Group Spreading: A Protocol for Provably Secure Distributed Name Service

Baruch Awerbuch
Department of Computer Science
Johns Hopkins University
3400 N. Charles Street
Baltimore, MD 21218, USA
baruch@cs.jhu.edu

Christian Scheideler
Department of Computer Science
Johns Hopkins University
3400 N. Charles Street
Baltimore, MD 21218, USA
scheideler@cs.jhu.edu

Abstract

In order to enable communication between a dynamic collection of peers with given ID's, such as "machine.cs.school.edu", over the Internet, a distributed name service must be implemented on top of this collection of peers that translates given names into IP addresses, by implementing a dynamic distributed database for concurrent IP lookups.

This paper shows that this very fundamental task can be implemented in a completely decentralized way in the presence of arbitrary massive Byzantine attacks while expending only poly-logarithmic overhead for retrieval, protection, and maintenance. To accomplish this, this paper introduces a new methodology that essentially maintains a random distribution of all (honest and Byzantine) peers in an overlay network for any sequence of arrivals and departures of nodes up to a certain rate, under a reasonable assumption that Byzantine peers are a significant minority. Keeping nodes randomly distributed allows the system to form reliable quorums for join, leave, and lookup operations so that for a polynomial number of steps, *any* of these operations can be executed reliably by *any* honest peer, with high probability.

This demonstrates that scalable peer-to-peer systems are not limited to music-swapping and can potentially perform important applications that require rigorous security guarantees.

1 Introduction

1.1 The problem and the results

In order to enable communication between a dynamic collection of peers with given ID's, such as "machine.cs.school.edu", over the Internet, a distributed name service must be implemented on top of this collection of peers that translates given names into IP addresses, by implementing the following operations:

- $p.\mathsf{Join}(q, \mathsf{ID})$: peer p in the system receives a request to join the system from a peer q with identity ID.
- $p.\mathsf{Leave}()$: peer p leaves the system.
- $p.\mathsf{Lookup}(\mathsf{ID})$: peer p wants to obtain the IP address of the peer q in the system with $\mathsf{ID}(q) = \mathsf{ID}$.

These operations must be implemented so that they can be run *concurrently* and *reliably* in an *asynchronous* environment in spite of massive insider attacks, i.e. arbitrary Byzantine behavior by a large number of peers that are *part* of the service (for a detailed model see Section 2.1). Protecting against insider attacks seems a formidable problem, but we can show in a rigorous way:

Theorem 1.1 *For any $c > 0$ and any sequence of operations lasting for at most $O(n^c)$ steps during which at most a $O(1/\log N)$ fraction of the nodes in the system can be adversarial at any time, the join/leave rate of honest nodes is $O(1/\log N)$ (i.e. up to $O(N/\log N)$ honest nodes may join or leave in a time unit), and the join/leave rate of adversarial nodes is $O(1/\log^2 N)$,* all *operations initiated by honest peers are correctly executed with probability at least $1 - O(1/n^c)$, and with communication cost of $O(\log^{O(1)} N)$ bits per operation, where n is the minimum number and N is the current number of honest nodes in the system during the attack.*

In other words, for a polynomial number of time steps, even a large fraction of Byzantine peers will have *no* effect on the operation of honest nodes, with high probability, and this can be achieved with only a *polylogarithmic* overhead. Previous solutions to this problem required *linear* overhead (see below).

In the rest of this paper, we proceed with first explaining the significance and basic approach of our result and then proceed with presenting our protocols and proofs. Since these are very complex, we restrict ourselves to giving only intuitions in the main part of this manuscript. The details can be found in the appendix. Though the appendix is quite lengthy, we felt we had to add it to give sufficient evidence that our techniques indeed work.

1.2 Significance of the problem

Scalability or efficiency of a distributed system is measured in its ability to retrieve the data and to maintain internal data structures with small overhead; a rule of thumb for a scalable system is that the above overhead must grow at most poly-logarithmically in its size. Security or reliability of a distributed system is measured in its ability to withstand massive and malicious attack, including Byzantine behavior of its components.

Achieving security and scalability at the same time is a decades old open problem in the field of distributed computing. In unreliable centralized systems, such as communication channels and storage systems, the trade-off between efficiency and reliability has been extensively studied in the context of coding theory, namely is expressed by the trade-off between the number of errors corrected by the code and the number of error-correcting bits. It is only natural to pose such a fundamental question in the distributed context.

The additional formidable complication in distributed systems is the need to maintain virtual communication channels between the peers in the system. Server-based architectures are not an option here since they are not scalable. Hence, mechanisms are needed that allow the peers to maintain an overlay network in a distributed way and without a server, also known as *peer-to-peer overlay networks*.

It is intuitively obvious that an overlay network supporting any service needs to support joining, leaving and routing between the peers, and that without a secure and scalable implementation of such a network, the field of scalable and reliable distributed services does not really exist.

1.3 Existing work

Classical distributed computing methods [12, 3, 13, 17] use Byzantine agreement and two-phase commit approaches with inherently *linear* redundancy and overhead to maintain a safe state.

The *proactive security* approach in [16, 11, 10, 2, 7] uses different coding techniques to protect unreliable data in *reliable* networks; applying these methods in our context still yields linear overhead.

Fixed topology networks as in [6], will work only for non-Byzantine peers, and only allow fail-stop faults; the construction cannot handle malicious behavior of even a few malicious players.

The reliability of *hash-based peer-to-peer overlays* (or DHT's) such as Chord [19], Pastry [18], and Tapestry [20] hinges on the assumption that the IDs given to the nodes are pseudo-random, so that they can cope with a constant fraction of the nodes failing concurrently, with only logarithmic overhead. While this may seem to perfectly defeat massive attacks under these randomness assumptions, DHT's cannot handle even small-scale adaptive adversarial attacks involving the selection of adversarial IP addresses (to get close to desired IDs). One such "sybil" attack is described in [5]. Remarkably, the attackers do not need to do anything complex such as inverting the hash function; all that is needed is to get hold of a handful (actually, logarithmic) number of IP addresses so that IDs can be obtained that allow to disconnect some target from the rest of the system. This can be accomplished by a linear number (i.e. $O(N)$) of offline trial/errors. For similar attacks, see [4].

Random or unpredictable placement of data in a logarithmic size subset of locations (as in Freenet) ensures that data is difficult to attack, but also makes it *difficult to retrieve*. Specifically, data retrieval of randomly placed data requires a linear number of queries, which is definitely unscalable.

2 Basic approach and results

A common technique to ensure reliability, which is also the basis of our approach, is to organize the peers in completely interconnected *quorums*. However, instead of just forming a single quorum, we will form an overlay network of quorums of logarithmic size to ensure that Join, Leave, or Lookup operations by honest peers cannot only be executed reliably but also efficiently. Taking this into account, our basic goal can be phrased in the following way:

Design overlay network operations so that for any *arrival-departure sequence of honest and adversarial nodes over t time steps in which the adversarial nodes never represent more than an ϵ fraction of the honest nodes in the overlay network and the number of join/leave requests of honest nodes does not exceed some specified rate, any overlay network operation initiated by an honest node will be executed successfully, with high probability.*

We call an overlay network *survivable* if it can ensure this property for any $t = poly(n)$, where n is the smallest number of nodes in the overlay network during the adversarial attack. Our goal will be to formulate basic conditions and protocols for the network operations to fulfill the survivability condition. Notice that we have to add the term "with high probability" above, because a priori, it is not possible to distinguish between honest and adversarial nodes. So no absolute guarantees can be given, unless we put all nodes into a single quorum, which is highly inefficient and therefore out of question. Furthermore, for bounded degree networks a rate of $O(1/\log N)$ departures or arrivals seems to be optimal because for each arrival of a

peer $\Omega(\log N)$ work appears to be necessary for a survivable system. Hence, an arrival rate of $O(1/\log N)$ would translate into a work rate of $O(1)$ for the system, which is at the limit of what it can sustain.

2.1 Basic assumptions

In order to perform a rigorous security analysis, we need a set of basic assumptions. These assumptions are general enough that we expect them to be of interest beyond this paper and to aid in the design and analysis of secure overlay networks in the future.

ϵ-bounded adversaries

We consider a peer to be *adversarial* if it belongs to an adversary or it is simply unreliable. Otherwise, a peer is called *honest*. We assume that honest peers cannot be taken over by the adversary. However, the adversary has a collection of own peers that it can integrate into the network. We allow arbitrary adversaries with bounded resources, i.e. an adversary can own at most an ϵ-fraction of the peers in the system at any time. Such adversaries are called ϵ-*bounded*. A priori, adversarial nodes cannot be distinguished from honest nodes.

These assumptions need some justification. Suppose that *any* honest peer can in principle be taken over by the adversary. Then it would be impossible to form reliable quorums of small size, because as soon as the adversary is part of a quorum, and therefore knows all peers in a quorum, it could simply turn all of them into adversarial peers, thereby turning this quorum into an adversarial quorum. If quorums cannot be secured, we do not see any way of establishing a reliable overlay network service. However, weaker models for the ability of an adversary to take over honest peers are imaginable. For example, only some *fixed subset* of the honest peers may be vulnerable to an adversary (for example, because they have not installed the latest security patch). In this case, we simply view these peers as adversarial peers, so our results still hold in this case.

The other major assumption we make is that adversaries are *bounded*. Imagine that an adversary can have an unbounded number of peers in the overlay network. Then they could easily dominate every quorum, and therefore the system cannot be secured. Hence, it is reasonable to assume that an adversary can only have a bounded number of peers in the system at any time. However, the *identities* of these peers may be chosen out of some *unbounded* set of identities that the adversary has at its disposal, so that it does not help honest peers to remember which peers behaved adversarial in the past.

Trusted gateway peers

We assume that a limited number of so-called *gateway peers* is available that are searchable over the Internet and always part of the overlay network. Gateway peers are like normal peers, with the only difference that they are the only ones allowed to integrate new peers. For our protocols to work we assume that the gateway peers are honest. However, the protocols would even work if all gateway peers are adversarial as long as no honest peer leaves and the adversarial nodes only represent an ϵ fraction of the system.

Also here, some explanations are in order. We came up with gateway peers because somehow new peers have to be able to find out about peers already in the system. Assumptions like "peers contact random peers in the system" are not realistic because how should a peer find this random peer? Hence, a server or service that is accessible via the Internet must be available. The easiest solution here would be to have a trusted server that knows a collection of peers in the system. Though the server may be trusted, the peers in the system it knows about may *not* be trusted (a priori, honest peers cannot be distinguished from adversarial peers). But if a new honest peer is referred to an adversarial peer, there is certainly no way of guaranteeing its correct integration into the overlay network. Hence, for a trusted server to be effective in integrating

new peers, the server should be *part* of the peer-to-peer system (and not just refer to peers), because in this case guarantees for the correct execution of join requests can be given by us. Now, we may not want to assume powerful servers because otherwise we may not have an open peer-to-peer system any more (because someone has to pay the bill). Hence, the server or servers that give access to the peer-to-peer network may just be as powerful as the other peers, with the only difference that they are trusted. These represent our so-called gateway peers.

Now, one may say that if the gateway peers are honest, why not let them control the overlay network of the peers so that it is secure. There are two good reasons to not let them do this. First, if the overlay network were controlled by gateway peers, it would not be scalable (see the limited lifetime issues we will address below) because we assume the gateway peers to represent only a negligible fraction of the peers in the system. Second, the overlay network would have no chance of surviving the situation that gateway peers are not available (due to some DoS attack) or are even adversarial. Our protocols would survive such a situation as long as honest peers do not leave because their operation does *not* depend on the gateway peers. Gateway peers are *only* needed to make sure that new honest peers have a chance to enter the system.

Honest peers are reliable

We use an asynchronous system model in which every honest peer has roughly the same internal clock speed (i.e. up to a small constant factor) but there is no global time. At any point in time, any message sent by an honest peer v to another honest peer w will arrive at w within a unit of time. (Other message transmissions may need any amount of time.)

Undoubtedly, computers do have roughly the same internal clock speed today (otherwise, we could not rely on them for scheduling appointments). Message delays between honest peers have to be bounded because time is a very critical issue for dynamic overlay networks. If no upper bound can be given for message delays between honest nodes, i.e. we have a completely asynchronous system, we strongly believe that it is impossible to make overlay networks secure (see the limited lifetime issues below). As long as only a small fraction of messages between honest nodes violate our delay bound, our protocols would still work correctly, but for simplicity we assume that a delay bound can be guaranteed.

