LOCKSS – Lots of Copies

Preserving Peer Replicas By Rate-Limited Sampled Voting

P. Maniatis, M. Roussopoulos, TJ Giuli, D. Rosenthal, M. Baker, Y. Muliadi

(Stanford U.)

In *Proc. of SOSP 2003*

Motivation

- Academic publishing is moving to the web
- Rental provides no guarantee of long-term access
- Librarians see this as their responsibility
- *How to preserve access to journal and other archival information published on the web?*
  - Ensuring long-term access
  - Guarantying authenticity of document copies

Answer

- LOCKSS: Lot Of Copies Keep Stuff Safe
- A digital preservation system that models the physical document system approach
- Having many copies ensures the long-term survival of the documents
- Peer-to-peer opinion polls guarantee the authenticity of the documents

Digital preservation systems

- Must resist random failures and deliberate digital attacks for a **long time**
- Have unusual requirements:
  - Lack of central control
  - Must avoid long-term secrets like encryption keys
- Can make some operations very time consuming without sacrificing usability
- Must be very cheap to build and maintain
  - No high-performance hardware (RAID)
- Need not to operate quickly
  - Should prevent rather than expedite changes
- Must properly operate for decades without central control

Design principles

- Cheap storage is unreliable:
  - Write-once media are a least as unreliable as disks
- No long-term secrets:
  - Too hard to preserve; too hard to recover from leak
- Use inertia:
  - Prevent change, do not make it too easy
- Avoid third party reputation:
  - Too vulnerable to slander or subversion (eBay problem)
- Intrusion detection is intrinsic:
  - Not done by extrinsic system
- Assume a strong adversary:
  - Attackers will be able to use very large numbers of hosts

Existing LOCKSS system

- Makes it appear to uses that pages remain available at their original URL even when they are gone
- Libraries participate in a P2P system, running persistent web caches that
  - ***Collect*** documents by crawling journal websites
  - ***Distribute*** by acting as limited proxy cache for the library's patrons
  - ***Preserve*** by cooperating with other caches to detect and repair damages
- Caches cooperate
  - Sample of peers vote on the hash of a specified part of the contents
  - Provide peers with confidence in content ***authenticity*** and ***integrity***

Why polls?

- On-line journals
  - Do not sign the materials they publish
  - Do not provide manifest enumerating the files forming a paper, issue or volume
- Crawling is unreliable
- NO completely reliable storage medium exists
  - All media can be stolen or destroyed
- Better to put our trust in number of replicas

Existing LOCKSS system

- Peers vote on large ***archival units*** (AU)
  - Year run of a journal
- Each peer will hold a different set of Aus
  - No universal library
- A peer that loses a poll has a bad AU
  - Call series of increasingly specific partial polls to locate damage
- Once damage is located/detected, provide site having a damaged copy with a good one if the site has participated in a previous poll
  - *Prevents free-loading*
- Peers only supply materials to peers that can prove they had the material in the past
  - *Prevents theft*
- Inexpensive
  - One PC with 3x180GB disk can preserve 210 years of largest journal

The new opinion poll protocol

- Assumes no common-mode failure
- Two classes of peers
  - Malign peers – conspiring peer trying to subvert the system
  - Loyal Peers is a non-malign peer
    - Damaged (has bad AU)
    - Healthy (has correct AU)
- Overall goal – To ensure that loyal peers have a high probability to be in a healthy state

A poll and its outcomes

- A LOKSS peer
  - Calls a poll much more frequently than any anticipated rate of random damage
  - Invites into its poll a random subset of peers
- Poll outcomes
  - Landslide win
    - Votes overwhelmingly agree with peer’s version of AU
    - Do nothing
  - Landslide loss
    - Votes overwhelmingly disagree with peer’s version of AU
    - Repair peer’s version of AU (by updating it)
  - Inconclusive poll
    - Require human intervention

Roles and voting membership

- Poll initiator
  - Only beneficiary of the outcome
- Poll participants
  - Need not find out the result of polls
  - Inner and outer circle
    - Inner circle participants – Selected by the poll initiator from its Reference List; only their votes count
  - Outer circle participants – Nominated by inner circle participants and selected by poll initiator
    - Could be invited into further inner circles

Sybil-Attack preventions

- Sybil attack: Use an unlimited number of forged identities to subvert a system
- Prevention schemes:
  - Infrequent voting (Limits the rate of change in the system
  - Bimodal distribution of system states (increase the chance to trigger alarms)
  - Require each peer to expend significant computing power for each step
    - Computing the hash for an AU
  - Churn (to be explained later)

Poll initiation and poll effort proof

- Initiator sends to each inner circle peer a Poll message containing a fresh public key
- Inner circle peers reply with Poll Challenge
- For each Poll Challenge it has received, initiator produces some computational effort that is provable via a pool effort proof and sends it in a Poll Proof message
- Nominate and Vote messages follow

