Teleconferencing Session Management Engine

Ted Ko

Bellcore Applied Research

Master's Thesis supervised by Abel Weinrib and John Wroclawski

Outline

- Overview of Context and Goals
- Session Description Language
- Centralized Model
  - --- **API boundaries and control flow**
  - --- **Implementation issues**
  - --- **Touring Machine (tm) example**
- Distributed Model
  - --- **Language modifications**
  - --- **Implementation issues**
- Research Issues
  - --- **Generality**
  - --- **Dynamicity**
  - --- **Scalability**

Context

"Managing Shared Ephemeral Teleconferencing State: Policy and Mechanism"

by Scott Shenker, Abel Weinrib, and Eve Schooler

- Teleconferencing style determined by policy choices
- Most systems limit styles based on mechanism
- Abstraction language separates policy and mechanism

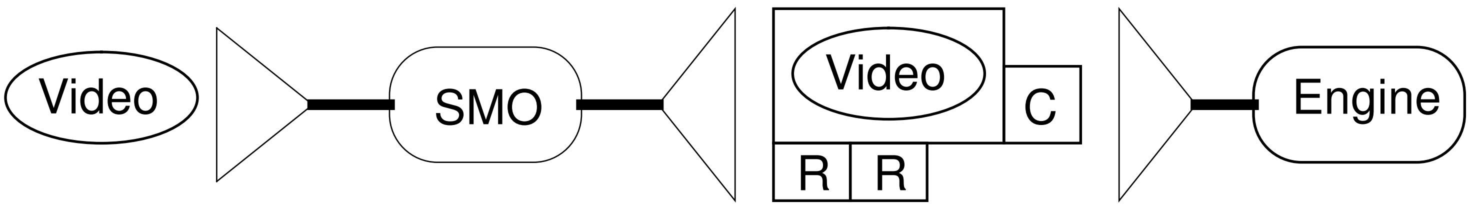
Session Abstraction

Session Description:

- Variables
- Members
- Rules
- Consistency designations

Policies are executed based on this description.

The Session Engine is intended to be a tool used by a session management object (SMO).



Research Issues

- Implementation concerns

  --- Limitations
  --- Tradeoffs in complexity and function

- Other types of conferencing generality

  --- Basics of teleconferencing state
  --- Different call models

- Usefulness of flexible control policy

  --- Comparison to lightweight specific applications

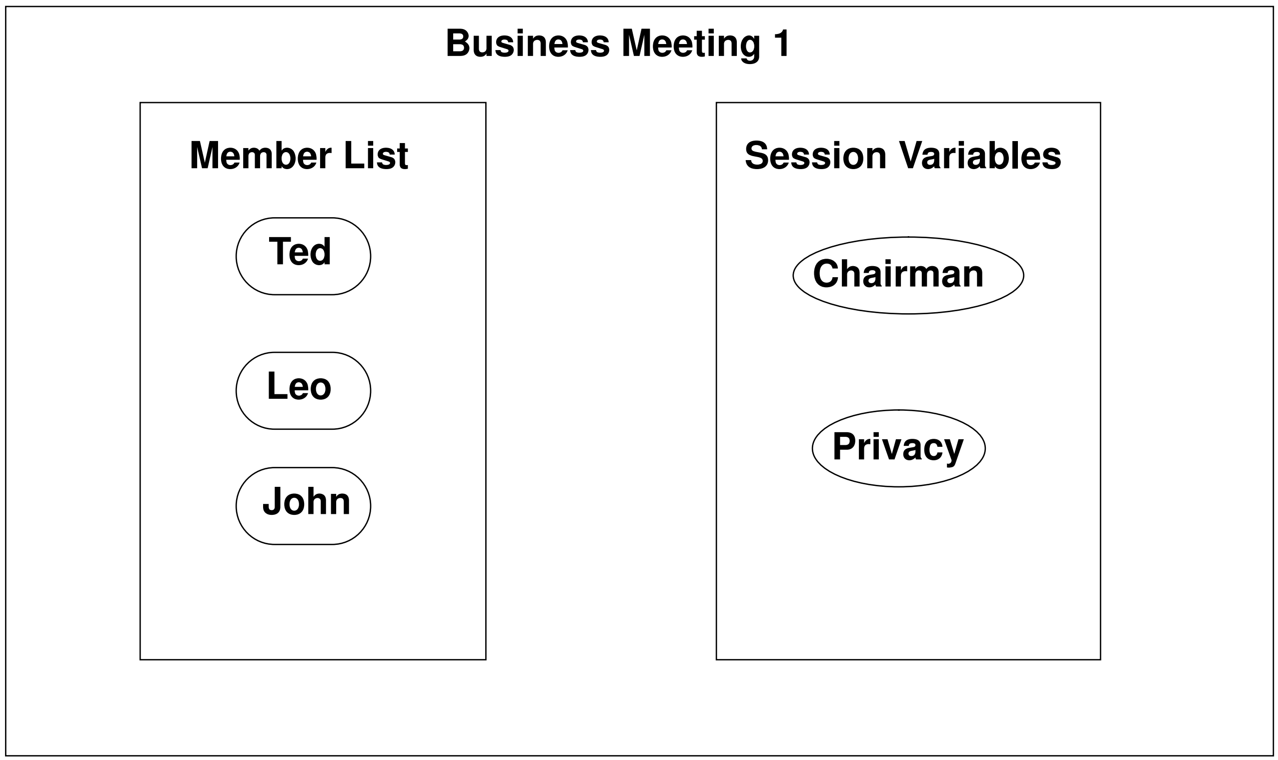
- Usefulness of flexible consistency policy