Honest peers have unbounded bandwidth

We assume that honest peers have unbounded bandwidth, i.e. an honest node can receive and send out an unbounded number of messages in a unit of time.

Though this assumption appears to be very strong, it does *not* protect against legal attacks (i.e. smart attacks against the protocol) but only against brute-force denial-of-service attacks. Hence, it still remains a challenging task to perform a security analysis. As long as adversarial nodes do not transmit unnecessary packets (i.e. they adhere to the protocol), the number of messages an honest node will have to deal with in a time unit will normally be very low in our protocols (i.e. logarithmic in the number of nodes in the system) so that our protocols are practical despite the unlimited bandwidth assumption. We expect that designing secure overlay networks for peers of bounded bandwidth will be a very challenging task, which is unfortunate since in reality this is the case. Nevertheless, we do not view it as impossible, and therefore our result in this paper may be seen as a first step in this direction (especially because *no* provably secure design for overlay networks was known before, apart from our recent work in [**?**]).

Security assumptions

We assume that a certification authority is available to issue certified names to peers that want to enter the system. This prevents peers from taking over the identities of other peers, which is important for any name

service to work correctly, but it does *not* prevent adversarial peers from registering under adaptively chosen names that are different from names of the honest peers. An honest peer only establishes connections to peers with correctly certified names.

Finally, we need some assumptions about how messages are passed. We assume that the source of a message cannot be forged so that adversarial nodes cannot take over the identity of honest nodes. Also, a message sent between honest nodes cannot be deleted, or altered by the adversary. However, the adversary may know about every message sent in the system. Furthermore, we assume the existence of a bit commitment scheme (e.g, [14, 9]) that allows nodes to commit to a certain key without revealing any part of it. This is important for our ID generation strategy.

Notice that the source issue can actually be solved easily without cryptography as long as adversaries cannot hijack IP addresses or listen to communication between honest nodes: if a message arrives from IP address x, then the receiver y asks x for a confirmation that contains a secret (for example, a random key). Only if y receives an acknowledgement from x containing the secret, y will accept the message. The assumption that messages cannot be inspected, deleted, or altered by the adversary is realistic in our case because we assume the nodes of our overlay network to sit at the edge of the Internet, and therefore peers cannot inspect communication between other peers.

2.2 The Chord overlay network

Next, we review the construction of the Chord overlay network [19], which is used in our impossibility results as well as in our secure overlay network construction.

Suppose that we have a system currently consisting of a set V of n nodes, and further suppose we have a (pseudo-)random hash function $h : V \to [0, 1)$ that maps nodes to real values in the interval $[0, 1)$. The basic structure of Chord is a doubly-linked cycle, the so-called *Chord ring*, in which all nodes are ordered according to their hash values. In addition to this, every node v has edges to nodes $p_i(v)$, called *fingers*, with $p_i(v) = \text{argmin}\{w \in V \mid h(w) \geq h(v) + 1/2^i\}$ for every $i \geq 1$.

One can show that Chord has a diameter of $O(\log n)$ and an expansion of $\Omega(1/\log n)$ w.h.p. [1], making it robust in distributed environments under random faults but not against adversarial behavior.

2.3 Main results

Next we summarize our main results in this paper.

Lower bounds

We show that survivability cannot be maintained by predictable overlay networks, i.e. networks in which for some fixed arrival and departure sequence the topology will always be the same.

Theorem 2.1 *No predictable overlay network can be survivable.*

We also show that none of the suggested hash-based peer-to-peer systems can be survivable (in their basic form), even if the hash function for the node identification numbers is chosen at random (instead of using some fixed SHA-1 function), or all nodes (including the adversarial nodes) use completely random identification numbers. The basic problem is that a node can potentially be in the system for an unlimited amount of time.

As we will see, any overlay network that wants to be survivable has to act in a proactive fashion in order to protect itself, as e.g., in [16, 11, 10, 2, 7]. In particular, it must be possible to exclude nodes from the overlay network, even against their will. If this were not possible, then any node could potentially stay in the overlay network for an unlimited amount of time, causing tremendous security problems (as we will see

later). We accomplish this in an *oblivious* way by setting a maximum lifetime after which every node has to leave the system.

Upper bounds

We then present a strategy, called *Group Spreading* that is survivable against adversarial peers. It should be applicable to all DHT-based peer-to-peer overlay networks suggested in the literature, but in order to simplify the presentation, we just concentrate on the Chord network given above.

The core idea of the Group Spreading technique is to use a simple mechanism that enforces the selection of a random ID for every successful ID request of a peer that wants to join. This simple mechanism may fail if adversarial peers are involved in it. To prevent them from causing problems for honest peers, every honest peer will keep a group of $\Theta(\log N)$ nodes in the system so that with high probability sufficiently many nodes of a peer can rejoin the system as the lifetime of its old nodes expires. Keeping nodes in clusters of size $\Theta(\log N)$ and enforcing a limited lifetime then makes sure that for a polynomial number of steps every cluster will have a majority of honest nodes, with high probability. There are many tough details that have to be solved to handle this strategy in a dynamic overlay network with Byzantine nodes. For example, the group spreading protocol has to run *while* nodes join and leave the system. Also, Byzantine nodes make it extremely difficult for nodes to agree on the membership of a cluster, especially when clusters have to be reorganized due to a changing number of nodes in the system. Nevertheless, we can show:

Theorem 2.2 GROUP SPREADING *survives up to a* $\Theta(1/\log N)$ *fraction of malicious nodes with a communication cost of* $\log^{O(1)} N$ *bits per operation if the join/leave rate of honest nodes is* $O(1/\log N)$ *and the join rate of adversarial nodes is* $O(1/\log^2 N)$.

2.4 Probabilistic tools

We will frequently use the Chernoff bounds.

Lemma 2.3 (Chernoff [8]) *Consider any set of independent random variables* $X_1, \ldots, X_n \in \{0, 1\}$. *Let* $X = \sum_{i=1}^{n} X_i$ *and let* $\mu = \mathrm{E}[X]$. *Then it holds for all* $\delta \in [0, 1]$ *that*

$$\Pr[X \leq (1 - \delta)\mu] \leq e^{-\delta^2 \mu/2}$$

and for all $\delta \geq 0$ *that*

$$\Pr[X \geq (1 + \delta)\mu] \leq e^{-\min[\delta, \delta^2]\mu/3}$$

3 Non-survivable overlay networks

In this section we prove that predictable overlay networks and hash-based overlay networks (i.e. networks in which the ID of a node is determined by a hash function) are not survivable. Furthermore, we show that being able to enforce a limited lifetime is crucial for the survivability of systems based on a virtual space, like hash-based systems.

3.1 Predictable overlay networks

An overlay network is *predictable* if for any fixed arrival and departure sequence the topology will always be the same. Notice that all hash-based overlay networks with a fixed hash function are predictable.

We start this section by demonstrating that *no* predictable overlay network can be survivable under our definition of survivability.

Theorem 3.1 *Consider an arbitrary predictable overlay network of maximum degree d that allows N peers to join or leave in T time units. Then there is a join/leave sequence of $2N$ honest nodes so that an ϵ-bounded adversary with $\epsilon \geq d/N$ can surround an honest peer in $O(T)$ steps.*

Proof. The proof is relatively easy. First, $2N$ peers join to create a network of size $2N$, and afterwards the first N peers that joined the network leave. This takes $O(T)$ time steps. Consider now any peer in the resulting network, say v, and let $w_1, \ldots, w_d$ be its neighbors. Then, consider the join/leave sequence of honest peers that is like the sequence above but without $w_1, \ldots, w_d$. Assign the join events for $w_1, \ldots, w_d$ to the adversary. Then we arrive at the situation that v is completely surrounded by adversarial nodes. This sequence *always* works because the overlay network is predictable. Hence, the theorem follows. □

3.2 Hash-based overlay networks

Hash-based overlay networks are vulnerable to adversarial attacks even if the hash function is chosen at random, and it is a one-way hash function. The mere fact that nodes do not change their location over time turns them into "sitting ducks". To illustrate how an attack on hash-based approaches would look like, consider the Chord system.

Take any peer u in Chord with hash value $x \in [0, 1)$ (see figure 1). By generating a set A of adversarial peers with hash values $x - \epsilon$, $x + \epsilon$, and $x + 1/2^i + \epsilon$ for all relevant i where ϵ is sufficiently small, u will have no peer pointing to it any more, and all peers u is pointing to belong to A. Hence, if the peers in A leave, u will be disconnected from the system. Notice that even a relatively modest adversary can come up with such a set A, even if the hash function is not invertible. It just has to try enough values (which is easily possible with SHA-1; the fact that the hash values may depend on IP addresses is not a limitation, because with IPv6 there will be plenty of them available – even for private users). Also, notice that an adversary just has to know the value x to start an attack on u. Once an adversary has managed to carve out a peer from the system, it may park it in a bogus peer-to-peer system so that the peer does not notice being removed from the original system. In this way, the adversary can remove a large number of peers with a relatively modest amount of effort.

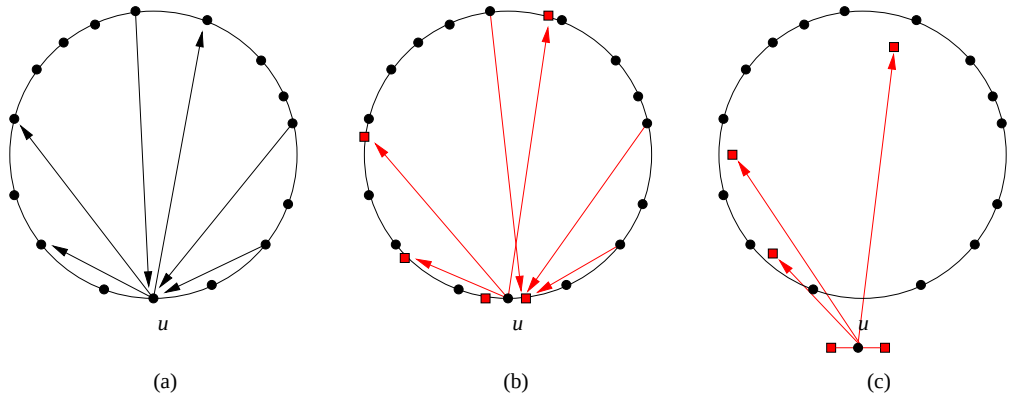


Figure 1: Removing a peer from the Chord system.

As we see next, also truly random IDs do not help as long as no node can be excluded from the system against its will, even if there is a secure mechanism for enforcing such an ID on *every* node that joins the system.

3.3 Problems with unlimited lifetime

All hash-based systems are based on the concept of a virtual space. The basic idea underlying these systems is that nodes are given virtual locations in some space, and the overlay network is constructed based on these virtual locations. That is, depending on its virtual location, a node aims to maintain connections to other virtual locations and does this by establishing pointers to the nodes closest to these locations. See, e.g. [15] for a general framework behind this approach.

Let us consider any hash-based overlay network in which instead of using a hash function, each node gets a truly random ID. Suppose that for some node v we wish to attack, there is still some region R of volume ϵ left so that if a node obtains an ID in R, then it will receive an edge to or from v. In order to get a node into region R, a single adversarial node, on expectation, only needs $1/\epsilon$ attempts. While the adversary tries to occupy other regions necessary to exclude v, it *holds on* to those of its nodes that already successfully made it into a relevant region.

Since for any fixed area of volume $1/n^2$, the probability that at least one out of n honest nodes chooses an ID in this area is at most $1/n$, an adversary will manage to take over a region R relevant for v in at most n^2/M rounds, with high probability, where M is the number of adversarial nodes. Hence, the overlay network will not survive any polynomial number of steps, and is therefore not survivable according to our definition.

The sampling approach above can also be used to gain the majority of nodes in any region, thus causing the approach of just assigning random IDs to nodes and clustering them by grouping close-by nodes together to fail. Hence, unlimited lifetime can result in a fast degradation of randomness.

4 Outline of the Group Spreading Protocol

In this section, we give an outline of the Group Spreading Protocol. The details can be found in the appendix. In the following, let N be the current number of honest nodes in the system and M be the current number of adversarial nodes in the system.

