

CS603: Distributed Systems

Lecture 19: Security Issues in DHTs

Next Tuesday

- Be prepared to talk for 5 minutes
- If you want to use slides, send them to me by Monday 11:59
- You can not be negative without offering solutions on how to address the negative comments

Final Project

- Deadline (for now) is Thursday May 4
- No classes week April 24 - 28
- 2 hour session of projects demonstration for everybody. (this will compensate for no classes in week April 24 -28)
- The 2 hour session is for everybody, it can be as late as Friday May 5
- Starting with tomorrow till April 6, everybody should meet with me to have a defined-detailed project with specified outcomes

Distributed Hash Tables

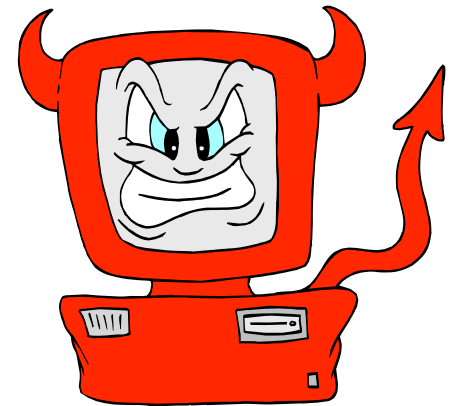
- Decentralized distributed systems that partition a set of keys among participating nodes with the goal to efficiently route messages to the unique owner of any given key
- **Key space**: ownership of keys is split among the nodes according to some partitioning scheme that maps nodes to keys
- **Overlay network**: nodes self organize in an overlay network; each node maintains a set of links to other nodes (its neighbors or routing table). Overlay and routing information is used to locate an object based on the associated key

Structured vs. Unstructured Overlays

- Structured overlay networks:
 - **Constrain the neighbor selection set**
 - A small subset of nodes meeting presubscribed conditions are eligible to become neighbors
 - The goal here is to bound the cost of locating objects and the number of network hops
- Unstructured overlay networks:
 - **Neighbor set selection is not constrained**
 - The goal here is maximizing performance in terms of throughput and latency

Security Issues in P2P Systems

- Are deployed over public open networks
 - ➡ Vulnerable to malicious attacks coming from **outside the overlay** network
Deployment of authentication mechanisms help significantly to avoid the attack
- Push trust to end-nodes: anybody can be part of the overlay
 - ➡ Vulnerable to malicious attacks coming from **inside the overlay** network (**Byzantine attacks**)
A different game: attacker can do anything, information can not be trusted



DHT Assumptions

- Nodes identifiers are assigned uniformly and randomly
- Each application object is assigned a key
- Each key is assigned to a root node for that key
- Objects are replicated and contained on replica roots, mapped by a replication function
- A node maintains
 - a routing table
 - a neighbor set which contains nodes whose nodeIds are close to that node
- Routing is performed based on the key, the routing table and the neighbor set

Communication

- Network communication (not-overlay), routing based on IP, transport communication UDP/TCP;
- Overlay communication, routing takes place based on P2P routing tables that relate to the mapping key-object;

Attacker Model

- A fraction f node can behave arbitrarily, including colluding
- Separate coalitions of colluding nodes are unaware of each other
- Attacker can not interfere with communication on network layer, cryptographic techniques are used as protection (authentication, integrity, confidentiality)
- Attacker can interfere with overlay level routing
- Messages between correct nodes and over good paths can be delayed, but eventually arrive

What Can Go Wrong?

- A small fraction of malicious nodes can prevent correct message delivery throughout the overlay.
- How? Can you describe specific attacks?
- P2P system can not be completely closed systems, they need to support open environments with mutually distrusting parties
- How to achieve this? Is it possible?
- When receiving the requested data, how do you know it is indeed the correct one and it was not modified by the node that stored it?

Secure Routing Requirements

Fundamental component of a solution to attacks against P2P.

GOAL:

When a non-faulty node sends a message with key k , the message reaches all non-faulty members of the set of replica roots corresponding to k with very high probability. Ensures that:

