CORPS: Building a Community Of Reputable PeerS in Distributed Hash Tables

Erika Rosas[1], Olivier Marin[1] and Xavier Bonnaire[2]

[1]*Laboratoire d'Informatique de Paris 6, Université Pierre et Marie Curie*
INRIA-CNRS
4 place Jussieu, 75005 Paris, France
[2]*Departamento de Informática, Universidad Técnica Federico Santa María*
Avenida España 1680, Valparaíso, Chile
Email: Erika.Rosas@lip6.fr, Olivier.Marin@lip6.fr, Xavier.Bonnaire@inf.utfsm.cl

Building trust is a major concern in Peer-to-Peer networks as several kinds of applications rely on the presence of trusted services. Traditional techniques do not scale, produce very high overhead or rely on unrealistic assumptions. In this paper, we propose a new membership algorithm (CORPS) for Distributed Hash Tables which builds a community of reputable nodes and thus enables the implementation of pseudo-trusted services. CORPS uses a reputation-based approach to decide whether a node can be a member of the group or not. We demonstrate the benefits of this approach and evaluate how much it improves the reliability of a trusted routing service.

Keywords: Peer-to-Peer, Trust, Reputation Systems, Secure Routing, Membership Algorithm

Received ; revised

1. INTRODUCTION

The fully distributed nature of Peer-to-Peer (P2P) networks creates a scalable, fault tolerant and self-organized system with the potential to involve millions of nodes. However, the lack of central control, the considerable number of peers and the high dynamism of the network make it very hard to build trust among peers.

To deliver a valuable service in P2P applications, it is important to trust that the participants will act as requested. For instance in file sharing applications, a peer must trust that others will not upload a virus. In P2P storage applications, a client must trust that the designated peers will indeed save the information, and even more importantly that other peers will forward messages correctly so that communication can be successful.

Building trust is especially complex since a P2P network includes untrusted nodes from an open environment, such as the Internet. Users from all parts of the world interact and share their resources without knowing each other. Untrusted nodes may be faulty, malicious, and act together to attack the network. Consequently, the quality of service of applications may be deteriorated due to message overhead or data loss.

There are two categories of P2P networks: structured and unstructured ones. Our work is oriented on the former, and especially on Distributed Hash Tables (DHT), which provide efficient key lookups, high data availability and persistence.

Among the existing solutions for building trust in P2P networks, reputation systems [1] and accountability [2] have shown to be very good and efficient approaches.

Accountability detects and exposes faulty nodes by creating non-repudiable records of every node's actions. PeerReview [2], for example, is able to detect even a single misbehaviour since it is based on deterministic actions. However, the actions of every node have to be periodically checked by other nodes, who must replay the protocol using the input in the log. If the protocol is computationally complex, this results in a heavy cpu load. In addition, accountability does not detect malicious behaviours that are not protocol related or not verifiable deterministic transactions.

Reputation systems assess the past history of a peer by gathering feedback from nodes with previous interactions with this peer. This evaluation is an estimation of the peer's future behaviour in a context related situation [1]. Reputation systems are easily applicable, since there is no need to know which protocol the nodes are following. However, this means that the feedback information does not constitute an irrefutable proof of misbehaviour; it also makes it nearly

impossible to detect a one time attack. If a node fails once, it can redeem itself and improve its reputation by carrying out correct transactions.

Reputation systems only provide the ability for a node to find the reputation of a given node in order to decide whether or not to do a transaction with it.

The goal of this paper is to propose an efficient and simple way to build a group of trusted nodes – called *Trusted Ring* – within a DHT. This allows to coalesce reputable nodes in order to provide a pseudo-trusted service: a service that is almost fully trustworthy, since the nodes that process requests exhibit a very high probability of being trustworthy.

CORPS (Community Of Reputable PeerS) builds the *Trusted Ring* using a reputation system to decide whether a node can be a member of the group or not. The main contributions of our work are:

- A new approach to build trustworthy services in P2P networks that is based on a reputation system.
- A membership algorithm to build a scalable *trusted ring* within a DHT that allows to find reputable peers efficiently.
- An example of how to build a pseudo-trusted service by way of the *Trusted Ring* and an evaluation of the reliability of the service.

Section 2 explains the properties of the underlying systems: DHTs and reputation systems. The design of the CORPS algorithm including the join and leave procedure, as well as the maintenance techniques are given in Section 4. A pseudo-trusted routing service is presented in Section 5 to illustrate the benefits of our approach.

Section 6 presents a theoretical evaluation, some simulations results and costs in order to evaluate the performance of CORPS. As security issues are a major concern in P2P networks, a discussion in Section 7 shows the ability of CORPS to resist some traditional attacks like the Sybil attack [3] or collusion of nodes. We also give some hints about the portability of CORPS over existing DHTs.

Finally the conclusion presents an overview of the results and benefits of our approach, and gives some directions for future research in trusted P2P services.

2. BACKGROUND

We present in this section the underlying systems of our approach. First, a description of the type of P2P overlay we are interested in, and then, an overview of Reputation Systems that are used to identify trustworthy peers.

2.1. DHT-based P2P overlay networks

P2P networks are autonomous, self-organised and highly scalable systems that have the potential to grow up to millions of nodes. P2P nodes share their resources, collaborate and interact directly. We will focus on DHTs, which provide efficient key lookups, high data availability and persistence.

DHT networks are systems that maintain a strong topology, for example a ring in the case of Chord [4], Pastry [5] or Tapestry [6]. We assume the use of Pastry throughout this work as the underlying P2P overlay, but the use of any other ring-like topology, such as Chord or Tapestry, is straightforward. Please refer to Section 7.2 for more details about the portability of our proposition to other DHT networks.

Every node in Pastry is assigned a unique nodeID in a space of identifiers of 128-bits, generated using a cryptographic hash [5]. The nodeID determines the position of the node in the circular namespace, that is the Pastry ring. In Pastry, a key k is a value in the same namespace of the nodeIDs.

The neighbours of a node in Pastry are stored in two sets: (1) the *leaf set* contains the L numerically closest nodes in the ring, organised in $L/2$ clockwise closest nodes, and $L/2$ counter-clockwise closest nodes. (2) *the neighbour set* contains the K closest nodes in terms of latency. To maintain these two sets of nodes, a keep alive message is used every time Δt to detect node failures. When a node X realizes that a node in its leafset has fail, X starts an update procedure to repair its leafset. A typical value for the leafset size is 16 nodes.

The Pastry routing algorithm is a prefix based algorithm that routes a message to the numerically closest node of a given key k in the ring. Figure 1 shows an example of the routing process. The Pastry routing table stores on the n^{th} row the IP address of nodes which nodeID share the first n digits with that of X [5]. The algorithm forwards the messages to a node from its routing table that shares at least one more digit with the key k than the current node. If no such a node can be found and the current node does not know any other numerically closest to the key k, then the current node is the closest and the routing ends.

This algorithm allows to route a message to a given key k in $\mathcal{O}(\log(N))$ hops, where N is the network size [5].

2.2. Reputation Systems

Reputation Systems mitigate the problem of malicious nodes in P2P networks, trying to build trust among the nodes. The key idea of a reputation system is to predict the future behaviour of nodes based on feedback about their past transactions [1]. A transaction is application dependent, for example forwarding a message in the network, buying an item in e-commerce services, share or store files, etc. After a transaction, the client node emits a recommendation that evaluates the behaviour of the other peer. The aggregation of these recommendations leads to a reputation value.

A reputation system built on top of a DHT has the

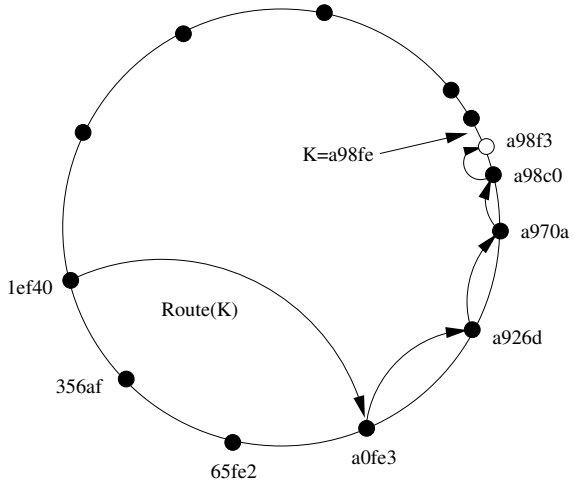

FIGURE 1. Routing example in Pastry: Route of key $a98fe$ arrive to the node with the closest nodeID $a98f3$.

ability to compute a global reputation value for every node. Indeed all the recommendations about a single node can be handled consistently at a common location: either by a specific node or by a set of nodes. Among existing reputation systems for DHTs, we can cite: PeerTrust [7], WTR [8], Eigentrust [9], PowerTrust [10].

