Robust Distributed Name Service

Baruch Awerbuch[*]
baruch@cs.jhu.edu

Christian Scheideler[*]
scheideler@cs.jhu.edu

Abstract

This paper suggests a method called Trust-but-Verify that may turn DHT-based peer-to-peer systems into systems that are provably robust against even massive adversarial attacks.

1 Introduction

The Internet was originally designed for the purpose of being extremely robust against hardware attacks, such as natural disasters or wars. However, software attacks (such as viruses, worms, or denial-of-service attacks) have become increasingly severe over the past few years, whereas hardware attacks are negligible. Thus, for any distributed application to run reliably on the Internet, it is of utmost importance that it is robust against adversarial software attacks. This is especially important for critical applications such as name service, i.e. a service that translates names such as "machine.cs.school.edu" into IP addresses so that machines can communicate with each other.

The current way name service is provided in the Internet is server-based. However, server-based architectures are vulnerable to attacks. A much more robust alternative appears to be the recently emerged peer-to-peer paradigm with its strictly decentralized approach. Unfortunately, despite the appeal of a decentralized approach, it appears to be a daunting task to develop peer-to-peer networks that are robust against adversarial attacks. Most of the proposed solutions suggest the use of classical methods such as certification authorities or cryptography to secure peer-to-peer systems. However, it should be clear that without being overly restrictive, a certification authority will be ineffective in banning adversarial peers. Also, in an open environment, cryptographic methods will not be able to keep adversarial peers out. Thus, one should abandon classical methods for securing peer-to-peer systems and instead search for mechanisms that ensure reliability *despite* the presence of adversarial peers.

The goal of this paper is to demonstrate that it seems posssible to design *completely open* peer-to-peer systems that are provably robust against *arbitrary Byzantine attacks* of a large fraction of its peers.

1.1 Security goal

In order provide a distributed name service, the following operations have to be implemented:

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

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. An overlay network is called *reliable* if any of the three network operations Join, Leave, and Lookup can be executed reliably by any honest peer. The goal of this paper is to demonstrate that efficient and reliable overlay networks can be constructed under certain simplifying assumptions.

1.2 Fundamental principles for reliability

To achieve efficiency and reliability, we believe that the following fundamental principles have to be fulfilled.

[*]Department of Computer Science, Johns Hopkins University, 3400 N. Charles Street, Baltimore, MD 21218, USA

Regions instead of individual peers. Certainly, a system in which the correct execution of an overlay network operation depends on the correct behavior of individual peers is not survivable. Hence, quorums of peers have to be formed to check each other's behavior and therefore ensure the reliable execution of operations. However, forming separate quorums of peers (i.e. grouping the peers in 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 peers can create different views of the current situation in honest peers, it is hard to find a consensus on a cluster operation between the honest peers, creating inconsistencies. To avoid these problems, we suggest not to form clusters but instead allow each honest peer to decide by itself which *regions* of peers it considers to be safe. In our case, the ID space of the peers will be the interval $[0, 1)$, and a region may be any subinterval of this interval. Overlay network operations initiated by a peer will be executed on a region level.

Flexible instead of fixed regions. We will adopt the approach in DHT-based systems of using a pseudo-random one-way hash function for assigning IDs in $[0, 1)$ to the peers. One-way hash functions protect against identity theft but they cannot protect against adversarially selected hash values to obtain the majority of peers in some region. Hence, any region-based organization that wants to be survivable in an open environment must be flexible in the region size. In other words, it does not suffice just to pick regions of size $O((\log n)/n)$ to ensure survivability but region sizes have to be adapted to the scale of the attack on the peer-to-peer system. Ideally, this should be done so that the region size is kept at a minimum possible size necessary to cope with the attack, i.e. for scalability reasons it should be avoided just to group all the peers into a single region.

