Pp. 299-314 of the *Proceedings*

Secure routing for struct peer overlay net

To appear in the Fifth Symposium on Operati Implementation (OSDI 20

Miguel Castro[1], Peter Druschel[2], Ayalvadi Ganes Dan S. Wallach[2]

[1]*Microsoft Research Ltd., 7 J J Thomson Avenue,*
{mcastro,ajg,antr}@microso

[2]*Rice University, 6100 Main Street, MS 132, Hous*
{druschel,dwallach}@cs.ri

Abstract:

Structured peer-to-peer overlay networks provide a su large-scale, decentralized applications, including distr communication, and content distribution. These overla can route messages correctly even when a large fractio network partitions. But current overlays are not secure malicious nodes can prevent correct message delivery problem is particularly serious in open peer-to-peer sy autonomous parties without pre-existing trust relation resources. This paper studies attacks aimed at prevent

in structured peer-to-peer overlays and presents defen
describe and evaluate techniques that allow nodes to j
routing state, and to forward messages securely in the

1 Introduction

Structured peer-to-peer (p2p) overlays like CAN [16],
Tapestry [21] provide a self-organizing substrate for la
applications. These systems provide a powerful platfo
variety of decentralized services, including network st
and application-level multicast. Structured overlays al
object in a probabilistically bounded, small number of
requiring per-node routing tables with only a small nu
the systems are scalable, fault-tolerant and provide eff

However, to fully realize the potential of the p2p para
must be able to support an open environment where m
with conflicting interests are allowed to join. Even in
large scale, it may be unrealistic to assume that none o
been compromised by attackers. Thus, structured over
variety of security attacks, including the case where a
nodes act maliciously. Such nodes may mis-route, cor
routing information. Additionally, they may attempt to
nodes and corrupt or delete objects they are supposed
system.

In this paper, we consider security issues in structured
describe attacks that can be mounted against such ove
they support, and present the design of secure techniq
attacks. In particular, we identify *secure routing* as a k
combined with existing, application-specific security t
secure, decentralized applications upon structured ove
(1) a secure assignment of node identifiers, (2) secure
and (3) secure message forwarding. We present techni
problems, and show how using these techniques, secu
efficiently despite up to 25% of malicious participatin
that the overhead of secure routing is acceptable and p
malicious nodes.

The rest of this paper is organized as follows. Section
structured p2p overlays, specifies models and assump
routing. Sections 3, 4 and 5 present attacks on and sol
identifiers to nodes, routing table maintenance and me
respectively. Section 6 explains how the overhead of s
minimized by using self-certifying data. Finally, Secti
and Section 8 provides conclusions.

2 Background, models an

In this section, we present some background on struct
like CAN, Chord, Tapestry and Pastry. Space limitatio
detailed overview of each protocol. Instead, we descri
structured p2p overlay networks that we use to keep th
any particular protocol. For concreteness, we also giv
point out relevant differences with the other protocols
and assumptions used later in the paper about how fau
Finally, we define secure routing and outline our solut

Throughout this paper, most of the analyses and techn
of our abstract model, and should apply to other struct
otherwise noted. However, the security and performan
fully evaluated only in the context of Pastry; a full eva
other protocols is future work.

2.1 Routing overlay model

We define an abstract model of a structured p2p routin
capture the key concepts common to overlays like CA
Pastry.

In our model, participating nodes are assigned uniforn
from a large *id space* (e.g., the set of 128-bit unsigned
specific objects are assigned unique identifiers, called
id space. Each key is mapped by the overlay to a uniq
root. The protocol routes messages with a given key to

To route messages efficiently, each node maintains a *r*
other nodes and their associated IP addresses. Moreov
neighbor set, consisting of some number of nodes wit
node in the id space. Since nodeId assignment is rand
represents a random sample of all participating nodes.

For fault tolerance, application objects are stored at m
overlay. A *replica function* maps an object's key to a s
the set of *replica roots* associated with the replica key
of participating nodes in the overlay. Each replica roo

Next, we discuss existing structured p2p overlay proto
our abstract model.

\psfig{file=table.eps,width=1.0\textwidth}

Figure 1: Routing table of a Pastry node with nodel

base 16, represents an arbitrar

\psfig{file=ring.eps,width=1.0\textwidth}

Figure 2: Routing a message from node

depict live nodes in Pastry's circular

2.2 Pastry

Pastry nodeIds are assigned randomly with uniform d
128-bit id space. Given a 128-bit key, Pastry routes ar
the live node whose nodeId is numerically closest to t
keeps track of its neighbor set and notifies application

Node state: For the purpose of routing, nodeIds and k
sequence of digits in base (is a configuration par

A node's routing table is organized into rows a

entries in row of the routing table contain the IP add
nodeIds share the first digits with the present node's

digit of the node in column of row equals . The

corresponds to the value of the th digit of the loca

empty. A routing table entry is left empty if no node w
prefix is known. Figure 1 depicts an example routing

Each node also maintains a neighbor set (called a ``lea
of nodes with nodeIds that are numerically closest to

with larger and smaller nodeIds than the curre

constant for all nodes in the overlay, with a typical val
, where is the number of expected nodes

ensures reliable message delivery and is used to store
objects.

Message routing: At each routing step, a node seeks
node in the routing table whose nodeId shares with the
one digit (or bits) longer than the prefix that the key

node's id. If no such node can be found, the message i
nodeId shares a prefix with the key as long as the curr
closer to the key than the present node's id. If no appro
the routing table or neighbor set, then the current node
the message's final destination.

Figure 2 shows the path of an example message. Anal
number of routing hops is slightly below , with

around the mean. Moreover, simulation shows that the
crash failures.

To achieve self-organization, Pastry nodes must dynar
state, i.e., the routing table and neighbor set, in the pre
node failures. A newly arriving node with the new noc

by asking any existing Pastry node to route a specia
The message is routed to the existing node with noc
then obtains the neighbor set from and constructs
rows from the routing tables of the nodes it encounter
to . Finally, announces its presence to the initia
which in turn update their own neighbor sets and rout
overlay can adapt to abrupt node failure by exchangin
( ) among a small number of nodes.