Reputation systems have to deal with malicious nodes that: do not participate, collude with other malicious nodes, and emit false recommendations. There are techniques to mitigate the impact of these attacks, such as the ones presented in TrustGuard [11] and WTR [8]. Nevertheless, to our knowledge, none of the existing solutions to promote trust in P2P can be 100% effective in detecting and blocking these attacks.

3. PROBLEM FORMULATION

We consider a DHT network in which each node has an associated reputation. In this section we formalize the problem, we state assumptions, and desired properties of the system.

3.1. System Model

Our underlying P2P overlay consists in a Distributed Hash Table, such as Pastry [5]. Any node may fail or leave the system at any time.

We assume that a unique nodeID is assigned to a node which enters the DHT. NodeIDs are uniformly distributed in a 160-bits nodeID space. We suppose that each node has a self-generated public/private key pair. The nodeID is generated by computing the Secure Hash Function SHA-1 of the node's public key. When entering the DHT, the new node must provide its public key and its nodeID.

3.2. Design Goals

Our aim is to provide trust within the network by building a group of pseudo-trusted nodes: the *Trusted Ring*. The group must be self-organised and scalable. Any node of the DHT should be able to access a pseudo-trusted node in the *Trusted Ring*, but the possibility of not finding such a node remains.

The target for our proposition are best-effort systems which tolerate recoverable faults. The goal of the *Trusted Ring* is to improve the way these systems operate by providing an efficient method for finding nodes that perform correctly.

We intend for our system to achieve the following properties:

- Availability: After a start-up phase the *Trusted Ring* should be as available as possible.
- Robustness: The system must cope with malicious behaviours of nodes.
- Convergence: After a start-up phase, all the trustworthy nodes should be part of the *Trusted Ring*.
- Performance: The system should generate a low traffic overhead.

Our system must not introduce more vulnerabilities to the underlying system. An important design principle is to avoid bottleneck problems. Limiting the awareness of nodes to a partial knowledge of the network is important in order to provide load balance and scalability.

3.3. Trust Model

We consider a probabilistic model of trust based on reputation. The reputation value $R(X)$ is the probability that node X will be honest in the future. This reputation value is computed according to a list of recommendations emitted by nodes that have already carried out transactions with X.

After each transaction, a node emits a recommendation about its peer. A node may lie: it may emit negative recommendations about a peer that behaves correctly, or positive recommendations about a malicious peer. Several nodes may collude to increase or decrease the reputation value of another node. These problems generate a deviation between the computed reputation of the node and its real behaviour. We consider that this deviation depends on the function used to compute the reputation value and on the percentage of malicious nodes within the system.

In the following, we assume there is a reputation system in the overlay structure with the following properties:

- Every node X has an associated reputation value $R(X)$ which represents the probability that X is an honest node.
- $R(X)$ is computed using the recommendations emitted by nodes that have completed a transaction with X. Bad recommendations have a stronger effect on $R(X)$ than good ones. It should be more difficult for a node to increase its reputation value than to decrease it.

- For every node X, $R(X)$ is highly available in the DHT.

To avoid nodes that lie about a reputation value, our model will consider a reputation system that computes the reputation of nodes concurrently on different nodes. They decide individually if that node is reputable or not. We use a voting scheme in case of disagreement. On the whole, assuming that there is a smaller percentage of malicious nodes in the network, the result avoids false statements about reputation.

3.4. Formalization

Formally, the goal of this paper is to construct a set of pseudo-trusted nodes, TS such that:

- $TS = \{n_1, \ldots, n_i, \ldots, n_N\}$, and for each node $n_i \in TS$, $R(n_i) \geq \rho$ where ρ is a threshold with $\rho \in [0 \ldots 1]$.
- Let N be the real number of honest nodes in the overlay, and t the time. Then:

$$\lim_{t>T}(Card(TS)) = N \quad (1)$$

In other words, after a time T, the number of nodes in TS tends to be N.

- After time T, if honest nodes tend to have the maximum reputation value, $\lim_{t>T} R(n_i) = 1$, then, the sum of their reputation values will be equal to the number of honest nodes.

$$\lim_{t>T} \sum_{i=1}^{N} R(n_i) = N \quad (2)$$

In all the following sections, we will call a node $n_i \in TS$ a trusted node, even if there still remains a probability that this node's actual reputation is smaller than the threshold ρ. For more details please refer to Section 6.1.

4. BUILDING THE TRUSTED RING

In this section we describe the CORPS: our membership algorithm used to build the *Trusted Ring*. The *Trusted Ring* provides an efficient way to contact a trustworthy node in order to involve it in a pseudo-trusted service.

4.1. Architectural principles

The general view of our system is a double ring: our *Trust Ring* nested into the normal DHT. The join, leave and update operations – explained hereafter – respectively allow a node to enter the *Trusted Ring*, to leave it, and to update its local vision of this ring. A monitoring procedure allows to check the capacity of a member of the *Trusted Ring* to remain into the ring, and a remove procedure expels a node from the ring.

Figure 2 shows the basic structure of the system. Some nodes are identified as trusted and some as normal.

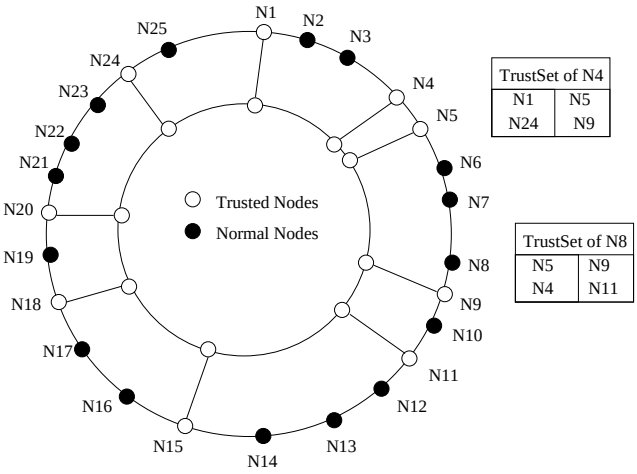

FIGURE 2. Example of *trustset* structure

Every node in the DHT maintains a *trustset*. The *trustset* is a set of D numerically closest trusted nodes, with $D/2$ clockwise nodes, and $D/2$ counter-clockwise ones. This concept is similar to the *leafset* in Pastry, but instead of collecting the nodeIDs of the numerically closest nodes it references the numerically closest nodes among those that are actually trusted. The trusted nodes which have a node X in their *trustset* will be called *managers* of X. All the normal nodes that will have X in their *trustset* will be called the *followers* of X.

A node's ability to be part of the *Trusted Ring* must be regularly assessed. In order to decide if a node can enter the *Trusted Ring*, we use the underlying reputation system. We define a threshold ρ, such that node X should be part of the Trusted Ring if and only if $R(X) > \rho$. The value of ρ depends on the level of security the application requires. The closer ρ is to 1, the higher the confidence in the node's trustworthiness. However, it is essential to reach a compromise between this confidence and the number of nodes inside the *Trusted Ring* in order to provide an efficient service.

The *Trusted Ring* is managed in a way similar to the Pastry overlay, explained in Section 2. Every node in the *Trusted Ring* has a *trustset* – an equivalent to the leafset which references reputable nodes.

4.2. Initializing the TrustSet

The initialization process for a new node that joins the DHT network consists in building its *trustset* from scratch. When the *join* in the Pastry ring is terminated, the node gathers *trustsets* from the nodes in its leafset. If this process does not return any trusted node, then there are no available trusted nodes in the numeric neighbourhood of the node: the *Trusted Ring* is temporarily unavailable for this node. We analyse in Section 6.3.4 the minimum ratio of trusted nodes that propels the *Trusted Ring* towards full availability.

4.3. Joining the Trusted Ring

After each transaction, the reputation of a node can change. Once the reputation of a node X reaches the threshold ρ, X sends a `Join message` to all the nodes in its *trustset*, which will become its *managers*. The managers of X use the reputation system to check if the reputation of X is effectively greater than the threshold ρ. If it is, then each manager adds X to its own *trustset*. A node that lies about its ability to enter in the *Trusted Ring* cannot enter to the *trustsets* of the others nodes, and remains alone.

The *Join* protocol introduces new trusted nodes into the *Trusted Ring*, but the changes remain transparent to normal nodes. For this reason, the joining process requires an *announcement phase*. In this phase a new trusted node sends its *trustsets* to every node in its leafset. Malicious nodes can send *false announcements*, declaring they are trusted nodes. To avoid this behaviour when a node receives new announcements must check the reputation of the new trusted node.