4.1 Basic approach

We start with some basic definitions. A *peer* is a user or machine that wants to participate in the peer-to-peer system, and a *node* is a logical unit in the system. A peer may have multiple nodes in the system. However, honest peers will limit their nodes to $O(\log N)$. A node is called *honest* if it belongs to an honest peer. We assume that honest nodes execute our protocols in a faithful and reliable way. Adversarial nodes may do *anything*. (Recall that we only have to worry about legal attacks because honest nodes have infinite bandwidth.)

Organize nodes in regions

Certainly, a system in which the correct execution of an overlay network operation depends on the correct behavior of individual nodes is not survivable. Hence, quorums of nodes have to be formed to check each other's behavior and therefore ensure the reliable execution of operations. However, forming separate quorums of nodes (i.e. grouping the nodes into disjoint clusters) in a distributed system is not an easy task. The problem here is that the number of clusters cannot be kept fixed if one wants the system to be scalable. Hence, once in a while clusters have to be created, deleted, split, or merged. However, since adversarial nodes can create different views of the current situation in honest nodes, it is hard to find a consensus on a cluster operation between the honest nodes, creating inconsistencies. To avoid these problems, we do not form clusters of nodes but instead allow each honest node to decide by itself which *regions* of nodes it

considers to be safe. In our case, the ID space of the nodes will be the interval $[0, 1)$, and a region may be any subinterval of length $1/2^r$ for some $r \in \mathbb{N}$ starting at an integer multiple of $1/2^r$. Overlay network operations initiated by a node will be executed on a region level. To make sure that these can be executed reliably and efficiently, we require honest nodes to maintain regions of size $\Theta((\log N)/N)$. The *home region* of a node v (i.e. the region containing $\mathsf{ID}(v)$) is denoted by R_v, and the set of nodes that v knows in R_v is denoted by C_v.

Use a limited lifetime

As we saw above, an unlimited lifetime can result in a fast degradation of randomness. Hence, we will only allow nodes to stay in the system for $O(\log N)$ steps. This is done by maintaining some parameter $L = \Theta(\log N)$ at each node and using the rule that every connection to a node in the system is dropped once that node reaches an age of L. The value of L will differ slightly between the honest nodes but we will show that this difference can be kept very small.

Use multiple nodes for each peer

Since we allow adversarial nodes to be part of the system, some of the regions used for generating new random IDs will contain adversarial nodes. In such a case, our ID generation protocol may not succeed. Thus, to make sure that sufficiently many ID generation attempts will succeed to rejoin the network, each honest peer will keep $\Theta(\log N)$ nodes at random places in the network. As we will see, the adversary will not be able (with high probability) to place nodes in each of these regions because our ID generation protocol will make sure that whenever a new ID is generated successfully, it is random.

Spread lifetimes of honest nodes

Every peer p will aim to maintain a set of nodes so that for each of the next L time units there is exactly one node of p that will depart in that unit. Together with the fact that the nodes of p have random IDs, we will see that this results in the remaining lifetimes of the honest nodes being well-spread in every region, with high probability. This allows us to pursue the following approach.

Participation in overlay network operations is not transferable

For security reasons, we do not allow a node to take over the role of another node in an overlay network operation. Hence, if the lifetime of a node expires before it completes an overlay network operation, its state will be lost. To make sure that this does not endanger the correct execution of overlay network operations, the following rule is used:

Whenever a new overlay network operation is started, an honest node v (that has completed its integration) first checks whether its remaining lifetime is sufficiently large to complete it (given that the node initiating the operation is honest). If so, it participates in it, and otherwise not.

With this rule, no honest node will depart in the middle of an overlay network operation, which could potentially create problems for this operation. Since the lifetimes of honest nodes are sufficiently well spread in every region, it holds that whenever a new overlay network operation is started, there will always be a sufficient number of honest nodes in a region that can participate in it, with high probability, so that the operation can be executed correctly.

4.2 Protocols

Next we summarize the protocols we will use to achieve the basic approach above. The following protocols have to be implemented for the user-level operations:

- **Join protocol**: This lets a new peer join the system.
- **Leave protocol**: This lets a peer leave the system.
- **Lookup protocol**: This performs the lookup of a name in the system.

Also protocols for internal operations will be needed:

- **Insert protocol**: This inserts a name entry into the system.
- **Rejoin protocol**: This lets nodes of a peer rejoin the system.
- **Region protocol**: This updates the home region of a node.
- **Lifetime protocol**: This removes a connection whenever the lifetime of a neighbor expires. Also, it controls the departure of the own nodes.

Next we discuss the basics of each of the protocols.

4.3 The Region protocol

Recall that a *region* in $[0,1)$ is an interval of length $1/2^r$ for some $r \in \mathbb{N}$ starting at an integer multiple of $1/2^r$.

Region decomposition

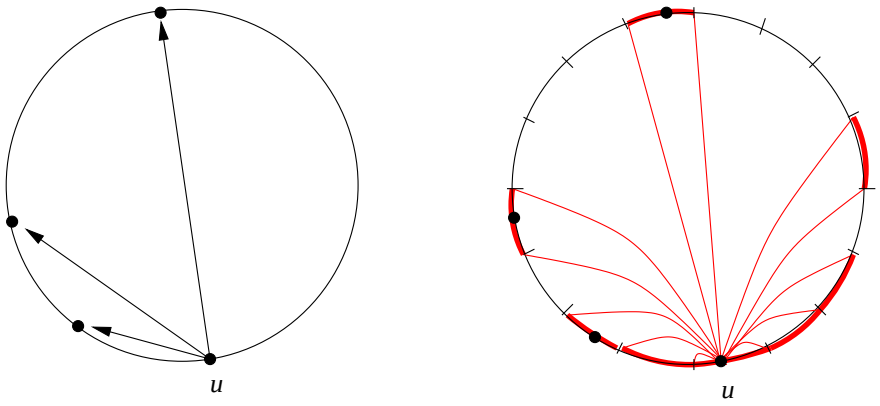
In Chord, every node v maintains pointers to nodes closest to $\mathsf{ID}(v)+1/2^i$ for all i. We need to transform this pointer structure into a more secure version. Towards this goal we proceed as follows. In our construction, v maintains some integer $r_v \in \mathbb{N}$, called *range* of v. The value of r_v may be different among the nodes, but as we shall see, the differences are small.

Given r_v and $\mathsf{ID}(v)$, we define v's *origin* as $o_v = \lfloor \mathsf{ID}(v) \rfloor_{1/2^{r_v}} + 1/2^{r_v+1}$, where $\lfloor \cdot \rfloor_x$ rounds to the nearest integer multiple of x from below. v's *region decomposition* of $[0,1)$ as the partition of $[0,1)$ into sub-intervals $[j/2^{r_v}, (j+1)/2^{r_v}]$ with $j \in \mathbb{N}$. Now, instead of pointing to node w with $\mathsf{ID}(w)$ being the closest match to $\mathsf{ID}(v) + 1/2^i$, we define the region $R^+_{v,i}$ as the unique sub-interval from v's region decomposition that contains $o_v + 1/2^i$ and the region $R^-_{v,i}$ as the unique sub-interval that contains $o_v - 1/2^i$. The set of all these regions, together with the region $R_{v,0}$ (or simply R_v) containing $\mathsf{ID}(v)$, is called $\mathcal{R}_v$. v aims to keep pointers to every node w with $\mathsf{ID}(w) \in \mathcal{R}_v$. See Figure 2 for an illustration.

The set of all nodes in a region $R \in \mathcal{R}_v$ which v has pointers to is called its *view* of R and denoted by C^v_R. If R is v's home region, i.e. the region covering v itself, we will also denote C^v_R by C_v.

Range computation

$\mathcal{R}_v$ will remain the same throughout v's lifetime. However, v will inspect its home region to find out whether future nodes inserted by v need a range different from v. For this v needs the Region protocol. Let $C_v(r)$ denote that unique region of size $1/2^r$ covering node v, and let $\gamma > 0$ be a fixed parameter (which is the same for all nodes in the system). Then the protocol simply works as follows:



(a) (b)

Figure 2: The left picture shows a node u with three fingers in the original Chord network, and the right picture shows the region decomposition of u together with the regions to which u has pointers.

Select r so that $|C_v(r)|$ is as close as possible to $\gamma(r - \log r)$.

The next lemma bounds the deviation of the range values among the nodes in an ideal situation.

Lemma 4.1 *If γ (as specified above) is a sufficiently large constant, then under the assumption that each node v has a random $\mathsf{ID}(v)$ and that each node v knows the location of every other node in its region R_v, the ranges of the nodes are by at most 1 apart with high probability.*

Proof. Let N be the current number of nodes in the system. Suppose for the moment that we use the rule that ideally γr (instead of $\gamma(r - \log r)$) nodes should be in a region of size $r/2^r$ (instead of $1/2^r$). Consider some region R of size $r/2^r$ in which the number of nodes is $(1+\epsilon)N \cdot r/2^r$. Then it holds that

$$\gamma r = \frac{(1+\epsilon)N \cdot r}{2^r} \quad \Leftrightarrow \quad r = \log \frac{(1+\epsilon)N}{\gamma} \, .$$

This implies that there are

$$\frac{(1+\epsilon)N \cdot r}{2^r} = \gamma \log \frac{(1+\epsilon)N}{\gamma}$$

nodes in a region of size

$$\frac{r}{2^r} = \frac{\gamma}{1+\epsilon} \cdot \frac{\log(\frac{(1+\epsilon)N}{\gamma})}{N} \, .$$

For any constant $0 < \delta < 1$, one can choose a constant γ so that according to the Chernoff bounds, for ϵ outside of $[-\delta, \delta]$ such a region will not exist with high probability. Hence, with high probability,

$$r \in \left[\left\lfloor \log \frac{(1-\delta)N}{\gamma} \right\rfloor , \left\lceil \log \frac{(1+\delta)N}{\gamma} \right\rceil \right]$$

which is an interval of size at most 1 if δ is sufficiently small. Substituting now r by $\log(2^r/r)$, we get our original rule that we want to have $\gamma(r - \log r)$ honest nodes in a range of size $1/2^r$, and also for this r, the deviation among the nodes is at most 1 with high probability. This completes the proof. □

Notice that Lemma 4.1 still holds if the honest nodes have different views but every honest node knows (almost) every other honest node and the fraction of adversarial nodes is sufficiently small in a region, because then the views of the honest nodes only differ by a small constant factor. Keeping the join/leave rate of honest nodes limited to $\epsilon/\log N$ for some sufficiently small ϵ, a deviation by an additive 1 can even be ensured over L steps, where L is the maximum lifetime of a node. This simplifies the protocol design. However, with slightly more complex protocols one can also tolerate larger deviations, thereby allowing higher join/leave rates. More details are in the appendix.

4.4 The Lifetime protocol

The lifetime L_v of a node v is determined at the point when it is integrated into the system, using the formula

$$L_v = \lambda \cdot r_v$$

for some constant λ that is the same for all nodes. r_v is v's range when v joins the system. From our remarks in the previous subsection we can conclude that L_v will deviate by at most λ among the honest nodes. L_v represents the maximum possible lifetime of v. The peer p owning v will use the following rule for assigning a lifetime to v bounded by L_v:

p maintains a queue Q_p containing all of its current nodes. The node in slot i still lives for i time steps before it has to leave the system. p's aim will be to continuously generate new nodes with random IDs, but only if it can find an empty slot i in Q_p for some prospective node v with maximum lifetime L_v where $i \leq L_v$. If so, p will go ahead with a random ID generation for v and, if successful, will assign v to slot i. Then p will integrate v into the system.

The ID generation and integration will be done with the help of the Rejoin protocol. p will start sufficiently many rejoin operations so that the lifetime of a new node v will always be sufficiently close to L_v, i.e. Q_p will always contain sufficiently many nodes.

The lifetime rule and Lemma 4.1 make sure that p will have $\Theta(\log N)$ nodes in the system at any time and the remaining lifetimes of the honest nodes are uniformly distributed. Both are important prerequisites for the overlay network operations to be executed correctly and efficiently. In each time unit, p takes the node in front of the queue and executes the Leave operation for it.

Besides its own lifetime, an honest node v will also keep track of the lifetime of its connections, i.e. for every link to some node w, v will keep a time stamp T_w which approximates the time point when w joined the system. Once the current time step t exceeds $T_w + (1+\tau)L_w$, v will cut its connection to w, where $(1+\tau)$ is the maximum factor by which the clocks of two honest nodes deviate.