2.3 CAN, Chord, Tapestry

Next, we briefly describe CAN, Chord and Tapestry, v
differences relative to Pastry.

Tapestry is very similar to Pastry but differs in its appr
nodes and in how it manages replication. In Tapestry,
namespace are not aware of each other. When a node's
an entry for a node that matches a key's th digit, the
node with the next higher value in the th digit, modu
table. This procedure, called *surrogate routing*, maps
the node routing tables are consistent. Tapestry does n
neighbor set, although one can think of the lowest pop
routing table as a neighbor set. For fault tolerance, Tap
produces a set of random keys, yielding a set of replic
the id space. The expected number of routing hops in

Chord uses a 160-bit circular id space. Unlike Pastry,
only in clockwise direction in the circular id space. In
routing table in Pastry, Chord nodes maintain a routing
pointers to other live nodes (called a ``finger table''). T
table of node refers to the live node with the smalles
. The first entry points to 's successor, and su
nodes at repeatedly doubling distances from . Each n
pointers to its predecessor and to its successors in th
successor list represents the neighbor set in our model
function maps an object's key to the nodeIds in the ne
i.e., replicas are stored in the neighbor set of the key's
expected number of routing hops in Chord is .

CAN routes messages in a -dimensional space, wher
routing table with entries and any node can be re
hops on average. The entries in a node's routing table

dimensional space. CAN's neighbor table duals as bot
neighbor set in our model. Like Tapestry, CAN's repli
keys for storing replicas at diverse locations. Unlike P
CAN's routing table does not grow with the network s
hops grows faster than in this case.

Tapestry and Pastry construct their overlay in a Intern
reduce routing delays and network utilization. In these
entries can be chosen arbitrarily from an entire segme
increasing the expected number of routing hops. The p
initializing the routing table to refer to nodes that are
topology and have the appropriate nodeId prefix. This
routing [17]. However, it also makes these systems vu
shown in Section 4.

The choice of entries in CAN's and Chord's routing ta
The CAN routing table entries refer to specific neighb
dimension, while the Chord finger table entries refer t
space. This makes proximity routing harder but it prot
exploit attacking nodes' proximity to their victims.

2.4 System model

The system runs on a set of nodes that form an over
described in the previous section. We assume a bound
of nodes that may be faulty. Faults are modeled using
Byzantine failure model, i.e., faulty nodes can behave
all necessarily be operating as a single conspiracy. Th
partitioned into independent coalitions, which are disj
( ). When , all faulty nodes may
cause the most damage to the system. We model the c
into multiple independent coalitions by setting .
work together to corrupt the overlay but are unaware
We studied the behavior of the system with ranging
different failure scenarios.

We assume that every node in the p2p overlay has a st
be contacted. In this paper, we ignore nodes with dyna
addresses, and nodes behind network address translati
p2p overlays can be extended to address these concern
traditional network hosts.

The nodes communicate over normal Internet connect
two types of communication: *network-level*, where no
without routing through the overlay, and *overlay-level*
through the overlay using one of the protocols discuss
use cryptographic techniques to prevent adversaries fr

network-level communication between correct nodes.
control over network-level communication to and fror
can compromise overlay-level communication that is
Adversaries may delay messages between correct nod
message sent by a correct node to a correct destinatior
no faulty nodes is delivered within time with proba

2.5 Secure routing

Next, we define a secure routing primitive that can be
techniques to construct secure applications on structur
sections show how to implement the secure routing pr
network models that we described in the previous sect

The routing primitives implemented by current structu
best-effort service to deliver a message to a replica ro
With malicious overlay nodes, the message may be dr
be delivered to a malicious node instead of a legitimat
these primitives cannot be used to construct secure ap
inserting an object, an application cannot ensure that t
legitimate, diverse replica roots as opposed to faulty n
roots. Even if applications use cryptographic methods
malicious nodes may still corrupt, delete, deny access
all replicas of an object.

To address this problem, we define a secure routing pr
primitive ensures that when a non-faulty node sends a
message reaches all non-faulty members in the set of
high probability. is defined as the set of nodes that
the set of replica keys associated with , a live root no
replica key. In Pastry, for instance, is simply a set o
numerically closest to the key. Secure routing ensures
eventually delivered, despite nodes that may corrupt,
and (2) the message is delivered to all legitimate repli
nodes that may attempt to impersonate a replica root.

Secure routing can be combined with existing security
maintain state in a structured p2p overlay. For instanc
stored on the replica roots, or a Byzantine-fault-tolera
BFT [4] can be used to maintain the replicated state. S
the replicas are initially placed on legitimate replica ro
message reaches a replica if one exists. Similarly, secu
build other secure services, such as maintaining file m
distributed storage utility. The details of such services
paper.

Implementing the secure routing primitive requires th
securely assigning nodeIds to nodes, securely maintai
securely forwarding messages. Secure nodeId assignn
cannot choose the value of nodeIds assigned to the no
Without it, the attacker could arrange to control all rep
mediate all traffic to and from a victim node.

Secure routing table maintenance ensures that the frac
appear in the routing tables of correct nodes does not
fraction of faulty nodes in the entire overlay. Without
correct message delivery, given only a relatively smal
Finally, secure message forwarding ensures that at lea
to a key reaches each correct replica root for the key v
Sections 3, 4 and 5 describe solutions to each of these

3 Secure nodeId assignme

The performance and security of structured p2p overla
fundamental assumption that there is a uniform rando
cannot be controlled by an attacker. This section discu
the attacker violates this assumption, and how this pro

3.1 Attacks

Attackers who can choose nodeIds can compromise th
overlay, without needing to control a particularly large
example, an attacker may partition a Pastry or Chord
complete and disjoint neighbor sets. Such attackers m
victim nodes by carefully choosing nodeIds. For exam
every entry in a victim's routing table and neighbor se
a Chord overlay. At that point, the victim's access to th
completely mediated by the attacker.