In Figure 2, if N8 is a new trusted node, it sends a `Join message` to its *trustset*: N4, N5, N9, N11. These nodes check if N8 really has a reputation such that $R(N8) > \rho$, and forward this information to the nodes in their leafset. N7, N10 and the other nodes in the leafset of the managers will become the followers of N8. Each manager maintains a list of followers for all the nodes in its own *trustset*.

When node A adds a new trusted node X in its *trustset*, it contacts X's managers and subscribes to X's *follower-list*.

4.4. Monitoring scheme and Remove procedure

The nodes in a P2P network can change their behaviour as they carry out new transactions. Some nodes may exhibit oscillating behaviours, which can generate a significant yo-yo effect with respect to the reputation value. Applying a *Join* procedure to these nodes would cause a useless overhead. The reputation of X is checked by its managers periodically, every Δt. The value of Δt is a parameter of the system; in our experiments we used the same value as that used by Pastry to update the leafset. While the reputation of X is within $[\rho - \alpha, 1]$ then the node is still considered as trusted. α is a tolerance parameter of the monitoring process. If $\rho - \alpha$ is too small, the *Trusted Ring* may endure a high level of churn. If $\rho - \alpha$ is too big, then nodes with a reputation value slightly lower than the threshold should be able to remain as trusted nodes. A good value for α should be chosen according to the effect of bad recommendations at the reputation system level. If bad recommendations have an important effect on the reputation decrease, and it is much more difficult for a node to increase its reputation, then a small value can be chosen for α.

When a manager detects that $R(X) < \rho - \alpha$, it alerts all the followers of X that X can not be considered as trusted any more, and removes X from its *trustset*. If the size of the *trustset* is D with $D = 3k + 1$, then when a node A receives at least $2k + 1$ remove alerts from $2k + 1$ distinct trusted nodes, it removes X and updates its *trustset*.

When a node in the DHT fails and becomes unavailable, this is detected by the normal leafset maintenance algorithm of Pastry. When node Z detects that a node X has gone, it alerts all the nodes in its *trustset*. If X was a member of the *trustset*, then the nodes of the *trustset* will alert all the followers of X (`Failed message`) that X has failed.

4.5. Leaving the Trusted Ring

When a trusted node X wants to leave the DHT, it sends a `Leave message` to all the nodes in its *trustset*. The managers of X forward the message to all the followers of X and each follower runs an *Update* procedure.

4.6. *Trustset* update

When a node leaves the *Trusted Ring*, because of a normal departure, or because of a remove procedure (`Leave message`, `Remove message`, and `Failed message`), the followers of X must update their *trustsets*. The update procedure is similar to the leafset maintenance in Pastry. Every follower contacts the last available node in its *trustset*, clockwise or counter-clockwise according to the position of the leaving node, and asks for its *trustset*. Then each follower replaces the missing node with the numerically closest node from this *trustset*.

4.7. Start-up phase

If the newly trusted node has an empty *trustset* it will start a new *Trusted Ring* by itself. The outcome of this process is that several ring structures can coexist in the same P2P network unbeknownst to one another.

Self-organisation and convergence come into play when two *Trusted Ring* structures find each other and merge. When a node Z discovers a node X from another ring, then X (and the nodes in its *trustset*) can be inserted into the *trustset* of Z, and X can replace a node in Z's *trustset* if it is numerically closer to Z. Both rings merge by way of node Z.

4.8. Finding a trusted node

With the information in their *trustset*, each node knows the location of their D numerically closest trusted nodes. Finding a trusted node for node X simply consists in looking into its *trustset*. If its trustset is empty, then there is no trusted node available for X. Node X will try to initialize its *trustset* again to search for new information as described in Section 4.2.

Considering that nodes are randomly distributed in the nodeID space (honest and malicious ones), it is highly unlikely for any node to have an empty *trustset* during the stable phase of the *Trusted Ring*. For more details about the stabilization phase, please see Section 6.3.4.

4.9. Incentive mechanism

Our *Trusted Ring* is composed of the "*elite*" of the P2P network. Any node can access it, including malicious nodes and freeriders that do not participate actively to the system. If any node can use the system and if there aren't enough nodes in the *Trusted Ring*, then the latter becomes a bottleneck and the load will increase and discourage nodes from participating.

We introduce a way of encouraging nodes to participate in the *Trusted Ring* by giving priority to those nodes when using the pseudo-trusted service. Requests from nodes inside the *Trusted Ring* will be answered faster. We apply weighted fair queuing [12] on the routing of request at every node, with higher weight for requests from trusted nodes.

A further incentive to improve the reputation value of the nodes is introduced. We allow access to the *Trusted Ring* only to nodes that have at least done some work in the network. When a trusted node is contacted by a peer X, it first verifies the reputation of that peer. If all nodes start with an initial reputation value of $R(X) = \beta$, node that can access the Trusted Ring have to prove a $R(X) > \beta$ – meaning that in the worst case X has made some good work in the network.

5. EXAMPLE OF A TRUSTED ROUTING

In this section, we give an example of how to use the *Trusted Ring* for a trusted routing service. Section 6 will cover a theoretical evaluation of the *Trusted Ring*, as well as the benefits in the case of trusted routing.

We do not use the traditional routing algorithm at the Pastry level, but the one proposed in [13] called Tracer Routing. The goal of the Tracer Routing algorithm is to route a message M to the numerically closest node of a given key k, similarly to Pastry. However, the Tracer Routing algorithm uses a public/private key scheme to identify the first malicious node that was encountered in case of routing failure. The key idea of Tracer Routing relies on the following:

- Each node of the routing path informs the initiator of the routing about the next hop, and signs the message with its private key.
- When a node receives a routing message, it acknowledges the message to the initiator and forwards a signed message to the next hop.
- The message content is encrypted by the initiator private key.

The initiator is able to verify the next hop of the routing, if the message does arrive to the next hop and if the message has been altered. Moreover, if no acknowledgement is received, the initiator can determine which node in the path is faulty. For more details on the Tracer Routing, please refer to the work of Xiang and Jin [13]. Note that Tracer Routing has a higher cost than traditional Pastry routing in terms of the number of hops, but it is still in $O(log(N))$ where N is the size of the DHT.

Trust is specific to a domain: to achieve pseudo-trusted routing the reputation system has to compute the reputation based on how a node behaves while forwarding messages. The nodes will achieve a high reputation by correctly routing messages for other nodes. In this context, the initiator will emit a good recommendation value of 1 for each node that routes the message correctly, and a bad recommendation value of 0 for the first faulty node in the routing path. Thus nodes that behave correctly will increase their reputation in the context of routing, and malicious nodes will decrease theirs.

After the start-up phase, the *Trusted Ring* will contain almost all the nodes that behave correctly while routing. We can then easily implement a trusted routing service within the *Trusted Ring*, by using the same routing algorithm as Pastry

Since the *Trusted Ring* is also a DHT, we suppose that every trusted node manages two routing tables. The first one is the normal routing table of the Tracer Routing, and the second one corresponds to a routing table that contains only trusted nodes. The maintenance of the trusted routing table is similar to Pastry, and the replacement of a node is optimistic.

The next section gives an evaluation about the benefits of using our *Trusted Ring* for trusted routing, including an analysis regarding the probability of routing failures compared to the normal routing service.

6. EVALUATION

In this section we present a theoretical evaluation of CORPS, as well as a set of simulations and performance results.

6.1. Theoretical evaluation

We suppose in the following that the underlying reputation system makes an error ε when classifying a node X with a reputation $R(x) \geq \rho$, where $\rho \in [0 \cdots 1]$, and $\varepsilon = f(\rho)$. In other words, classifying a node X as honest because its reputation is greater than ρ has a probability of error ε.

