Autonomically Improving the Security and Robustness of Structured P2P Overlays

Gerald Kunzmann
Institute of Communication Networks
Munich University of Technology (TUM)
gerald.kunzmann@tum.de

Andreas Binzenhöfer
Institute of Computer Science
University of Würzburg
binzenhoefer@informatik.uni-wuerzburg.de

Abstract

Recent research efforts have shown that peer-to-peer (p2p) mechanisms incorporate a potential that goes well beyond simple file sharing. Compared to the classic client-server architecture, these systems do not suffer from a single point of failure.

However, there is still the danger that an adversary is able to attack a specific subpart of the system. This is especially true for structured p2p networks like Chord. A well targeted attack could cause disruptions in its global ring structure and result in severe performance degradations, loss of resources or major malfunctions.

In this paper we introduce a self-protecting approach to prevent such disruptions before they actually happen. However, since it is practically impossible to avoid all failures and attacks, we also present self-repairing algorithms, which are able to automatically detect disruptions and initiate appropriate countermeasures to reestablish the structure of the overlay.

1. Introduction

As the Internet is growing, the drawbacks of the classic client server architecture become an increasing problem. Current developments in distributed systems prove that the p2p paradigm has the potential to overcome such drawbacks. In contrast to the client server relationship, the inherent structure of p2p networks naturally resembles the connections between communicating groups. They are highly scalable since new users automatically add new resources to the system. Most importantly, they do not suffer from a single point of failure.

In this context, structured p2p networks are particularly appealing to companies in order to enable new business applications. Due to the well defined structure of the overlay, those systems are able to offer search guarantees as well as a limited search time delay [1]. However, the functionality of a deployed system heavily depends on the maintenance of its structure. A disruption of the overlay structure can cause anything from a degraded performance or a limited functionality up to the point of a total collapse of the system.

Most structured p2p networks are based on distributed hash tables (DHT). DHTs store <key, value> pairs among participants in a decentralized distributed system. The pairs can be looked up efficiently by routing the query request to the pair's owner. Additionally, DHTs are designed to be highly scalable and fault-tolerant.

The ring structure of the most researched structured p2p system Chord [2] is especially vulnerable to attacks since each disruption of the overlay can cause a disconnection of the overlay ring. In the worst case the network is split into two separate rings, which are not aware of each other. Such disconnections cannot only be caused by malicious attackers but also by churn, i.e. by the frequency at which new users join and leave the system. There are different proposals of how to handle churn in a structured p2p network [3], however, it is impossible to entirely avoid failures in the system.

To increase the stability of Chord-based p2p systems, we present a novel self-protecting approach which is able to detect possible problems at an early stage and to react accordingly. However, while it is certainly important to try to prevent attacks and failures, one cannot entirely avoid them. As experience shows, distributed systems will encounter failures and

0-7695-2699-3/06/$20.00 (c) IEEE

one should design for it. Therefore we additionally aim at the recovery from failures rather than at failure-avoidance alone. Our self-repairing algorithms are able to automatically detect disruptions and security problems and will initiate redundant countermeasures to reestablish the structure of the overlay.

The remainder of this paper is structured as follows. We summarize related work in Section 2. Section 3 gives a brief overview of Chord with a focus on aspects relevant to this paper. Section 4 identifies some security issues of the Chord protocol. In Section 5 we show how to recover from disruptions in the overlay structure and describe an approach of how to try to avoid them in Section 6. Section 7 finally concludes the paper.

2. Related Work

There are different kinds of security concerns in DHT-based p2p networks. Most research so far concentrates on misbehaving nodes that do not implement the protocol correctly or which simply cannot be trusted. [4] gives a good overview of security problems which are inherent to large p2p systems. The focus is on adversary peers which mislead legitimate nodes by providing them with false information. The authors concentrate on attacks against the routing and against the data storage system.

[5] also studies attacks aimed at preventing correct message delivery in structured peer-to-peer overlays and presents defenses to these attacks. A secure routing algorithm is proposed which allows tolerating up to 25% malicious nodes while providing good performance when the fraction of compromised nodes is small.

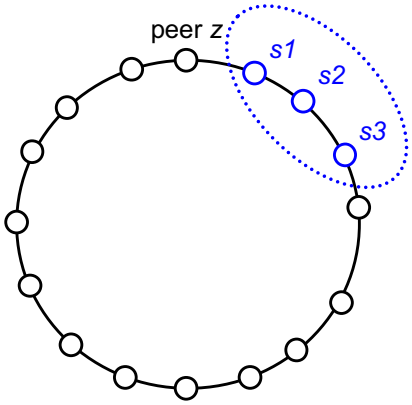

Figure 1: Successors of peer z

Several security threats like the well known Sybil attack [6] are addressed in [7]. Its main focus is on a quantitative analysis of routing anomalies that can be caused by malicious nodes returning incorrect lookup routes. Finally [8] studies what kind of information an adversarial node can learn about another node in the same network through the simple observation of network traffic.