4.5 The Rejoin protocol

Besides queue Q_p, peer p also maintains a queue S_p. For every node v that p has newly integrated into the system, p will check whether $|S_p| < L_v/3$. If so, p will assign v to the lowest available slot in S_p. In each time unit, p will remove the lowest slot from S_p and initiates the Rejoin protocol in the node, u, stored in that slot. Based on the available space in Q_p, u will execute (up to) three times in parallel the following three stages, which represent the Rejoin protocol:

1. **Random ID generation**: Node u contacts all nodes in its home region R_u. (Recall that the set of these nodes is denoted by C_u.) Each honest node $v \in C_u$ that has sufficient lifetime left to complete the Rejoin operation chooses a random value $x_v \in [0,1)$ and sends $h(x_v)$ to all nodes in C_u, where h is the result of a bit commitment scheme. This allows v to commit to a value without revealing it. v then waits for $L_u/3$ steps, and during these steps v accepts commitments from other nodes. Once this is over, and u requests v to reveal its key (because there is room in Q_p), v will send x_v to all nodes

in C_u. Once v has received all x_w's of the $h(x_w)$'s it has received earlier, it computes $x = \bigoplus_w x_w$ and sends x back to u. If u has received at least $|C_u|/5$ times the same x, u adds a new node with ID x and maximum lifetime $L(r) = \ell \cdot r$ to the slot reserved for that node in Q_p, where r is the *current* range view of u.

2. **Authorization**: Afterwards, u sends an authorization request to all nodes in C_u that sent x to u. The nodes will then send a message to the unique region of size $1/2^r$ owning x. Once the region has replied with an acknowledgement containing its set of nodes, say C, the nodes in C_u forward C to u so that p knows whom to contact in order to integrate a new node u' with $\mathsf{ID}(u') = x$ into the system. (There may be discrepancies in the view of C, but we will address them in Section A.)

3. **Integration**: Finally, p will contact C to obtain information about the connections of the nodes in C so that it can establish all connections necessary for u'. (Notice that the authorization makes sure that C knows u' and u' knows C. Furthermore, the integration will ensure that the neighbors of C know u' and u' knows the neighbors of C so that u' can be fully integrated.)

Once a node completes the integration stage, it is called *mature*. *An honest node will only participate in other overlay network operations once it is mature.* To make sure that the rejoin rate is limited to $O(1/\log N)$, each node (except for the gateway nodes) is allowed to execute the Rejoin protocol at most once during its lifetime. Rejoin requests beyond that number will not be accepted by the other nodes. Notice that each rejoin request needs $\Theta(\log N)$ steps to be processed, requiring an adversarial node to stay in the system for $\Omega(\log N)$ steps if it wants its request to succeed.

4.6 The Join protocol

Peers may not know anyone in the peer-to-peer system when they want to join it. Hence, we assume that a limited number of so-called *gateway peers* are available that are searchable over the Internet and always part of the system. Gateway peers are like normal peers, with the only difference that they can initiate a potentially infinite number of rejoin operations.

Each new peer p contacts one of these gateway peers, say q. q then executes the Rejoin operation at each of its nodes to get p integrated into the system. (To make sure that this can not be abused by adversarial nodes, i.e. the arrival rate of adversarial nodes is bounded, each peer p that wants to join may have to solve a computational challenge or a Turing test before being admitted by the honest nodes.)

4.7 The Leave protocol

The Leave protocol is fairly simple. Once a peer p decides to leave, no new nodes will be inserted into the network by p and p waits until the lifetime of all of its nodes currently in the network has expired.

4.8 The Insert protocol

To be able to explain the Insert protocol, we first describe how to select a path in our framework.

Consider an operation $\mathsf{Insert}(z)$ from node v in the original Chord system for some key $z \in [0, 1)$. Let o_v be the origin of v as defined in Section 4.3, and let the IDs of the virtual locations traversed by Insert in Chord form the path

$$p(v, z) = (x_0 = o_v, x_1, \ldots, x_{\ell-1}, x_\ell = z)$$

Since this path narrows down on z in an exponential way, paths in Chord have a length of $O(\log n)$, w.h.p., which is important for our strategies to work. In the Group Spreading framework, this path translates into a sequence of regions $R_0, R_1, \ldots, R_\ell$ where R_i, $i < \ell$, is the unique region of size $1/2^{r_v}$ that contains x_i

and R_ℓ is the unique region containing z. This is also called the r_v-*path* of (v, z). Correspondingly, the Lookup protocol for some system node v and name (resp. hash value of the name) z and IP address i works as follows:

1. Search for the region owning z: v first sends the insert request to all nodes in R_v who will then send it further along an r_v-path till the requests reach the region R responsible for z.

2. Insert (z, i) in R: all honest nodes in R keep the entry (z, i) stored till they leave (i.e. their lifetime expires).

Nodes will not pass their data on to other nodes when they leave. Hence, information has to be refreshed to prevent it from being washed out. Therefore, every honest peer has to initiate the Insert protocol every $L/3$ steps. Though this creates extra overhead, it has the advantage that adversarial nodes cannot accumulate an unbounded number of name entries in the system, protecting it from getting overloaded. Also, this mechanism makes a Delete protocol obsolete.

4.9 The Lookup protocol

The Lookup protocol for some system node v and key z works in two stages:

1. Search for the region owning z: v first sends the lookup request to all nodes in R_v who will then send it further along an r_v-path till the requests reach the region responsible for z.

2. Delivery of IP address: The nodes in the region owning z will send the IP address associated with z back to v, using again an r_v-path.

4.10 Basic invariants

Among other aspects, the following invariants have to be preserved for the Group Spreading protocol to be survivable. We call a node a *legal member* if at least one honest node in the system has a pointer to it.

1. Every legal member has a random ID, and this random ID was generated at most $O(\log N)$ steps before the node became a legal member.

2. For every legal member v, all honest nodes that have pointers to v have the same view of $\mathsf{ID}(v)$ and the age of v (up to minor deviations), and this view has not changed since v joined the system.

3. For every mature honest node v it holds that v is connected to all other mature honest nodes in its regions.

More details can be found in the appendix.

References

[1] B. Awerbuch and C. Scheideler. Chord++: Low-congestion routing in chord. Technical report, See http://www.cs.jhu.edu/∼scheideler, 2003.

[2] R. Canetti, R. Gennaro, A. Herzberg, and D. Naor. Proactive security: Long-term protection against break-ins. *RSA CryptoBytes*, 3(1):1–8, 1997.

[3] Castro and Liskov. Practical byzantine fault tolerance. In *OSDI: Symposium on Operating Systems Design and Implementation*. USENIX Association, Co-sponsored by IEEE TCOS and ACM SIGOPS, 1999.

[4] S. Crosby and D. Wallach. Denial of service via algorithmic complexity attacks. In *Usenix Security*, Aug 2003.

[5] J. R. Douceur. The sybil attack. In *In Proc. of the IPTPS02 Workshop, Cambridge, MA (USA)*, 2002.

[6] A. Fiat and J. Saia. Censorship resistant peer-to-peer content addressable networks. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2002.

[7] Y. Frankel, P. Gemmell, P. D. MacKenzie, and M. Yung. Optimal resilience proactive public-key cryptosystems. In *IEEE Symposium on Foundations of Computer Science*, pages 384–393, 1997.

[8] T. Hagerup and C. Rüb. A guided tour of Chernoff bounds. *Information Processing Letters*, 33:305–308, 1989/90.

[9] S. Halevi and S. Micali. Practical and provably-secure commitment schemes from collision-free hashing. In *CRYPTO 96*, pages 201–215, 1996.

[10] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung. Proactive public key and signature systems. In *ACM Conference on Computer and Communications Security*, pages 100–110, 1997.

[11] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing or: How to cope with perpetual leakage. *Lecture Notes in Computer Science*, 963:339–??, 1995.

[12] L. Lamport. The weak Byzantine generals problem. *Journal of the ACM*, 30(3):669–676, 1983.

[13] L. Lamport and N. Lynch. Distributed computing. Chapter of Handbook on Theoretical Computer Science. Also, to be published as Technical Memo MIT/LCS/TM-384, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1989.

[14] M. Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.

[15] M. Naor and U. Wieder. Novel architectures for p2p applications: the continuous-discrete approach. In *SPAA Proceedings*, 2003.

[16] R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. In *In Proceedings of the 10th Annual ACM Symposium on Principles of Distributed Computing*, pages 51–59, 1991.

[17] R. D. Prisco, B. W. Lampson, and N. A. Lynch. Revisiting the paxos algorithm. In *Workshop on Distributed Algorithms*, pages 111–125, 1997.

[18] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Heidelberg, Germany, 11 2001.

[19] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01*, San Diego, CA, 8 2001.

[20] B. Y. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. In *UCB Technical Report UCB/CSD-01-1141*, 2001.

A Details of the Group Spreading Protocol

In the following, we concentrate on a formal presentation of those protocols that are crucial for the Group Spreading approach, which are the Insert, Lookup, and Rejoin protocol. This is followed by an analysis of the survivability of the system.

A.1 Protocol notation

Since our protocols are highly interactive, we found it more convenient to extend the notation in the cryptography literature rather than using abstract protocol notation such as Estelle, Esterel, or Promela. Our notation is rather simple. If node v wants to send messages $M_1, \ldots, M_k$ to w, we write "send to $w : (M_1, \ldots, M_k)$". The execution of a process on node v is specified by v $[\langle \text{conditions} \rangle] : \langle \text{action} \rangle$. The conditions specify the events that trigger the execution of the action. A label of the form (ℓ) in the conditions specifies that a correct message of label (ℓ) must have been received. By "correct" we mean that the form, contents, and signatures in the message are correct (as far as this can be verified by the node).

A.2 The Insert and Lookup protocol

The protocols for Insert and Lookup can be found in Figure 3.

A.3 The Rejoin protocol

The different stages of the Rejoin protocol can be found in Figures 4, 5, and 6.

A.4 Analysis

We start with some notation. A node is called *legal* if at least one honest node in the system has a pointer to it. A node is called *mature* once it completes the (re)join operation. Notice that a node may be legal before it is mature. The *age* of a node is the time that has passed by since it initiated stage 3 of the Rejoin protocol. A view of region R is called *legal* if it contains only legal nodes in R and it contains all mature honest nodes in R. For any time point t, we define $\ell_t = \min_v L_v$ and $L_t = \max_v L_v$ where the $\min$ and $\max$ is taken over all mature honest nodes that are in the system at time t. If t is clear from the context, we will omit it. A system is called *safe* at time t if in every region R with $|R| \geq (\gamma \log N)/N$ for some sufficiently large constant γ

1. the number of honest nodes that are legal members in R is at least $(2/3)\gamma \log N$ and at most $(4/3)\gamma \log N$,
2. for any interval I in $[0, L]$ of size $\ell/2$, the number of mature honest nodes in R that have an age in I is at least $(1/3)\gamma \log N$, and
3. the number of adversarial nodes that are legal members in R is at most $(1/30)\gamma \log N$.

The main theorem of this paper is:

Theorem A.1 *The Group Spreading protocol makes Chord survivable.*

A.5 Proof of Theorem A.1

For this we first need to show:

Lemma A.2 *For* any *join/rejoin/leave sequence of honest and adversarial nodes over a number of time steps polynomial in n in which*

- *the number of honest nodes is at least n at any time,*
- *the join/rejoin/leave behavior of honest nodes satisfies the protocol conditions,*
- *the gateway nodes are honest,*
- *the rate of join/leave requests of honest nodes is at most $\epsilon/\log N$,*

- *the rate of join requests of adversarial nodes is at most $\epsilon/\log^2 N$, and*
- *the adversarial nodes never represent more than an $\epsilon/\log N$ fraction of the honest nodes in the overlay network,*

the peer-to-peer system constructed by the Group Spreading protocol will be safe, with high probability.

Suppose that this is true. Then we can show that all overlay network operations initiated by honest peers are executed successfully, which would imply Theorem A.1. For this we first have to define what we mean by "successful".

- A join request by some peer p is called *successful* if p manages to integrate at least ℓ nodes into the system within $O(\log N)$ steps that satisfy the Invariants in Section A.6.
- An insert request by some peer p is called *successful* if the node u of p initiating that operation manages to store p's name x in $O(\log N)$ steps in every honest node in the region R of size $1/2^{r_u}$ containing x that was mature when u initiated the operation.
- A lookup request by some peer p for some name x is called *successful* if p receives the IP address of the peer with name x within $O(\log N)$ steps.