  --- Scalability

Description Language

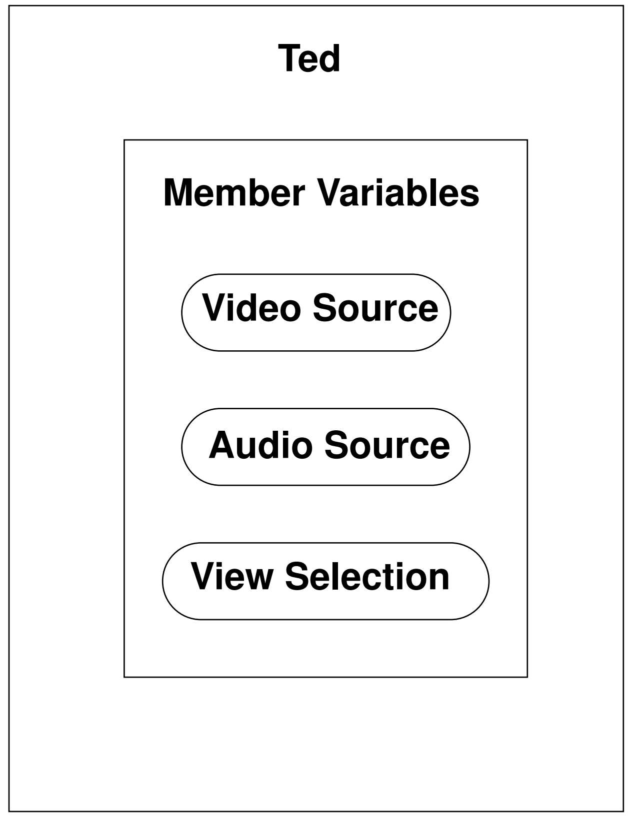
SESSION: session-name ; <create-type> ; <Member List> ; <Session Variables>