In contrast to the above, the main contribution of our work is to prevent malicious nodes from destroying the structure of the overlay network and to develop self-repairing mechanisms to recover the structure in case of a disruption.

3. A brief introduction to Chord

This section gives a brief overview of Chord with a focus on aspects relevant to this paper. A more detailed description can be found in [2].

In general, a DHT assigns each peer in the overlay an m-bit identifier using a hash function such as SHA-1. Chord builds a ring topology (clockwise marked with numbers from 0 to 2^m), where the position of a peer on this ring is determined by a peers *m*-bit identifier. If the ring structure is lost, the functionality of the Chord algorithm can no longer be guaranteed. Therefore a peer stores information about its *r* immediate successors on the ring. Figure 1 shows the successor list for a peer z and $r = 3$ successors. It consists of s1, s2, and s3, the three immediate successors of peer z. If the immediate successor s1 of peer z goes offline, peer z can still contact the next closest peer s2 of its successor list. As stated in [2], in an *N*-node system, $r = \log_2(N)$ peers are sufficient to ensure that each peer knows the id of its closest living successor.

The hash function also assigns keys to data (resources or keywords). According to Chord, a key k is assigned to the first node whose identifier is equal to or follows (the identifier of) k in the identifier space. This node is called the *successor* of key k.

A peer could look up another peer or key by passing the query around the circle using its successor pointers. To accelerate searches each peer also maintains pointers to other peers, which are used as shortcuts through the ring. Those pointers are called fingers, whereby the *i*-th finger in a peers finger table contains the identity of the first peer that succeeds the nodes own id by at least 2^{i-1} on the Chord ring. That is, peer z with hash value id_z has its fingers pointing to the first peers that succeed $id_z + 2^{i-1}$ for $i = 1$ to m, where 2^m is the size of the identifier space.

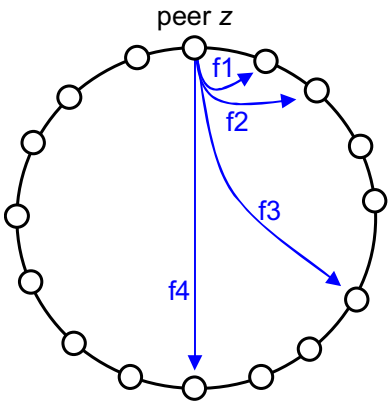

Figure 2: Fingers of peer z

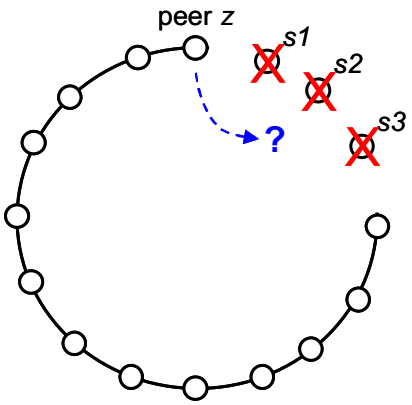

Figure 3: Concurrent failure of p's successors

Figure 2 shows fingers f1 to f4 for peer z. Using these finger pointers, a lookup requires only about $O(\log_2 N)$ hops. Searches return a correct answer as long as each node knows its correct successor. Fingers are only used to speed up lookups. A detailed mathematical analysis of the search delay in Chord rings can be found in [1].

4. Security Concerns (and Detection)

Loss of all successors

Erroneous successors can lead to erroneous lookups. In the worst case, they can even cause disruptions in the overlay topology. Chords ring structure can encounter two different kinds of serious damage. First, if a peer loses all of its successors, the ring will break open. Second, the ring structure may be split into two halves or two separate sub rings.

In this section we discuss different offensive scenarios that result in such overlay disruptions. In particular, we identify different threats and their impacts.

Due to churn or a well directed denial of service (DoS) attack on at least r successive nodes on the Chord ring, peer z, that precedes the affected part of the ring, will no longer be able to contact any of its successors. In fact, it can be shown, that the probability to lose all successors due to churn is not negligible [9]. After sending several ping messages to these offline or attacked nodes, a timer expires and the nodes are removed from z's successor list. Consequently, the ring structure breaks open as depicted in Figure 3 ($r = 3$). A peer can easily detect such a break in the ring as soon as it discovers its list of successors to be empty.