Vote verification and tabulation

- Verification
  - If proof of effort is incorrect, vote is invalid, peer is black listed
  - If proof is correct, and hash matches, it is valid and agreeing
  - If proof is correct, and hash mismatches, it is valid and disagreeing
- Tabulation
  - Agreeing votes < threshold (landslide loss), the initiator needs to repair its copy
  - Agreeing votes > threshold (landslide win), the initiator updates its reference list and schedules the next poll
  - Otherwise, raise an alarm (inconclusive)
- No able to get quorum (enough valid votes) for a while
  - Raise an alarm (inter-poll interval)

After the poll

- Repair
  - Need to detect inconsistencies between the voting information and the repaired AU
  - If initiator cannot complete the repair process, raise the corresponding alarm
- Reference list update
  - Remove all disagreeing peers and some randomly chosen agreeing peers from the inner circle
  - Resets the expiration time for the remaining peers
  - Insert all outer circle peers whose votes were valid and agreeing
  - Insert randomly chosen entries from friends list up to a churn factor

From the invitee’s perspective

- Poll solicitation
  - Decide if to participate and challenge the initiator
- Poll effort verification
  - Verify poll effort proof and nominate a random set of peers to the initiator’s outer circle
- Vote construction
  - Vote is hash of AU and interleaved with provable computational effort
  - Vote computation is divided in rounds, each with computational effort and the hashed portion double in size
  - A subsequent challenge is dependent on the previous challenge
- Repair solicitation
  - After the vote, the initiator may ask for help to repair its AU

Protocol objectives

- Overall objectives
  - Prevent an adversary from swaying an opinion poll in his favor
  - Or waste loyal peers' resources
- Thus, the protocol must
  - Prevent adversary from gaining a foothold in a poll's initiator reference list
  - Make it expensive for adversary to waste another peer's resources
  - Make it likely that the adversary 's attack will be detected on time

Mechanisms used to get it …

- Effort sizing (along inertia – large changes require large efforts)
  – Use memory-bound computations
  – An initiator needs to expend more effort than the cumulative effort it imposes on the voters (computation >> verification)
- Timeliness of effort (avoiding third party reputation)
  – Only proofs of recent effort can affect the system
  – Need to expend resources to maintain foothold

Mechanisms used to get it …

- Rate limiting (another application of inertia)
  - Loyal peers call polls autonomously and infrequently
  - The rate of progress for an attack is limited by victims, not by attackers
- Reference list churning
  - Avoid depending on a fixed set of peers
    - They become easy targets
  - Avoid depending on entirely on random peers
    - They can launch Sybil attacks
  - With friends list
    - Attackers can gain foothold on the outer circle list but not the friends list

Mechanisms used to get it

- Obfuscation of protocol state (assuming a powerful adversary)
  - Encrypt all but the first protocol message exchange
  - All loyal peers invited to a poll, even those who decline to vote, must go through the motions of the protocol
    - Can’t deduce the number of loyal peers who are involved in deciding the outcome of a poll
- Alarms (intrusion detection is inherent to the system)
  - Protocol raises an alarm if a poll is inconclusive, suspects local spoofing, hasn’t been able to complete a poll for a while
  - Raising an alarm is expensive to discourage a rational adversary

Adversary attacks

- Platform attacks
  - Can take over a fraction of peers instantaneously
  - Not discussed in the paper but accounted for in the evaluation
- Protocol attacks – Play against the LOCKSS protocol, some examples
  - Stealth modification
    - Replace good AUs with bad ones without being noticed
  - Nuisance
    - Raise many alarms to waste resources and dilute alarms credibility
  - Attrition
    - Prevent loyal peers from repairs by wasting resources elsewhere
  - Theft
    - Obtain published content without paying
  - Free-loading
    - Obtain services without supplying services in return

Counter-attack techniques

- Adversary foothold in a reference list
  - Need to wait for invitation to vote
  - Need to behave well for a long time before the attack (without raising alarms)
- To deal with adaptive adversary (deciding what to do after collecting all information
  - Defend by requesting commitments on future protocol steps
    - Ask random sample bits (verified) before each poll
    - The repair AU must match the initial bits
- Avoid hijacking
  - Randomly retransmit PollChallenge msgs trying to get to the initiator