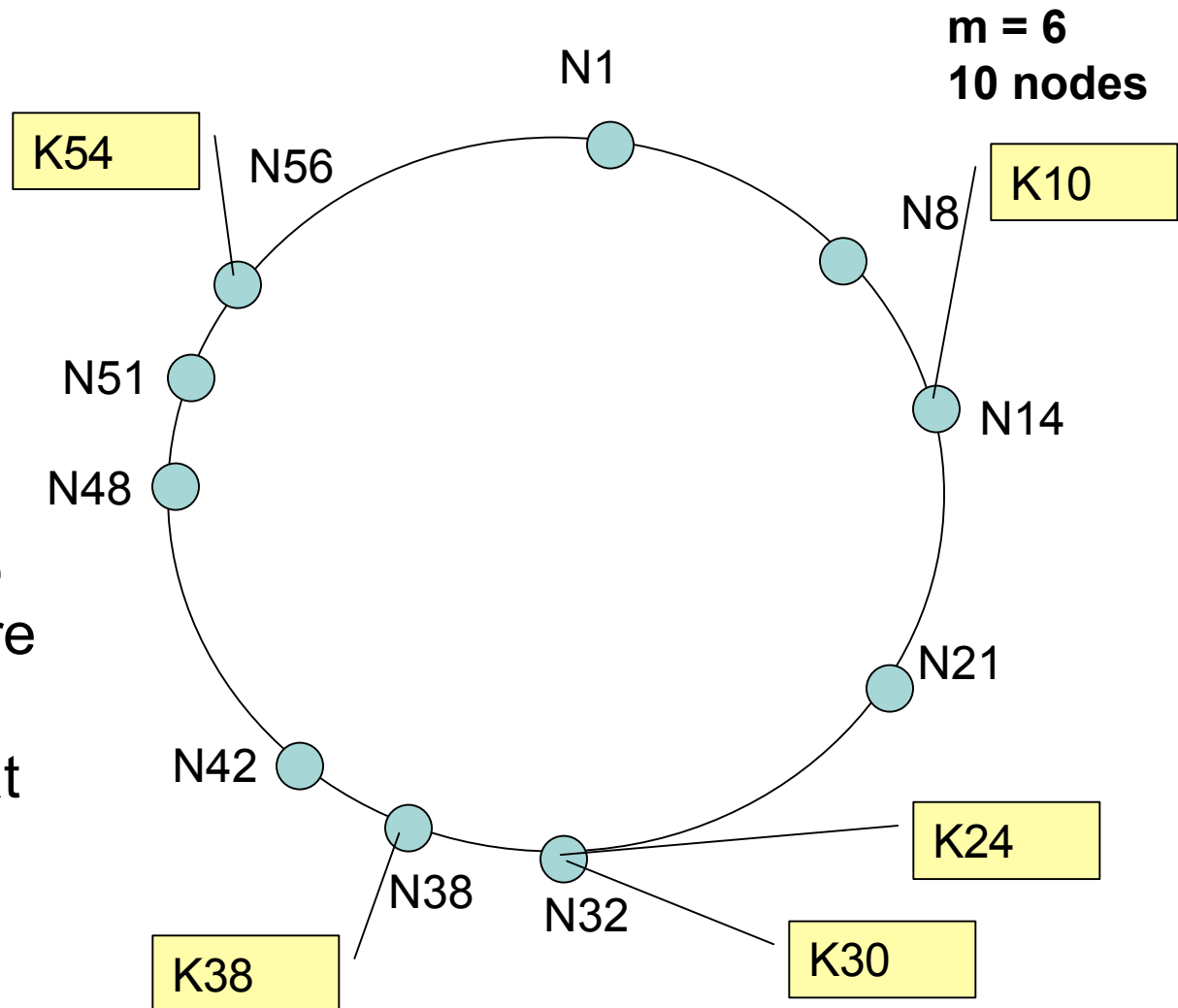
1.) Message is *eventually* delivered
2.) Message is delivered to all correct replica roots

Building Blocks

- 1. Secure assignment of node identifiers**
- 2. Secure routing table maintenance**
- 3. Secure message forwarding**

Node ID Assignment Implications

- Node ID determines where will be placed in the structured overlay
- Determines who the neighbors are going to be
- Determines what objects a node will hold



Attacks Based on Node ID Assignment

- What if attacker can choose the ID of a node?
 - Surround a victim node
 - Partition a p2p network
 - Can control what object will be a replica for
 - Holding objects allow an attacker to delete, corrupt or deny access to objects
- Can an attacker choose the ID of a node?
 - In some systems ID is randomly generated
 - In some systems ID is the hash of the IP address

Sybil Attack

- Attack particular to peer networks, a malicious attacker takes/forges multiple identities
- Result: attacker controls a significant part of the system while correct nodes are not aware of this, they see different identities
 - destroy cohesion of the overlay
 - observe network status
 - slow down, destroy overlay
 - DoS
- How to ensure/validate distinct identities refer to distinct entities?

Evaluating Identity

- Straightforward form of identity: secure hash of a public key
- How to evaluate/learn the identity of other entities
 - Use a trusted agency (learn from trusted source)
 - A node has a direct way of validating other nodes - direct validation (learn directly)
 - Using other untrusted agencies - indirect validation (learn from others)
- Which one is best?