create type : unanimous, majority, any, none



Session Member

MEMBER: member-name ; consistency ; <Member Variables>

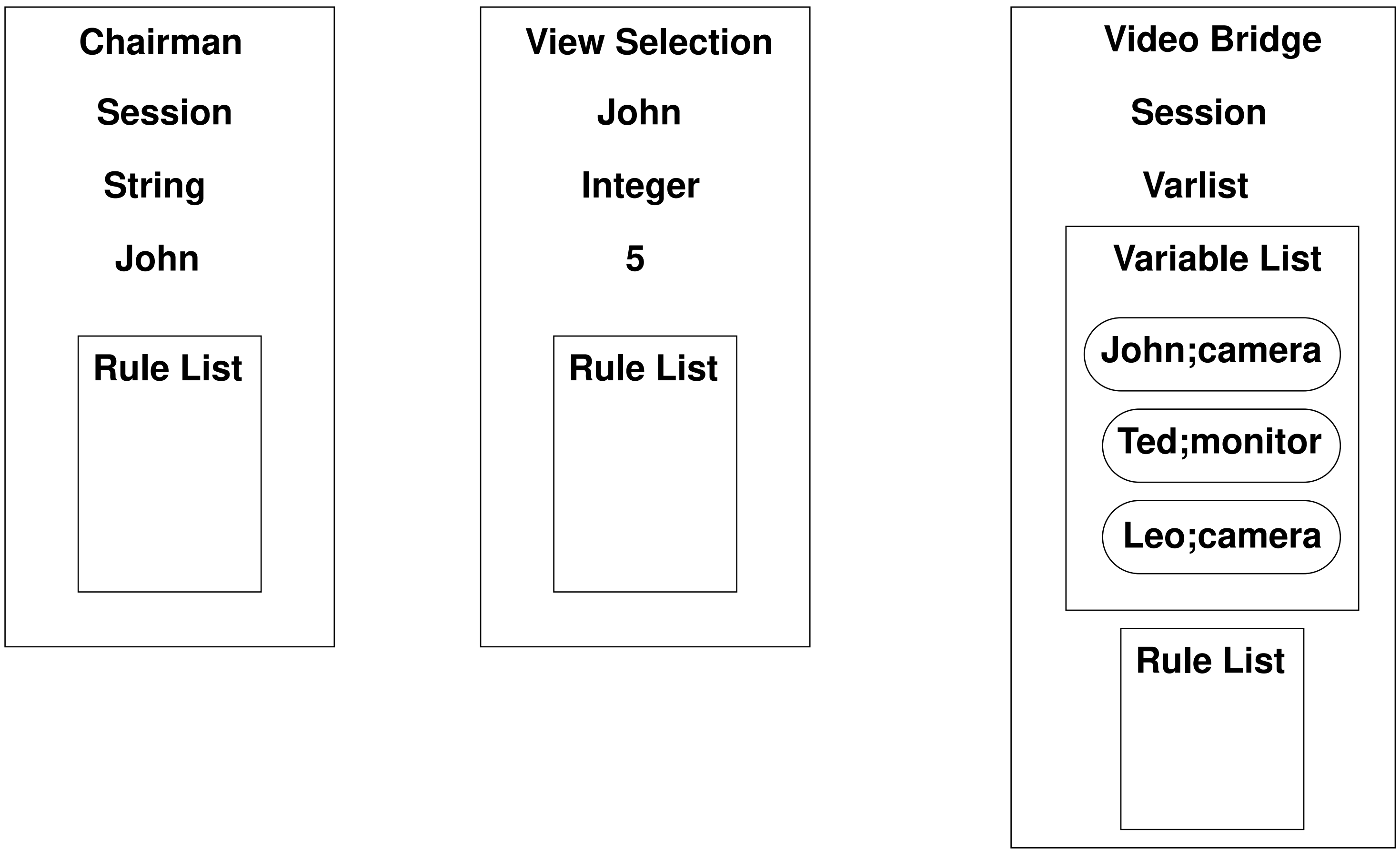


Variable

VARIABLE: variable name ; variable owner ; consistency ; <vartype> ;

[value, <variable list>] ; <Rule List>

vartype: string , integer , floating , varlist



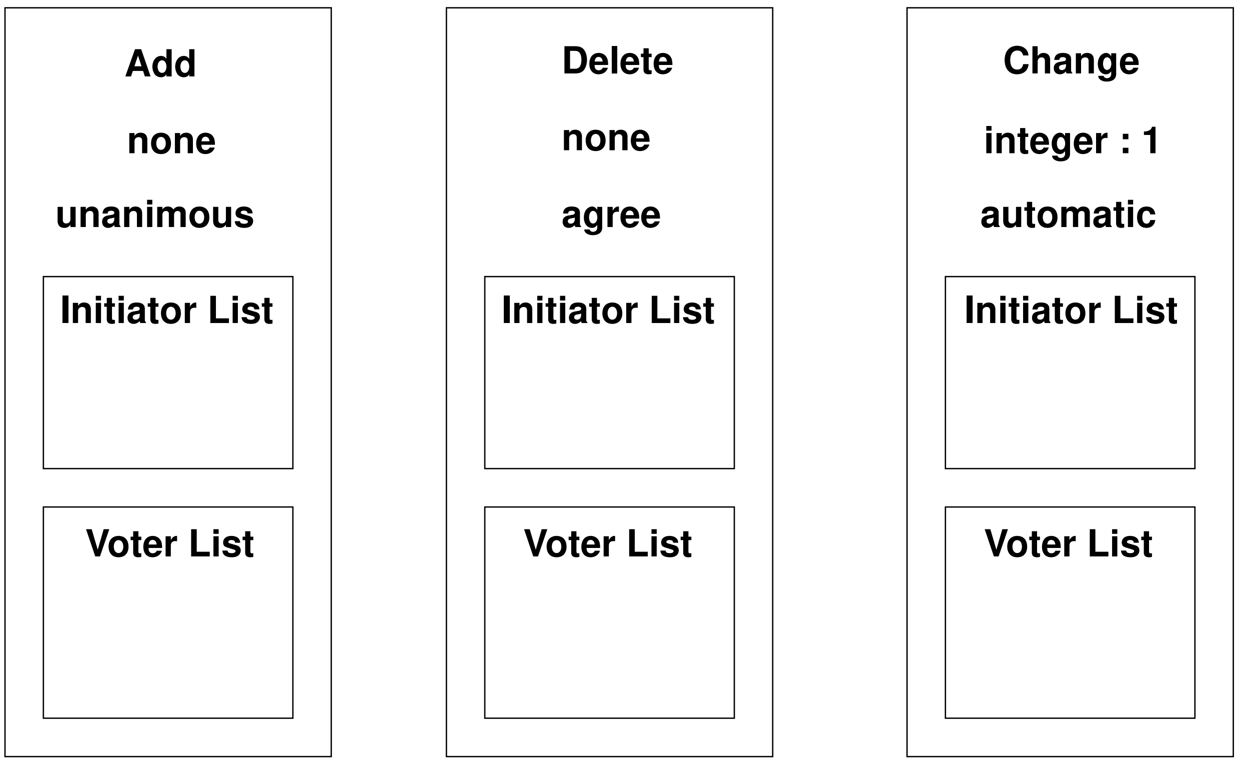
Rule

RULE : <operation type> ; <value type>; [value] ; <vote type> ; [fraction] ;
<Initiator List> ; <Voter List>

