Nodes not participating for the desired tag
= Not used lookup query
4
3
2
1
F
J
E
I
D
H
C
G
B
A
ID-Space of Comademlia Network

Fig. 2. The Comademlia lookup process for a certain combination of tags.

An example for the lookup process is shown in figure 2. In this example node A computes its $R_{intersection}$, which does not contain C but B. Although C would be nearer to the desired ID, C does not wish to participate in routing for at least one required tag and is thus not part of A's routing table for this specific combination of tags. Instead A queries B with a lookup that contains the desired ID and the accompanying tags of the content. Node B, naturally, only returns nodes which accept all tags for the desired data as well. The lookup process then continues iteratively as is usual with standard Kademlia.

In order to route queries and data each node needs knowledge about the policies of their contacts. A node communicates its current policy to other nodes by adding the policy as often as possible, optimally always, to other messages being sent. Any node can also just query any other node for its policy if needed, e.g. if it is suspected that some policy information might be stale. If the maximum number of tags is a network parameter, the policies can be represented by a bitstring, where every tag has a certain position and can just be set to one or zero, to indicate if a policy accepts the tag or not, respectively.

IV. Evaluation

To test the viability of the prototype we conducted several simulations. In the following, we describe the setup, execution and evaluation of the results in detail.

A. Setup

To simulate our architecture we used the event-based network simulator PeerfactSIM.KOM [10]. We implemented Comademlia as an application for the Kademlia overlay for PeerfactSIM.KOM. All changes to the original Kademlia network, that were outlined in the previous section, were implemented. We also implemented the Kademlia overlay by abiding closely to the original Kademlia paper [5].

B. Simulation Design

We used different simulations to evaluate the behavior and performance of Comademlia. We configured the Comademlia network to distinguish between 50 different tags, since we used real world data from the top 50 tags on the Q&A website stackoverflow.com, which had still significant popularity differences, to model a popularity distribution. Furthermore, all simulations were conducted without churn, since the focus of this work is on the impact that compliance management has on the performance of a peer-to-peer network. Adding churn should not change the relative performance of a peer-to-peer network with compliance management compared to one without compliance management.

For our main simulation we simulated 256 nodes, which are organized in 50 ”groups”, which aggregate nodes that have similar policies. Every group was assigned a uniformly distributed number of nodes between one and 10. We also created 50 different pieces of data which were to be distributed and looked up inside the Comademlia network during the simulation. To configure the main simulation as closely to a real scenario as possible we used external data to model the policy and tag distributions. Tag distribution in a real application will follow some kind of popularity model, with the most popular tag used quite often and the least popular tag used very sparingly. To get real life data, we queried the amount of tags each question on stackoverflow.com gets assigned and how the tags are distributed overall. We found that the number of tags per question is Gaussian distributed with a mean around three. Minimum and maximum number of tags where one and five respectively, since those are a hard cap on stackoverflow.com.

For the overall popularity distribution we gathered the total amount of usages for the top 50 tags on stackoverflow.com. The numbers can be found in figure 3. We used the inverse of the same overall popularity distribution to determine the policies of the node groups. This is sensible because typically very popular content will be accepted by the majority of nodes, while unpopular content probably will be rejected by a large part. The nodes were configured to accept a random number of tags, which was Gaussian distributed with a mean of 40 and a standard deviation of three.

We ran the main simulation ten times with different seeds for 24 hours of simulated time. For every run-through the nodes first joined the network without conducting any other actions, besides determining their policy. After all nodes joined the network, a random node with matching policy was being assigned as the owner of one of the data-pieces. This node tried to store the data inside the Comademlia network. The procedure was repeated until all data-pieces were assigned an owner once. Please note that the owner of the data did not store the data itself, unless its ID was close enough according to the storage algorithm. Following the storage of the data, the retrieval phase starts performing one action per simulated minute and continues until the simulation is finished. During every retrieval phase action, nodes from a randomly selected node group, whose policy allows for it, try to retrieve a random, existing piece of data from the network.

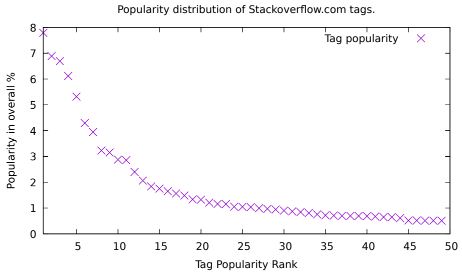

Fig. 3. Popularity tag distribution according to stackoverflow.com tags.

C. Evaluation Results

In this section we present and discuss the results from the main simulation and some follow-up simulations, which we conducted to answer questions brought up by the initial results. Further results can be found at a dedicated web-page [8] online.

1) Accessibility: One of the measured metrics was the accessibility, meaning the number of lookup requests that were successful when searching for existing data. For nine out of the ten seeds, simulated during the main simulation, the accessibility was a full 100%, meaning every node that sent a lookup request for some existing value received a valid response with the desired data. In the remaining case 6842 out of 7026 (97,4%) sent lookups were successful. We investigated what lead to the non-perfect retention rates in some edge cases, and can conclude that it is caused by a network partition for certain tags.