Direct and Indirect Validation (Untrusted Sources)

- Utilize computational tasks to validate distinctness;
 - validate distinctness of two entities by getting them to perform some task (for example a computational puzzle) that a single entity could not
 - can not assume homogeneous resources, only minimum; faulty entity could have more than minimum
 - The goal is to make it practical impossible for an adversary to have challenges issued simultaneously, limit the number of identities he can forge

Direct Validation Limitations

- Even with severely resource constraints, a faulty entity can counterfeit a constant number of identities
- Each correct entity must simultaneously validate all the identities it is presented otherwise a faulty entity can counterfeit an unbounded number of identities

Indirect Validation

- A sufficiently large set of faulty entities can counterfeit an unbounded number of identities
- All entities in the system must perform their identity validations concurrently, otherwise a faulty identity can counterfeit a constant number of multiple identities

Certified (Secure) NodeID Assignment

- Delegate ID generation to trusted CAs
- Bind IPs with nodeIDs such that colluding attackers can not exchange certificates
- Nodes must pay for certificates to prevent attackers from buying many "correct" certificates
- Works for static IP addresses
- Does not solve all problems: what happens if the IP changes?
- What happens if the trusted CA is not available or can not be reached?

Certified (Secure) NodeID Assignment

- How about distributed ID generation with periodic renewal of distributed IDs
 - Addresses single point of failure
 - Requires techniques to moderate the rate at which attackers can acquire node IDs

Routing Table Maintenance

- Routing table contains information about where to 'look next'
- Table is updated based on information from other nodes

$m = 6$

Finger Table for N8

N8+1	N14
N8+2	N14
N8+4	N14
N8+8	N21
N8+16	N32
N8+32	N42

finger [k] = first node that succeeds $(n+2^{k-1}) \bmod 2^m$

Attack Against Maintaining Routing Table

- Attackers can easily supply ‘malicious’ updates or can return incorrect lookup
 - point to faulty or non-existent nodes
 - fake the closest node
 - lie about next hop
- Result: lookup will fail (denial of information to a node) or the lookup algorithm will have sub-optimal performance

Secure Routing Table Maintenance

- Constrained Routing Tables: Identify invariants in the system and look for violations of the invariants
- Maintain two routing tables - one that uses proximity information and one that constrains entries to "specific" values
 - Proximity routing used in normal operation
 - Constrained routing used when failures occur:
- Other proposed solutions involve anonymous auditing

Secure Bootstrapping

- How to securely bootstrap the routing table?
 - A new node, n , picks a subset of bootstrap nodes to query and join the network
 - n uses the bootstrap information to initialize its constrained routing table

Attacks on Forwarding

- Simply ignore forwarding messages, route to the wrong node
 - Failed if ANY one in routing is faulty
 - Probability of routing successfully to a replica root is $(1-f)^{h-1}$
 - h is the number of average hops for delivering a message
 - h depends on the overlay

Secure Message Forwarding

- Ensure that with high probability, at least one copy of a message reaches every correct replica root
- Collect the prospective set of replica roots from the prospective root node
- Apply **routing failure test** to determine if routing worked correctly, If no, use ***redundant and/or iterative routing.***

Testing Routing

- Route Failure Test:
 - Average density of nodes per unit of “volume” in the id space is greater than the average density of faulty nodes
 - Compares density of nodes in the neighbor set of the sender with the density of nodes close to the replica roots of the destination key
 - Have sender contact all prospective roots
 - Timeout to detect ignoring routing msgs, selecting the appropriate threshold not easy
- Use redundant routing when test fails
 - Neighbor set anycast - sends copies of message towards destination until they reach a node with the key in its neighbor set.
- How about false positives and false negatives when performing the routing failure test?
- Redundant routing has high overhead?

Iterative Routing

- Alternative to redundant routing
- Every lookup answer goes back to the requester that can verify that the next hop gets him closer (using the distance function) to the node hosting the object associated with the requested key
- Iterative routing is more secure, but more expensive

What Does Secure Routing Buy Us?

- Prevents attacks at join time: secure nodeID assignment and bootstrapping
- Ensure that when a correct node sends a message for a particular key, the message reaches all correct replica roots for the key with very high probability.
- What about the data? We need other mechanisms, for example self-certifying data

Self-Certifying Data

- Client can check data and only needs to rely on routing when certification check fails.
- Reduces the reliance on the redundant, secure routing primitive (you still need secure forwarding otherwise there is no data to verify in the first place)
- Uses concepts like proactive signature sharing or group keys/signatures.
- Self-certifying data can eliminate the overhead of secure routing in common cases

Reading

- Security Considerations for Peer-to-Peer Distributed Hash Table. Emil Sit, Robert Morris
- Secure routing for structured peer-to-peer overlay networks. Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, Dan S. Wallach