operation type : add , change, delete

value type : string, integer, floating, none

vote type : agree, unanimous, majority, fraction, anyone, automatic



Initiators and Voters

INITIATOR : <initiator type> ; initiator name

initiator type : member , group, owner, anyone, designation

VOTER : <voter type> ; voter name

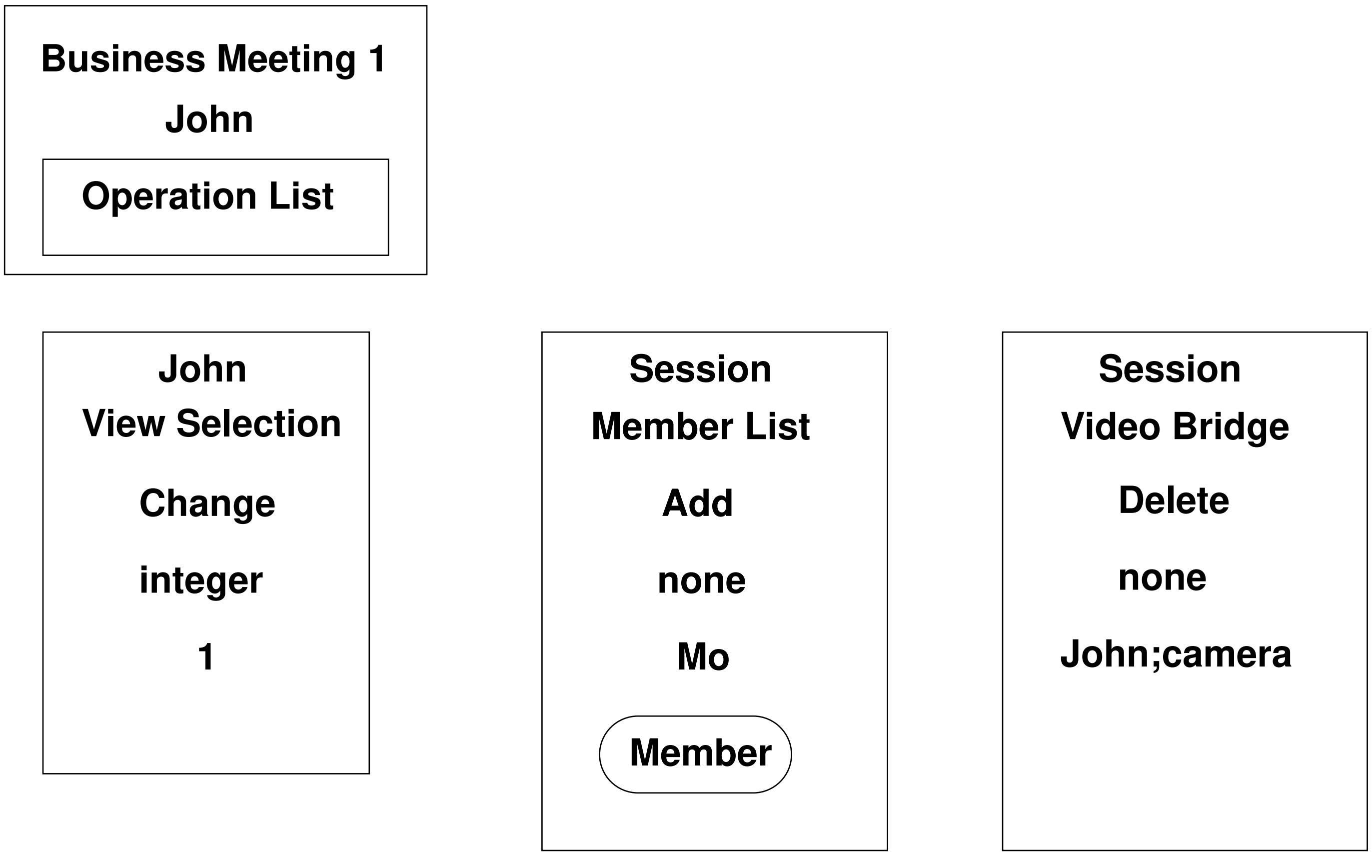
voter type : member, group, owner, designation, target, towner, lowners

Add	Delete	Change
none	none	none
unanimous	agree	agree
Initiator List group:M	Initiator List member:Leo	Initiator List owner
Voter List group: M	Voter List target	Voter List designation: Chairman