Stealth modification attack strategy

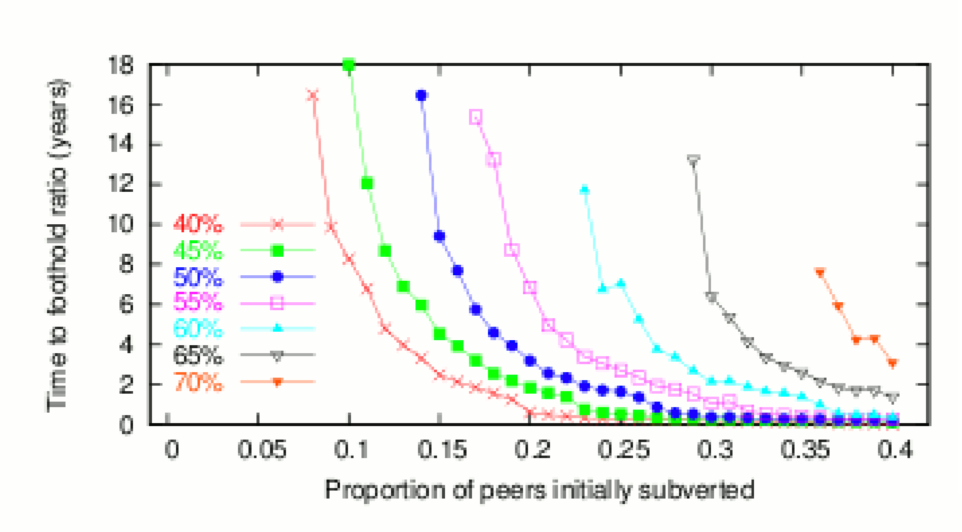
- Two phases
  - Lurk to build a foothold in loyal peers' reference lists
  - Attack
- Need to have the majority of votes
- Need to have loyal peers < the alarm threshold
- An adversary
  - Needs to wait for an initiator to call for votes
  - Needs to go through many rounds of voting without triggering an alarm
  - Needs to expend effort to maintain the foothold in the reference list

Evaluation

- For real
  - LOCKSS program initiated by Stanford University Libraries
    - Software under development since 1999
  - First beta version released in 2001
  - Production version released in April 2004 as Open Source (http://www.sourceforge.com)
- Simulation
  - Running LOCKSS for 30 years (event-based simulation)
  - 1000 peers
    - To initiate the friends list (and thus the reference list)
      - Clusters of 30 peers
      - 29 peers in the initial friends list
        - » 80% from the local cluster
  - 20 years of lurking, 10 years of attacking
  - Two sets of simulations (attacks instructed by lurking results)

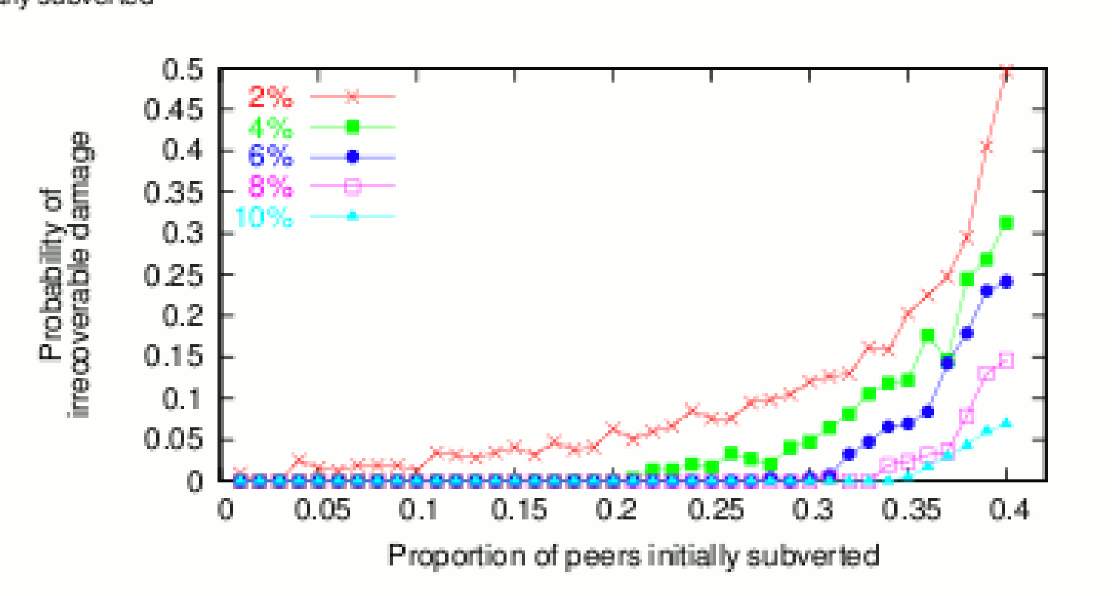
Results

Low rates of false alarms in the absence of attacks



Time taken by the lurking phase to a foothold; with low subversion levels adversary needs nearly 20 years to get 40% foothold ratios

Can sustain up to 1/3 of the peers subverted (with 10% churn)



Conclusions

- Work has shown you can use
  - Massive replication
  - Rate limitation
  - Inherent intrusion detection
  - Costly operations

  to build an archival system capable of resisting attacks by powerful adversaries over decades