Secure routing for structured peer-to-peer overlay networks (by Castro et al.)

Shariq Rizvi
CS 294-4: Peer-to-Peer Systems

The problem

- P2P systems: resilient but not secure
- Malicious nodes:
  - fake IDs
  - distort routing table entries
  - prevent correct message delivery

"Techniques to allow nodes to join, to maintain routing state, and to forward messages securely in presence of malicious nodes"

In retrospect

- Unlike Byzantine solutions
  - specific to P2P systems
  - no exact agreement needs to be reached
- Unlike the Sybil attack
  - resort to central authentication for IDs

The model

- *N* nodes
- Bound *f* on the fraction of faulty nodes
- Bound *cN* on the number of faulty nodes in coalition
- Every node has a static IP address

"Secure routing ensures that when a non-faulty node sends a message to key *k*, the message reaches all non-faulty members is the set of replica roots R_k with very high probability"

Sub-problems

- Securely assigning IDs to nodes
  - attacker may capture all replicas for an object
  - attacker may target a particular victim
- Securely maintaining routing tables
  - attackers may populate with faulty entries
  - most messages are routed to faulty nodes
- Securely forwarding messages
  - even with proper routing tables, faulty nodes can corrupt, drop, misroute messages

Certified nodeIDs

- Offline certification authorities
  - assign random IDs to nodes
  - certificate binds the ID to public key and IP
    - attacker cannot swap IDs between his nodes
    - bad for dynamic address assignment, host mobility, or organizational changes
    - CAN nodeIDs change when nodes join and depart
  - Avoid giving multiple IDs to one entity
    - charge for each certificate – increases cost of attack
    - bind IDs to existing trustworthy identities

Secure routing table maintenance

- Should have at most fraction *f* of faulty entries
  - worst for row 0 of the Pastry routing table
  - during node joins, probability of getting a faulty entry is $(1 - f) \times f + f \times 1 > f$
- Impose constraints on the table entry
  - required to be closest to some point in ID space
  - like Chord

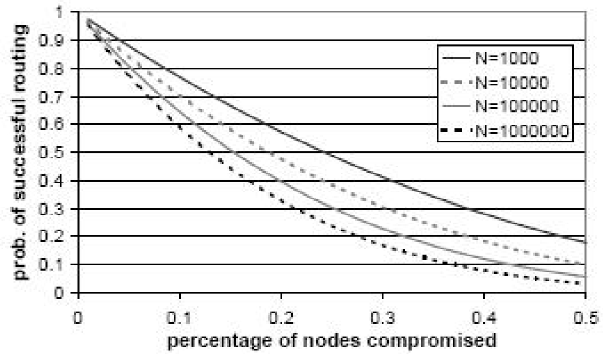
One solution for Pastry: two routing tables

- Normal locality-aware routing table
- A constrained routing table
  - Row *l*, column *d* entry for node *i*:
    - shares a prefix of length *l* with *i*
    - has *d* as its *(l+1)* st digit
    - closest nodeID to the point p: p satisfies above properties and has remaining digits same as *i*
- New state initialization algorithm exploiting an interesting property

Secure message forwarding

- Probability of routing successfully between two non-faulty nodes is $(1-f)^{h-1}$
  - h is $\log_{2^b}(N)$ for Pastry
- Probability of routing correctly to a non-faulty replica root is $(1-f)^h$
- Tradeoff: increasing *b* decreases the number of hops but also increases the amount of state information

Probability of routing to a correct replica b=4



Proposed Solution

- Has to ensure that with high probability, one copy of the message reaches each replica root

1. Route message to the key
2. Root node returns prospective set of replica roots
3. Did routing work? (failure test)
   - Yes: use these as replica roots
   - No: use expensive redundant routing

Routing failure test

- Takes a key and the set of prospective replica roots
- Returns negative if the set of roots is likely to be correct for the key; otherwise positive
- If no set is returned, returns positive
- Works by comparing the density of nodeIDs in the sender's neighborhood set with the density of nodeIDs close to the replica roots of the destination key

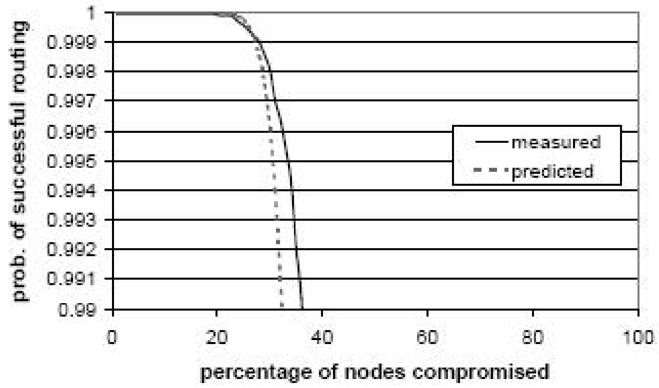
The test for Pastry

- In Pastry, the replica set is a subset of the neighbor set of the key's root
- Let μ_p be the average numerical distance between the consecutive nodeIDs in p's neighbor set
- To test root neighbor set {id0, id1, …}, for a key *x*, p checks that:
  1. All nodeIDs in the set have valid certificates, the middle one is the closest nodeID to *x*, and the nodeIDs satisfy the definition of a neighbor set
  2. The average numerical distance μ_{rn} between consecutive nodeIDs in this set < μ_p x γ
- γ decides the tradeoff between false positives and false negatives

Redundant routing

- If the failure test returns positive
- Use constrained routing table
- P sends the message to key *x* via different members of its neighborhood set
  - messages take diverse path (longer paths?)
- Any non-faulty node that receives the message and has the root of *x* in its neighborhood set, sends its certificate to p
- p collects such certificates in a list; sends the list to all nodes in the list. Process iterates upto 3 times
- p computes the closest neighbor's to *x*

Performance of redundant routing 100,000 nodes, b=4, l=r=32



Etc.

- Tolerates upto 25% malicious nodes well
- Self-certifying data – nodes can check the authenticity of returned objects
  - reduces need for redundant routing