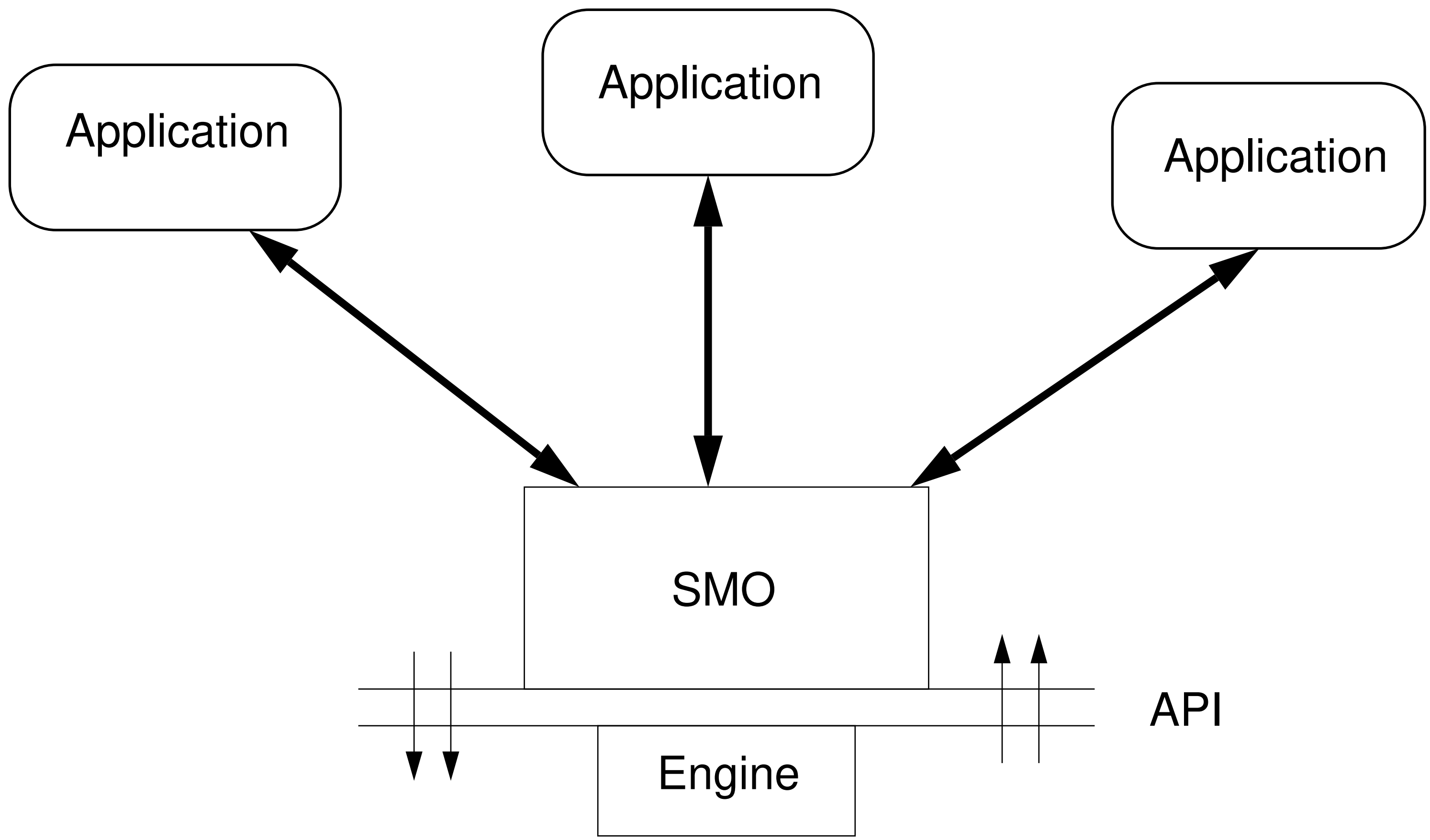
Operation and Operation Set

OPERATION : variable owner ; variable name ; <operation type> ;
<value type> ; [value] ; keyword ; [Variable, Member]

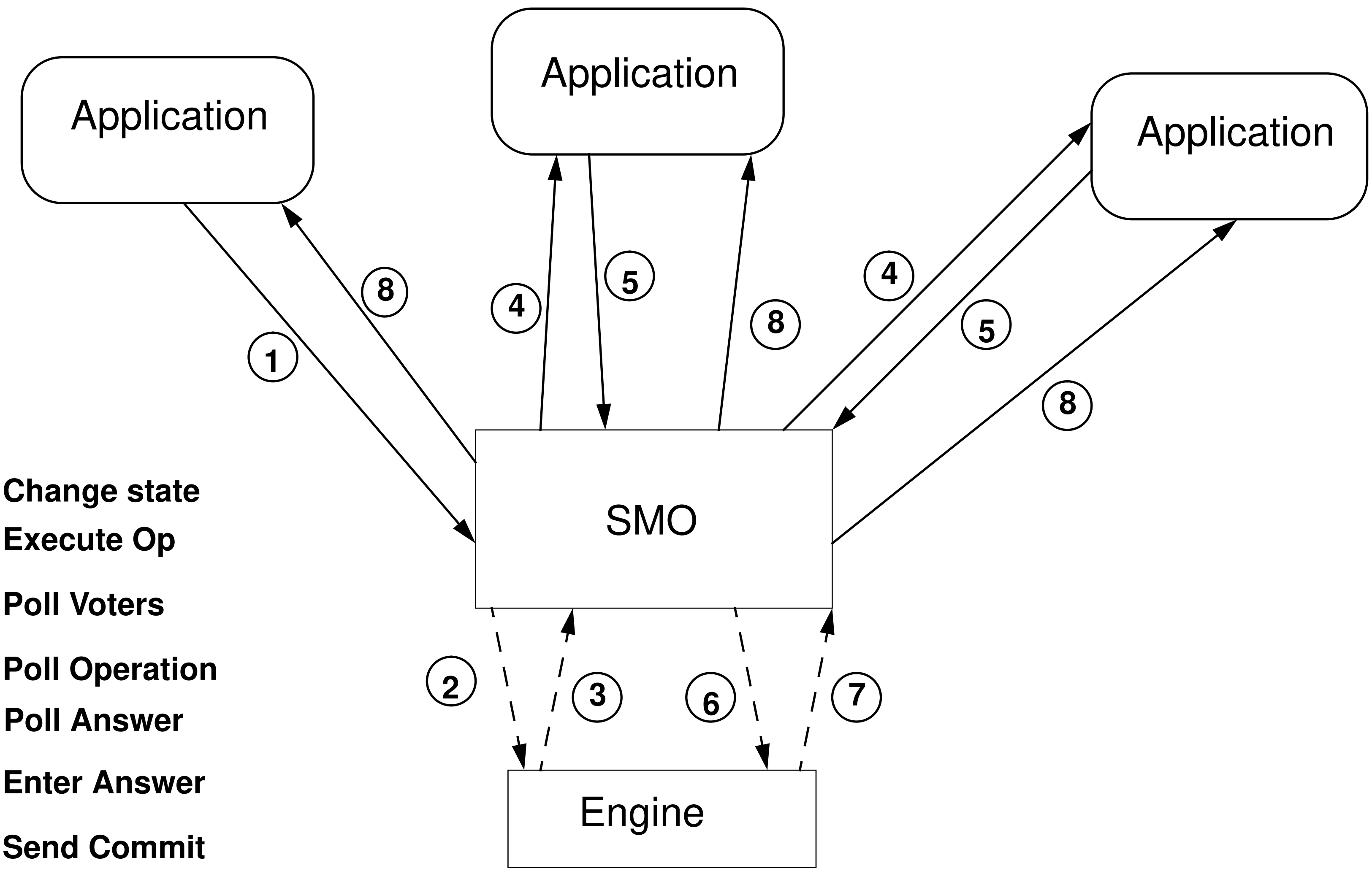
OPERATION SET : session name ; initiator ; <Operation List>



