A survey of Peer-to-Peer Security Issues

Dan S. Wallach
dwallach@cs.rice.edu
Rice University

Presented by: Jamal S. Bajaber
jbajaber@gmu.edu

Acknowledgment

Some of the followings slides are borrowed or adapted from slides made by the author of the paper.

Outline

- Background
- Pastry
- System model
- Routing security
- Fairness
- Trust
- Conclusions

Background

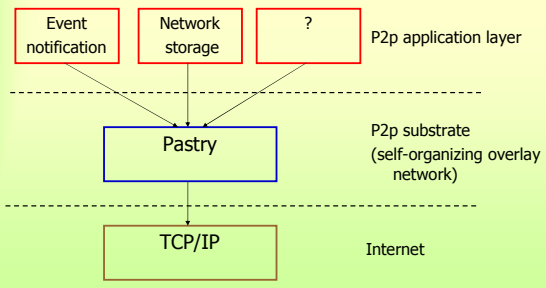
Peer-to-peer systems

- Unstructured (Napster, Gnutella)
- Structured (Can, Chord, Pastry, Tapestry)

Common issues

- Organize, maintain overlay network
  - Node arrivals
  - Node failures
- Resource allocation/load balancing
- Resource location
- Locality (network proximity)

Architecture



Outline

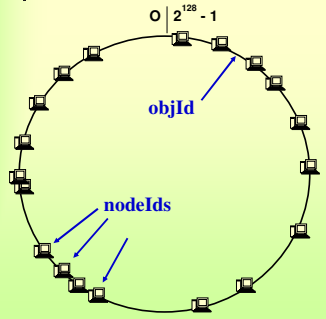
- Background
- Pastry
- System model
- Routing security
- Fairness
- Trust
- Conclusions

Pastry

Generic p2p location and routing substrate

- Self-organizing overlay network
- Consistent hashing
- Lookup/insert object in < $log_{16}N$ routing steps (expected)
- *O(log N)* per-node state
- Network locality heuristics

Pastry: Routing



Consistent hashing

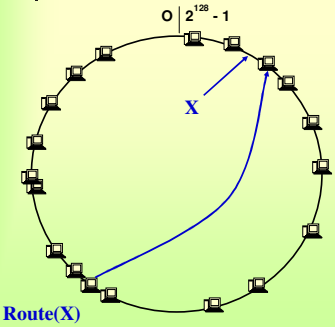
128 bit circular id space

nodeIds (uniform random)

objIds (uniform random)

Invariant: node with numerically closest nodeId maintains object

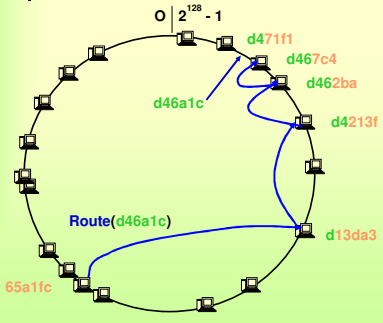
Pastry: Object insertion/lookup



Msg with key *X* is routed to live node with nodeId closest to *X*

Problem: complete routing table not feasible

Pastry: Routing



Properties
- log16 N steps
- O(*log N*) state

Outline

- Background
- Pastry
- System model
- Routing security
- Fairness
- Trust
- Conclusions

System model

- A set of N nodes
- Faulty nodes
  - f ($o \le f < 1$)
  - Independent coalition sets
    size bounded by cN ($1/N \le c \le f$)
  - $c = f$ ➔ most damage to the system
- Static IP address

System model

- Two types of communication
  - Network-level
    nodes communicate directly
    [Cryptography - to protect from adversaries]
  - Overlay-level
    Messages are routed through the overlay
    [Secure Routing Primitive]
- An adversary has complete control over network-level communication to and from nodes it controls

Outline

- Background
- Pastry
- System model
- Routing security
- Fairness
- Trust
- Conclusions

Security issues

Peer-to-peer systems

- Structured (Can, Chord, Pastry, Tapestry)
  - Application
    - File sharing systems