Let n be the size of the *Trusted Ring*. The probability to have k misclassified nodes in the Trusted Ring, that is k malicious nodes is:

$$P_{k_{malicious}} = \left(\begin{array}{c} n \\ k \end{array} \right) \varepsilon^{n-k}(1-\varepsilon)^k \qquad (3)$$

Then, the probability to have at most k malicious nodes in a *Trusted Ring* of size n is:

$$P_{\leq k} = \sum_{i=1}^{k} \left(\begin{array}{c} n \\ k \end{array} \right) \varepsilon^{n-i}(1-\varepsilon)^{i} \quad (4)$$

6.1.1. Trusted Routing Failure

A trusted routing fails when the message cannot be delivered to the closest trusted node of a given key k. This may happen when at least a node on the routing path fails. Let h be the number of hops to route a message in the *Trusted Ring*. As the trusted routing algorithm uses the same prefix based routing as Pastry, h is such that $h = O(log(n))$ where n is the *Trusted Ring* size. Then the probability for a routing to succeed – that is all nodes on the path behave correctly – is given by:

$$P_{r-succeed} = (1-\varepsilon)^{h} \quad (5)$$

Given this result, the probability to have to make one more routing knowing that $r-1$ have already failed is given by the following geometric law:

$$P_{r} = P_{r-succeed} \times (1 - P_{r-succeed})^{r-1} \quad (6)$$

with

$$\lim_{r\to\infty} \sum P_{r} = 1 \quad (7)$$

Then the probability to succeed after at most r routings is:

$$P_{atmost-r} = \sum_{i=1}^{r} P_{r-succeed} \times (1 - P_{r-succeed})^{i-1} \quad (8)$$

and the probability to require more than r routings to succeed is then:

$$P_{more-r} = 1 - P_{atmost-r} \quad (9)$$

$$E(r) = \frac{1}{P_{r-succeed}} \quad (10)$$

Table 1 compares the expected number of tries required to reach a key k successfully with Pastry and with pseudo-trusted routing. We suppose that we are in a context where 30% of the nodes will reach the ρ threshold. The first column represents the percentage of malicious nodes in the network. The rest of the nodes are regular nodes that are not really malicious, but they won't reach the ρ threshold. We also suppose that only malicious nodes do not correctly route messages in traditional Pastry.

Without any malicious node, the traditional Pastry Ring and the *Trusted Ring* are obviously equivalent as there is no node against which to discriminate.

Figure 3 compares the number of attempts require to achieve a successful routing with Pastry and with our *Trusted Ring* assuming 30% of malicious nodes. $E(r)$ is always smaller for the *Trusted Ring* than for Pastry alone, but the difference between both exponentially increases when the percentage of malicious nodes increases. The greater the number of malicious nodes, the greater the benefit of the *Trusted Ring* as it allows to discriminate against malicious nodes.

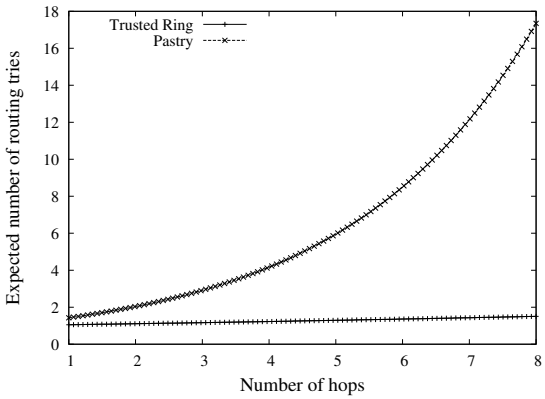

FIGURE 3. Number of expected routing attempts with respect to the number of hops

To our knowledge, none of the existing reputation systems that compute a global reputation value, can handle more than 30% of malicious nodes in the network. They will become unstable and then collapse, unable to distinguish malicious nodes from normal ones. In such a case, the *Trusted Ring* will also collapse.

6.1.2. Trusted Set Evaluation

Every node's *trustset* contains D numerically closest trusted nodes, with $D/2$ clockwise and $D/2$ counter-clockwise nodes. We suppose that after a start-up phase, the *Trusted Ring* reaches a stable state in which it comprises almost all the trusted nodes of the network. During the stable phase, each node has a high probability of owning a *trustset* that is full.

The worst case for our solution is when the number of malicious nodes is just lower or equal to 30%. In a 5 millions nodes network, with 1.5 millions of real trusted nodes, according to equation 8, the probability to have to make at most 9 tries for a successful routing is 6.42×10^{-6}. At the same time, the expected number of tries is 1.5. If we consider a typical size of 16 nodes for the *trustset*, then according to equation 4, the probability to have at most $16 - 9 = 7$ malicious nodes of 16 is 1.6×10^{-8}, which is a very low probability.

Thus, using a *trustset* of 16 nodes as starting points is more than sufficient to ensure successful routing *almost* all the time.

Note that the probability for all the nodes of the *trustset* to be malicious, assuming a maximum classification error for the underlying reputation system of 5%, is 1.52×10^{-21}. Hence the probability for having a fully erroneous *trustset* is theoretically possible, but practically infeasible.

	Pastry			Trusted Ring			
M%	**Size**	$\lceil O(log_{2^b} N) \rceil$	$E(r)$	**Size**	$\lceil O(log_{2^b} N) \rceil$	ε	$E(r)$
	10^5	5	1.29	30×10^3	4	0.01	1.04
5%	5×10^6	6	1.36	1.5×10^6	6	0.01	1.06
	500×10^6	8	1.50	150×10^6	7	0.01	1.08
	10^5	5	2.25	30×10^3	4	0.03	1.13
15%	5×10^6	6	2.65	1.5×10^6	6	0.03	1.20
	500×10^6	8	3.66	150×10^6	7	0.03	1.28
	10^5	5	5.95	30×10^3	4	0.05	1.22
30%	5×10^6	6	8.50	1.5×10^6	6	0.05	1.36
	500×10^6	8	17.35	150×10^6	7	0.05	1.5

TABLE 1. The number of expected tries to succeed a routing

6.1.3. Discussion about the reputation system error ratio

Nearly all existing reputations systems make errors when computing and classifying honest nodes in the network. This is mainly due to the following reasons:

- The reputation of a node can be computed with a low amount of recommendations, and does not necessarily represent the real behaviour of the node. Reputation systems with some kind of risk management like WTR [8] and TrustGuard [11] can mitigate this effect. Using an initial reputation value of 0.5 as in WTR can also help to reduce this problem, requiring a minimum number of recommendations to be able to reach the ρ threshold in order to join the *Trusted Ring*.
- A node can be affected by typical attacks like the collusion of malicious nodes, and this can affect a node's reputation such that the node appears with a better reputation than expected – associated recommendations can be rumours or false recommendations. Existing reputation systems can mitigate this kind of malicious behaviour by weighting the recommendations with the emitter's reputation.

Considering a maximum error rate of 5% is a typical value for a reputation system. In some cases it may be over-estimated (for more details, please refer the results obtained for the WTR reputation system [8]). This error hardly depends on the total number of malicious nodes in the network, and decreases when the ratio of malicious node decreases. The less malicious nodes there are in the system, the easier it is to discriminate against them.

6.2. Performance

Our system does not depend on the size of the P2P network and is built on top of a structured overlay. For these reasons it inherits its scalability, self organisation and locality properties.

In the following, we will call *direct message* a message sent knowing the IP of the peer. We will call *reputation messages* the messages sent to the reputation system in order to acquire the reputation value of a node. Our evaluation is based on WTR [8] and PeerTrust [7], where one reputation message implies approximately $\log(N)$ messages, with N the number of nodes in the network.

The total number of messages sent during a join process is $(D+L)*(1+\log(N))$, with D the number of nodes in the *trustset*, L the number of nodes in the leafset and N the number of nodes in the network. This total accounts for $(D + L)$ direct messages to the nodes in the leafset of the new trusted node and $(D+L)*\log(N)$ messages from the reputation messages. Experimentally, the value of L is typically configured as 16 in Pastry. With the goal of achieving the same properties of robustness we set D equal to 16. If we consider a network of one million of nodes, the total number of messages during the join process will be 224.

The number of messages exchanged during the removal of a node is $D * F$, with D the number of managers that send messages to the follower list of size F. Our simulation experiments show that the size of the follower list remains approximately the same as D. Given a follower list size of 16, the number of messages during a removal process will be 256. If nodes do not subscribe to all the follower lists of the managers, but randomly choose $D/2 + 1$ of them, then this number can be reduced to 144.

It is important to note that this number is small in the P2P context, as it is distributed among the nodes of the *trustset*. In addition, after the start-up phase the number of `Join messages` should significantly decrease, and the number of `Leave` and `Remove messages` will likely be small.

The number of maintenance messages for each trusted node is $D * \Delta t$, since you have to monitor every child node, every time. We do not need to use $\log(N)$, since after the first time, we can contact the same node to do the reputation request. For some reputation systems, for instance WTR [8] and PeerTrust [7], the maintenance messages can be eliminated by monitoring the reputation information from the source. WTR uses reputation managers to save the recommendation of nodes. We think a good improvement to the original solution, will be to use this reputation manager to

detect suspicious behaviours.

The parameters that define the level of security of the *Trusted Ring* are:

- ρ: The boundary of the reputation value. The higher this value, the harder it is to be part of the *Trusted Ring*.
- α: the tolerance factor. The higher this value, the higher the possibility to have untrusted nodes in the *Trusted Ring*.
- D: Defines the amount of knowledge about the rest of the network: how many trusted parties a node knows. This parameter makes the system more robust against malicious attacks.