Centralized Model



Flow of Control



1 - Change state

2 - Execute Op

3 - Poll Voters

4 - Poll Operation

5 - Poll Answer

6 - Enter Answer

7 - Send Commit

8 - Commit Op

General Implementation Issues

- Special consideration of member operations

  --- Adding and Deleting members
  --- Member exit

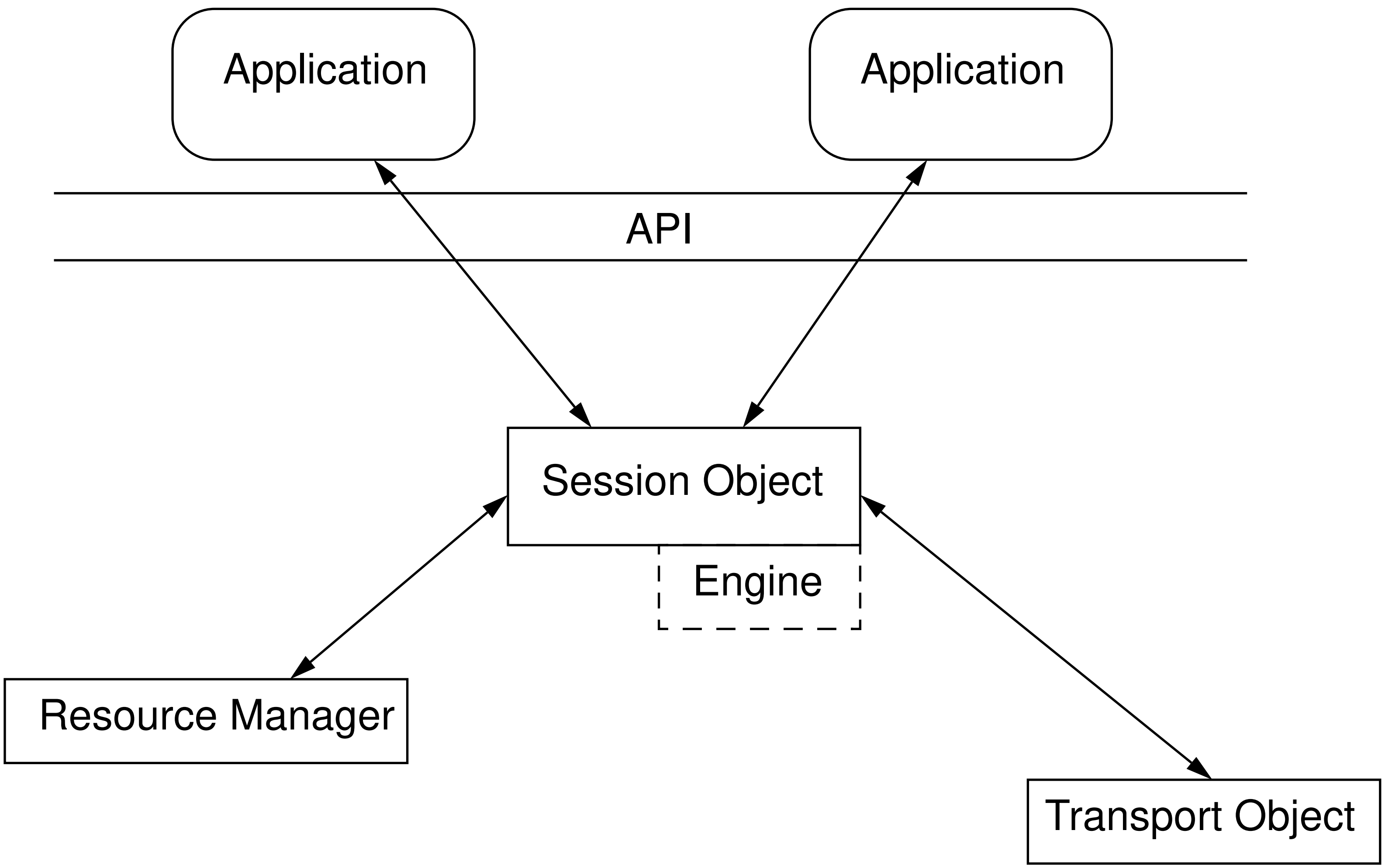
- Limited specification of member groups
- Granularity of rules

  --- Rule per variable value
  --- Limited types of voting

- Logical combination of voting

  --- Rules in operation, logical OR
  --- Operations in set, logical AND

Touring Machine



Touring Machine Session

Sample Session Create message:

(sessionCreate 9876 “bob:app:s1” 345679 (addCon “vCon” “video”)
(addSource “vCon” “bob:app:camera” “joe:app:camera”)
(addSink “vCon” “bob:app:monitor” “joe:app:monitor”))

