



Conservation vs. Consensus in Peer-to-Peer Preservation Systems

Prashanth P. Bungale

Division of Engineering and Applied Sciences
Harvard University

Joint work with Geoffrey Goodell and
Prof. Mema Roussopoulos

Peer-to-Peer Preservation

■ Preservation:

- A peer is interested in preserving *archival units* (AUs)
- Each AU replicated on a subset of peers
- Other peers' aid and resources used to achieve goal

■ Challenges:

- Natural degradation in storage media: *bit-rot*
- Catastrophic events
- Human errors
- *Attacks* by adversaries to change the data preserved
- Providing incentives

Context: LOCKSS

- A p2p system for preserving access to archival information published on the Web
- In production use in over 80 libraries worldwide
- A large number of independent, low cost, persistent web caches
- Cooperate to detect and repair damage
 - Via voting in “*opinion polls*”
- Two approaches to providing preservation: Consensus and Conservation

Consensus

- *Goal*: all peers come to a *uniform agreement* over time
- Each peer believes its AU version may be *questionable*
 - Uses *aggregate opinion* of other peers to resolve doubt
 - Sometimes involves *replacing* its current copy
 - Requires frequent peer participation due to lack of self-confidence

Conservation

- *Goal*: each peer retains indefinitely the exact copy of AU it holds initially
- Each peer believes its AU is the “right” version
 - Always attempts to *conserve* this copy – even if other peers disagree
- Two modes of operation
 - *Time-of-need*
 - When it suffers a damage to its AU
 - Only time when it seeks the help of other peers to recover
 - *Self-Sufficiency*
 - Does nothing to its AU

Framework for Design Considerations

- Trust in the source of the AU
 - Low trust => Consensus
- Trust in the means of procuring the AU
 - Low trust => Consensus
- Frequency of bit-rot
 - High frequency => Conservation
- Frequency of human error
 - High frequency => Consensus
- Resource relevance to participants
 - Low relevance => Conservation

Framework (cont'd)

- Presence of adversaries

- Tolerance for stealth-modification

- Is it tolerable for some of the peers to have an incorrect AU sometimes?

- Tolerance for nuisances

- Can the users tolerate frequent nuisances?

Framework applied to LOCKSS

- Trust in the source of the AU and trust in the means of procuring the AU: *low*
- Frequency of bit-rot: *extremely low*
 - Assumed to be once in 200 machine years on an average
- Frequency of human error: *can be high*
- Resource relevance to participants: *high*
 - As libraries often subscribe to the same AUs from the publishers
- Presence of adversaries:
 - Unlimited computation power and unlimited identities
 - Tolerance for stealth-modification: *medium*
 - Tolerance for nuisances: *medium*

LOCKSS - Design

- Obtaining full consensus is difficult
- Each peer holds
 - Reference list of peers it has discovered
- Periodically (faster than rate of bit rot)
 - Takes a sample of the reference list
 - Invites the chosen peers to send a hash of their copy of the document

LOCKSS - Design (cont'd)

- Peer compares votes with local copy
- If landslide agreement, the peer is happy
- If landslide disagreement, the peer repairs
 - To repair, the peer gets the copy of somebody who disagreed and then reevaluates the same votes
- If poll is inconclusive, the peer raises alarm
 - Alarms are built-in intrusion detection

LOCKSS - Evaluation*

■ Stealth-modification Attacks

- Worst-case increase of probability of accessing a damaged copy was just 3.5%
 - With up to 40% of the peer population being adversarial

■ Nuisance attacks

- Results showed that the system could only moderately defend against nuisance attacks

* ***Preserving Peer Replicas By Rate-Limited Sampled Voting.*** Petros Maniatis, Mema Roussopoulos, TJ Giuli, David S. H. Rosenthal, Mary Baker, and Yanto Muliadi. *Proceedings of the 19th ACM Symposium on Operating System Principles (SOSP)*. October 2003. Bolton Landing, NY.

Conservation Approach: Sierra

- Conservation based alternative to LOCKSS
- Borrows techniques from LOCKSS
 - Calling opinion polls using a sample of the peer population
- But, does not blindly rely on the results of the polls
- Aims to achieve better defenses against attacks
 - But, at the expense of making restrictive assumptions

Framework applied to Sierra

- Trust in the source of the AU and trust in the means of procuring the AU: *high*
- Frequency of bit-rot: *low*
- Frequency of human error: *low*
- Resource relevance to participants: *high*
- Presence of adversaries:
 - Unlimited computation power and unlimited identities
 - Tolerance for stealth-modification: *zero-tolerance*
 - Tolerance for nuisances: *low*

Sierra - Design

- Store AU plus hash of AU
- Peers call polls periodically as in LOCKSS
 - If the stored AU and hash *match*
 - *Self-sufficiency* state => Ignore the result of the poll
 - **Blacklist** disagreeing voters
 - If the AU and local hash *do not match*
 - Enter *time-of-need* state and remain in this state for next n polls
 - If **minority threshold** no. of peers agree with each other, raise alarm
- Voters only vote if in *self-sufficiency* state and decline the poll invitation otherwise

Sierra - Analysis

■ Stealth-modification Attacks

- The adversary is **unable** to carry out successful attacks even at one peer
 - Even with up to **60%** of the peer population being adversarial

■ Nuisance attacks

- The system offers **strong defense** against nuisance attacks
 - Because it is vulnerable to nuisance attacks only when a bit-rot occurs

Conclusion

- Preservation is not straightforward
- Consensus Approach: LOCKSS
- Conservation Approach: Sierra
- Trade-off between defense against:
 - Inadvertent damages
 - Bit-rot, catastrophic events and human errors
 - Intentional damages
 - Adversarial attacks



Questions?

E-mail:

{prash, goodell, mema} @ eecs.harvard.edu