We prove these properties in several claims and lemmata. We start with two statements that we cannot show here since we first have to introduce some invariants for these. Their proofs can be found in Section A.6.

Lemma A.3 *As long as the system is safe, any join request by some honest peer is successful.*

The next statement gives a bound on the deviation of N, the number of nodes in the system.

Claim A.4 *For any constant $0 < \rho < 1$ there is a constant $0 < \epsilon < 1$ so that the number of nodes in the system changes by a factor of at most $(1 + \rho)$ in L steps, where L is the maximum over all L_t during these steps.*

Hence, to simplify the statement of the claims below, we will assume in the following that as long as we just look at time intervals of size at most L (where L is the maximum over all L_t during these steps), N is fixed. However, whenever the exact value of N matters in the proofs, we will consider the deviations, as done in the following claim.

Claim A.5 *As long as the system is safe it holds that for any region R of size at least $(\gamma \log N)/N$ and any node with a legal view of R, the number of honest nodes in R that were mature $t \leq L/2$ steps ago is at least $|C_R^v|/5$ and the number of adversarial nodes in R is at most $|C_R^v|/20$.*

Proof. From the safeness condition we know that the number of mature honest nodes in R that were already in R $t \leq L/2$ steps ago is at least $(1/3)\gamma \log N$ for the number N that was true t steps ago. According to Claim A.4, this number is still at least $1/(3(1 + \rho)) \cdot \gamma \log N$ after at most $t \leq L/2$ steps, compared to the current N. Because at the current time the number of honest nodes is at most $(4/3)\gamma \log N$ and the number of adversarial nodes is at most $(1/30)\gamma \log N$, it follows that if ρ is sufficiently small, the number of honest nodes in R that were mature t steps ago is still at least $|C_R^v|/5$.

The upper bound on the adversarial nodes follows from the fact that the number of honest nodes is at least $(2/3)\gamma \log N$ and the number of adversarial nodes is at most $(1/30)\gamma \log N$. □

Claim A.6 *The adversarial nodes cannot forge an insert or lookup request from an honest node.*

Proof. Suppose that the adversarial nodes make some honest node v accept a request M from some other honest node u. Further suppose that v is the first honest node accepting M and it is in a region R_k representing the kth region along the path of the request from u. If $k > 1$, then v must have received M from at least a $1/5$ fraction of the nodes in its view of $R_{k-1}(r)$ where $r \in \{r_v, r_v + 1\}$. However, due to the safeness of the system, there must have been an honest node v' in $R_{k-1}(r)$ that has already accepted M before v, creating a contradiction. Hence, assume that $k = 1$. In this case, v would have accepted a message M from u although it did not come from u. But according to our model, which assumes that the source of a message cannot be forged, this cannot happen, also creating a contradiction. □

Claim A.7 *As long as the system is safe and the distance d the request has to travel to reach some region R is at most $L/3$, it takes at most d steps for an insert or lookup request initiated by a mature honest node to reach all (remaining) honest nodes in R that were mature when the request was initiated, and the request arrives at these nodes in an unchanged form.*

Proof. We prove the lemma by induction over the number of hops. If some honest node v initiates some request M, then it follows from our delay assumptions that it takes at most 1 step until all honest nodes in R_v receive M. So suppose that it has already been shown that it takes at most k steps until all mature honest nodes in R_k, the kth region along the path of the request, received the original M from all honest nodes in R_{k-1} that were mature when the request was initiated.

Suppose that the request was initiated at step t and let γ be chosen so that $1/2^{r_v} = (\gamma \log N)/N$. From Claim A.5 we know that the number of mature honest nodes in R_{k-1} and any sufficiently large subregion of it that were in the system at time t is at least a fraction of $1/5$ of all the nodes in any legal view of that region. Hence, at least a $1/5$ fraction forwards the original M in at most k time steps to all the mature honest nodes in R_k. Since the adversarial nodes represent at most $1/20$ of the nodes in R_{k-1} and any of its sufficiently large subregions at any time, it follows that when an honest node v waits until it has received the same request from at least a $1/5$ fraction of nodes its view of the previous region of the request (which includes R_{k-1} or one of its subregions), every honest node in R_k that was already mature at time t will accept the original message M in at most k steps. Therefore, it takes at most $k+1$ steps until all mature honest nodes in R_{k+1} receive and accept the original request M. This completes the induction and therefore the proof. □

Two remarks are in order for this claim. First, notice that it is really sufficient to just guarantee at most $(1/30)\gamma \log N$ adversarial nodes at any time since honest nodes will *only* consider messages in any threshold rule that are from nodes that are still legal members of the system. Second, one may think that for every region along the path the honest nodes will accept a message M at basically the same time. However, since we do not assume a lower bound on the transmission delay of messages, in the worst case the adversarial nodes can cause time differences between the messages from honest nodes that grow linearly with the distance, forcing us to resort to honest nodes that were mature when the request was initiated.

Lemma A.8 *As long as the system is safe, any insert operation executed by an honest node is successful within $O(\log N)$ steps.*

Proof. Suppose that some honest node u initiates an insert operation for name x. From Claim A.7 it follows that an insert request only takes $O(\log N)$ time to reach its destination in its original form. There, x will be stored in all honest nodes that were mature when the request was initiated, and hence the insert request is successful. □

Lemma A.9 *As long as the system is safe, any lookup operation executed by an honest node is successful within $O(\log N)$ steps.*

Proof. Recall that at latest every $L/3$ steps a peer p will refresh its name x. Since refreshing the name via an insert operation only takes $O(\log N)$ steps, it follows from Claim A.5 that for the next $L/3$ time steps the fraction of mature honest nodes storing x compared to the total number of nodes in any view of an honest node of the region of x is at least $1/5$. Hence, it follows from Claim A.7 that any lookup request for some name x will successfully reach the name of x and its answer will successfully travel backwards in $O(\log N)$ steps. □

A.6 Proof of Lemma A.2

We prove the lemma by induction on the time the system has run so far. We assume that the system initially consists only of honest nodes, and the number of these nodes is at least n, the parameter for our probability bounds. In this case, it is easy to show that the system has been safe so far. So we can proceed with the induction step. For the induction step to work, we want to show apart from safeness that the following invariants are fulfilled:

1. Every legal member has a random ID, and this random ID was generated at most $O(\log N)$ steps before the node became a legal member.

2. For every legal member v, all honest nodes that have pointers to v have the same view of $\mathsf{ID}(v)$ and the age of v (up to minor deviations), and this view has not changed since v joined the system.

3. For every mature honest node v it holds that v is connected to all other mature honest nodes in its regions.

Our strategy is to show that if the system has been safe and the invariants have been true so far, the invariants also hold for subsequent (re)join/leave operations, which then can be used to show that the system remains safe, with high probability.

We need a couple of claims to prove the invariants, which are based on the assumption that the system has been safe so far.

Claim A.10 *For every honest node v accepting an ID x (by executing step (6b)) in an ID generation stage of some node u it holds that x must be random and the generation took at least $(1-\tau)L_u/3$ steps and at most $(1+\tau)L_u/3 + O(1)$ steps from v's point of view.*

Proof. According to the ID generation protocol, v must have contributed a random key x_v to x, and once v revealed x_v it did not accept any further commitments from other nodes. Hence, $x = \bigoplus_w x_w$ (where $\oplus$ denotes the bit-wise XOR) must be random. Since v waits for at least $(1-\tau)L_u/3$ steps before revealing x_v and at most $(1+\tau)L_u/3 + O(1)$ steps before it does not execute (6b) any more, the claim follows. □

Claim A.11 *For every honest node v adding an entry $(x, u, r, j, T_v, 1)$ to J_v it holds that there must have been an honest node w accepting x in the ID generation stage at most $O(\log N)$ steps before.*

Proof. Suppose there is such a node w, and suppose that w is part of region R_i. Review stage 2 of the Rejoin protocol. Then w must have added an entry $(x, u, r, j, T_w, 0)$ to J_w before. Continuing with this backwards in time, we distinguish between two cases.

If $i = 0$ (i.e. the new ID x is in $R_u(r)$), then w must have received message (v, auth, $(s,t), x, C_v(r), u, r, j$) from at least $3|C_w(r_u)|/20$ nodes in $C_w(r_u)$. Hence, there must have been an honest node w accepting x in stage 1.

So assume that $i > 0$. In this case, w received message (v, auth, $(s,t), x, A_v, u, r, j$) from at least $|C^w_{R_{i-1}}(r+1)|/5$ nodes in $C^w_{R_{i-1}}(r+1)$. (w can really check this because due to our discrepancy assumption on the rates, $r \geq r_w - 1$.) In this case, there must have been an honest node v in $R^w_{i-1}(r+1)$ that successfully received an authentication message from a previous region. Continuing with this argument by induction, we end up with some honest node w in region R_0 that must have received message (v, auth, $(s,t), x, C_v(r), u, r, j$) from at least $3|C_w(r_u)|/20$ nodes in $C_w(r_u)$. Hence, again there must have been an honest node w accepting x in stage 1.

It remains to bound the runtime of this process. For this we first have to show that for any node honest $w \in R_i$, the set A_w computed in (3) is always a legal view of $R_u(r)$. This can be done by induction on i. For $i = 0$, w received A_v from at least $3|C_w(r_u)|/20$ nodes v in $C_w(r_u)$. If w includes a node w' in A_w, then w' must be in A_v for at least $|C_w(r_u)|/10$ of these v's, and therefore w' must be part of a view of $C_v(r)$ of some honest node v. Thus, w' must be a legal member. On the other hand, every mature honest node in $C_v(r)$ will be included in A_w because an honest node in $C_w(r_u)$ sends an authentication message to w only if it is mature, and these mature honest nodes know all other mature honest nodes in $R_u(r)$.

Now, w only changes $(x, u, r, j, T_w, 0)$ in J_w to $(x, u, r, j, T_w, 1)$ if it has received messages (v, accept, (u, j)) from at least $3|A_w|/20$ nodes in A_w. Since A_w is a legal view of $R_u(r)$, this means it has received an accept message from at least one honest node in $R_u(r)$. But an honest node in $R_u(r)$ only sends an accept message if the time that passed by till it executed (2) in stage 2 is at most $\delta \log N$. Furthermore, an honest node only executes (2) if it accepted the new ID (in (6b) of stage 1) at most δ steps before. Hence, the time between the point when w received the message from the honest node and the point where an honest node accepts x in the ID generation is at most $O(\log N)$. Since w also uses the rule $\bar{T} - T_w < \delta \log N$ to convert $(x, u, r, j, T_w, 0)$ into $(x, u, r, j, T_w, 1)$ and there must have been an honest node sending an accept message to w after $\bar{T}$, the claim follows. □

Claim A.12 *For every honest node v establishing a connection to some new node u with ID x and lifetime t in its region it holds that v must have added an entry $(x, u, r, j, T_v, 1)$ for u in J_v at most $O(\log N)$ steps before.*

Proof. Follows immediately from the condition in step (2) of stage 3. □

Claim A.13 *For every mature honest node v establishing a connection to some new node u that did not add an entry for u in J_v it holds that v must have received a message from an honest node v' that added an entry $(x, u, r, j, T_v, 1)$ for u to J_v at most $O(\log N)$ steps before.*

Proof. Honest nodes only establish connections to new nodes in step (4) of stage 3. Hence, consider some honest node v doing this. Then it must have received messages (w, new_node, u', x, r, j) from at least $3|C^v_{R_{u'}}(r+1)|/20$ nodes w in $C^v_{R_{u'}}(r+1)$ for some $r \leq r_v + 1$. In this case, at least one of the nodes w must be honest, and this node must have executed step (2) in stage 3, i.e. it has an entry $(x, u, r, j, T_w, 1)$ for u in J_w. Also, there must be an honest node whose message arrived at v after $\bar{T}$, and therefore the delay between some honest w executing (2) and v executing (4) is at most δ. Furthermore, w only sends messages (w, new_node, u', x, r, j) if the time since it added $(x, u, r, j, T_v, 1)$ for u to J_v is at most $\delta \log N$. Hence, the time bound of the claim holds. □

Claim A.14 *For every new honest node v establishing a connection to some node u with ID x and age t it holds that there is a mature honest node that established a connection to u in (2) of stage 3 at least $(1-\tau)t - O(1)$ steps before and there is a mature honest node that established a connection to u in (2) of stage 3 at most $(1+\tau)t + O(1)$ time steps before.*

Proof. Recall that honest nodes take the generation of other honest nodes into account when accepting an ID and an age (see step (6) in stage 3). An honest node v will always pick the minimum generation for which it can receive from at least $1/5$ of the nodes of its view of u's region the same x for u. Using the safeness property, v never has to go beyond a generation of 3, where the generation 0 are the honest nodes w that added u to J_w. v is of generation i if it considered a $1/5$ fraction for generation $i-1$. So suppose that v is of generation i. Because of the safeness condition it follows that there must have been an honest node w of generation $i-1$ reporting an age below v's median for u and an honest node w' of generation $i-1$ reporting an age above v's median. Let us ignore for the moment deviations in the time clocks and message delays. In this case, continuing with the argument above for earlier generations, there must have been an honest node w of generation 0 reporting an age below v's median and an honest node w' of generation 0 reporting an age above v's median. Hence, when taking deviations in the time clocks and message delays into account, w must have established a connection to u at most $(1+\tau)t + O(1)$ steps before and w' must have established a connection to u at least $(1-\tau)t - O(1)$ time steps before, which completes the proof. □

This claim yields the following claim.