As Chord lookups are only performed clockwise, the peer is not able to search for its new successor. Therefore performing a rejoin if a peer looses all of its successors is no feasible solution to this kind of disruption.

The consequence of a loss of all successors is a transient routing state. That is, some nodes might no longer be reachable, while others might not be able to forward search queries correctly.

Partitioning of the overlay

Another threat to the network is a partitioning of the overlay structure, i.e. the ring breaks into two or more separate overlays [2]. This scenario is likely to occur if gateways between physically separated networks fail. Chords stabilization mechanism updates all erroneous successor pointers. After a certain mean time to repair two or more consistent sub rings emerge. Lookups can still be performed correctly in all new Chord rings, but due to the partitioning, not all data stored in the original overlay will still be available in all sub parts. A company running a global DHT application, for example, will no longer be able to access all data stored in the DHT, if one plants access point fails. Running a DoS attack on nodes that have a critical location in the physical network is sufficient to damage the whole network.

In mobile ad-hoc networks (MANETs), network splits are even a common issue. The overlay is likely to be partitioned due to frequent and fast node movement, node failures and MANETs that are out of each others range. Successive splits without any countermeasures finally result in many sparely populated subnets.

There exist mechanisms (c.f. Section 6) that reduce the risk of a ring split, but are not able to avoid them entirely. Moreover, the above examples clearly indicate that the overlay protocol must be able to recover from a partitioned network. Therefore, we introduce some efficient mechanisms that are able to detect and merge sub rings in Section 5.

5. Recovery

Recovery from a partitioning of the overlay

If an overlay is split into several partitions, but the nodes are still connected in the physical network, it is likely that there are still fingers in each partition that point to nodes in other parts of the network. Lookups will pass through different sub rings and in the end return an erroneous result. Nodes can use their finger entries and information gathered during lookups to learn about nodes in other partitions. By inserting all other appropriate nodes into their own successor list, the separate rings will merge automatically.

However, in scenarios where no physical connections between separate sub rings exist, as pictured in the previous section, the partitions cannot be merged. Fingers pointing to nodes in other parts cannot be contacted and the algorithm that updates the node's fingers removes these entries after a while. If the physical connection between two rings is re-established, nodes will not learn about the other ring by themselves.

A simple approach is to run a periodic rejoin at every node. In doing so, each node starts a lookup for its direct successor via the bootstrap service. It makes no difference if the bootstrap mechanism is a local or remote cache of available nodes or a single server. The proceeding is the same as with a node that joins the network. If the bootstrap service by chance returns a node from another partition, this information can be used to merge both rings. In our simulation environment we observed that two rings merge within a few minutes if at least one node learns about any node in the other ring. The main drawback of this approach is that each node periodically has to perform a rejoin operation and therefore stresses the bootstrap service. The shorter the rejoin period, the faster two different rings can be detected, but the more the bootstrap mechanism is stressed. Therefore, this algorithm will not scale for huge overlay networks.

In a more efficient variant of this mechanism only one well-defined peer in each ring, e.g. the peer with the smallest ID sends a periodic message to the bootstrap server. A peer assumes it is the responsible peer if its predecessor has a higher ID than the node itself. The bootstrap server notices separate rings as soon as it receives messages from different peers. By informing all involved peers, a merging process can be started. As only one peer per ring sends periodic messages this variant is highly scalable. Also, the frequency of performing this algorithm can be increased significantly resulting in a much faster detection of sub rings.

Recovery from loss of all successors

If the ring breaks due to a failure of *r* successive nodes the peer preceding the disrupted part of the ring is not able to contact any of its successors. As discussed in the previous section a standard lookup for the nodes successor will also not return any result. We present a modified search algorithm that is capable of performing lookups regardless of disruptions. The key functionality is an algorithm that redirects a lookup request in counterclockwise direction if the lookup skipped one or more nodes. We call this method *redirection mechanism*. It can also be used in normal operation if a lookup request skips the keys correct successor and is received by the wrong succeeding peer. As soon as a peer recognizes that a search overshot its target, it applies our redirection mechanism.

A node y can easily detect that a lookup did overshoot the correct successor, if it receives a lookup message for a key k located between the initiator of the lookup and itself, but node y is not k's successor. Using its predecessor, node y is able to redirect the message towards the correct successor. The message may also be redirected over several nodes until the correct node is reached.