1.3 Basic approach

We require that any execution of a Join, Leave, or Lookup operation (by honest peers), as well as the storage of data, has to be handled on a region level. To ensure that regions (and therefore the network) work reliably, we must make sure that in every region relevant to an honest peer, the honest peers are in the majority. Taking this into account, our goal above can be rephrased in the following, more formal way:

Design overlay network operations so that for any *arrival-departure sequence of honest and adversarial peers over t time steps in which the adversarial peers never represent more than an ϵ fraction of the peers in the overlay network, the adversarial peers will not gain the majority in any region relevant for honest peers, 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 peers 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 peers. So no absolute guarantees can be given, unless we put all peers into a single region, which is highly inefficient and therefore out of question.

1.4 Existing work

There is fair amount of work on the subject. The reliability of *hash-based peer-to-peer overlays* (or DHT's) such as Chord [12], Pastry [10], and Tapestry [13] hinges on the assumption that the IDs given to the peers are pseudo-random, so that they can cope with a constant fraction of the peers 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 of offline trial/errors. For similar attacks, see [4].

Much of the active sabotage attacks, such as forging the correctness and authenticity of data as well as misrouting, can be addressed using cryptographic techniques such as selfcertifying path names [8], end-to-end acknowledgements, etc. These techniques allow the system to detect and ignore nonauthentic data, avoid peers who incorrectly route, etc. Blocking of routing packets appears to be the most difficult to monitor. Examples of routing attacks are discussed in Castro et al. [3], and examples of data-related attacks are described by Sit and Morris [11], who give a series of examples of particular weaknesses in existing hash-based networks and discuss potential defenses against some of these problems. [3] describe attacks that can be mounted against such overlay networks and the applications they support, and present the design of secure techniques that may help thwart some of these attacks in certain practical scenarios (such as using certified IDs).

Below we describe some of the more theoretical approaches.

Classical distributed computing methods [6, 2, 7, 9] use Byzantine agreement and two-phase commit approaches with inherently $\Omega(n)$ redundancy and overhead to maintain a safe state.

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.

Provable randomness is the method pursued in [1]. This results in fairly complex protocols, using various cryptographic assumptions.

2 The Trust-but-Verify approach

We start with a collection of basic assumptions explaining the model of our approach.

2.1 Basic assumptions

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 may have an unlimited number of IDs for its peers but it can only own at most an ϵ-fraction of the peers in the system at any time. Such adversaries are called *ϵ-bounded*. A priori, adversarial peers cannot be distinguished from honest peers.

We assume that every honest peer that wants to join the system knows some honest peer already in the system and thus can initiate the Join operation by one of these peers. This is a reasonable assumption, because if a new honest peer does not know any honest peer in the system, there is certainly no (reliable) way that it can join the system.

We consider asynchronous systems in which every honest peer has roughly the same clock speed but there is no global time. Since honest peers are considered reliable, we assume that at any point in time, any message sent by an honest peer p to another honest peer q will arrive at q within a unit of time. (Other message transmissions may need any amount of time.) Furthermore, honest peers have unbounded bandwidth, i.e. an honest peer can receive and send out an unbounded number of messages in a unit of time. The latter assumption allows us to ignore denial-of-service attacks, but it does *not* simplify the task of securing an overlay network against legal attacks (i.e. attacks exploiting security holes in its protocols). As long as adversarial peers do not transmit unnecessary packets, the number of messages an honest peer will have to deal with in a time unit will normally be low so that our protocols are practical despite this assumption. Designing provably secure overlay networks for honest peers with bounded bandwidth is very challenging and needs further research.

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, 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. The *identification number* of a peer p, $\mathsf{ID}(p)$, in our overlay network is determined by a pseudo-random one-way hash function mapping names to real values in $[0, 1)$. The pseudo-randomness makes sure that IDs of honest peers are random, and the one-way property makes sure that if the name of a peer cannot be taken over, it is also hard to take over its ID. Finally, we need some assumptions about how messages are passed. We assume that the source of a message cannot be forged so that adversarial peers cannot forge messages from honest peers. Also, a message sent between honest peers cannot be deleted or altered by the adversary. However, the adversary may know about every message sent in the system.