(sessionChange 234 “bob:app:s1” 56677 (delClient “joe:app”))

Engine Translation:

Session : “bob:app:s1” ; unanimous ; <Member List> ; <Session Variables>

Member List : “MemberList” ; “Session” ; * ; * ;
{add ; none ; agree ; (group:M) ; (target) }
{delete ; none ; agree ; (group:M) ; (target) }

SET: “bob:app:s1” ; “bob:app” ; [“MemberList” ; “Session” ; delete ; none ; “joe:app”]

Distributed Model

Consistency Designations

C = Critical D = Delta E = Eventual N = None

X = C & E Y = D & E Z = C & D

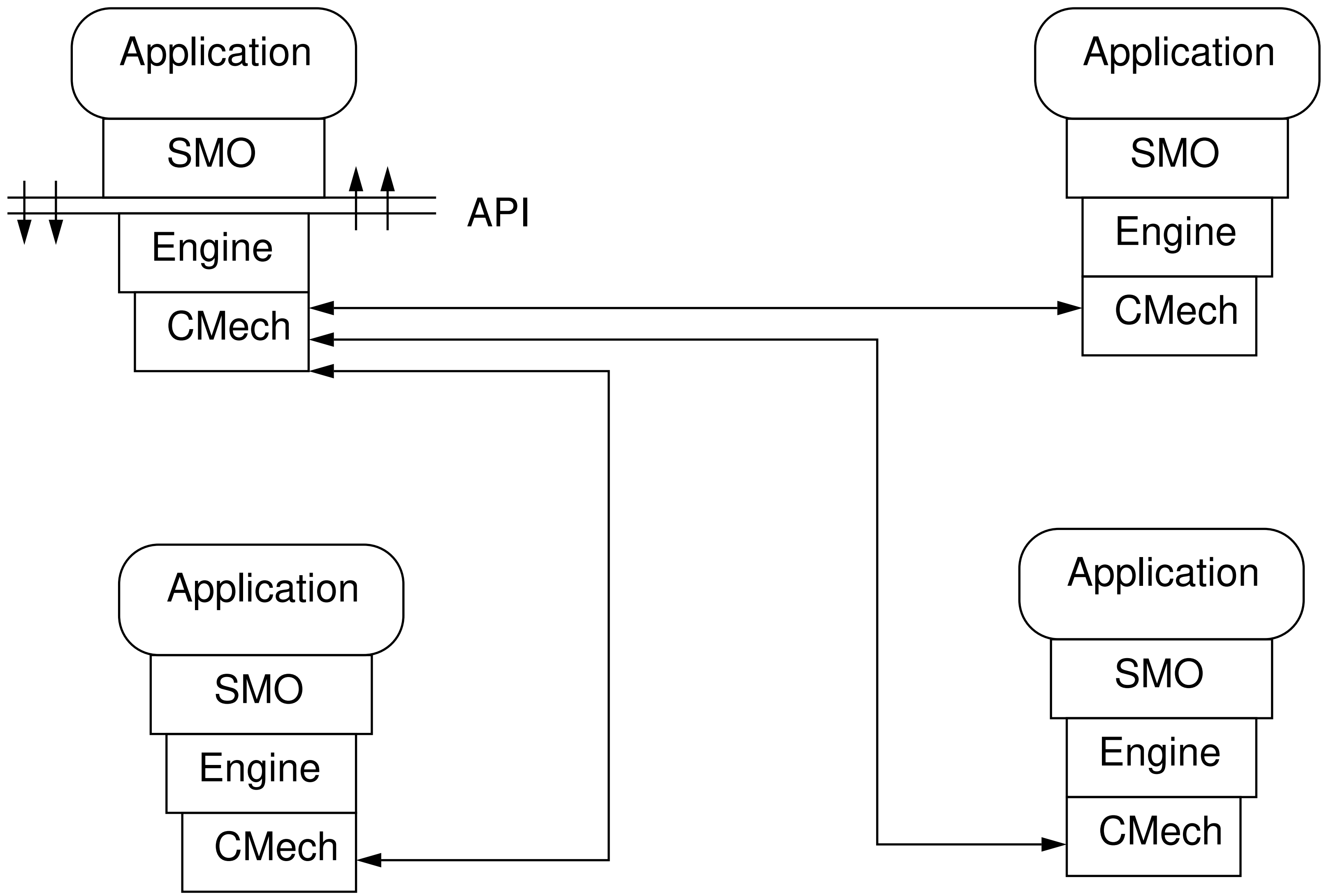
Members can be C, E, N, X Variables can be C, E, N, X, Y, Z