In case of an open ring the node preceding the disruption can use the redirection mechanism to repair the overlay disruption. It simply sends a lookup message for its own ID+1 to the closest available finger. In general, this is the smallest finger that is situated outside the nodes former successor list. As shown in Figure 4, this node will then redirect the message in counterclockwise direction until the message arrives at the other end of the disruption. This peer no longer possesses a valid predecessor as all of its preceding peers have failed. Therefore, it assumes that the initiator of the message is its new predecessor. For the same reason it assumes that it is responsible for the searched ID and answers the lookup. The initiator of the message inserts the sender of the request in its successor list and initializes a stabilization procedure with its new successor. The disruption is repaired and correct routing is reestablished.

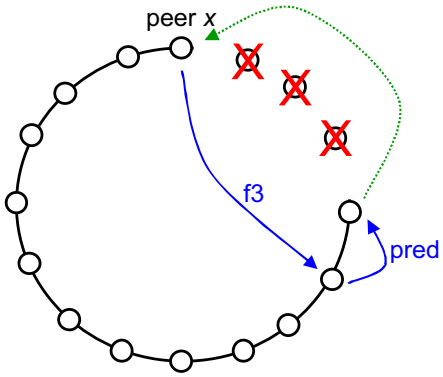

Figure 4: Automatic disruption recovery, initialized at the beginning of the break

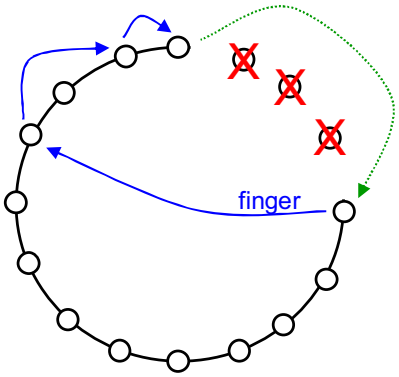

Figure 5: Automatic disruption recovery, initialized at the end of the break

If a peer also stored enough predecessors (i.e. maintained a symmetric neighbor list) a similar recovery mechanism could be used by the peer at the end of the disruption. A symmetric neighbor list consisting of *r* successors and *r* predecessors is also useful in achieving a more stable overlay [10, 11] and a more efficient replication algorithm. In the following we therefore assume symmetric neighbor lists.

Then, a node that has lost all of its predecessors initiates a lookup for its own ID (see Figure 5). The lookup will traverse the ring until it arrives at the node at the beginning of the disruption. If this node is not yet aware of the disruption it tries to forward the lookup message to one of its successors. As all successors have failed, the node will receive no acknowledgments and after a certain while delete all successors from its list. A node that is aware of the disruption, as it has lost all successors, inserts the sender of the lookup message into its own list of neighbors. It then forwards the lookup to its new successor and starts stabilizing with it.

If both nodes at the edges of the rings broken part run a recovery algorithm the disruption is detected faster and can be repaired with higher probability. In the worst case one redundant lookup message is routed through the ring.

Note that if symmetrical routing [12] is applied, the redirection mechanisms is no longer necessary. Both nodes at the edges of the disruption can initiate a symmetrical lookup for their own ID.

Recovery using token based stabilization

[11] proposes a Token Ring [13] like algorithm that replaces Chord's stabilization messages by tokens, which are sent in both directions around the ring. This way, a more stable overlay can be achieved. In normal operation nodes are in the state *repeat*, i.e. they forward all incoming tokens to the next node on the ring. A node that is situated at one end of a broken ring does no longer receive token messages from the disrupted part of the ring. Therefore, it changes to the state *active monitor* and starts generating periodic tokens. All tokens contain ID and IP address of its initiator. Acknowledgements prevent tokens from being lost as nodes fail. The token is passed through the ring until it reaches the peer at the other end of the broken part. There, the information about the tokens initiator can be used to repair the ring disruption. The initiator is inserted into the empty neighbor list and a stabilization process is initiated. However, this algorithm does not scale with the ring size as the token is forwarded from node to node, requiring *N* times the average transmission time to circulate the ring.

6. Avoidance

Regarding the correctness of the Chord overlay, we observed that the probability of disruptions can noticeably be reduced by some simple modifications to Chord's stabilization algorithm.

To avoid a disruption in the ring structure, nodes should prevent an empty successor list. If the number of entries reaches a critical minimum, nodes can fill their successor list with any active node they known (e.g. finger entries) or learn about (e.g. from received messages). The redirection mechanism will still guarantee correct lookups.

To increase the correctness of the overlay structure nodes can also decrease their stabilization period. The more often stabilization messages are sent, the more up-to-date the neighbor entries. We suggest an adaptive mechanism that increases the stabilization period if

the number of known successors shrinks or if the overlay structure is measured to be more dynamic. Additionally the size of the neighbor list can be adjusted adaptively to the current churn rate in the network. However, the more often stabilization messages are sent and the more successors are included in the messages, the more bandwidth is required. Nodes should pay attention to their current resource usage to avoid performing a DoS attack on themselves.