Claim A.15 *For every node u, all honest nodes v that established a connection to u have views of the age of u that are in $[(1-\tau)t - O(\log N), (1+\tau)t + O(\log N)]$ for some t. If u is honest, then all honest nodes v that established a connection to u have views of the age of u that are in $[(1-\tau)t - O(1), (1+\tau)t + O(1)]$ where t is u's own view of its age, i.e. u initiated the integration stage t steps ago.*

Proof. We start with the first statement. Certainly, an honest node v only accepts an integration request by some node u in (2) of stage 3 if v added an entry for u into J_v at most $O(\log N)$ steps before. If v did this, then we know from Claim A.11 that there must have been an honest node w that accepted the ID x for u at most $O(\log N)$ steps before. w only accepts the ID (i.e. it executes (6b) in stage 1) if it has already executed (3a) in stage 1, which it only does once $|K_{u,j}| \geq |C_w(r_u)|/5$. In this case, w must have received commitments from $\geq 3|C_w(r_u)|/20$ mature honest nodes in $C_w(r_u)$ in a time interval of size at most $T = (1+\tau)L_u/3 + O(1)$. Thus, due to the safeness of the system, every mature honest node v that was already around T steps ago will have received at least $|C_v(r_u)|/10$ commitments from other mature honest nodes before w executes (3a). This means that every one of these mature honest nodes will set $i_u = 1$ at most δ steps after w executes (3a). There are at least $|C_{v'}(r_u)|/5$ of these nodes from the view of every mature honest node v'. Hence, every honest node v' that was not mature T steps before will either receive $i_u = 1$ from at least $|C_{v'}(r_u)|/10$ honest nodes in step (7) of stage 3 or (5a)-messages from at least $|C_{v'}(r_u)|/10$ honest nodes in stage 1. In either case, v' sets $i_u = 1$. Hence, no honest node will accept another rejoin request from u $O(1)$ steps after w executes (3a). Furthermore, any honest node can only accept an ID after w revealed its key, and honest nodes wait for at least $(1-\tau)L_u/3$ steps before revealing their keys and at most $(1+\tau)(L_u/3 + \delta)$ before aborting the protocol. Hence, the time deviation among the honest nodes that accept ID x can be at most $O(\delta)$, which together with Claim A.14 finishes the proof of the first statement.

The second statement follows from the fact that an honest node u sends an integration message to all honest nodes v with an entry about u in J_v within one time unit. Hence, all of these nodes start their counters with a difference of at most one time step. □

As a result of the claims above, we get that the invariants 1 and 2 are fulfilled. Thus, it remains to prove invariant 3.

Claim A.16 *For every honest node u that successfully completes the ID generation stage for some ID x it holds that at least $3|C_u|/20$ mature honest nodes capable of completing the rejoin operation accepted x, and the ID generation stage took at most $O(\log N)$ steps. Furthermore, if R_u only contains legal members that are honest, then u is guaranteed to succeed with the ID generation.*

Proof. Recall the safeness condition and Claim A.5. Suppose that an honest node u has initiated an ID generation. Since invariant 3 was fulfilled for u when it did this, u contacted mature honest nodes that can participate in the join operation that represent at least $|C_u|/5$ of the nodes of u's view of R_u throughout the execution of the rejoin operation. Hence, if u's region only contains honest nodes, u is guaranteed to succeed. In general, if u successfully completes the ID generation stage, then it has received answers with the same ID x from at least $|C_u|/5$ nodes in R_u within $O(\log N)$ steps. Thus, according to the safeness condition, at least $3|C_u|/20$ mature honest nodes in R_u accepted x, completing the proof. □

Claim A.17 *If an honest node u successfully completes the ID generation, then it also successfully completes the authorization stage in $O(\log N)$ steps.*

Proof. From the previous claim we know that if u successfully completes the ID generation stage for some ID x, then at least $3|C_u|/20$ mature honest nodes capable of completing the rejoin protocol accepted x. Hence, once u starts the authorization stage, at least $3|C_u|/20$ honest nodes w will execute (2).

Thus, all mature honest nodes w in $C_{v,1}(r)$ and $C_{v,2}(r)$ (see the protocol) will receive at least $3|C_u|/20 > |C_w(r_u)|/10$ authentication messages of the correct form for u and therefore forward the authentication message to $C^w_{R_1}(r+1)$. Since there are at least $|C^{w'}_{R_1}(r+1)|/5$ honest nodes forwarding the message further, all honest nodes w' in R_2 will receive at least $|C^{w'}_{R_1}(r+1)|/5$ authentication messages of the correct form for u. Thus, they will accept the message and send it further. Hence, by induction, at least $|C_{R_x}(r+1)|/5$ mature honest nodes v in both subregions of x's region R_x of size $1/2^{r+1}$ capable of completing the rejoin operation will add an entry for u in J_v.

Now, all of these honest nodes will report their views of $R_x(r+1)$ backwards. Consider any mature honest node w in $R'_1(r+1)$. First of all, w will receive views of (its relevant subregion) $R_x(r+1)$ (of R_x) from at least $|C_{R_x}(r+1)|/5$ honest nodes in $R_x(r+1)$. Hence, w will produce a view C and send it further to all mature honest nodes in $R'_2(r+1)$. C is guaranteed to be legal because every node w' can only be added to C if it occurs in at least $3|C_{R_x}(r+1)|/20$ views, and therefore in a view of an honest node in $R_x(r+1)$. On the other hand, every mature honest node w' will be in C because w will consider views from at least $3|C_{R_x}(r+1)|/20$ mature honest nodes in R_x and all of these, by invariant 3, know all other mature honest nodes in $R_x(r+1)$. Hence, C is legal and contains all honest nodes w' in $R_x(r+1)$ that added an entry for u into $J_{w'}$. By induction, this can be shown for all honest nodes on the path backwards to u. Hence, every honest node v that sent $(2a)$ will compute sets $A_{v,1}$ and $A_{v,2}$ for the two subregions of R_x that are legal.

Because there are at least $3|C_u|/20 > |A_w|/10$ nodes v that sent $(2a)$ from the viewpoint of every honest node w in R_x (note that A_w is a legal view of C_u), and according to Claim A.7 it only takes $O(\log N)$ steps for the routing part to succeed, every honest node w that added an entry for u to J_w will execute $(6a)$. Since there must be at least $3|A_{v,1}|/20$ and at least $3|A_{v,2}|/20$ such nodes from the viewpoint of every honest node in R_u, all the honest nodes that executed $(5a)$ will also execute $(7a)$. Hence, u will execute (8) and will compute legal views $A_{u,1}$ and $A_{u,2}$ of the two subregions of R_x, i.e. they contain all w in R_x that added an entry for u into J_w. □

Claim A.18 *For every new honest node u with a successful authorization stage it takes at most $O(1)$ steps until u has a pointer to every mature honest node in its region.*

Proof. If the authorization stage is successful, then u contacts all nodes in its view of R_x to integrate u'. Since these contain at least $|C_{R_x}|/5$ mature nodes v having an entry for u in J_v, and all of these are in A_u, each of these nodes

will accept u', and all mature honest nodes in the regions relevant for u' will accept u'. On the other hand, u' will receive back views of its relevant regions from at least $3|C_{R_x}|/20$ honest nodes in R_x and will establish pointers to all mature honest nodes in its regions. Up to this point, the integration only took $O(1)$ steps, and at its end, every mature honest node in u''s region has a pointer to u'. Thus, u' will be accepted by every new honest node entering the system afterwards.

Now, u' waits for $O(1)$ time steps (see step (3)). The reason for this is that u' first wants to make sure that every new honest node entering the system will now establish a connection to u' and u' will also establish a connection to this node. Hence, u' only has to worry about honest nodes that came into the system before that time point, including honest nodes that joined the system at the same time as u'. To solve this problem, u' sends a request for views to all nodes in its regions once it is sure that it passed the first stage of the integration. Since u' will receive back views of at least $3|C_R|/20$ mature honest nodes of each of its relevant regions R, it will establish pointers to all mature honest nodes in R and those that have already passed the first stage of the integration, i.e. u' fulfills the conditions in invariant 3 of being mature. Also this part of the integration stage only needs $O(1)$ steps. □

Combining these claims yields invariant 3. Thus, the invariants are fulfilled. So given that the invariants are true, it remains to show that the system is safe. For this we also need a couple of claims.

Claim A.19 *Every adversarial node can only initiate one successful rejoin operation during its lifetime.*

Proof. If a node u wants a rejoin operation to be successful, then at least one honest node w must add an entry for u in J_w because otherwise the integration stage will not be executed by any honest node. However, if there is such a node w, then by induction backwards on the routing path it follows that there must have been an honest node v in R_u executing step (3) in the authorization stage. But this can only happen if at least one honest node v in R_u executed step (6a) in the ID generation stage, which can only happen if v executed step (3a) in the ID generation stage before. Thus, at least $|C_v(r_u)|/5$ nodes committed to some key, and all honest among these have to reveal their keys in approximately the same time for an honest node to execute (6a). Hence, their initiations of step (2) must have been more than $(1-\tau)L/3$ steps before v reaches its time limit of $(1+\tau)(L/3+\delta)$. It follows from the arguments in Claim A.15 that then any honest node will only accept a rejoin request from u up to $(1-\tau)L/3-\delta$ steps before v reaches its time limit. However, in this case all key commitments of these nodes will count for the current rejoin execution, i.e. u cannot enforce another successful rejoin operation, completing the proof. □

Claim A.20 *Given that the system has been safe and the invariants hold, the number of adversarial nodes in any region R of size $(\gamma \log N)/N$ for a sufficiently large γ is at most $(1/30)\gamma \log N$.*

Proof. We need to combine several arguments:

1. According to Claim A.15, the lifetime of every legal member must be bounded by $\lambda' \log N$ for some constant λ' close to λ in the lifetime bound.
2. According to Claim A.19, every node can only initiate one rejoin operation during its lifetime that will be accepted by sufficiently many honest nodes in its region, and a rejoin operation is only successful if that node stays in the system for at least $(\lambda \log N)/3$ steps (up to some minor deviations).
3. There can be at most $\epsilon N/\log N$ adversarial nodes in the system at any time.
4. At most $\epsilon N/\log^2 N$ new adversarial nodes can join the system in a time step.