2.2 Outline of Trust-but-Verify approach

The basis of the trust-but-verify approach is the assumption that the hash value of every honest peer is like an independent, random value in $[0, 1)$, but hash values of adversarial peers may be any values different from the values of honest peers. The honest peers will organize in regions $R \subseteq [0, 1)$ they consider to be safe (i.e. the honest peers are in the majority) and will form a hypercubic pointer structure between these regions. If an honest peer feels that one of its regions is no longer safe, it will change it to a larger region. On the other hand, if an honest peer feels that a subregion within one of its regions is now safe, it will move to the subregion. Each honest peer will continuously probe peers it knows in its regions. If some peer does not respond, the honest peer will drop its connection to it. Since we assume honest peers to be reliable, this will not happen to honest peers. Messages are routed along safe regions to make sure that adversarial peers cannot alter, delay, or delete them. Next we give a more detailed description of how to select regions, how to interconnect the peers, and how to join, leave, and search in the overlay network.

2.3 Region decomposition

Consider an infinite complete binary tree T with root r. We use T to decompose $[0, 1)$ into a hierarchy of intervals. For this, let each edge to a left child have label 0 and each edge to a right child have label 1. For each node v in T, we define its label $\ell(v)$ to be the bit string resulting from the edge labels when going along the unique path from r to v. (I.e. the leftmost node in level 4 of T has label 0000, and its sibling has label 0001.) For any node label $\ell = (x_1x_2x_3\ldots)$, we define $b(\ell) = \sum_{i\geq 1} x_i 2^{-i}$. Using this terminology, we can associate a unique region in $[0, 1)$ with each tree node v at level i in T: $R_v = [b(\ell(v)), b(\ell(v)) + 1/2^i)$. In the following, a region always means an interval of size $1/2^i$ for some integer $i \geq 0$ starting at an integer multiple of $1/2^i$. I.e. every region has a unique tree node.

Given a peer p and a region R, let C_R^p denote p's *view* of R, i.e. the set of peers p is connected to in R. p's view of R is called *reliable* if the number of adversarial peers in C_R^p is at most $|C_R^p|/4$. Given that all honest peers have random IDs, the following result holds, where n is the current number of honest peers in the system.

Theorem 2.1 *Let $0 < \epsilon \leq 1/8$ and $c > 0$ be constants. If for some peer p and region R, $|R| \geq (c \log n)/n$ and $|C_R^p| \leq (1 + \epsilon)|R| \cdot n$, C_R^p contains all honest peers in R, and c is sufficiently large compared to ϵ, then p's view of R is reliable, with high probability.*

The theorem directly follows from the well-known Chernoff bounds because if $|R| \geq (c \log n)/n$ for a sufficiently large c, then the number of honest nodes in R is at least $(1-\epsilon)|R| \cdot n$, with high probability. In this case, the fraction of adversarial nodes in C_R^p can be at most $2\epsilon \leq 1/4$ if $\epsilon \leq 1/8$.

Hence, honest nodes p should, as a prerequisite, try to maintain connections to regions R so that $|R| \geq (c \log n)/n$ and $|C_R^p| \leq (1+\epsilon)|R| \cdot n$. If this is true, we say that p views its region to be reliable, or the region is *safe*. There are several issues that have to be addressed in order to maintain these safeness requirements. For example, honest peers need a good approximation of n, and they need to be able to send region queries in order to expand their region view if necessary. Also, as we will see, just maintaining safe regions is not sufficient for reliable communication as required for estimating n.

2.4 Overlay network

The *level* of a region R is defined as the level of its tree node and denoted by ℓ_R, where the root of the decomposition tree has level 0.