Attackers who can choose nodeIds can also control ac
attacker can choose the closest nodeIds to all replica k
object, thus controlling all replica roots. As a result, th
corrupt, or deny access to the object. Even when attac
they may still be able to mount all the attacks above (
large number of legitimate nodeIds easily. This is kno

Previous approaches to nodeId assignment have eithe
randomly by the new node [5] or compute nodeIds by
node [20]. Neither approach is secure because an attac
to choose nodeIds that are not necessarily random, or
hashes to a desired interval in the nodeId space. Partic
even modest attackers will have more potential IP add

there are likely to be nodes in a given p2p network.

3.2 Solution: certified nodeIds

One solution to securing the assignment of nodeIds is
central, trusted authority. We use a set of trusted certif
assign nodeIds to principals and to sign *nodeId certifi*
nodeId to the public key that speaks for its principal a
ensure that nodeIds are chosen randomly from the id s
forging nodeIds. Furthermore, these certificates give t
infrastructure, suitable for establishing encrypted and
between nodes.

Like conventional CAs, ours can be offline to reduce t
signing keys. They are not involved in the regular ope
with valid nodeId certificates can join the overlay, rou
repeatedly without involvement of the CAs. As with a
CA's public keys must be well known, and can be inst
software itself, as is done with current Web browsers.

The inclusion of an IP address in the certificate deserv
p2p protocols, such as Tapestry and Pastry, measure th
nodes and choose routing table entries that minimize d
multiple legitimate nodeId certificates could freely sw
controls, it might be able to increase the fraction of att
node's routing table. By binding the nodeId to an IP ad
an attacker to move nodeIds across nodes. We allow m
per IP address because the IP addresses of nodes may
otherwise, attackers could deny service by hijacking v

A downside of binding nodeIds to IP addresses is that,
changes, either as a result of dynamic address assignm
organizational network changes, then the node's old ce
invalid. In p2p systems where IP addresses are allowe
nodeId swapping attacks may be unavoidable.

Certified nodeIds work well when nodes have fixed n
Chord, Pastry, and Tapestry. However, it might be har
assignment in CAN. CAN nodeIds represent a zone in
is split in half when a new node joins [16]. Both the n
the nodeId of the joining node change during this proc

3.2.1 Sybil attacks

While nodeId assignment by a CA ensures that nodeId
also important to prevent an attacker from easily obtai
nodeId certificates. One solution is to require an attack

certificates, via credit card or any other suitable mech
cost of an attack grows naturally with the size of the n
nodeId certificates cost $20, controlling 10% of an ov
$2,000 and the cost rises to $2,000,000 with 1,000,00
attacks is even higher; it costs an expected $20,000 to
particular point in the id space in an overlay with 1,00
attacks economically expensive, these fees can also fu

Another solution is to bind nodeIds to real-world iden
money. In practice, different forms of CAs are suitabl
identity-based CA is the preferred solution in ``virtual
organization that already maintains employment or m
identity checks. In an open Internet deployment, a mo
suitable because it avoids the complexities of authenti

None of the known solutions to nodeId assignment ar
network is very small. For small overlay networks, we
members of the network are trusted not to cheat. Only
critical mass, where it becomes sufficiently hard for a
resources to control a significant fraction of the overla
allowed to join.

3.3 Rejected: distributed nodeId g

The CAs represent points of failure, vulnerable to bot
Also, for some p2p networks, it may be cumbersome
money or prove their real-world identities. Therefore,
construct secure p2p overlays without requiring centra
identity checks. Unfortunately, fully decentralized no
have fundamental security limitations [10]. None of th
can ultimately prevent a determined attacker from acq
nodeIds.

However, several techniques may be able to, at a mini
which an attacker can acquire nodeIds. One possible s
prospective nodes to solve crypto puzzles [15] to gain
approach that has been taken to address a number of d
attacks [13,8]. Unfortunately, the cost of solving a cry
to the slowest legitimate node, yet the puzzle must be
slow down an attacker with access to many fast mach
effectiveness of any such technique.

For completeness, we briefly describe here one relativ
generate certified nodeIds in a completely distributed
The idea is to require new nodes to generate a key pai
SHA-1 hash of the public key has the first bits zero
operations required to generate such a key pair is . T
cryptography allow the nodes to use a secure hash of t
This hash should be computed using SHA-1 with a di

MD5 to avoid reducing the number of random bits in
they performed the required amount of work to use a
information that would allow others to reuse their wo
to achieve the desired level of security.

It is also possible to bind IP addresses with nodeIds to
that exploit network locality. The idea is to require no
order to be able to use a given nodeId with an IP addr
nodes to find a string such that SHA-1(SHA-1(*ipad*
to zero. Nodes would be required to present such an
to be accepted by others.

Finally, it is possible to periodically invalidate nodeId
entity broadcast to the overlay a message supplying a
for the hash computations. This makes it harder for an
nodeIds over time and to reuse nodeIds computed for
overlay. However, it requires legitimate nodes to perio
and communication to maintain their membership in t

4 Secure routing table ma

We now turn our attention to the problem of secure ro
routing table maintenance mechanisms are used to cre
neighbor sets for joining nodes, and to maintain them
routing table and neighbor set should have an average
entries that point to nodes controlled by the attacker (
attackers can increase the fraction of bad entries by su
which reduces the probability of routing successfully

Preventing attackers from choosing nodeIds is necess
it is not sufficient as illustrated by the two attacks disc
solutions to this problem.