Hence, the maximum number of join and rejoin requests that can be issued by adversarial nodes within $\lambda' \log N$ time steps is at most

$$(\lambda' \log N) \cdot \left(\frac{\epsilon N}{\log^2 N} + \frac{\epsilon N/\log N}{(\lambda \log N)/3} \cdot 3 \right) = \frac{\lambda' \epsilon (1+9/\lambda) N}{\log N} \leq \frac{\epsilon' N}{\log N}$$

if ϵ is sufficiently small compared to λ. Hence, at any time, there can be at most $\epsilon' N/\log N$ many IDs the adversarial nodes can choose from. On expectation, at most $\epsilon' \gamma$ of these fall into an range R of size $(\gamma \log N)/N$. According to invariant 1, every legal member must have an chosen independently at random. Hence, it follows from the Chernoff bounds that there can be at most $(1/30)\gamma \log N$ adversarial IDs in R with high probability and therefore at most $(1/30)\gamma \log N$ adversarial nodes in R at any time. □

Claim A.21 *As long as the system has been safe and the invariants hold, it holds for any honest peer p and any constant $\delta > 0$ that there is an $L = \lambda \log N$ for a sufficiently large constant λ so that p needs at most $(1+\delta)L$ attempts to integrate L nodes into the system, with high probability.*

Proof. Notice that in the ID generation stage an honest peer will only ask the nodes to reveal their random keys if the new node for p to be integrated into the system has an available slot in Q_p. Since in each step a node is removed from Q_p, p needs to integrate L new nodes within L steps. We want to bound the attempts necessary for integrating these nodes. Since every attempt is started in an honest node u with a random ID and adversarial nodes can generate at most $\epsilon' N/\log N$ IDs within L steps, the probability that for a region size of $(\gamma \log N)/N$ there is an adversarial node in the region of u is at most $2\epsilon' \cdot \gamma$. If $\epsilon' > 0$ is sufficiently small, this can be reduced to some $q \leq \delta/(6(1+\delta))$. Suppose now that p uses $(1+\delta)L$ attempts. Then the expected number of attempts that fail is at most $(1+\delta)L \cdot q \leq \delta L/6$, and therefore the expected number of different honest nodes that fail is at most $\delta L/6$. The different honest nodes have independent, random IDs, it follows from the Chernoff bounds that with high probability at most $\delta L/3$ nodes will fail if λ is sufficiently large. Now, each failure of an honest node may cause up to 3 failed attempts. Hence, at most δL attempts with fail, with high probability, which means that p needs at most $(1+\delta)L$ attempts to integrate L new nodes, with high probability. □

Claim A.22 *For any region R of size $(\gamma \log N)/N$ for a sufficiently large constant γ, the number of honest nodes in R is at most $(4/3)\gamma \log N$ w.h.p.*

Proof. The tricky aspect in the proof is that although the IDs of nodes are random, the adversarial nodes can create a bias by determining which of the rejoin attempts by honest peers should succeed and which not. Fortunately, one can show that this influence is relatively small.

From Claim A.21 we know that for every peer p currently having L_p nodes in the system at most $(1+\delta)L_p$ attempts for some small constant $\delta > 0$ were necessary for this, w.h.p.. Hence, given that all of these attempts were successful, there would be at most $(1+\delta)N$ peers in the system instead of N. Hence, the maximum expected number of honest nodes in a region R of size $(\gamma \log N)/N$ is at most $(1+\delta)\gamma \log N$. If $\delta \leq 1/6$ and γ is sufficiently large, then it follows from the Chernoff bounds that at most $(4/3)\gamma \log N$ honest nodes can be in R with high probability. □

Claim A.23 *For any region R of size $(\gamma \log N)/N$ for a sufficiently large constant γ, the number of honest nodes in R is at least $(2/3)\gamma \log N$ w.h.p.*

Proof. From the proof of Claim A.21 we can conclude that for every peer p currently having L_p nodes in the system at least $(1-\delta)L_p$ of these are based on attempts in some completely honest region, w.h.p. Hence, the expected number of honest nodes in a region R of size $(\gamma \log N)/N$ is at least $(1-\delta)\gamma \log N$. If $\delta \leq 1/6$ and γ is sufficiently large, then it follows from the Chernoff bounds that at least $(2/3)\gamma \log N$ honest nodes must be in R with high probability. □

Claim A.24 *For any region R of size $(\gamma \log N)/N$ for a sufficiently large constant γ, it holds: The number of honest nodes in R with a lifetime of at least $L/2$ (with L being an upper bound on the lifetime) is at least $(1/3)\gamma \log N$, w.h.p.*

Proof. First, suppose that none of the attempts of the honest peers to integrate new nodes has failed. Because every peer has a node departure for every time step and the nodes have random IDs, it would follow in this case that for any interval of size $L/2$ in the lifetime interval $[0, \ell]$ (where ℓ is a lower bound on the lifetime) there are on expectation at least $(1/2)\gamma \log N$ honest nodes in R. However, the adversarial nodes can create a bias on the lifetimes of nodes that we need to bound.

Again, from Claim A.21 we know that for every peer p currently having L_p nodes in the system at most $(1+\delta)L_p$ attempts for some small constant $\delta > 0$ were necessary for this. Hence, the lifetime of a node resulting from an attempt can deviate by at most δL steps. Thus, for any interval of size $L/2$ there are on expectation only at least

$$\frac{L/2 - 2\delta L}{L}\gamma \log N \geq (1/2 - 2\delta)\gamma \log N$$

honest nodes in R that will surely end up in the desired interval. This is still fine for our lower bound if $\delta < 1/12$. In this case, if γ is a sufficiently large constant, then it follows from the Chernoff bounds that at least $(1/3)\gamma \log N$ honest nodes with a lifetime in an interval of size $L/2$ are in R, with high probability, which completes the proof. □

Finally, we can also prove Claim A.4 stated above.

Claim A.25 *For any constant $0 < \rho < 1$ there is a constant $0 < \epsilon < 1$ so that the number of honest nodes in the system changes by a factor of at most $(1 \pm \rho)$ in L steps, where L is the maximum over all L_t during these steps.*

Proof. Several factors can influence N:

1. new peers may not yet have at least ℓ nodes in the system,
2. peers that want to leave may have less than ℓ nodes in the system
3. adversarial peers may select their number of nodes freely, as long as it does not exceed the $\epsilon/\log N$ bound, and
4. honest peers that are mature (i.e. their join operation has completed) and that are not in the process of leaving may have a variable number of nodes in the system, depending on their success.

For our consideration of these items, we will slightly abuse the notation by assuming that $\log N$ is a fixed value though it is not. We can do this since it turns out that the deviation in N is actually small, and therefore insignificant when considering $\log N$.

We start with the first two items. Due to our bounds on the arrival and departure rate of peers and our time bounds for performing a join or leave operation, we know that at any time there can be at most an $L \cdot \epsilon/\log^2 N = \epsilon'/\log N$ fraction of honest peers that are leaving or entering the system. Since each of these has between 0 and L nodes, the maximum deviation they can cause is $L \cdot \epsilon'/\log N = \epsilon''$, where $\epsilon'' > 0$ can be an arbitrarily small constant depending on ϵ.

Next we consider item 3. According to our model, there can be at most an $\epsilon/\log N$ fraction of adversarial nodes in the system at any time. Hence, the deviation they can contributed is bounded by $\epsilon/\log N$.

It remains to consider item 4. For this we need to show that with high probability, every mature honest peer will have at least $(1-\delta)\ell$ nodes in the system at any time. Since the maximum number of nodes of an honest peer is L, $L - \ell \leq \lambda$, and $\ell \geq \lambda' \log N$ for some constant λ', it follows in this case that the maximum deviation caused by item 4 is at most $(\delta\ell + \lambda)/\ell = \delta + \lambda/(\lambda' \log N)$. This can be bounded by an arbitrarily small constant $\epsilon''' > 0$ depending on ϵ and λ. So we need to show:

Claim A.26 *For any constant $\delta > 0$ there is a constant $\lambda > 0$ (used for the lifetime) so that for any mature honest peer p and any time step, p will have at least $(1-\delta)\ell$ nodes in the system at that time, with high probability.*

Proof. Suppose that for some peer p and time t, p has less than $(1-\delta)\ell_t$ nodes in the system. Then we follow p backwards in time till it had at least ℓ_{t_0} nodes in the system, which it had because it successfully joined the system. This means that in a time interval of size $T = t - t_0$, p has only been successful for at most $(1 + \lambda/\log n)T - \delta\ell_t$ join attempts. However, in T steps p will initiate $3T$ join attempts. Since $T \geq \delta\ell_t$, it follows that

Use backwards argument in time, it follows from arguments similar to Claim A.22 that this is not possible, with high probability, if the constant λ is sufficiently large. □

Setting $\rho = \epsilon'' + \epsilon''' + \epsilon/\log N$ gives the bound in Claim A.4. □

Combining the claims above yields the safeness of the system.

$\mathsf{Insert}(z, i)$:
// *We assume:*
// $s = \lfloor \mathsf{ID}(u) \rfloor_{1/2^r}$, *the starting point of* u*'s region*
// $t = \lfloor x \rfloor_{1/2^r}$, *the starting point of* z*'s region*
(1) u: send to all $v \in C_u$: (u, insert, $\mathsf{ID}(u), r_u, (z, i)$)

(2) v [upon receiving (1) from a node $u \in C_v(r_u)$ or (2a) from $\geq |C^v_{R_{i-1}}(r)|/5$ nodes in $C^v_{R_{i-1}}(r)$, $r \in \{r_v, r_v + 1\}$]:
if $r_v - 1 \leq r_u \leq r_v + 1$ **then**
if $v \notin [t, t + 1/2^{r_u})$ **then**
(2a) send to all $w \in C^v_{R_{i+1}}(r_v)$: (v, insert, $x, r_u, (z, i)$)
else
// v *stores* (z, i) *in its data base* D_v
$D_v \leftarrow D_v \cup \{(z, i)\}$

$\mathsf{Lookup}(z)$:
(1) u: send to all $v \in C_u$: (u, lookup, x, r_u, z)

(2) v [upon receiving (1) from a node $u \in C_v(r_u)$ or (2a) from $\geq |C^v_{R_{i-1}}(r)|/5$ nodes in $C^v_{R_{i-1}}(r)$, $r \in \{r_v, r_v + 1\}$]:
if $r_v - 1 \leq r_u \leq r_v + 1$ **then**
if $v \notin [t, t + 1/2^{r_u})$ **then**
(2a) send to all $w \in C^v_{R_{i+1}}(r_v)$: (v, lookup, x, r_u, z)
else
if $(z, i) \in D_v$ **then**
(2b) send to all $w \in [t, t + 1/2^r)$: (v, reply, $x, r_u, (z, i)$)

(3) w [upon receiving (2b) or (3a) from at least $|C^w_{R'_{i-1}}(r)|/5$ nodes in $C^w_{R'_{i-1}}(r)$, $r \in \{r_v, r_v + 1\}$]:
if $r_v - 1 \leq r_u \leq r_v$ **then**
if $w \notin [s, s + 1/2^r)$ **then**
(3a) send to all $w' \in R'_{i+1}(r_w)$: (w, reply, $x, r_u, (z, i)$)
(3b) **else** send to u: (w, reply, (z, i))

(4) u: [upon receiving (3b) from $\geq |C_u|/5$ nodes with the same (z, i)]:
return i

Figure 3: The $\mathsf{Insert}(z, i)$ and $\mathsf{Lookup}(z)$ protocol initiated by node u. $(R_i)_{i \geq 0}$ is the sequence of regions traversed towards the region of z, and $(R'_i)_{i \geq 0}$ is the sequence of regions traversed back to the region of $\mathsf{ID}(u)$ according to the Chord routing strategy. Notice that the view of R_i may depend on the receiving and sending nodes. If a node v receives sufficiently many messages for the same request M from R_i w.r.t. a range $r \in \{r_v - 1, r_v\}$, it will accept it and forward M to all nodes in its view of R_{i+1}.

