“PeerReview”: Practical Accountability for Distributed Systems

Andreas Haeberlen1,2, Petr Kouznetsov1, Peter Druschel1
- Max Planck Institute for Software Systems
- Rice University

Questions that “PeerReview” addresses

- How to detect faults?
- How to identify the faulty nodes?
- How to convince others that a node is (not) faulty?

Source of this slide: SOSP'07 slide by authors.
Claims to..

- Detect "Byzantine faults" and identify a "faulty" node.
  - Byzantine faults – when a faulty component not only behaves erroneously but also inconsistently when interacting with other components.
- Never convicting a "correct node".
- Done by:
  - Keeping secure record for each message sent & received by each node.
  - Nodes can sign their messages.
  - Nodes get periodically checked by other correct nodes.

Limitations

- Correct node's actions are deterministic.
- Cannot expose a faulty node that keeps quiet and never sends a message that a correct node would not send.
  - Keeping quiet – packet loss or processing delay.
- Can only detect fault after a correct node is causally affected.
- Cannot detect every instance of misbehavior – limits its scalability. So, offers probabilistic detection guarantee.
- Message signing introduces overhead – larger for shorter messages and LAN.
  - Overhead grows linearly with # of nodes that need to check a given node's action to be sure that at least one of the inspecting nodes or inspected node is correct.
Assumptions

- Each node has a cryptographic key pair that can be linked to an unique node identifier.
- Correct nodes' behavior deterministic & communication between correct nodes eventually succeeds.

High level view of “PeerReview”

- Secure record of all nodes’ action
- Inspect recorded information and detect faulty behavior.
- An ideal fault detector should guarantee
  - Whenever a node becomes faulty, it should be exposed
  - No correct node is ever exposed by a correct node.
- Some terminology:
  - Correct node: that follows a given protocol.
  - Faulty node: Does something that a correct node would never have done given a situation.
Problem statement

- Detectably faulty (DF) – ex. Node 'I' sends an incorrect message 'm' that causally affects a correct node.
  - The nodes that send msgs caused by 'm' => 'Accomplices of I wrt m'.
  - A node may appear as "correct" to different individual correct nodes but still be "DF". Ex. 1d
- Detectably ignorant (DI):

Consistent hashing – used to map witnesses to a node
- Each node witnesses other $k$ nodes whose node IDs are closest to its own.
- Choosing parameters:
  - Choosing witness set, too large no scalable, too small it can miss some faults.
  - Msg complexity $O(Ψ^2)$
  - Taudit & Tbuf – buffering authenticator – determine maximum time to detection. Smaller => faster detection, but increases msg overhead as small pieces of logs are transferred rather than large chunks.
  - Audit – on demand rather than periodic.
Applications

- Overlay multicast
  - So called free-loader problem
  - Detect nodes that may want to tamper with contents
  - Witnesses are selected randomly