4.1 Attacks

The first attack is aimed at routing algorithms that use
information to improve routing efficiency: attackers m
the fraction of bad routing table entries. For example,
assumed allows an attacker to control communication
it controls. When a correct node sends a probe to es
with a certain nodeId, an attacker can intercept the pro
closest to reply to it. If the attacker controls enough
Internet, it can make nodes that it controls appear clos

the probability that they are used for routing. The atta

maximal fraction of colluding nodes) is small even if

This attack can be ruled out by a more restrictive com nodeId certificates bind IP addresses to nodeIds (see S faulty nodes can only observe messages that are sent t this attack is prevented. But note that a rogue ISP or c offices around the world could easily perform this atta routers appropriately. The attack is also possible if the indirection that the attacker can control, e.g., mobile I

The second attack does not manipulate proximity info the fact that it is hard to determine whether routing up overlay protocols like Tapestry and Pastry. Nodes rece they join the overlay and when other nodes join, and t from other nodes in their routing table periodically to delays. In these systems, attackers can more easily su always point to faulty nodes. This simple attack cause table entries to increase toward one as the bad routing More precisely, routing updates from correct nodes po probability at least whereas this probability can be

updates from faulty nodes. Correct nodes receive upd with probability at most and from faulty nodes

Therefore, the probability that a routing table entry is least , which is greater than . This

subsequent update, causing the fraction of faulty entri

Systems without strong constraints on the set of node table slot are more vulnerable to this attack. Pastry an constraints at the top levels of routing tables. This fle determine if routing updates are unbiased but it allow exploit network proximity to improve routing perform impose strong constraints on nodeIds in routing table closest nodeIds to some point in the id space. This ma proximity to improve performance but it is good for s choose the nodeIds they control, the probability that a closest to a point in the id space is .

4.2 Solution: constrained routing

To enable secure routing table maintenance, it is impo constraints on the set of nodeIds that can fill each slot example, the entry in each slot can be constrained to b point in the id space as in Chord. This constraint can b independent of network proximity information, which attackers.

The solution that we propose uses two routing tables:
proximity information for efficient routing (as in Past
constrains routing table entries (as in Chord). In norm
table is used to forward messages to achieve good per
used only when the efficient routing technique fails. V
to detect when routing fails.

We modified Pastry to use this solution. We use the n
routing table and an additional *constrained* Pastry rou
aware routing table of a node with identifier , the slo
contain any nodeId that shares the first digits with
st digit. In the constrained routing table, the entry
point to the closest nodeId to a point in the domain.
shares the first digits with , it has the value in the
same remaining digits as .

Pastry's message forwarding works with the constrain
modifications. The same would be true with Tapestry.
initialize and maintain the routing table were modifie

All overlay routing algorithms rely on a *bootstrap no*
state of a newly joining node. The bootstrap node is r
message using the nodeId of the joining node as the k
faulty, it can completely corrupt the view of the overl
node. Therefore, it is necessary to use a set of diverse
to ensure that with very high probability, at least one
nodeId certificates makes the task of choosing such a
cannot forge nodeIds.

A newly joining node, , picks a set of bootstrap node
route using its nodeId as the key. Then, non-faulty bo
forwarding techniques (described in Section 5.2) to o
joining node. Node collects the proposed neighbor
nodes, and picks the ``closest" live nodeIds from each
neighbor set (where the definition of closest is protoc

The locality-aware routing table is initialized as befor
nodes along the route to the nodeId. The difference is
 picks the entry with minimal network delay from th
for each routing table slot.

Each entry in the constrained routing table can be init
forwarding to obtain the live nodeId closest to the des

This is similar to what is done in Chord. The problem

with (recall that controls the number of colum
Tapestry and Pastry). To reduce the overhead, we can
that, by induction, the constrained routing tables of th
point to entries that are close to the desired point . T
routing tables from the nodes in its neighbor set and u
constrained routing table. From the set of candidates t
it picks the nodeId that is closest to the desired point f
of this process, informs the nodes in its neighbor se

We exploit the symmetry in the constrained routing ta
to update their routing tables to reflect 's arrival: c
set of candidates for each entry to determine which ca
routing table entries to point to . It informs those can

To ensure neighbor set stabilization in the absence of
informs the members of its neighbor set whenever it c
retransmits this information until its receipt is acknow

5 Secure message forward

The use of certified nodeIds and secure routing table
constrained routing table (and neighbor set) has an ave
random entries that point to nodes controlled by the at
constrained routing table is not sufficient because the
probability of successful delivery by simply not forwa
the algorithm. The attack is effective even when is s
section describes an efficient solution to this problem.

5.1 Attacks

All structured p2p overlays provide a primitive to sen
absence of faults, the message is delivered to the root
average of routing hops. But routing may fail if any
route between the sender and the root are faulty; fault
message, route the message to the wrong place, or pre
Therefore, the probability of routing successfully betw
a fraction of the nodes is faulty is only: , v

The root node for a key may itself be faulty. As discus
tolerate root faults by replicating the information asso
nodes -- the *replica roots*. Therefore, the probability o

correct replica root is only: . The value of

is in CAN, in Chord, and

We ran simulations of Pastry to validate this model. T probability of success slightly lower than the probabil simulations (because the number of Pastry hops is slig average [3]) but the error was below 2%.

Figure 3 plots the probability of routing to a correct re using the model) for different values of , , and quite fast when or increase. Even with only 10% the probability of successful routing is only 65% wher Pastry overlay.

Figure 3: Probability of routing to a c

In CAN, Pastry, and Tapestry, applications can reduce increasing the value of or . Fewer hops increase th correctly. For example, the probability of successful d 100,000 nodes is 65% in Pastry when and 75% also increases the cost of routing table maintenance; a success requires an impractically large value of . Ch , which results in a low probability of success, e. 42% under the same conditions.

5.2 Solution: detect faults, use div

The results in Figure 3 show that it is important to dev securely. We want a *secure routing primitive* that take key and ensures that with very high probability at leas reaches each correct replica root for the key. The ques efficiently.