// u: *node that initiates Rejoin protocol*

(1) u : send to all $v \in C_u$: (u, join)
wait for $L_u/3$ steps
// *compute current range* r
$r \leftarrow$ Region(u)
// $L(r)$*: lifetime depending on current* r
if $|Q_p| < L(r)$ **and** (no node u' of p has $state_{u'} = critical$) **then**
// *start of critical section, only executed by one node at a time,*
// *but Rejoin operations are scheduled by peer, so no problem*
for $j = 1$ to $L(r) - |Q_p|$ **do**
reserve slot $|Q_p| + j$ in Q_p for jth join operation
(1a) send to all $v \in C_u$: (u, reveal, j)
// *give honest nodes time to compute* x_j*'s*
wait for δ steps
release all reservations that have not entered (1) of stage 2

(2) v [upon receiving (1)]:
// $i_u \in \{0, 1\}$*: rejoin indicator for* u*, see also stage 3*
// N*: estimated by* v *via* r_v
if ID$(v) \in R_u$ **and** $i_u = 0$ **and** (remaining life of v is $\geq (1 + \tau)(L_u/3 + \delta \log N)$) **then**
$i_u = 1$ // u *cannot rejoin again*
for each $j \in \{1, 2, 3\}$:
$x_{v,j} \xleftarrow{R} [0, 1)$ // *generate random key*
$K_{u,j} = \emptyset$ // *initialize set of other keys*
// $C_v(r_u)$*:* v*'s view of* u*'s region*
// $h(\cdot)$*: result of bit commitment scheme*
(2a) send to all $w \in C_v(r_u)$: (v, commit, $h(x_{v,j}), (u, j)$)
wait for $(1 + \tau)(L_u/3 + \delta)$ steps
if $(6b)$ has not been executed yet **then** abort protocol
// *some node in* R_u *(including* u*) must be adversarial*

(3) v [upon receiving (1a) with parameter j]:
wait until at least $(1 - \tau)L_u/3$ steps are over since (2) started
if $|K_{u,j}| \geq |C_v(r_u)|/5$ **then**
(3a) send to all $w \in C_v(r_u)$: (v, key, $x_{v,j}, (u, j)$)

(4) v [upon receiving $(2a)$ from a node $w \in C_v(r_u)$]
if ($(2a)$ has been executed) **and** ($(2a)$ from w received $\leq \delta$ steps before receiving (1)) **and**
$((3)$ has not been executed yet) **then**
store $(w, h(x_{w,j}))$ in $K_{u,j}$

(5) v [upon receiving $(2a)$ from $\geq |C_v(r_u)|/10$ or $(5a)$ from $\geq |C_v(r_u)|/10$ nodes in $C_v(r_u)$]:
// *adversarial* u *may try to contact only subset of honest nodes*
if $i_u = 0$ **then**
wait for δ steps // *to give* u *sufficient time to initiate (2)*
$i_u = 1$ // *no more rejoin requests accepted from* u
(5a) send to all $w \in C_v(r_u)$: (v, no_rejoin, u)

(6) v [upon receiving $(3a)$ from a node $w \in C_v(r_u)$ and having executed $(3a)$]:
if $h(x_{w,j})$ matches $(w, h(x_{w,j}))$ in $K_{u,j}$ **then**
replace $(w, h(x_{w,j}))$ by $(w, x_{w,j})$
if no $(w', h(x_{w',j}))$ left in $K_{u,j}$ **then**
(6a) $x_j \leftarrow x_{v,j} \oplus (\bigoplus_{(w', x_{w',j}) \in K_{u,j}} x_{w',j})$ // $\oplus$ *is the bit-wise XOR*
(6b) send to u: (v, key, x_j, j)

Figure 4: Stage 1 of the Rejoin protocol for node u. We assume that the clocks of the nodes deviate in speed by at most a $(1 \pm \tau)$ factor for some $0 < \tau << 1$ and that $\delta > 0$ is a sufficiently large constant.

// r: *range computed by u in stage 1*

(1) u [upon receiving the same (6b) of stage 1 for some j from $\geq |C_u|/5$ nodes]:
if reservation not released for j **then**
insert u' into reserved slot in Q_p

(1a) send to all $v \in C_u(r)$: (u, authenticate, x, r, j)

(2) v [upon receiving $(1a)$ and having sent $(6b)$ in stage 1 with x]:
if at most δ time since $(6b)$ of stage 1 **then**
$s \leftarrow \lfloor \mathsf{ID}(v) \rfloor_{1/2^{r+1}}$ // *starting point of source region*
$t \leftarrow \lfloor x \rfloor_{1/2^{r+1}}$ // *starting point of destination region*
// $C_{v,1}(r)$, $C_{v,2}(r)$: *node sets in two halves of* $R_u(r)$

(2a) send to all $w \in C_{v,1}(r) \cup C_{v,2}(r)$: ($v$, auth, $(s,t), x, u, r, j$)

(3) w [upon receiving $(2a)$ from $> |C_w(r_u)|/10$ nodes v in $C_w(r_u)$ or $(3a)$ from $\geq |C^w_{R_{i-1}}(r+1)|/5$ nodes v in $C^w_{R_{i-1}}(r+1)$ for some $r \leq r_w + 1$]:
if $(2a)$-messages received **then** $A_w \leftarrow C_w(r)$
else
// A_v: *cluster in message of* v, $C \in \{C_w(r_u), C^w_{R_{i-1}}(r+1)\}$
$A_w \leftarrow \{v' \in \bigcup_v A_v : \ |\{v \mid v' \in A_v\}| \geq |C|/10\}$
if $w \notin [t, t + 1/2^r)$ **then**

(3a) send to all $w' \in C^w_{R_{i+1}}(r+1)$: ($w$, auth, $(s,t), x, A_w, u, r, j$)
else
// N: *estimated by w via r_w*
if remaining lifetime of $w \geq \delta \log N$ **then**
// w *may allow p to join its region and remembers its current time* T_w
$J_w \leftarrow J_w \cup \{(x, u, r, j, T_w, 0)\}$

(3b) send to all $w' \in [t, t + 1/2^r)$: (w, reply, $(s,t), x, C_w(r+1), u, r, j$)

(4) w' [upon receiving $(3b)$ or $(4a)$ from at least $|C^w_{R'_{i-1}}(r+1)|/5$ nodes in $C^w_{R'_{i-1}}(r+1)$ with $r \leq r_{w'} + 1$]:
// A_w: *cluster in message of* w
$A_{w'} \leftarrow \{w'' \in \bigcup_w A_w : \ |\{w \mid w'' \in A_w\}| \geq 3|C^{w'}_{R'_{i-1}}(r+1)|/20\}$
if $w' \notin [s, s + 1/2^r)$ **then**

(4a) send to all $w'' \in C^{w'}_{R'_{i+1}}(r+1)$: ($w'$, reply, $(s,t), x, A_{w'}, u, r, j$)
else

(4b) send to all $v \in C_{w'}(r_u)$: (w', reply, $(s,t), x, A_{w'}, u, r, j$)

(5) v [upon receiving $(4b)$ from $\geq |C_{v,1}(r)|/5$ nodes in $C_{v,1}(r)$ and $\geq |C_{v,2}(r)|/5$ nodes in $C_{v,2}(r)$]:
if $\leq \delta \log N$ time since execution of (2) **then**
$A_{v,1} \leftarrow \{w' \in \bigcup_w A_w : \ |\{w \in C_{v,1}(r) \mid w' \in A_w\}| \geq 3|C_{v,1}(r)|/20\}$
$A_{v,2} \leftarrow \{w' \in \bigcup_w A_w : \ |\{w \in C_{v,2}(r) \mid w' \in A_w\}| \geq 3|C_{v,2}(r)|/20\}$

(5a) send to all $w \in A_{v,1} \cup A_{v,2}$: ($v$, accept, (u,j))

(6) w [upon receiving $(5a)$ from $> |A_w|/10$ nodes in A_w]:
$\bar{T} \leftarrow$ median of arrival times of messages
if $(x, u, r, j, T_w, 0) \in J_w$ **and** $T_w - \bar{T} < \delta \log N$ **then** // T_w: *current time of* w
change entry to $(x, u, r, j, T_w, 1)$

(6a) send to all $v \in A_w$: (w, accept, (u,j))

(7) v [upon receiving $(6a)$ from $\geq 3|A_{v,1}|/20$ nodes in $A_{v,1}$ and $\geq 3|A_{v,2}|/20$ nodes in $A_{v,2}$ and having sent $(5a)$]:

(7a) send to u: (v, join, $x, A_{v,1}, A_{v,2}, j$)

(8) u [upon receiving $(7a)$ from $\geq 3|C_u|/20$ nodes in $v \in C_u$ with the same x]:
$A_{u,1} \leftarrow \{w \in \bigcup_v A_{v,1} : \ |\{v \mid w \in A_{v,1}\}| \geq |C_u|/10\}$
$A_{u,2} \leftarrow \{w \in \bigcup_v A_{v,2} : \ |\{v \mid w \in A_{v,2}\}| \geq |C_u|/10\}$

Figure 5: Stage 2 of the Rejoin protocol. In this authentication stage, $(R_i)_{i \geq 0}$ represents the sequence of regions traversed towards the region of x, and $(R'_i)_{i \geq 0}$ is the sequence of regions traversed back to the region of $\mathsf{ID}(v)$ according to the Chord routing strategy. Notice that the view of R_i may depend on the receiving and sending nodes.

(1) u: **for each** $i \in \{1,2\}$:
send to all $v \in A_{u,i}$: (u, integrate, u', x, r, j)
// u': *new node to join* A_u

(2) v [upon receiving (1)]:
// T_v: *current time of* v
// S_v: *set of data about all nodes in the view of* v, *contains tuples* $(w, x_w, r_w, T_w, g_w, i_w)$
// (T_w: *step when* w *joined,* $g_w \geq 0$: *generation,* $i_w \in \{0,1\}$: *rejoin indicator for* w)
// u *authorized and authorization not too old?*
if $(x, u, r, j, T_u, 1) \in J_v$ **and** $T_v - T_u < \delta \log N$ **and** $r_v - 1 \leq r \leq r_v + 1$ **then**
$S_v \leftarrow S_v \cup \{(u', x, r, T_v, 0, 0)\}$
// $\mathcal{R}_v(r+1)$: *set of all pointer regions of* v *relevant for* u'
(2a) send to all $w \in \bigcup_{R \in \mathcal{R}_v(r+1)} C_R^v$: ($v$, new_node, u', x, r, j)
(2b) send to u': (v, welcome, $(C_R^v)_{R \in \mathcal{R}_v(r+1)}$)

(3) u' [upon receiving $(2b)$ from $\geq 3|A_{u,1}|/20$ nodes in $A_{u,1}$ and $\geq 3|A_{u,2}|/20$ nodes in $A_{u,2}$]:
$S_{u'} = \emptyset$
for each $R \in \mathcal{R}_{u'}$:
$C_{R,1}^{u'} \leftarrow \{v \in \bigcup_w C_R^w : |\{w \in A_{u,1} \mid v \in C_R^w\}| \geq |A_{u,1}|/10\}$
$C_{R,2}^{u'} \leftarrow \{v \in \bigcup_w C_R^w : |\{w \in A_{u,2} \mid v \in C_R^w\}| \geq |A_{u,2}|/10\}$
$C_R^{u'} \leftarrow C_{R,1}^{u'} \cup C_{R,2}^{u'}$
for each $w \in C_R^{u'}$:
$S_{u'} = S_{u'} \cup \{(w, ?, ?, ?, ?, 0)\}$
// *wait until properly integrated into all honest nodes in* $\mathcal{R}_{u'}$ *by* $(2a)$ *messages*
wait for δ steps
(3a) send to all $v \in \bigcup_{R \in \mathcal{R}_{u'}} C_R^{u'}$: ($u'$, integrate, x, r, j)
$r_{u'} \leftarrow r$ and $L_{u'} \leftarrow \ell \cdot r$
wait for δ steps
remove all entries in $S_{u'}$ that still have a "?" in them
// u' *is now mature*

(4) v [upon receiving $(2a)$ from $\geq 3|C_{R_{u'}}^v(r+1)|/20$ nodes $w \in C_{R_{u'}}^v(r+1)$ with the same (u', x, r, j) for some $r \leq r_v + 1$]:
$\bar{T} \leftarrow$ median of arrival time of messages
if $T_v - \bar{T} \leq \delta$ **then**
$S_v \leftarrow S_v \cup \{(u', x, r, T_v, 1, 0)\}$

(5) v [upon receiving $(3a)$]:
// $t_w^v = T_v - T_w$: w*'s age from* v*'s view*
(5a) send to u': (v, welcome, $(w, x_w, r_w, t_w^v, g_w, i_w^v)_{w \in C_v(r)}$)

(6) u' [upon receiving $(5a)$ from some $v \in R \in \mathcal{R}_{u'}$ and $(5a)$ from $\geq |C_R^{u'}|/5$ nodes $w \in R$ with the same (v, x_v, r_v) and $g_v \leq g$ for the minimum safe generation g]:
// $g = t_v^{u'}/(L_{u'}/3)$ *is safe because* $\geq |C_R^{u'}|/5$ *mature honest nodes for this*
$t_v^{u'} = \text{median}_w\{t_v^w\}$
$S_{u'} \leftarrow S_{u'} \cup \{(v, x_v, r_v, T_{u'} - t_v^{u'}, g+1, i_v)\}$

(7) u' [upon receiving $(5a)$ from $\geq |C_R^{u'}|/10$ nodes $w \in R$ with $i_u^w = 1$]:
$i_u = 1$

Figure 6: Stage 3 of the Rejoin protocol for new node u'. A_u is the new cluster for u', i.e. the cluster u computed in stage 2. Again, δ is a sufficiently large constant.