Managing Shared Ephemeral Teleconferencing State: Policy and Mechanism

(a.k.a. An update on the agreement algorithm)

Scott Shenker <shenker@parc.xerox.com>
Abel Weinrib <abel@bellcore.com>

March 30, 1994

Outline

What's new

Session state and operations

Policy specification

Mechanisms built on reliable communication service

Mechanisms built on unreliable communication service

Draft paper available:

ftp.isi.edu:confctrl/docs/agree.ps

What's New?

New policy specification supports

- "tight" control of some state by some members
- "loose" control of some state (with eventual consistency for some members)

smooth continuum from tight control (with poor scaling properties) to loose control (with good scaling)

New hybrid mechanism

- locking for tight control
- announcements with ordering of messages by time stamps for loose control

Mechanism for unreliable network transport

- announcements with suppression and back-off

Session State and Operations

Session state $\{S_i\}$

- members M
- eventual members M_e
- critical members M_c
- policy variables
  - $\{I(O_i)\}$, $\{V(O_i)\}$, E, C, D
- everything else
  - media descriptors
  - encryption key
  - common names of members
  - who pays the bill
  - etc.

Operation $op=O_i \bullet ... O_j$

- O_i acts on S_i

Expressing Policy

Initiator

- $I(O_i)$ specifies the set of members that can initiate operation O_i

Voting rules

- $V(O_i)$ function of vector of members' votes from (YES, NO, ABSTAIN, NO-REPLY)

Eventual Consistency

- Strong Eventual Consistency—"all eventual members agree on all state variables in E a time T after the last change operation"
- Weak Eventual Consistency—"the probability that the eventual members have different views asymptotically vanishes"

Eventual state is the result of a causal ordering of operations for both

Consistent Voting

"whenever votes are cast on a change to state in D, then all members in M_c agree on the state in C"

- C is the critical state that is viewed consistently
- D is the delta state that is changed
- M_c lists the critical members that vote with the consistent view

M_c in C, only critical members may initiate changes to state in C or D

Example

A private conference controlled by a set of key members:

- members M (=eventual members M_e)
- critical members M_c
- media stream descriptors
- encryption key
- common name of each member
- next-to-speak
- policy specification
  - M_c in C and in D
  - M (=M_e) in D
  - adding a member requires unanimous vote of the critical members
  - deleting a member initiated by a critical member, but requires the assent of the member
  - media stream descriptors are eventually consistent state, and requires unanimous vote of all members
  - encryption key is eventually consistent state, and requires no voting
  - only a member may change its common name, and it is changed unilaterally (the voting rule is null)
  - next-to-speak is not eventually consistent state, changes can be initiated by any member, but require a majority vote

Implementing Policy

Computational infrastructure

- local clock
  - bounded clock skew
- buffer to store operations

Communications infrastructure

- reliability
  - reliable (TCP-like)
    - all messages are delivered in bounded time
    - order is preserved for messages from a sender to a receiver
  - unreliable
    - probability that k consecutive messages from a sender to a receiver all fail to arrive is $O(k^{-x})$
- addressing
  - explicitly named list
  - shared bus (multicast)

Mechanism—Reliable Communications

Static Membership

Messages:

- `Poll(Id: id, Operation: op)`
- `Lock(Id:id)`
- `Poll_Lock(Id: id, Operation: op)`
- `Announce(Id: id, Operation: op, Time: timestamp)`
- `Commit(Id: id, Operation: op, Time: timestamp)`
- `Release(Id: id, Operation: op)`
- `Response(Id: id, Response: response)`

Locking used for changes to state in C or D

- guarantees Consistent Voting condition

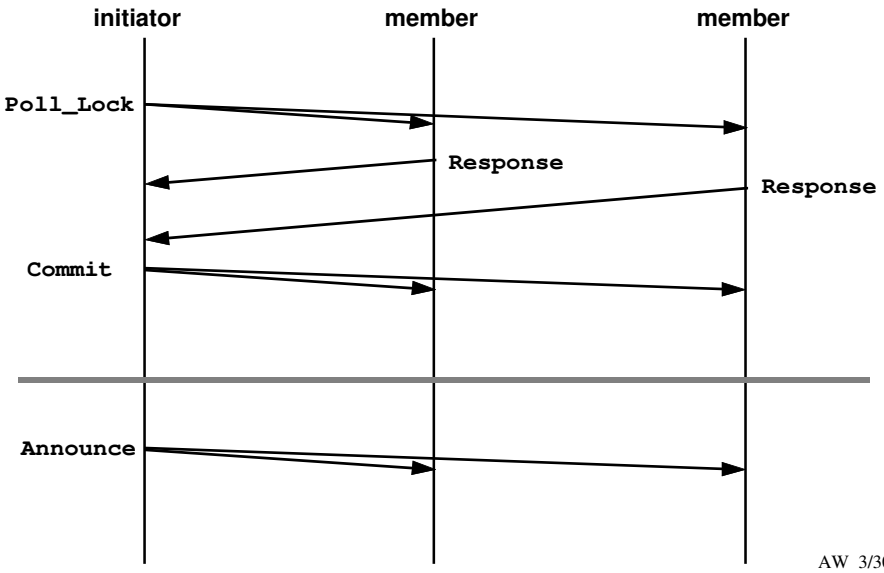
Announces used for changes to state not in C or D