Our approach is to route a message efficiently and to a
determine if routing worked. We only use more expen
the failure test returns positive. In more detail, our sec
message efficiently to the root of the destination key u
routing table. Then, it collects the prospective set of re
prospective root node and applies the *failure test* to th
the prospective replica roots are accepted as the correc
message copies are sent over diverse routes toward th
that with high probability each correct replica root is r
describing how to implement the failure test. Then we
and why we rejected an alternate approach called itera

5.2.1 Routing failure test

The failure test takes a key and a set of prospective re
returns negative if the set of roots is likely to be correc
returns positive. Of course, routing can fail without th
of prospective replica roots. The sender detects this by
sends a message. If it does not receive a response befo
failure test returns positive triggering the use of redun

Detecting routing failures is difficult because a coaliti
to be the legitimate replica roots for a given key. Since
determined by the structure of the overlay, a node who
must rely on overlay routing to determine the correct s
message is routed by a faulty node, the adversary can
replica root set that contain only nodes it controls. Fur
that the adversary just happens to legitimately control
roots. This problem is common to all structured p2p o

The routing failure test is based on the observation tha
nodeIds per unit of ``volume'' in the id space is greate
faulty nodeIds. The test works by comparing the dens
set of the sender with the density of nodeIds close to t
destination key. We describe the test in detail only in t
simplify the presentation; the generalization to other o
Overlays that distribute replica keys for a key uniform
use this check by comparing the density at the sender
between each replica key and its root's nodeId.

In Pastry, the set of replica roots for a key is a subset o
root node, called the key's *root neigbor set*. Each corre
average numerical distance, , between consecutive
The neighbor set of contains live nodes: , the
nodeIds less than 's, and the nodes with the close
To test a prospective root neighbor set,

1. all nodeIds in have a valid nodeId certificate
   is the middle one, and the nodeIds satisfy the de

2. the average numerical distance, , between co
   satisfies:

If satisfies both conditions, the test returns negativ
positive. The test can be inaccurate in one of two way
when the prospective root neighbor set is correct, or it
when the prospective set is incorrect. We call the prob
and the probability of false negatives . The paramete
between and . Intuitively, increasing decreases

Assuming that there are live nodes with nodeIds un
id space (which has length), the distances bet
approximately independent exponential random varia
. The same holds for the distances between consecu
that can collude together but the mean is . It
and are independent of . They only depend on the
fraction of colluding nodes because faulty nodes only
nodes that they collude with.

Under these assumptions, we have derived the followi
and (see detailed derivation in the Appendix):

These expressions can be used to compute and nu
the following closed-form upper bounds for and :

where is the number of distance samples used to co
distance samples used to compute , and . Th

The analysis shows that and are independent of
the test's accuracy can be improved by increasing the
used to compute the means. It is easy to increase the n
compute by augmenting the mechanism that is alre
neighbor sets. This mechanism propagates nodeIds th
from a neighbor set to the other members of the set; it
nodeIds further but we omit the details due to lack of
the number of samples used to compute because o
describe below. Therefore, we keep .

Figure 4: Routing failure test: probability
( ) and negatives ( ). The predicted cu
indistinguishable from the simulation mea
upper bounds are not tigh

We ran several simulations to evaluate the effectivene
The simulations ran in a system with 100,000 random
values of and for different values of with
at the sender is , and the number of root neighb
shows predicted values computed numerically, the up
measured in the simulations. The predicted curves ma
almost exactly but the upper bounds are not very tight
obtained when , which is equal to with

5.2.1.1 Attacks:

There are several attacks that could invalidate the ana
failure test. First, the attacker can collect nodeId certi
the overlay, and use them to increase the density of a
Second, the attacker can include both nodeIds of node
correct nodes in a prospective root neighbor set. Both
probability that messages reach all correct replica roo
to counter in overlays that distribute replica keys over
roots have no detailed knowledge about the nodeIds c

These attacks can be avoided by having the sender co
neighbors to determine if they are live and if they hav
omitted from the prospective root neighbor set. To im
prospective root returns to the sender a message with
a list with the secure hashes of the neighbor sets repor
root neighbors, and the set of nodeIds (not in the pros
are used to compute the hashes in this list. The sender
consistent with the identifiers of the prospective root
prospective root neigbor the corresponding neighbor

In the absence of faults, the root neighbors will confir
can perform the density comparison immediately. For
this happens with probability , where
distribution [6] and is the number of root neighbors
prospective root neighbor set, the routing failure test
communication before the density check can be run.
strategy to deal with this case. Currently, we consider
prospective root neighbors don't agree and use redund
worthwhile investing some additional communication
routing.

In addition to these attacks, there is a *nodeId suppress*
unavoidable and significantly decreases the accuracy
suppress nodeIds close to the sender by leaving the ov

Similarly, the attacker can suppress nodeIds in the roo
increases . Furthermore, the attacker can alternate b
honest nodes have no way of detecting in which mode

We ran simulations to compute the minimum error pr
without nodeId suppression attacks for different value
error increases fast with and it is higher than
samples at the sender. The nodeId suppression attack
probability of error for large percentages of comprom
probability of error is higher than 0.001 for ev
sender. Figures 5 and 6 show the results without and
attacks, respectively.

Figure 5: Routing failure test: minimum

without nodeId suppression attacks and v
samples.

Figure 6: Routing failure test: minimum er
nodeId suppression attacks and varying nu

These results indicate that our routing failure test is no
fortunately we can trade off an increase in to achiev
redundant routing to disambiguate false positives. We
the minimum that can be achieved for a target
, and different numbers of samples at the sender.
with nodeId suppression attacks.

Figure 7: Routing failure test: probability
for a false negative rate of 0.001 with no
attacks and varying number of s

The results show that the test is not meaningful for thi
nodeId suppression attacks. However, setting
sender enables the routing failure test to achieve the ta
value of and with , nodeId suppression attack
without nodeId suppression attacks the value of is c
routing is required 12% of the time.

5.2.2 Redundant routing

The redundant routing technique is invoked when the
The idea is simply to route copies of the message over
of the destination key's replica roots. If enough copies
diverse routes to each replica key, all correct replica r
copy of the message with high probability.