The aim of every honest peer p is to maintain the smallest region S_p containing $\mathsf{ID}(p)$ that is safe from p's point of view. S_p is also called p's *home region*. Also, a set of *active regions* $\mathcal{A}_p = \{A_i^p \mid i \in \mathbb{Z}\}$ is maintained so that for every $i \geq 0$,

1. A_i^p (resp. A_{-i}^p) is the smallest region containing $\mathsf{ID}(p) + 1/2^i$ (resp. $\mathsf{ID}(p) - 1/2^i$) that is safe from p's point of view and
2. at least $|C_{A_i^p}^p|/3$ (resp. $|C_{A_{-i}^p}^p|/3$) of the peers q in A_i^p (resp. A_{-i}^p) reported to p that $\ell_{S_q} \geq \ell_{A_i^p}$ (resp. $\ell_{S_q} \geq \ell_{A_{-i}^p}$).

Using the $1/3$ rule in condition 2 has two benefits: First, it guarantees that if an active region A is reliable, then there is at least one honest peer having its safe region inside A, which is useful for routing. Second, if the $1/3$ rule is violated at some point, then the majority of peers having a safe region larger than A is honest, allowing p to use a majority rule for view filtering to expand A to a region of higher level.

Notice that there can be different values i_1 and i_2 with $A_{i_1}^p \subseteq A_{i_2}^p$, so regions may sometimes be contained in other regions, but for any two active regions R_1 and R_2 of a peer p it must hold that either $R_1 \subseteq R_2$ (resp. $R_2 \subseteq R_1$) or $R_1 \cap R_2 = \emptyset$. Those regions R_1 with $R_1 \subseteq R_2$ only have to be maintained implicitly, so that p only has to explicitly store $O(\log n)$ regions at any point in time.

Besides S_p and $\mathcal{A}_p$, each honest peer p also maintains a region NS_p representing the new but not yet safe version of S_p and a set of regions $\mathcal{NA}_p$ representing the new but not yet safe version of $\mathcal{A}_p$. All new peers are only added to NS_p and $\mathcal{NA}_p$. If NS_p is safe at some point, S_p is replaced by NS_p. The same is done for regions in $\mathcal{NA}_p$. Hence, S_p and $\mathcal{A}_p$ are safe snapshots of the system. p views the regions S_p and $\mathcal{A}_p$ as *trusted* and the other regions NS_p and $\mathcal{NA}_p$ as *untrusted*.

p maintains connections to all peers in S_p (and some limited number of older versions of S_p), $\mathcal{A}_p$, NS_p, and $\mathcal{NA}_p$ and all peers q (it knows of) with p in S_q, $\mathcal{A}_q$, NS_q, or $\mathcal{NA}_q$, but p only forwards requests with a majority consensus in one of its trusted regions.

2.5 Maintaining safe and active regions

An honest peer p performs the following actions to maintain its safe and active regions:

1. NS_p is now safe. Then p sends (p, ℓ_{NS_p}) to all peers q it knows of that contain p in one of their untrusted regions and updates S_p to NS_p. (NS_p may be smaller or larger than S_p. If $\ell_{NS_p} < \ell_{S_p}$, p still keeps a backup of S_p for a limited time to avoid problems for regions $A \in \mathcal{A}_q$ containing p.)
2. NS_p is still unsafe. Then p moves one level upwards with NS_p. (We explain below how to realize that.)
3. A region $NA_i^p \in \mathcal{NA}_p$ is now safe. Then p sends (p, i, ℓ_{NS_p}) to all peers q it knows of that contain p in one of their untrusted regions and updates A_i^p to NA_i^p.
4. A set $NA_i^p \in \mathcal{NA}_p$ is still unsafe or violates the $1/3$ rule. Then p moves one level upwards with NA_i^p.

If the arrival rate of honest and adversarial peers is limited to $\epsilon/\log^2 n$, then the size of currently trusted regions is accurate enough to estimate n. Also, if the departure rate of honest peers is limited to $\epsilon/\log^2 n$, then the honest peers will be able to find new safe regions for S_p and some A_i^p quickly enough before the old versions of these sets become unreliable. More precisely, the following result can be shown.