6.3. Simulations

This section presents a set of simulation experiments designed to evaluate the performance of our *Trusted Ring*. For this purpose, we built an experimentation platform which allows to simulate the execution of our solution in a P2P network composed of up to 100,000 nodes. The platform was written in C and all the tests were performed on an Intel Core 2 Duo processor, 2.66 GHz, 4GB RAM, with Linux 2.6.28.

The simulation consists in a P2P network in which each node has a unique identifier computed with the SHA-1 function. As in Pastry, every node knows a leafset of nodes and takes part into a ring structure. Nodes use WTR [8] as the underlying reputation system.

The reputation information is generated by simulating random transactions among nodes, and emitting the resulting recommendations. Every transaction out of 10 requires a node to contact its trustset.

6.3.1. Behaviour Model

The nodes in the simulation have a fixed personality. To define the behaviour of each node, we use the recommendation values defined in WTR: a value of 1 characterises a very good transaction, a value of 0.75 a good transaction, a value of 0.5 a neutral result, 0.25 an incomplete transaction, and 0 a malicious transaction.

- *Honest nodes* obtain a recommendation value of 1 in 80% of the cases and a 0.75 recommendation value in 20% of the cases.
- *Regular nodes* obtain a recommendation value of 1 in 20% of the cases, 0.75 in 50% of the cases and 0.5 in 30% of the cases.
- *Malicious nodes* obtain a recommendation value of 0.5 in 20% of the cases, 0.25 in 30% of the cases and a recommendation value of 0 in 50% of the cases.

These values are based on typical probabilistic behaviours found in the literature about reputation systems, and make the simulation more realistic.

Parameter	Value
Default number of nodes	100,000
Size of the trustset	16
Size of the history	3
Trusted Nodes	30 %
Regular Nodes	50 %
Malicious Nodes	20 %
Reputation value boundary (ρ)	0.8
Churn	Y:1% - X: 10%

TABLE 2. Nodes and Network configuration

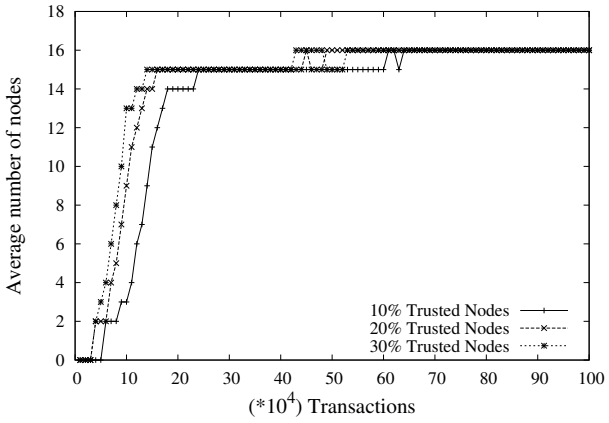

FIGURE 4. Size of the *trustset* on trusted nodes

6.3.2. Simulation Setup

We simulate a dynamic P2P network of N nodes under churn conditions. Every Y transactions (by default 1%), X random nodes (by default 10%) will leave the network and X others will initiate a join process, thus maintaining a constant number N of nodes in the system.

As in a real network a node can join the network at any time, and the same node can leave and join repeatedly.

With respect to Pastry, every simulation uses a value of 16 for the size of the leafset.

The default configuration is shown in table 6.3.2.

A *snapshot* of the system is taken every 1% of the total number of transactions. Each experiment is run 5 times and an average of the results is presented.

6.3.3. Convergence

We consider that our system converges when the trustworthy nodes are part of the *Trusted Ring* and the ring is consolidated: in other words, when the start-up process ends. To probe this characteristic, we measure in every snapshot the average size of the *trustset* of the honest nodes in the network. In every experiment, we change the percentage of honest nodes present in the network at the same time. Figure 4 shows the results.

The curves start at value 0 because at the beginning the nodes do not know any trusted node. Their upper bound is 16, the maximum size of the *trustset*. The

results show that the convergence is similar even when there are a small number of trusted nodes. Trusted nodes can easily find each other in this configuration. We can appreciate that the size of the *trustset* grows very quickly. After approximately 20×10^4 transactions the *trustset* of all the honest nodes is full. This is the number of transactions necessary for the reputation system to build a reputation value for each node, and in practice it means that every node has carried out at least 2 transactions.

6.3.4. Availability

The *Trusted Ring* is fully available when any node in the P2P network manages to query it successfully. In this experiment, if a node has a full *trustset*, it considers the *Trusted Ring* as 100% available. The availability depends on the number of honest nodes in the network, the number of malicious nodes and on the level of churn.

We conducted three experiments to analyse the behaviour of our system. First, we measured the average size of the *trustset* of the nodes that cannot belong to the *Trusted Ring* with different percentages of honest nodes. Figure 5 shows how fast common nodes build their *trustset*. Compared to the results of Figure 4 we can see that the construction of the *trustset* is much slower. The percentage of honest nodes has a significant impact on the acquisition of trusted nodes. As the nodes continue to make transactions, the information about trusted nodes starts spreading faster.

The system stabilises after approximately 50×10^4 transactions, which is approximately 5 transactions per node.

Experiments with several degrees of churn are presented in Figure 6. Every 10^4 transactions, a random selection of 10%, 20% or 30% of the nodes will leave the network while others will initiate a join process to maintain a constant number $(100,000)$ of nodes in the system. In this second experiment, we measure the same values as in the first one.

Up to a churn ratio of 20%, the system maintains its behaviour. With a 30% ratio the system becomes unstable: the size of the *trustsets* varies significantly and the traffic due to join and search messages increases considerably. Moreover, trusted nodes are not easy to find when they don't stay long enough to be part of the system. We can conclude that there is a limit with respect to the level of churn beyond which the *Trusted Ring* cannot maintain the information available to the nodes.

When a node requires a trusted node it searches in its *trustset*. Our third experiment measures the percentage of queries that the *Trusted Ring* can handle. We use the default configuration for this experiment. Figure 7 presents the results. In every snapshot the nodes send 1000 queries to the *Trusted Ring*. When the *trustsets* are not built yet, the *Trusted Ring* is unable to handle any of the queries. The service starts behaving

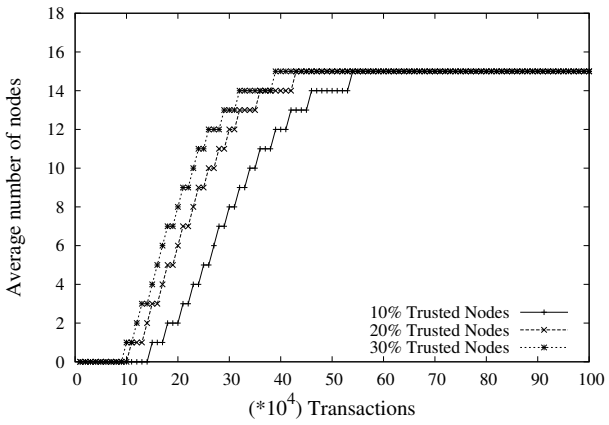

FIGURE 5. Size of the *trustset* on non trusted nodes

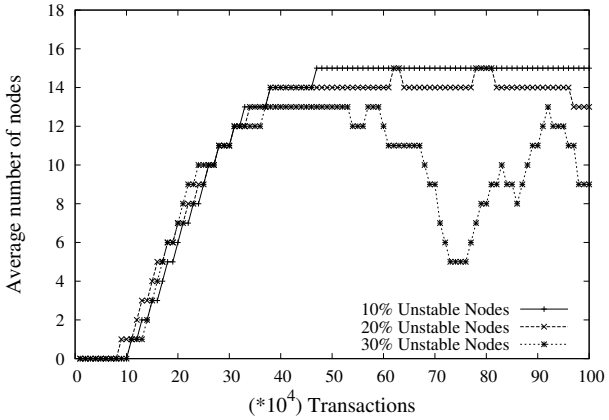

FIGURE 6. Size of the *trustset* on common nodes with respect to the churn ratio

satisfactorily when the information is distributed and the trusted nodes are detected.

The system stabilises after 50×10^4 transactions: the same results as in Figure 5.

6.3.5. Threshold ρ

The selection of ρ not only impacts the number of honest nodes allowed into the *Trusted Ring*, but also the number of malicious nodes that can enter it.

After the start-up phase, the reputation of the trusted nodes should be close to 1, as stated in the equation 2. However, malicious nodes emitting false recommendations about the honest ones make this result close but not equal to the number of honest nodes.