Most importantly, we stress that nodes should make use of all information they can gather about other nodes. They should check whether the sender of any message they receive fits in the list of neighbors or fingers. If the sender of the message is already part of a list, update the time last seen for this entry. Therefore, a node learns about new nodes without the need to wait for the next stabilization. Additionally, the necessary bandwidth for checking for the correctness of the finger entries can be reduced. Fingers with a very recent time last seen are skipped when fingers are updated.

We also suggest sending information about nodes that have failed to all neighbored nodes. Therefore, nodes can replace failed neighbors must faster. Yet, we dissuade from blindly trusting in information received from other nodes, as this information may be incorrect. So, nodes should verify the information, e.g., by sending a ping message to the responsible node. If recursive routing is applied, nodes exchange a lot of messages with their successors and fingers. Therefore, nodes are aware of failed contacts much faster.

Finally, we recommend using a symmetrical Chord variant with symmetrical neighbor lists [10] and symmetrical routing [12]. Additional symmetrical fingers can be achieved by exploiting the existing overhead [14]. Symmetrical routing enables nodes to search in both directions, so a simple disruption in the ring can be avoided.

7. Conclusion

Disruptions in structured p2p overlays cannot only be caused by well targeted attacks against specific nodes but also by churn, i.e. by the dynamic behavior of the participating peers.

In this paper we presented efficient mechanisms to actively prevent the loss of the overlay structure in both scenarios. Using some simple modifications to the standard algorithm a peer is able to exploit the existing overlay traffic to improve the stability of the overlay.

We also introduced a self-repairing mechanism, which is able to detect a disruption in the overlay network and to apply appropriate countermeasures. The algorithm was designed to be redundant in order to speed up the healing process and to improve its success rate.

Finally, we introduced a scalable solution to detect the existence of disjoint overlay partitions and showed how to automatically recombine them. Applying our modifications to a Chord-based p2p system can greatly improve its security and robustness.

8. References

[1] P.T.-G. Andreas Binzenhöfer. Delay Analysis of a Chord-based Peer-to-Peer
File-Sharing System. in ATNAC 2004. 2004. Sydney, Australia.

[2] I. Stoica, et al. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. in ACM SIG-COMM. 2001. San Diego, CA, USA.

[3] S. Rhea, et al. Handling Churn in a DHT. in USENIX 2004 Annual Technical Conference. 2004.

[4] E. Sit and R. Morris. Security Considerations for Peer-to-Peer Distributed Hash Tables. in 1st International Workshop on Peer-to-Peer Systems (IPTPS). 2002. Cambridge, MA.

[5] M. Castro, et al. Secure routing for structured peer-to-peer overlay networks. in OSDI '02. 2002. Boston, MA.

[6] J.R. Douceur. The Sybil Attack. in IPTPS02 Workshop. 2002. Cambridge, MA (USA).

[7] M. Srivatsa and L. Liu. Vulnerabilities and Security Threats in Structured Overlay Networks: A Quantitative Analysis. in 20th Annual Computer Security Applications Conference (ACSAC '04). 2004. Washington, DC.

[8] C.W. O'Donnell and V. Vaikuntanathan. Information Leak in the Chord Lookup Protocol. in Peer-to-Peer Computing (P2P 2004). 2004. Zurich, Switzerland.

[9] A. Binzenhöfer, D. Staehle, and R. Henjes, On the Stability of Chord-based P2P Systems. 2004, University of Würzburg.

[10] G. Kunzmann, A. Binzenhöfer, and R. Henjes. Analyzing and Modifying Chord's Stabilization Algorithm to Handle High Churn Rates. in 6th Malaysia International Conference on Communications (MICC) in conjunction with International Conference on Networks (ICON). 2005. Kuala Lumpur, Malaysia.

[11] G. Kunzmann, R. Nagel, and J. Eberspächer. Increasing the reliability of structured P2P networks. in 5th International Workshop on Design of Reliable Communication Networks (DRCN). 2005. Island of Ischia, Italy.

[12] V.A. Mesaros, B. Carton, and P.V. Roy. S-Chord: Using Symmetry to Improve Lookup Efficiency in Chord. in International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'03). 2003.

[13] Wikipedia, Token Ring. 2006.

[14] G. Kunzmann and R. Schollmeier. Exploiting the overhead in a DHT to improve lookup latency. in 11th Open European Summer School (EUNICE). 2005. University Carlos III of Madrid, Colmenarejo, Spain.