The issue is how to ensure that routes are diverse. One
members of the sender's neighbor set to forward the c
replica keys. This technique is sufficient in overlays t
uniformly over the id space (e.g., CAN and Tapestry).
overlays that choose replica roots in the neighbor set o
and Pastry) because the routes all converge on the key
For these overlays, we developed a technique called *n*
copies of the message toward the destination key unti
key's root in its neighbor set. Then it uses the detailed
has about the portion of the id space around the destin
correct replica roots receive a copy of the message.

To simplify the presentation, we only describe in deta
works in Pastry. If a correct node sends a message t
routing failure test is positive, it does the following:

(1) sends messages to the destination key . Each
different member of 's neighbor set; this causes the r
routes. All messages are forwarded using the constrai
include a nonce.

(2) Any correct node that receives one of the message
neighbor set returns its nodeId certificate and the nonc
to .

(3) collects in a set the nodeId certificate
the left, and the closest to on the right. Only
nonces are added to and they are first marked *penc*

(4) After a timeout or after all replies are received,
in to each node marked *pending* in and marks t

(5) Any correct node that receives this list forwards
nodes in its neighbor set that are not in the list, or it se
there are no such nodes. This may cause steps 2 to 4 t

(6) Once has received a confirmation from each of t
executed three times, it computes the set of replica roo

If the timeout is sufficiently large and correct nodes ha
each half of their neighbor set[1], the probability of reac
of is approximately equal to the probability that at l
messages is forwarded over a route with no faults to a
root in its neighbor set. Assuming independent routes,

where *binom* is the binomial distribution [6] with 0 su
the probability of routing successfully in each trial is
the extra hop for messages routed through a neighbor
of success for this technique depends on and is inde

We also ran simulations to determine the probability o
roots with our redundant routing technique. Figure 8 p
and the probability measured in the simulator for 100,
. The analytic model matches the results wel
probabilities. The results show that the probability of
for . Therefore, this technique combined with c

achieve a reliability of approximately 0.999 for

Figure 8: Model and simulation results fo
reaching all correct replica roots using redu
neighbor set anycast.

We studied several versions of redundant routing that
of success but perform differently. For example, the si
ensure that the nodeId certificates in belong to live

signing nonces by periodically signing their clock rea
synchronized clocks, and no signatures are necessary
source addresses. We are still exploring the design spa
possible to improve performance significantly by senc
at a time and using a version of the routing failure test
approach would also work well when reading self-cer

5.2.3 Putting it all together: performance

The performance of Pastry's secure routing primitive
routing failure test, the cost of redundant routing, and
redundant routing can be avoided. This section presen
and probability. For simplicity, we assume that all fau

the number of probes used by redundant routing is equ
), the number of samples at the source for routing

the number of nodes in the overlay is .

The cost of the routing failure test when it returns neg
delay and messages. The total number of bytes

IdSize HashSize IdCertSi

Using PSS-R [1] for signing nodeId certificates with 1
modulus for the node public keys, the nodeId certifica
the extra bandwidth consumed by the routing failure t
with and 2.9 KB with (plus the space use

When the test returns positive, it adds the same numbe
the extra delay is the timeout period.

The cost of redundant routing depends on the value of
when all of the root's neighbor set is added to in th
redundant routing adds extra message delay
messages. The total number of bytes in these message

IdSize IdCertSize SigSize

Using PSS-R for signing nonces, the signed nonce siz
bandwidth consumed in this case is 22 KB with
the space used up by message headers).

Under attack redundant routing adds a delay of at mos
the expected number of extra messages is less than
where is the expected number of
neighbor set that is added to in the first iteration. T
messages is less than 451 with and and
and . The total number of bytes sent under att
value except that the sender sends an additional
lists and the number of messages increases. This is an
and and 1 KB with and (plus
headers).

The probability of avoiding redundant routing is given
is the probability that the overlay routes the message t
the probability that there are no faulty nodes in the ne
is the false positive rate of the routing failure test. We
assumes that routing tables have an average of rand
assumption holds for the locality-aware routing table r
discussed in Section 4 and it holds for the constrained
attacks. We do not have a good model of the effect of

aware routing table but we believe that they are very l
of ( ).

The parameters and , should be set based on the de
can be expressed as the probability that all correct r
the message. The overlay size and the assignment of v
implicitly define a bound on . If this bound is exceed
example, we saw that with
routing is invoked 12% of the time for this value of

One can trade off security for improved performance l
and by decreasing to reduce the cost of the routing f
routing and to increase . For example, consider the f
with and , and (2)
and . Figure 9 plots the probability of a
these two scenarios for different values of . Without
invoked only 0.5% of the time in scenario (1) and 0.4
when the fraction of faulty nodes is small, the routing
performance significantly by avoiding the cost of redu

Figure 9: Probability of avoiding redund
scenarios: (1) with
and (2) with

5.2.4 Rejected: checked iterative routing

An alternative to redundant routing is iterative routing
Morris [19]: the sender starts by looking up the next h
setting a variable to point to this node; then, the sen
and updates to point to the returned value. The proc
is the root of the destination key.

Iterative routing doubles the cost relative to the more
but it may increase the probability of routing successf
sender to pick an alternative next hop when it fails to
This is not a strong defense against an attacker who ca
next hop. However, iterative routing can be augmente
whether the next hop in a route is correct.

Hop tests are effective in systems like Chord or Pastry
table because each routing table entry should contain
specific point in the id space. One can use a mechan
density checking that we used for the routing failure t
the average distance between consecutive nodeIds clo
to the distance between the nodeId in the routing table
. We ran simulations to compute the false positive and
approach with different values of (these rates are in
the minimum error for this hop test ( ) is equal t
and 256 samples to compute the mean at the se

The error is high because there is a single sample at th
our simulations indicate that iterative lookups using P
table with this hop check improve the probability of ro
example, the probability of routing successfully with
, and 256 samples to compute the mean at the se
0.3 to 0.4. But it adds an extra 2.7 hops to each route
positives.