Figure 8 shows how the sum of the reputation values of the trusted nodes varies with respect to the selected value ρ for the threshold. The percentage of honest nodes in the network is 30% and the percentage of malicious nodes emitting false recommendations is 20%. When choosing a low value of ρ as 0.6, the result of the equation 2 shows that there are trusted nodes in the *Trusted Ring* that are not honest. The opposite

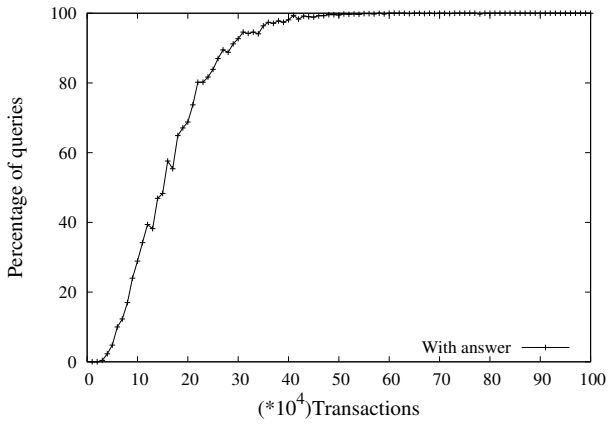

FIGURE 7. Hit ratio of the *Trusted Ring*

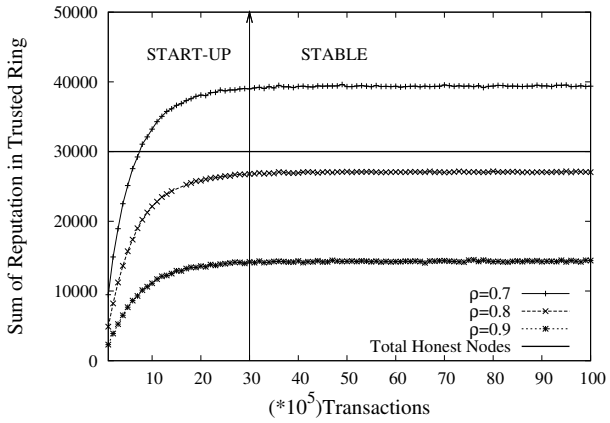

FIGURE 8. Convergence of the *Trusted Ring* size

happens when choosing a high value of ρ as 0.9, not all the honest nodes are allowed into the *Trusted Ring*, since the threshold is too high for the number of malicious nodes emitting false recommendations. For our configuration, a better option is $\rho = 0.8$, since the limit of the reputation of the nodes in the *Trusted Ring* is closer to the number of honest nodes in the DHT.

7. DISCUSSION

In this section we will discuss some security issues, the portability of our solution and give some example applications of pseudo-trusted services.

7.1. Security Issues

As explained in Section 6.1, the Sybil attack and Collusion attack are important threats to any P2P system and can degrade the reputation system. Malicious nodes can lie about the behaviour of others, or initiate fake transactions to improve their own reputation value. This section discusses the implications and possible solutions to mitigate them.

7.1.1. Sybil attacks

The Sybil attack [3] is a major concern if obtaining network identities is cheap. Such an attack allows to tamper the ratio of malicious nodes in the system as a single node acts under multiple identities: the ratio can then rise above the 30% upper threshold.

A common practise of the Sybil attack has a single malicious user create several identities with numerically close nodeIDs to be able to control a substantial part of the DHT. We assume that the generation of the nodeIDs can be verified by any node in the network, for instance the bootstrapping node or the node with the closest nodeID. Our proposition assumes that nodeIDs are computed from a public key, generated with a well-known public/private key algorithm. We also suppose that the number of bits of these keys is sufficient to avoid a brute force attack to reverse the SHA-1 function. During the *Join* procedure to the DHT, a node Z must provide several pieces of information to the verifier node X: its public key key_{Pub}, and the associated nodeID computed using $nodeID = SHA1(key_{Pub})$. X can easily verify the correctness of the nodeID with the public key. If they do not match, then X prevents Z from joining the ring.

Due to the mathematical properties of the SHA-1 hash function, it is practically impossible for a node to find two keys key_{Pub1} and key_{Pub2} such that $SHA1(key_{Pub1}) = SHA1(key_{Pub2})$. Therefore, it is also practically impossible for a node to find several public keys that give numerically close nodeIDs in the ring. Thus it is practically infeasible for a node to control a substantial sector of the ring.

However, a single node can still generate several nodeIDs distributed over the ring. This kind of Sybil attack does not allow to control a sector of the ring, but will increase failures in the DHT (Routing failures, etc.). Reputations systems like PeerTrust [7] or WTR [8] can efficiently mitigate the Sybil attack, but will collapse if the ratio of malicious nodes is too high. To our knowledge, no existing reputation system can control a large number of Sybil attacks in a DHT, and this is still a difficult topic within the P2P research community.

Other solutions to mitigate the presence of sybils are the cryptopuzzle [14]. The main idea of the crypto puzzles is to make obtaining nodeIDs exponentially difficult when a node needs a new one. Existing solutions usually rely on a set of servers to provide the nodeIDs, and are therefore not very scalable.

The impact of a Sybil attack over the *Trusted Ring* would mainly be at the level of the *trustsets*. The main idea is to be able to control various nodeIDs in a *trustset*. This is not an easy task: as shown above, it is practically infeasible to obtain numerically close nodeIDs that belong to the same *trustset*. Moreover, controlling a very large number of *trustsets* is practically impossible, as it will require to control a very large number of nodeIDs. This is infeasible due to the number

of transactions required from an attacker in order to establish a good reputation for all of its nodeIDs, and then insert them into the *Trusted Ring*.

7.1.2. Collusion of nodes

A collusion of nodes consists in an agreement made among various nodes to make a combined attack, or to artificially increase or decrease the reputation of a set of nodes. Increasing or decreasing the reputation of a node can be done using rumours or false recommendations. Some existing reputations systems like WTR can efficiently mitigate the effect of this kind of attack. Moreover, colluded nodes should be able to sustain the attack permanently in order to maintain a good reputation and remain in the *Trusted Ring*. This will have significant cost for the attackers, and being able to manage a great number of attackers is very unlikely.

7.2. Portability

Basically, any type of DHT can support the CORPS membership algorithm, as long as it provides some type of key lookup service. For example Chord [4] has a structure equivalent to the Pastry leafset called the successors list. Each node in Chord also maintains an equivalent of the routing table called the finger table.

An underlying reputation system with the same properties explained in Section 3.3 is also required. WTR [8] and PeerTrust [7] are two examples of this kind of reputation systems.

7.3. Target Applications

The aim of our *Trusted Ring* is to provide an efficient and easy way to build large scale trusted services. We have shown the benefits of our *Trusted Ring* for a trusted routing service in a DHT, and this approach can be used for a wide range of P2P applications. Here are some examples:

- Past [15] is a *storage management and caching system* that offers persistence over a P2P network. Each data in Past is stored in the DHT node whose nodeID is closest to the data identifier. Past can use the *Trusted Ring* by saving data *only on trusted nodes* in order to decrease the failure rate in the lookup process. In each lookup a node can contact a *trusted node* and ask to route to the trusted node that is closest to a key k.
- Applications that handle complex queries, like *Prefix Hash Tree* (PHT) [16] can exploit the CORPS. PHT is an indexing data structure which builds a logical binary tree of the data set over the DHT. This structure is distributed among the peers in the network. In order to look for data over the tree with a linear search, a node must send several requests to nodes within the tree, from the root node to the node where the data is saved. Each request routes a message in the DHT, and may fail due to malicious nodes that drop the messages. In addition, malicious nodes into the PHT structure can disturb the search process by cheating about its location on the binary tree. A PHT comprising only CORPS nodes can counter malicious nodes and improve the search process.
- Scribe [17] is a *publish/subscribe system* where nodes create topics that are saved in the DHT in the node with the closest nodeID to the topic identifier, called the rendez-vous node. Subscribers create a multicast tree by joining the DHT routing path to the rendez-vous node. The dissemination then covers the tree from the rendez-vous node. Scribe offers best-effort dissemination of events [17]. Scribe can use the CORPS nodes to save the topics and to manage subscriptions. When a node subscribes to a topic, it contacts a trusted node to route to the rendez-vous node. The path will only comprise reputable peers. In this way, Scribe can improve the reliability of the message dissemination and avoid malicious nodes that drop messages.

By changing the trust model we can build other trusted services, besides the pseudo-trusted routing presented here. For instance, if the reputation system decides whether a node is honest in an online marketplace, then the *Trusted Ring* can be composed of honest sellers, instead of honest forwarders of messages. We envision different ring structures for different trusted services.