- scales well

Buffering of messages based on time stamps used for changes to state in E (but not C or D)

- guarantees Strong Eventual Consistency and causality

Session Change



Changing Membership

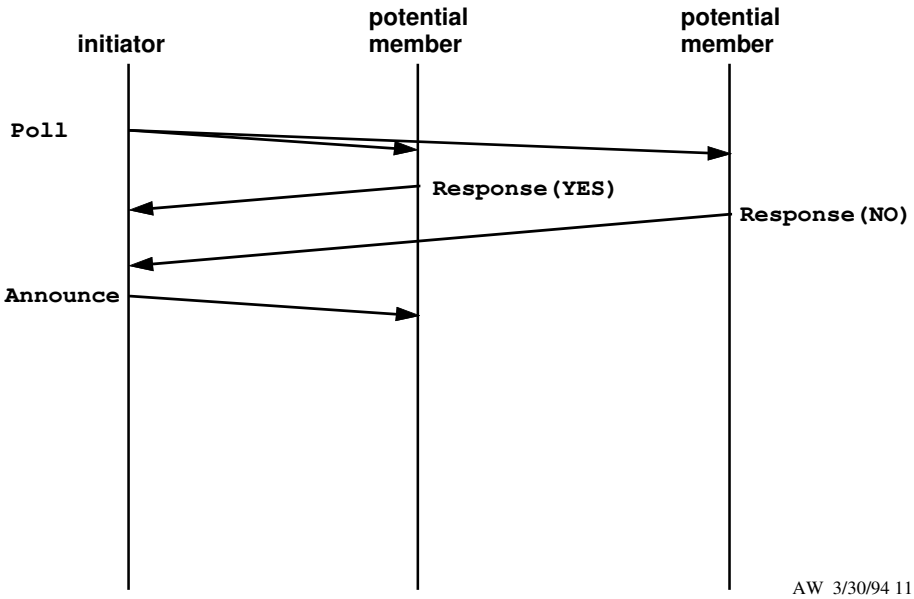
Adding a member

- the “candidate” contacts a member to act as initiator
- if successful, initiator
  - transfers its view of the state to the new member
  - forwards `Announce` or `Commit` messages for a fixed time to the new member

Session establishment

- initiator sends `Poll` message to all proposed members for an operation that builds the proposed session
- state includes voting rule that determines whether the session should be established as a function of how the proposed members respond
  - all members must agree to join
  - a particular set of members must agree
  - etc.
- if the proposal is successful, `Announce` messages are sent to the proposed members that voted YES

Session Establishment



Recovering from failures

"reliable communications" = "try hard, then time out"

Recover to a safe state (no session) when state becomes corrupted

- allow unilateral leaving by any member
  - change voting correctness to "votes are taken with plausible view of membership"

Mechanism—Unreliable Communications

(shared bus)

Restricted consistency policy

- Weak Eventual Consistency

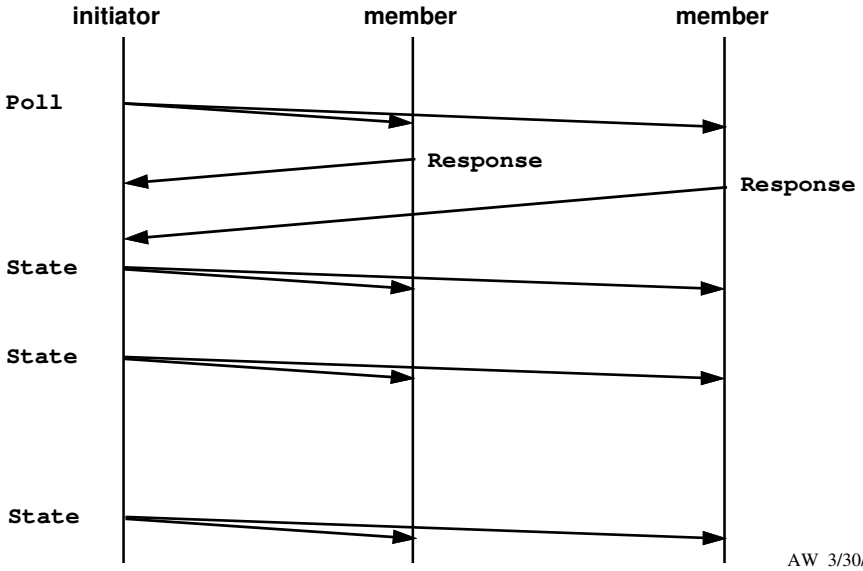
still support voting rules, initiator policies, causality

Messages

- `Poll(Id: id, Operation: op, Variable: i, Value: value_i, Variable: j, Value: value_j, ...)`
- `Response(Id: id, Response: response)`
- `State(Time: timestamp, Variable: i, Value: value_i, Variable: j, Value: value_j, ...)`

Mechanisms

- sequence of multicasts of `State` message by initiator of latest change to each piece of state
- suppression policy using timestamp values
- various back-off strategies to achieve good scaling

provides Weak Eventual Consistency with causal ordering of messages

Session Change