Security issues

Possible Attacks

Hard attacks

- erroneous responses to a request
  - **application level**: returning false data
  - **network level**: returning false routes
- traffic analysis
  - against systems provide *anonymous* communication
- Censorship
  - against systems provide high availability

Security issues

Possible Attacks

Other (softer) attacks

- fairness
  - disk space
  - network bandwidth
- trust
  - data
  - code

Routing security

- Secure routing ensures:
  - the message is eventually delivered
  - the message is delivered to all legitimate replica roots for the key
  - the replicas are initially placed on legitimate replica roots

Routing security

- Secure routing primitive:
  - Must deal with the following problems
    1. Secure nodeId assignment
    2. Secure routing table maintenance
    3. Secure message forwarding

Node ID assignment

- If you could choose nodeIds maliciously…
  - Control/censor all replicas of a document
    - Surround it in ID space
  - Control all outgoing routes from a node
    - Mediate a victim's access to the network
- NodeIds must be *random*

Simple solution

- Central authority assigns node IDs
  - Can also act as a certification authority
    - Corporate version: verify user-id / password
    - Commercial version: charge money
- Insufficient for small networks
  - Attacker could still control large % of nodes(Sybil Attack)
  - Moderate the rate at which nodeIds are given out

☛Small p2p networks must be trusted

Non-centralized solution?

- Preferable to avoid centralized nodes
  - Reliability, "spirit of P2P", etc.
- Some primitives we might use to build a solution
  - Bit commitment protocols
  - Solving hard problems (e.g., crypto puzzles)

Problems…

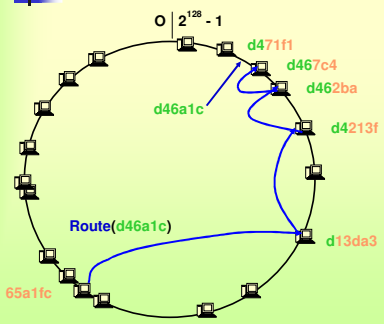
- Attacker with lots of {money, CPU time} can still take over.
- For now, stick with centralized solution.

Secure routing table maintenance

- routing tables and neighbor sets of correct nodes should have an average fraction of only *f* random entries point to faulty nodes in the entire overlay
- Attackers can increase the fraction
  - Locality-based attack
  - False routing updates

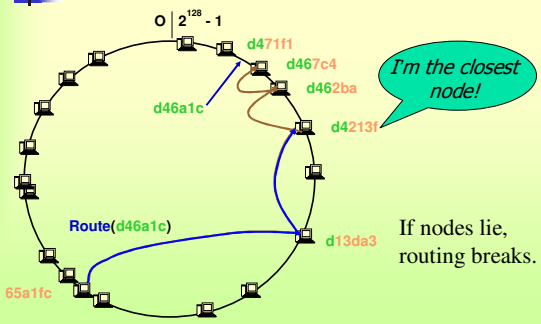
(more details in Castro et al [OSDI 2002])

Malicious routing



Routing trusts the intermediate nodes to be honest.

Malicious routing



If nodes lie, routing breaks.

Solving malicious routers

- Constrained routing
  - Two routing tables per node
    - One with locality, one "constrained"
      - Harder for attackers to corrupt constrained routing tables
    - First, try the normal route
    - If "suspicious", try
      - Diverse routes, using constrained routing table

Secure message forwarding

A faulty node in the route

- Dropping messages
- Routing messages to wrong nodes
- Pretend to be the replica root

Secure message forwarding

Solution

- Detect faults (*failure test*)
  - the test is not accurate
- Use divers routes (*redundant routing*)
  - Success ($f \leq 30\%$)

(more details in Castro et al [OSDI 2002])

Ejecting misbehaving nodes

- how to remove a malicious node from the overlay?!
- When a node accuses another of cheating, how to proof that?!
  - to avoid denial of service attack

Open problems!

Routing security

Secure routing primitive
- huge overhead!