7.4. Comparison with reputation systems

The proposed solution goes further than a reputation system. A reputation system only allows to check the reputation of a given node: it provides an opportunity for an application to decide whether or not to carry out a transaction with this node.

CORPS goes much further as it allows an application (i.e. any node) to quickly and directly find trusted nodes to carry out a transaction, with a very low probability for the transaction to fail. In fact, the probability for all the trusted nodes in a set returned by CORPS to fail is so low that, although all of them can still theoretically be malicious, it is practically impossible.

The maintenance of a trusted set on every node of the DHT allows to access several trusted nodes directly, without the cost of looking for such nodes. Hence, every node of the DHT has a local view of several trusted nodes from the nested *Trusted Ring*.

CORPS is a framework to build complex trusted applications over a DHT, as demonstrated with our example of trusted routing. In this sense, compared to existing solutions, CORPS is a step further towards building trusted applications over DHTs. It relies on a reputation system, but it is not one.

8. RELATED WORK

We aim to build trust in a P2P environment, using a membership algorithm to create a group of reputable peers. We use this solution to implement a pseudo-trusted routing algorithm in a DHT. In the following section we detail the related work.

8.1. Building Trust in P2P

The idea of building trust in large scale networks has already been proposed for different purposes [18, 19, 20]. Dent and Price present SCB [19], a system for the deployment of a trusted third party (TTP) service on clients. Their work assumes the existence of a highly resilient secure execution environment, with a hardware protection mechanism. This approach presents two main advantages: the placement of the TTP on local nodes, and reduced communication overheads. Dinh et al. use Trusted Platform Module (TPM) devices to provide trust in marketplaces [18], so that any attempt of misbehaviour will be discovered by checking monotonic counters.

The common factor among the cited solutions is that trust is built over trusted computing hardware [21], with physical and logical protection features. However, the existence of a secure hardware environment is a very strong assumption, especially in open large scale networks where nodes are heterogeneous. Our approach may not achieve as good a level of trust, but we believe the level obtained with a reputation system can be sufficient for best-effort systems, and that avoiding the use of specific hardware on every node is more realistic at present.

Traditional solutions to build trust use a centralized set of trusted servers, but these solutions do not scale up to a large number of nodes. Decentralized ways to identify malicious nodes and build trust are approaches based on accountability [2] and reputation systems [1]. These are very interesting solutions and our work can integrate any of these two approaches. Despite the fact that reputation systems can not decide with complete certainty if a node is malicious, we chose reputation systems since they are protocol independent and allow to categorize nodes easily.

CORPS focuses on finding trustworthy nodes, mitigating the impact of malicious nodes by giving a service with the most reputable ones. Other approaches focuses on finding the malicious nodes which can be ignored, isolated, or excluded from the network. Examples of this last approach are watchdogs [22] and anomaly detection [23]. We believe that approaches which aim at identifying malicious nodes do not apply convincingly for our model, since obtaining identities is cheap in most large scale P2P networks [24]: blacklisted nodes can easily rejoin the network.

Similarly to our approach, PowerTrust [10] selects the most reputable nodes to build trust in a P2P network. PowerTrust uses Chord [4] to build a distributed ranking mechanism to select the m most reputable nodes dynamically [10]. Their approach applies locality-preserving hashing to sort all nodes with respect to their global reputation value. Every node has reputation managers which store the current value into the underlying DHT. PowerTrust can find all the most reputable nodes by hashing the maximum reputation value. However, their approach has a major limitation: the top-m most reputable nodes are stored on the same node. This produces significant security and bottleneck issues.

In the absence of any infrastructure some approaches assume pre-trusted peers [9], or trusted relationships between peers such as those one can find in social networks [25]. These approaches clearly are better at solving the trust problem in these kinds of networks but pre-trusted peers or trusted relationships are not always available in open P2P networks, where connections do not imply trust. We aim for a general solution based on the assumption that there are always honest peers in a large network and we implement a way to find them and use them to build trustworthy applications.

Finally, from a probability point of view, if there are X% of malicious nodes in the network and if they are uniformly distributed in the identifier space, a random selection of r nodes will comprise X% of malicious nodes. Using redundancy, at least some of the r nodes will correctly perform the service. However, this implies an overhead that grows linearly with the number of malicious nodes. We describe some examples of this approach below.

8.2. Secure Routing

Securing the routing algorithm in a DHT has been addressed through redundancy. Redundancy sends multiple copies of the same message in order to improve the probability of success. However, it produces high message overhead. Diversity routing [26], multiple redundant router algorithm [27] and multiple path routing [28] are some examples of this type of approach. Diversity routing sends copies of the message along diverse routes to the destination nodes, using different neighbours of the source node [26]. Wallash approach [27] also send multiple copies to all neighbours of the source nodes. To avoid attacks based on locality, it uses a constrained routing table, allowing only the logically closest nodes to be inserted in the tables. The Cyclone multipath routing approach [28] increases the number of disjoint paths among the peers in the overlay. Our solution directly improves the normal routing so that the probability of success is higher.

Building a solution over a reputation system to improve the routing process is not new. Fetodova and Veltri [29] integrate a reputation system with the process of routing and data lookup for a DHT structure. The idea is to ignore the nodes with bad reputation during the routing and look-up process. CORPS does

the opposite: it searches for honest nodes to improve the service. Dewan et al. [30] use reputations based on the history of nodes relaying packets to ensure a packet will be successfully delivered in a mobile ad-hoc network. In this case the next hop is sent to the choice with highest reputation value. However, the highest reputation value may not be good enough. Our approach builds a ring structure so that all honest node are linked together, and can contact each other to deliver a trusted service.

Huang et al. present RouteGuard [31], a scheme that distinguishes between cooperative and uncooperative nodes in the routing process. Using a reputation system, the nodes fill their routing tables with cooperative nodes, thus isolating the uncooperative ones. Unlike our scheme, this approach does not allow to find new cooperative nodes. Moreover, this approach does not consider the dynamism of peer behaviour, which can cause fluctuations of the reputation values.

8.3. Membership Management

Several approaches [17, 32, 33] aim for membership management in P2P networks. Most of them target applications such as multicast and publish/subscribe, where a set of topics can be mapped onto different groups and a set of users subscribes to various topics. The most common solution is to build a tree for each group over the DHT network. This type of solution is not suitable for our problem since it relies heavily on a root node, which produces a unique point of failure/attack. Other solutions, such as meshes of connections among nodes in the group, or a gossip treatment of the membership as in Scamp [34], do not scale with respect to the number of nodes in the group and produce huge traffic overheads.

The construction of the *Trusted Ring* over a DHT overlay takes its inspiration from a Multi-Ring topology [35]. This work uses a multi-ring topology to build an overlay network for every group in a decentralised and scalable way. Nodes in the inner rings keep participating to the outer rings. DR-Chord [36] is another solution that uses a double ring where the Chord [4] ring is divided into two rings in order to improve the lookup operation and reduce the expected path length.

9. CONCLUSION

In this paper, we proposed a new approach towards providing trustworthy applications building over P2P systems: it consists in establishing a Community Of Reputable PeerS (CORPS) among the nodes. We present a probabilistic evaluation both of our solution and an example application built on top of it: the CORPS exhibits a very low probability of failure in the presence of malicious nodes. Simulations demonstrate the convergence of our membership algorithm, as well as its good availability after a stabilization phase.

Building trust in P2P networks is still a major concern, both in the industry and in the research community. A major effort is required in order to achieve efficient control traditional over traditional attacks such as the Sybil attack or the collusion of nodes.

From the application point of view, we are working on the feasibility of a pseudo-trusted certification authority. Its aim would be to certify, with a very high probability of success, whether a given transaction did or did not occur.

ACKNOWLEDGEMENTS

We would like to thanks Ph.D. María Cristina Riff for her help in the evaluation section of this paper. This research was supported by a doctoral scholarship CONICYT/INRIA.

REFERENCES

[1] Resnick, P., Kuwabara, K., Zeckhauser, R., and Friedman, E. (2000) Reputation systems. *Communications of the ACM*, **43**, 45–48.

[2] Haeberlen, A., Kouznetsov, P., and Druschel, P. (2007) PeerReview: practical accountability for distributed systems. *In Proceedings of the 21st ACM Symposium on Operating Systems Principles*, Stevenson, Washington, USA, 14-17 October, pp. 175–188. ACM, New York, NY, USA.

[3] Douceur, J. (2002) The Sybil Attack. *In Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS)*, Cambridge, MA, USA, 7-8 March, pp. 251–260. Springer-Verlag, London, UK.

[4] Stoica, I., Morris, R., Karger, D., Kaashoek, M., and Balakrishnan, H. (2001) Chord: A scalable peer-to-peer lookup service for internet applications. *In Proceedings of ACM SIGCOMM 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, San Diego, CA, USA, 27-31 August, pp. 149–160. ACM Press, New York, NY, USA.