The network partition is a residue effect from the Kademlia underlay and how the system builds and maintains its routing tables. Since the nodes that are added initially to a routing table are dependent on the bootstrap contact, separate networks can sometimes be formed for certain tags. Solving this partition problem could possibly be done by exploiting node lookups, that are not possible in the partitioned view, but can be made through contacts in other routing tables. After a partition is detected, the node can try to start node lookups in an area of the network where the nodes knowledge is not extensive enough. Since the node lookups are not constrained by policies, more nodes can be used for the lookup and potentially help resolve the partition.

Carrying on, we were interested in whether the accessibility rates depend on the number of accepted tags per node. To test the effect of node preferences on accessibility, we started

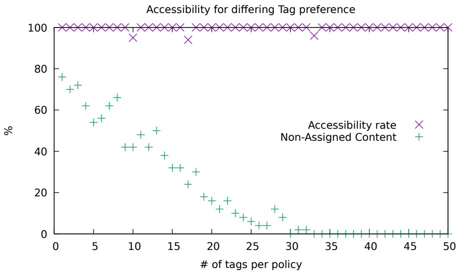

Fig. 4. The accessibility rates and percentage of content that could not be distributed inside the network due to no matching nodes for a series of simulations varying the number of tags accepted by the nodes.

a series of additional simulations, which were similar to the main simulation. In every simulation of the series the nodes policies had a fixed number of accepted tags. Starting from one and being incremented with every simulation. Additionally, we only simulated five instead of 24 hours, which was sufficient to compare accessibility. The results can be found in figure 4. As shown the accessibility is generally not influenced by the number of tags that are accepted by the participating nodes. However, there is data generated by the simulation, according to the same rules of the main simulation, that can not be stored inside the network, due to the fact that there is not a single node whose policy matches the tags on the data. For example if every node only accepts one tag, almost 80% of all generated data can not be placed inside the network. This is not very surprising, since the data we generate has on average three tags assigned to it. This further means that, depending on the distribution of tags on content, there are certain thresholds, which show how many tags have to be accepted by nodes on average to guarantee that the network can handle most data. For example, to be able to store about half of all possible data-pieces, the network in our simulations needs to have nodes that accept about six tags on average. To store about 80% of all possible content the nodes need to accept about 20 tags on average. Please note, that this are only the thresholds for this specific type of data. Other data with a different tag-distribution would produce other thresholds.

2) Network Complexity: We also measured the outgoing node degree of every node by taking all active, unique contacts from every routing table into account. The average node degree at the end of the simulations was always near the count of all nodes in the simulation. To confirm that this is not the default case for Comademlia we conducted further simulations. Our hypothesis was, that the node degree is dependent on the number of accepted tags per node, because every tag introduces an additional routing table, which heightens the number of contacts a node can potentially store. We compiled the average node degrees for the tag preference simulations from the previous section into figure 5. The results show, that the outgoing node degree scales linearly with the number of accepted tags per node. The results from the main simulation can thus be explained by the, in comparison to the node count, high number of allowed tags per policy.

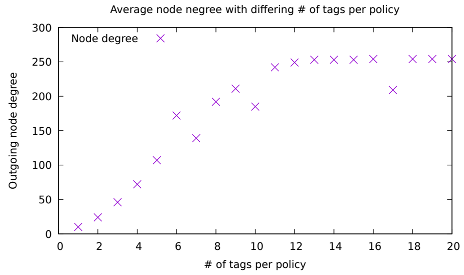

Fig. 5. The outgoing node degree for simulations using different numbers of allowed tags in the nodes policies.

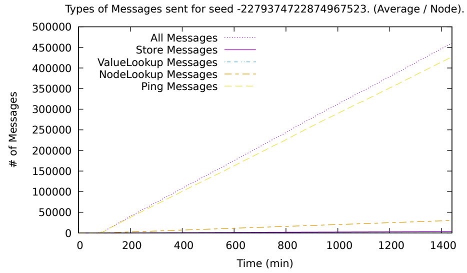

Fig. 6. The different types of Messages for one seed of the main simulation. Response type messages are not shown, since they are just mirrored in type.

3) Message Count: As a last metric we recorded the amount and type of sent messages during the simulation, which are shown in figure 6 for one of the seeds. Overall in the 24 hour simulated period the network generated 117,754,211 messages. The types of messages were very similar for all seeds and distributed (rounded) as follows: 0.7% store requests, 0.01% value lookups, 6.5% node lookups and 92.7% ping messages, which include requests for policy. We also tracked message responses separately, which trivially just mirrored the distribution of the messages. As a comparison we ran a Kademlia simulation which was kept as close to the main simulation's configuration as possible, which was done by omitting all policy and tag information[1]. The results show that the messages are comprised of 7.2% store requests, 0,2% value lookups, 72,4% node lookups and 20,2% ping messages. In sum over the 24 hour period the network generated 9,753,873 messages.