We tried to increase the number of samples by having
routing table row during each iterative routing step wi
the required slot. Unfortunately, this performs worse t
because the attacker can combine good and bad routin
high average density.

We also tried to combine checked iterative routing wi
technique that we described before. We used checked
neighbor set anycast messages in the hope that the imp
the iterative routes would result in an improvement ov
recursive routes. But there was no visible improvemen
iterative routes are less independent than the recursive
routing failure test combined with redundant routing i
for implementing secure routing.

6 Self-certifying data

The secure routing primitive adds significant overhea

In this section, we describe how the use of secure rout
using *self-certifying data*.

The reliance on secure routing can be reduced by stor
overlay, i.e., data whose integrity can be verified by th
to use efficient routing to request a copy of an object.
the object, it can check its integrity and resort to secur
integrity check fails or there was no response within a

Self-certifying data does not help when inserting new
when verifying that an object is not stored in the overl
secure routing primitive to ensure that all correct repli
Similarly, node joining requires secure routing. Never
can eliminate the overhead of secure routing in comm

Self-certifying data has been used in several systems.
cryptographic hash of a file's contents as the key durin
file, and PAST [18] inserts signed files into the overla

The technique can be extended to support mutable obj
guarantees. One can use a system like PAST to store s
descriptors that identify the set of hosts responsible fo
Group descriptors can be used as follows. At object cr
object uses secure routing to insert a group descriptor
that identifies the object. The descriptor contains the p
of the object's replica holders and it is signed by the o

The replica group can run a Byzantine-fault-tolerant r
BFT [4] and the initial group membership is the set of
the key. In this setting, read and write operations can b
client uses efficient routing to retrieve a group descrip
checks the descriptor's signature; if correct, it uses the
to authenticate the replica holders and to invoke a repl
fails to retrieve a valid descriptor or if it fails to authen
uses the secure routing primitive to obtain a correct gr
that the object does not exist. This procedure provides
guarantees (linearizability [11]) for reads and writes w
failure test in the common case.

Changing the membership of the group that is respons
is not trivial; it requires securely inserting a new grou
ensuring that clients can reliably detect stale group de
technique allows groups to change membership while
guarantees. Each group of hosts that stores replicas of
private/public key pair associated with the group. Wh
changes, each host in the new membership generates a
the hosts in the old membership use their old keys to s
containing the new keys, and then delete the old keys.

If this operation is performed by a quorum of replica h
the number of faulty group members is exceeded [4],

will not be able to collude to pretend they are the curr
form the quorum necessary to authenticate themselves

Group descriptors can be authenticated by following a
with an owner signature and has signatures of a quoru
subsequent membership change. The chain can be sho
from the owner or, alternatively, replicas can use proa
avoid the need for chaining signatures.

7 Related work

Sit and Morris [19] present a framework for performi
networks. Their adversarial model allows for nodes to
arbitrary contents, but assumes that nodes cannot inter
then present a taxonomy of possible attacks. At the ro
lookup, routing table maintenance, and network partit
security risks. They also discuss issues in higher-leve
storage, where nodes may not necessarily maintain the
storage replication. Finally, they discuss various class
attacks, including rapidly joining and leaving the netw
nodes to send bulk volumes of data to overload a victi
distributed denial of service attacks).

Dingledine *et al.* [9] and Douceur [10] discuss addres
large number of potentially malicious nodes in the sys
central authority to certify node identities, it becomes
whether you can trust the claimed identity of somebo
before communicated. Dingledine proposes to address
including the use of micro-cash, that allow nodes to b

Bellovin [2] identifies a number of issues with Napste
how difficult it might be to limit Napster and Gnutella
they can leak information that users might consider pr
queries they issue to the network. Bellovin also expre
``push'' feature, intended to work around firewalls, wh
distributed denial of service attacks. He considers Nap
to be more secure against such attacks, although it req
central server.

It is worthwhile mentioning a very elegant alternative
table maintenance and forwarding that we rejected. Th
node by a group of diverse replicas as suggested by L
are coordinated using a state machine replication algo
tolerate Byzantine faults. BFT can replicate arbitrary s
it can replicate Pastry's routing table maintenance and
Additionally, the algorithm in [14] provides strong co
overlay routing and maintenance.

However, there are two disadvantages: the solution is
faults, and it is less resilient than the solution that we
expensive because it requires an agreement protocol b
replicas should be geographically dispersed to reduce
faults, agreement latency will be high. Additionally, e
have less than of its nodes faulty. This bound on t
per group results in a relatively low probability of suc
probability that a replica group with replicas is corre
nodes in the Pastry overlay is compromised is
denotes the binomial distribution with successes, t
success . For example, the probability that a replica
the nodes compromised and 32 replicas is less than 93
probability of routing correctly with 100,000 nodes in

8 Conclusions

Structured peer-to-peer overlay networks have previou
model for nodes; any node accessible in the network
follow the protocol. However, if nodes are malicious
it is possible for a small number of nodes to comprom
applications built upon it. This paper has presented th
techniques for secure node joining, routing table main
forwarding in structured p2p overlays. These techniqu
which can be combined with existing techniques to co
robust in the presence of malicious participants. A rou
of efficient proximity-aware routing in the common ca
costly redundant routing technique only when the test
interference by an attacker. Moreover, we show how t
be reduced by using self-certifying application data. T
tolerate up to 25% malicious nodes while providing g
fraction of compromised nodes is small.

Acknowledgments

We wish to thank Robert Morris, Rodrigo Rodrigues,
shepherd David Wetherall and the anonymous referee
We also wish to thank Adam Stubblefield for many di
assignment problem. This work was supported in part
(003604-0079-2001) and NSF (CCR-9985332).

Bibliography

1

M. Bellare and P. Rogaway.
The exact security of digital signatures- How to
In *Advances in Cryptology - EUROCRYPT 96,*
Science, Vol. 1070. Springer-Verlag, 1996.