[5] Rowstron, A. and Druschel, P. (2001) Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. *In Proceedings of Middleware 2001, IFIP/ACM International Conference on Distributed Systems Platforms*, Heidelberg, Germany, 12-16 November, pp. 329–350. Springer-Verlag London, UK.

[6] Zhao, B., Kubiatowicz, J., and Joseph, A. (2002) Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. *Computer Communication Review*, **32**, 81.

[7] Xiong, L. and Liu, L. (2004) PeerTrust: supporting reputation-based trust for peer to peer electronic communities. *IEEE Transactions on Knowledge and Data Engineering*, **16**, 843–857.

[8] Bonnaire, X. and Rosas, E. (2009) WTR: A Reputation Metric for Distributed Hash Tables Based on a Risk and Credibility Factor. *Journal of Computer Science and Technology*, **24**, 844–854.

[9] Kamvar, S. D., Schlosser, M. T., and Garcia-Molina, H. (2003) The Eigentrust algorithm for reputation

management in P2P networks. *In Proceedings of the 12th International Conference on World Wide Web*, Budapest, Hungary, 20-24 May, pp. 640–651. ACM Press, New York, NY, USA.

[10] Zhou, R. and Hwang, F.-K. (2007) PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing. *IEEE Transactions on Parallel and Distributed Systems*, **18**, 460–473.

[11] Srivatsa, M., Xiong, L., and Liu, L. (2005) TrustGuard: countering vulnerabilities in reputation management for decentralized overlay networks. *In Proceedings of the 14th International Conference on World Wide Web*, Chiba, Japan, 10-14 May, pp. 422–431. ACM, New York, NY, USA.

[12] Demers, A., Keshav, S., and Shenker, S. (1989) Analysis and Simulation of a Fair Queueing Algorithm. *In Proceedings of the ACM Symposium on Communications Architectures and Protocols*, Austin, TX, USA, 19-22 September, pp. 1–12. ACM, New York, NY, USA.

[13] Xiang, X. and Jin, T. (2009) Efficient Secure Message Routing for Structured Peer-to-Peer Systems. *In Proceedings of the 2009 International Conference on Networks Security, Wireless Communications and Trusted Computing*, Wuhan, China, 25-26 April, pp. 354–357. IEEE Computer Society, Washington, DC, USA.

[14] Borisov, N. (2006) Computational Puzzles as Sybil Defenses. *In Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing*, Cambridge, UK, 2-4 October, pp. 171–176. IEEE Computer Society, Washington, DC, USA.

[15] Rowstron, A. and Druschel, P. (2001) Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. *In Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles*, Banff, Alberta, Canada, 21-24 October, pp. 188–201. ACM, New York, NY, USA.

[16] Ramabhadran, S., Ratnasamy, S., Hellerstein, J. M., and Shenker, S. (2004) Brief announcement: Prefix Hash Tree. *PODC: In Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing*, ST. John's Newfoundland, Canada, 25-28 July 368. ACM, New York, NY, USA.

[17] Castro, M., Druschel, P., Kermarrec, A., and Rowstron, A. (2002) Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in communications (JSAC)*, **20**, 1489–1499.

[18] Dinh, T. T. A., Chothia, T., and Ryan, M. (2009) A Trusted Infrastructure for P2P-based Marketplaces. *In Proceedings of the Ninth Internation Conference on Peer-to-Peer Computing*, Seattle, Washington, USA, 9-11 September, pp. 151–154. IEEE, Washington, DC, USA.

[19] Dent, A. W. and Price, G. (2005) Certificate management using distributed trusted third parties. In Mitchell, C. J. (ed.), *Trusted Computing*, pp. 251–270. IEE Press, London.

[20] Levin, D., Douceur, J. R., Lorch, J. R., and Moscibroda, T. (2009) TrInc: small trusted hardware for large distributed systems. *NSDI'09: In Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, Boston, Massachusetts, 22-24 April, pp. 1–14. USENIX Association, Berkeley, CA, USA.

[21] Trusted Computing Group. Trusted platform module specifications. online at https://www.trustedcomputinggroup.org/.

[22] Marti, S., Giuli, T. J., Lai, K., and Baker, M. (2000) Mitigating routing misbehavior in mobile ad hoc networks. *In Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, Boston, Massachusetts, USA, 6-11 August, pp. 255–265. ACM, New York, NY, USA.

[23] Wang, W., Man, H., and He, F. (2009) Collaborative anomaly detection for structured p2p networks. *In Proceedings of the Global Communications Conference*, Honolulu, Hawaii, USA, 30 November - 4 December, pp. 1–6. IEEE, Washington, DC, USA.

[24] Friedman, E. J. and Resnick, P. (2000) The Social Cost of Cheap Pseudonyms. *Journal of Economics and Management Strategy*, **10**, 173–199.

[25] Cutillo, L. A., Molva, R., and Strufe, T. (2009) Privacy preserving social networking through decentralization. *In Proceedings of the 2009 Sixth International Conference on Wireless On-Demand Network Systems and Services*, Snowbird, UT, USA, 2-4 February, pp. 145–152. IEEE, Washington, DC, USA.

[26] Castro, M., Druschel, P., Ganesh, A., Rowstron, A., and Wallach, D. S. (2002) Secure routing for structured peer-to-peer overlay networks. *OSDI '02: In Proceedings of the 5th Symposium on Operating Systems Design and Implementation*, Boston, Massachusetts, 9-11 December, pp. 299–314. USENIX Association, Berkeley, CA, USA.

[27] Wallach, D. (2003) A Survey of Peer-to-Peer Security Issues. *In Proceedings of the 2002 Mext-NSF-JSPS International Conference on Software Security: Theories and Systems*, Tokyo, Japan, 4-6 November, pp. 42–57. Springer-Verlag Berlin, Heidelberg.

[28] Artigas, M. S., Lopez, P. G., and Skarmeta, A. F. G. (2005) A Novel Methodology for Constructing Secure Multipath Overlays. *IEEE Internet Computing*, **9**, 50–57.

[29] Fedotova, N. and Veltri, L. (2009) Reputation management algorithms for DHT-base peer-to-peer environment. *Computer Communications*, **32**, 1400–1409.

[30] Dewan, P., Dasgupta, P., and Bhattacharya, A. (2004) On Using Reputations in Ad hoc Networks to Counter Malicious Nodes. *ICPADS '04: In Proceedings of the Tenth International Conference on Parallel and Distributed Systems*, Newport Beach, CA, USA, 7-9 July, pp. 665–. IEEE Computer Society, Washington, DC, USA.

[31] Guowei Huang, J. C. and Wei, L. (2010) RouteGuard: A Trust-Based Scheme for Guarding Routing in Structured Peer-to-Peer Overlays. *In Proceedings of the International Conference on Communications and Mobile Computing*, Washington, DC, USA, 12-14 April, pp. 330–334. IEEE Computer Society.

[32] Liang, J. and Nahrstedt, K. (2006) RandPeer: Membership Management for QoS Sensitive Peer-to-Peer Applications. *INFOCOM 2006: In Proceedings of*

the 25th IEEE International Conference on Computer Communications, Barcelona, Catalunya, Spain, 23-29 April, pp. 1–10. IEEE Computer Society, Washington, DC, USA.

[33] Ostrowski, K. and Birman, K. P. (2006) Scalable group communication system for scalable trust. *STC '06: In Proceedings of the First ACM Workshop on Scalable Trusted Computing*, Alexandria, Virginia, USA, 3 November, pp. 3–6. ACM, New York, NY, USA.

[34] Ganesh, A. J., Kermarrec, A.-M., and Massoulie, L. (2001) SCAMP: Peer-to-Peer Lightweight Membership Service for Large-Scale Group Communication. *In Proceedings of the Third Internation Workshop on Networked Group Communication*, London, UK, 7-9 November, pp. 44–55. Springer-Verlag, London, UK.

[35] Junginger, M. and Lee, Y. (2002) The Multi-Ring Topology - High-Performance Group Communication in Peer-to-Peer Networks. *P2P 2002: In Proceeding of the 2nd International Conference on Peer-to-Peer Computing*, Linköping, Sweden, 5-7 September, pp. 49–56. IEEE Computer Society, Washington, DC, USA.

[36] Shao-Shan, Y., Jiong, Y., Kamil, K., and Qi-Gang, S. (2007) DR-Chord-FAn Efficient Double-Ring Chord Protocol. *In Proceedings of the Sixth International Conference on Grid and Cooperative Computing*, Washington, DC, USA, 16-18 August, pp. 197–202. IEEE Computer Society.