The two main differences in Comademlia are the much higher number of messages overall and how they are distributed. The higher number of overall messages is mainly due to the fact, that not only one, but multiple routing tables are running their maintenance algorithms. One of those algorithm is the automatic refresh. The refresh interval designates how long a bucket of a routing table may be unused for any operation until it is actively refreshed by querying a random node that would fall in its prefix range. The count of ping messages heightens with every routing table as well due to the way contacts are added to the routing

[1] The full results can also be found online at the results page [8]

TABLE I
MESSAGES PER LOOKUP DATA FOR MAIN SIMULATION

Seed	Messages Total	per Lookup
-2279374722874967523	11333	1.61
-3478172347484860844	10995	1.56
-4075461766863512380	10799	1.54
-6967349957205617419	11097	1.58
-7863032748618955690	10709	1.52
3426883308801768639	12698	1.81
5661528113092291996	10845	1.54
6294736137111301708	11733	1.67
6580435484139318149	11190	1.59
8678612569865056655	10678	1.52
Overall	112077	1.60

tables. When a new contact is added to a routing table, some of the established contacts are pinged to check whether they are still alive and cooperating. The smaller number of store requests in the Comademlia network is explained by the fact that in a Kademlia network there are more potential nodes willing to store data, because Kademlia nodes are not restricted by policies as in Comademlia.

To complete the analysis of the message throughput we took a look, at how many messages are generated for a single content lookup. Every seed of the main simulation created 7026 value lookups. In table I the total and average amount of queries per lookup for the different seeds is shown. The table does not take responses into account, which are symmetrical. The overall average of 1.6 queries per lookup can be considered very good, since one query is the least possible amount for a lookup, which can only be achieved if every node knows every target node directly.

V. CONCLUSION AND OUTLOOK

In this paper we introduced the concept of compliance management in peer-to-peer networks. To get an understanding of the new type of network, we discussed its desired capabilities and goals. Based on those assumptions we proposed an architecture, which relies on categorized content and policy based routing. To answer the question whether such an architecture is feasible, we implemented a proof-of-concept inside an event-based network simulator and conducted several simulations. The outcomes of the simulations were evaluated and provided us with insight about how to achieve a better performance in the future. Furthermore, the evaluation showed that compliance management through policy based routing and storage in peer-to-peer networks is a feasible approach, and should be further investigated.

Building on this work, there are several things that have to be researched further. Firstly, there is no notion what optimal performance looks like for a compliance management network. Our evaluations only show us that such a network can function and some notions how to improve, but not how well it functions compared to a hypothetical optimum. In future research we will concentrate on finding a performance optimum for compliance management, which then can serve as a benchmark for future architectures. In its current form Comademlia nodes define their policies as a whitelist, meaning they explicitly state the tags which are accepted. Some research on the implications of a blacklist approach, if any, would be insightful. Furthermore, we will research different possibilities of integrating a fully functioning content tagging system into compliance management architectures, since tags are a crucial part of a successfully operating compliance management architecture, besides policy based routing and storage. Part of this research would not only be the question how to assign tags correctly, but also how to organize the tags between themselves. Comademlia uses an approach where the tags have no special structure, but other approaches, e.g. ontology-like structures, could possibly be utilized to enhance performance.

REFERENCES

[1] J. F. da Silva, L. P. Gaspary, M. P. Barcellos, and A. Detsch. Policy-based access control in peer-to-peer grid systems. In *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, pages 107–113. IEEE Computer Society, 2005.

[2] O. Görlitz, S. Sizov, and S. Staab. Pints: peer-to-peer infrastructure for tagging systems. In *IPTPS*, page 19. Citeseer, 2008.

[3] O. Görlitz, S. Sizov, and S. Staab. Tagster-tagging-based distributed content sharing. In *European Semantic Web Conference*, pages 807–811. Springer, 2008.

[4] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, et al. Oceanstore: An architecture for global-scale persistent storage. *ACM Sigplan Notices*, 35(11):190–201, 2000.

[5] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *International Workshop on Peer-to-Peer Systems*, pages 53–65. Springer, 2002.

[6] A. Muthitacharoen, R. Morris, T. M. Gil, and B. Chen. Ivy: A read/write peer-to-peer file system. *ACM SIGOPS Operating Systems Review*, 36(SI):31–44, 2002.

[7] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*, pages 329–350. Springer, 2001.

[8] A. Schneider. Simulation results for comademlia https://www.uni-duesseldorf.de/redaktion/index.php?id=102391, 2016.

[9] D. Smetters and V. Jacobson. Securing network content. Technical report, Citeseer, 2009.

[10] D. Stingl, C. Gross, J. Rückert, L. Nobach, A. Kovacevic, and R. Steinmetz. Peerfactsim. kom: A simulation framework for peer-to-peer systems. In *High Performance Computing and Simulation (HPCS), 2011 International Conference on*, pages 577–584. IEEE, 2011.

[11] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review*, 31(4):149–160, 2001.