Theorem 2.2 *If the adversary is ϵ-bounded for some sufficiently small constant $\epsilon > 0$ and the arrival and departure rate of honest and adversarial peers is at most $\epsilon'/\log^2 n$ for some sufficiently small constant $\epsilon' > 0$, then our protocols ensure that for any honest peer p, S_p and any $A \in \mathcal{A}_p$ are reliable at any time, with high probability.*

In order to prove this result, we have to describe how to grow the size of a region. This is handled by region requests. Before explaining them, we start with lookup requests.

2.6 Lookup requests

The peer p initiating a request M sends M to all nodes of its active region A_0^p. Every peer q that receives M will forward M to all peers q' it knows within A_0^p. Every peer q that receives M from at least half of the nodes in S_q sends it on to all nodes of its active region representing the next hop of M to its destination. There, the same spreading and checking is done as in p's active region until M reaches the destination.

Condition 2 of an active region and Theorem 2.2 make sure that if p is honest, there is always at least one honest peer at each hop that receives the message from a majority of nodes in its safe region so that the message is delivered in $O(\log n)$ steps.

2.7 Region requests

Region requests also use active regions to obtain a legal view of some requested region. A peer is called a *legal member* of the system if at least one honest peer is connected to it, and a view of a region R is called *legal* if it only contains legal members.

A region request that is still outside the requested region is handled like a lookup request. Once a region request for R is in R, the request is split into two requests, one for each subregion of R, and this division is continued at lower stages until the considered region R' is a subset of the region A_0^q of peer q handling the request. Then q sends its view of R' backwards, which is combined and filtered by majority rule upwards till the request gets back to its origin.

Every region request by an honest peer is processed correctly in $O(\log n)$ steps.

2.8 Estimating n

The estimation of n is done recursively, starting with each peer p reporting the size and level of A_0^p to all peers q it knows of in the next higher region containing p. Suppose now that up to level ℓ each honest peer p has computed an estimate of the number of peers in its home region (i.e. the region containing it) of level ℓ. If p reports its view to all peers it knows of in its home region of level $\ell - 1$, then every honest peer q will obtain views for the two regions of level ℓ contained in its home region of level $\ell - 1$, because q has active regions overlapping with each of these regions. Since q's active regions are safe, it can use the median rule once it has received views from a majority of peers in each region to compute an estimate for the size of its home region of size $\ell - 1$. This is continued upwards until every honest peer has an estimate of the number of peers in $[0, 1)$.

The estimation procedure takes $O(\log n)$ steps.

2.9 Join and leave

If a new peer p wants to join via some peer q in the system, it starts with requesting regions of lowest possible size, i.e. $(c \log n)/n$. If this does not provide a safe region for some A_i^p or S_p, p moves upwards with that region until it obtains a region satisfying the requirements of a safe or active region. For each such region that p obtains, p integrates itself into the system. After $O(\log^2 n)$ steps, p is fully integrated.

Leaving is simple: a peer may leave with or without notice. It is only important that the departure rate of honest peers does not exceed $\epsilon / \log^2 n$.

We intend to prove the following conjecture as the end result of all the conditions and protocols.

Conjecture 2.3 *The Trust-but-Verify method results in a survivable overlay network.*

References

[1] B. Awerbuch and C. Scheideler. Group Spreading: A protocol for provably secure distributed name service. Unpublished manuscript.

[2] Castro and Liskov. Practical Byzantine fault tolerance. In *OSDI*, 1999.

[3] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *OSDI*, 2002.

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

[5] J. R. Douceur. The sybil attack. In *IPTPS*, 2002.

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

[7] 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.

[8] D. Mazieres, M. Kaminsky, M. F. Kaashoek, and E. Witchel. Separating key management from file system security. In *SOSP*, pages 124–139, 1999.

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

[10] 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)*, 2001.

[11] E. Sit and R. Morris. Security considerations for peer-to-peer distributed hash tables. In *MIT workshop on Peer-Peer Systems*, 2001.

[12] 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*, 2001.

[13] 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.