What is the alternative?!

self-certifying data

Data whose integrity can be verified by clients

- Use efficient routing to request an object
- Check its integrity
- Integrity check fails / no response
  Use secure routning

Insertion object → use secure routing only!

Outline

- Background
- Pastry
- System model
- Routing security
- Fairness
- Trust
- Conclusions

Fairness

- Goal: fair use of
  - network storage
  - network bandwidth
- Possible policy
  - You can't use more than you give others

Storage(Quota Architectures)

Simple quota management
- Centralized server
  - Easy to keep policy consistent
  - Huge bottleneck
  - Single point of failure
- Smart cards
  - Quota information is distributed
  - Central issuing organization is required
  - Hacked card → infinite storage

Storage(Quota Architectures)

Distributed quotas

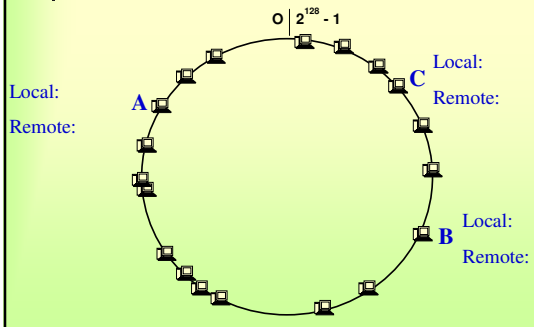
- Option 1: distributed quota managers
  - Comparable to smart card
  - Your leaf set maintains your quota records
    - Track nothing and endorse a request

Storage(Quota Architectures)

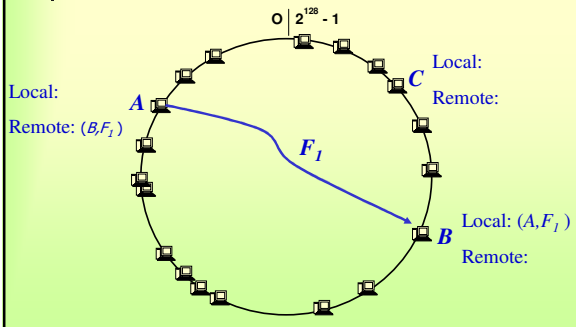
Distributed quotas

- Option 2: Economic system
  - Nodes *publish* accounting for storage
  - *Incentives* to be honest
  - *Auditing* to detect cheaters
  - *Punishment* for cheating
- Disk space is a lot like money
  - Let's build a disk space economy!