Conditions

1) Eventual members will eventually agree on the Eventual state

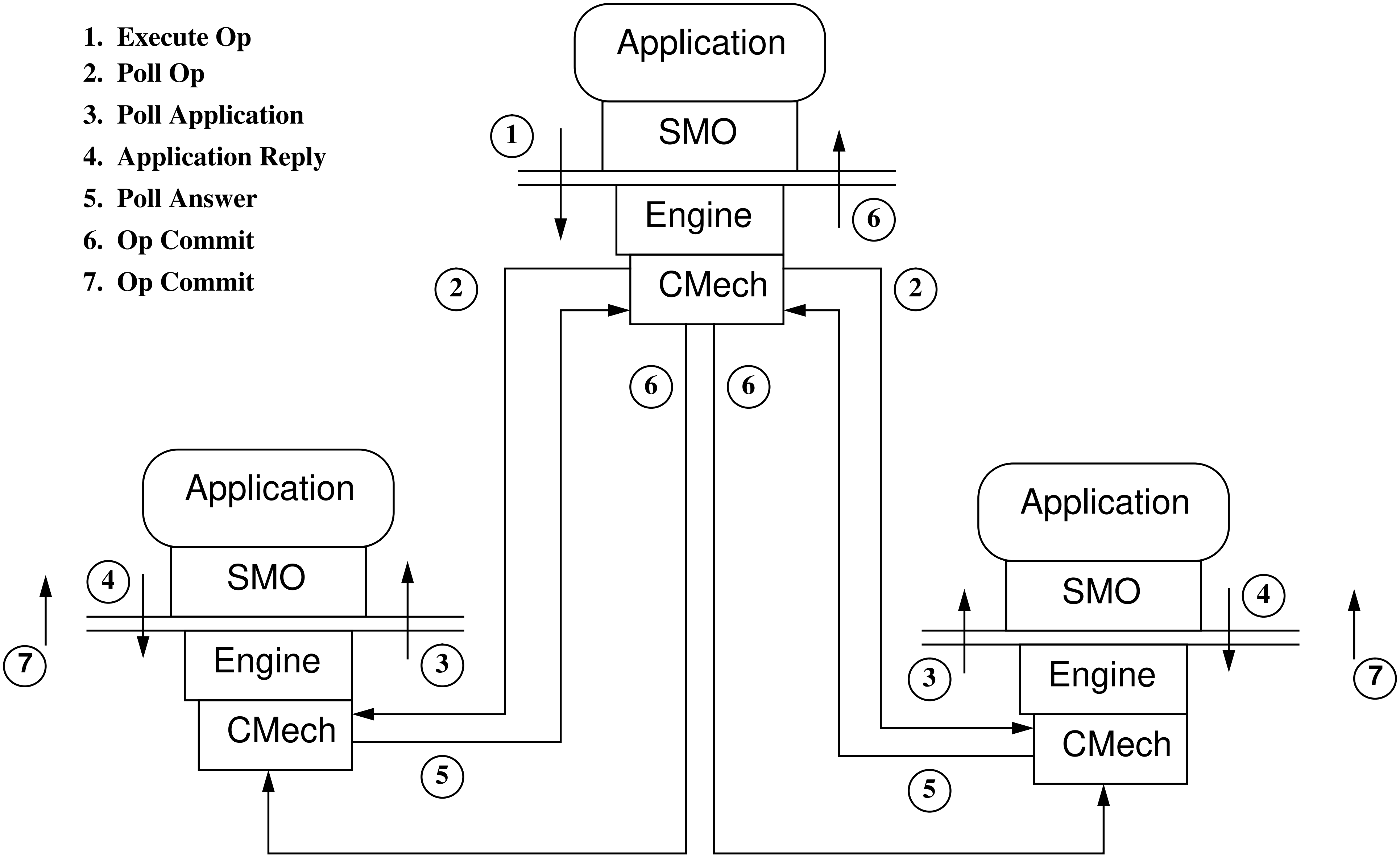
2) Critical members have a consistent view of the Critical state when voting on changes to the Delta state (requires locking)

Distributed Model



Flow of Control

1. Execute Op
2. Poll Op
3. Poll Application
4. Application Reply
5. Poll Answer
6. Op Commit
7. Op Commit



Implementation Issues

- Consistency designations set at variable not operation
- Locking done on entire critical state
- Mechanisms dependent on reliable control messaging

Conferencing Generality

Systems which are general try to express various types of objects in terms of a common base.

- End-system heterogeneity
- Different call models
- Session policies

End-System Heterogeneity

- Touring Machine 3

  --- policies defined in end applications

- IBM Distributed Computing Environment (DiCE)

  --- homogenous hardware & symmetric participation
  --- allows interaction at application level of detail

- Conference Control Channel Protocol (CCCP)

  --- defines fundamentals of conferencing
  --- Application, Member, Floor Control, Network

- These systems translate application to conferencing basics

- Engine envelopes description in control basics

Call Models

- A call model consists of operation stages or states
- Different systems execute unique call models
- Example: CCP and TM 2
- CCCP tries to solve this with finite state machines
- Engine is not responsible for this type of generality

Control Policy Generality

- Most systems have a small set of policy parameters
  - **--- CCP has specific flags (session, notification, etc.)**
- The Engine allows a wide range of policies within bounds

Question: What is considered sufficient? What is the necessary set of policies?

Question: Is control policy generality useful? What advantages does it have over lightweight, specific applications?

Dynamicity

Dynamicity refers to the ways in which a session may be modified while a conference is active.

- Changing rules
- Changing predefined modes of access
- Tearing down and re-creation of session

The Engine currently only supports new sessions for new policies.

Scalability

Scalability in terms of geographic area and number of members is the motivation for relaxing consistency conditions.

- Consistency control is easy for small conferences
  - **--- Eveything is treated as critical**
  - **--- Operations easily serialized**
- Relaxed conditions saves on communication
- Relaxed conditions allow more concurrency

Research Status

- Code for centralized model available
- User interface to centralized model based on Tcl / Tk
- Code for distributed model available
- Code has only been lightly tested
- Research findings will be written up in the thesis