2

Steve Bellovin.
Security aspects of Napster and Gnutella.
In *2001 Usenix Annual Technical Conference*, B
2001.
Invited talk.

3

Miguel Castro, Peter Druschel, Y. Charlie Hu, a
Exploiting network proximity in peer-to-peer ov
Technical Report MSR-TR-2002-82, Microsoft

4

Miguel Castro and Barbara Liskov.
Practical byzantine fault tolerance.
In *Proceedings of the Third Symposium on Ope*
Implementation (OSDI'99), New Orleans, Louis

5

Ian Clarke, Oskar Sandberg, Brandon Wiley, an
Freenet: A distributed anonymous information s
In *Workshop on Design Issues in Anonymity an*
320, July 2000.
ICSI, Berkeley, California.

6

Thomas H. Cormen, Charles E. Leiserson, and
Introduction to Algorithms.
MIT Electrical Engineering and Computer Scie

7

Frank Dabek, M. Frans Kaashoek, David Karge
Stoica.
Wide-area cooperative storage with CFS.
In *Proc. ACM SOSP'01*, Banff, Canada, Octobe

8

Drew Dean and Adam Stubblefield.
Using client puzzles to protect TLS.
In *10th Usenix Security Symposium*, pages 1-8,
2001.

9

Roger Dingledine, Michael J. Freedman, and D
Accountability measures for peer-to-peer syster

In *Peer-to-Peer: Harnessing the Power of Disru*
and Associates, November 2000.

10

John R. Douceur.
The Sybil attack.
In *Proceedings for the 1st International Worksh*
(IPTPS '02), Cambridge, Massachusetts, March

11

M. P. Herlihy and J. M. Wing.
Axioms for Concurrent Objects.
In *Proceedings of 14th ACM Symposium on Pri*
Languages, pages 13-26, January 1987.

12

A. Herzberg, M. Jakobsson, S. Jarecki, H. Kraw
Proactive public key and signature systems.
In *Proc. of the 1997 ACM Conference on Comp*
Security, 1997.

13

Ari Juels and John Brainard.
Client puzzles: A cryptographic defense against
In *Internet Society Symposium on Network and*
(NDSS '99), pages 151-165, San Diego, Califor

14

Nancy Lynch, Dahlia Malkhi, and David Ratajc
Atomic data access in content addressable netw
In *Proceedings for the 1st International Worksh*
(IPTPS '02), Cambridge, Massachusetts, March

15

Ralph C. Merkle.
Secure communications over insecure channels
Communications of the ACM, 21(4):294-299, A

16

Sylvia Ratnasamy, Paul Francis, Mark Handley,
Shenker.
A scalable content-addressable network.
In *Proc. ACM SIGCOMM'01*, San Diego, Califo

17

Antony Rowstron and Peter Druschel.
Pastry: Scalable, distributed object location and
to-peer systems.
In *Proc. IFIP/ACM Middleware 2001*, Heidelbe
2001.

18

Antony Rowstron and Peter Druschel.
Storage management and caching in PAST, a la
peer storage utility.
In *Proc. ACM SOSP'01*, Banff, Canada, Octobe

19

Emil Sit and Robert Morris.
Security considerations for peer-to-peer distribu
In *Proceedings for the 1st International Worksh*
(IPTPS '02), Cambridge, Massachusetts, March

20

Ion Stoica, Robert Morris, David Karger, M. Fr
Balakrishnan.
Chord: A scalable peer-to-peer lookup service f
In *Proc. ACM SIGCOMM'01*, San Diego, Calif

21

Ben Y. Zhao, John D. Kubiatowicz, and Anthon
Tapestry: An infrastructure for fault-resilient wi
Technical Report UCB//CSD-01-1141, U. C. Be

Appendix

This appendix describes an analytic model for the pro
negatives in the routing failure test.

We assume that there exist nodeIds distributed unif
interval of length . If is large and we look a
arbitrarily chosen location on this interval (for some
nodeIds is well approximated in distribution by a P
In particular, the inter-point distances are approximate
random variables with mean .

Let denote the exponential distribution with mean
exponential distribution with mean , where
nodes. Suppose are independent identically d
from one of these two distributions and we are require
distribution they are drawn from, e.g., can be
roots in Pastry and we are trying to determine if the se
only faulty nodes. An optimal hypothesis test is based
ratio to a threshold; by writing down the likelihood ra
equivalent to comparing the sample mean, denoted

We are in a situation where is unknown but we hav

(i.e., the samples that we collect from the nodeIds clo
space). We propose the following hypothesis test: cho
, for some constant , and accept/reject t
by comparing to this threshold. We now compute t
, and the false negative probability, , for this test.

Denote by and assume without loss of generalit
, define

and note that the are iid random variables. Let d
exponential random variables with mean . T
the event that . Thus,

where we write to denote probabilities when the
Recalling that has the gamma distribution with sha
parameter , we can rewrite the above as

where we used the change of variables an
equality. This expression can be used to compute n

We now derive a simple closed-form expression for a
bound shows that decays exponentially in the samp
the exact exponential rate of decay. For arbitrary
bound that

Now, if has an exponential distribution with mean

and for . Thus, for all

The tightest upper bound is obtained by minimising th

. The minimum is attained at . S

the bound,

We can derive an expression for the false negative pro

lines. Now, the are iid with distribution , i.e., they

with mean , and we are interested in the even

happens, then we fail to reject the hypothesis that the

where we write to denote probabilities when the

. In this case, has the same distribution as

distribution as , and we obtain using (1) tha

This allows us to compute numerically and by comb

the following closed-form upper bound

Footnotes

... set[1]

The neighbor set size should be chosen to ens

Generated by Sitaram Iyer (ssiyer@cs.rice.edu)

This paper was originally published in the Proceedings of the 5th Symposium on Operating Systems Design and Implementation, December 9–11, Boston, MA, US
Last changed: 6 Nov. 2002 aw