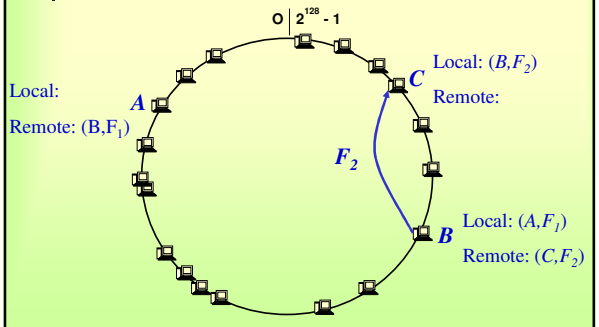
A simple disk economy



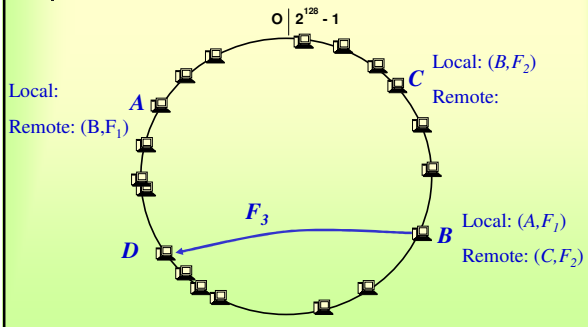
Each node tracks storage



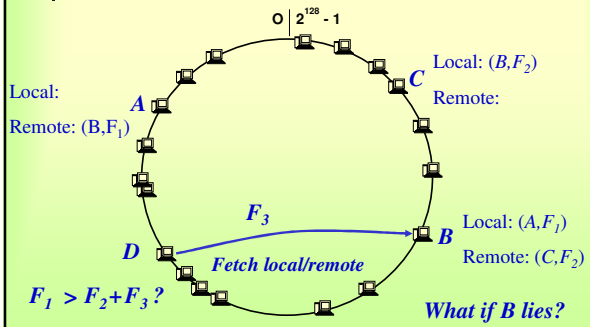
Each node tracks storage



Verify quota before storage



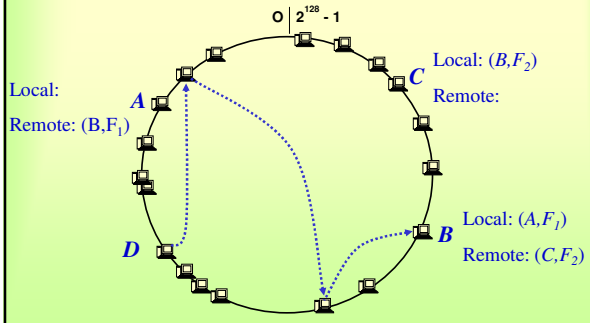
Verify quota before storage



Lying

- Inflate local list (claim you're giving more)
- Deflate remote list (claim you're using less)
  - Both let you use more on the network
- Incentives not to lie?

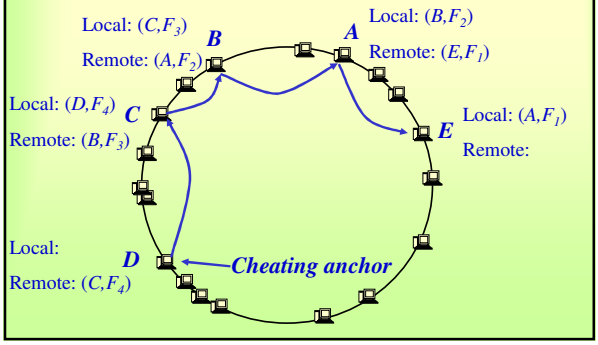
Need anonymous auditing



Auditing

- Alice stores file on behalf of Bob
  - Alice audits Bob's *remote list*
  - If Bob isn't "paying", Alice can delete the file
    - Disincentive to deflate remote list
    - Natural economic incentives to follow rules
- How to verify no inflation of the local list?

Cheating chains



Eliminating cheating chains

- Random audits will find cheating anchors
  - Verify that local books balance
  - Verify one level of indirection
- Local/remote list is a signed confession
  - Convince leaf set to eject the node

Fairness

Network bandwidth

- Micropayment system
  - Pay a token per a request
  - Gain a token when receive a request
- Problems
  - Scalability not clear yet
  - Checking token validity is more expensive
  - Requests may refused
    - Nodes had no need for more tokens
    - Make widely replicated data

Outline

- Background
- Pastry
- System model
- Routing security
- Fairness
- Trust
- Conclusions

Trust in P2P Overlays

Data

- The data being shared might not be trustworthy!
- Popularity-based ranking system
  - Need a popularity notion
    - Audit logs
    - Rank your files
    - Ran others files

Trust in P2P Overlays

Code

- applications can perform significant computations and consume vast amounts of disk storage
- Full privileges to access network and disks
- How trust is the code?!
- Need an architecture to safely execute untrusted code

Outline

- Background
- Pastry
- System model
- Routing security
- Fairness
- Trust
- Conclusions

Conclusions

Routing security
- Secure nodeId assignment
  - Requires trusted authority
  - Easy to build a public key infrastructure
- Secure routing tables
  - Requires diverse routes
  - Can use efficient techniques until suspicious
- Secure message forwarding
  - Requires costly techniques

Storage
- Good quotas need interesting primitives
  - "Open" books
  - Anonymous communication
  - Digital signatures
- Economic incentives keep nodes honest
  - Random auditing required to detect all cheaters

Network bandwidth
- Fair sharing still need efficient solution

Trust
- Ranking system required to ensure data popularity
- a general-purpose mobile code security architecture is needed to code trust