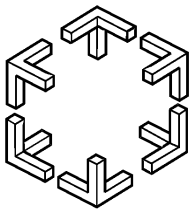
Distributed Computing and Systems
Chalmers university of technology



Lab 1
Bulletin Board System

Andreas Larsson

2011-01-21

Schedule

- The Ensemble system
  - Introduction
  - Architecture and Protocols
  - How does Ensemble achieve the group communication properties ?
- The Bulletin Board System

The Ensemble System

- A library of protocols that support group communication.
- Ensemble Provides
  - Group membership service
  - Reliable communication
  - Failure detector
  - Secure communication

Terminology

- Deliver a message
  - Send it upwards in the stack
    - From ensemble to the program that uses ensemble
    - Between layers within ensemble

Group membership service

- Endpoints
  - Abstraction for a communicating entity
  - Normally one per process
- Groups
  - Corresponds to a set of endpoints that communicates
  - Just a ***name*** for endpoints to use
- Views
  - A snapshot of the group membership at a specified point
    - May change from time to time
  - **Maintaining membership**

Reliable communication

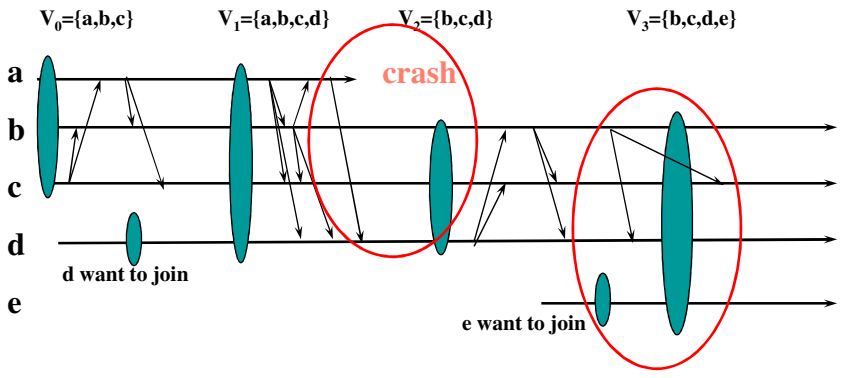
- Multicast communication
  - Messages are delivered by all group members
    - in the current view of the sender.
  - Possibly based on IP-multicast
- Point-to-Point communication
- Properties:
  - Virtual synchrony
  - Stability
  - Ordering

Virtual synchrony

- AKA: View-synchronous group communication

- Integrity
  - A correct process delivers a message at most once.
- Validity
  - A message from a correct process will be delivered eventually by that process
- Agreement
  - A message delivered by one correct process will be delivered by all correct processes

Virtual Synchrony
Examples of trouble

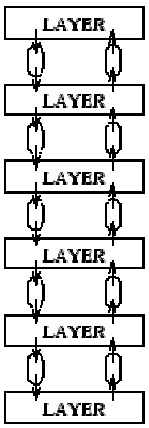


Schedule

- The Ensemble system
  - Introduction
  - Architecture & Protocols
  - How does Ensemble achieve the group communication properties ?
- The Bulletin Board System

Infrastructure

- Layered protocol architecture
  - All features are implemented as micro-protocols/layers
  - A stack/combination ~ a high-level protocol
- A new stack is created for a new configuration at each endpoint
- Ability to change the group protocol on the fly



Layers

- Layers are implemented as a set of callbacks that handle events passed to them.
  - Each layer gives the system 2 callbacks to handle events from its adjacent layers
  - Layers use 2 callbacks of its adjacent layers for passing events.
- Each instance of a layer maintain a private *local state.*

Stacks

- Combinations of layers that work together to provide high-level protocols
- Stack creation:
  - A new protocol stack is created at each endpoint of a group whenever the configuration (e.g. the view) of the group changes.
  - *All* endpoint in the same partition receive the same *ViewState* record to create their stack:
    - select appropriate layers according to the *ViewState*
    - create a new local state for each layer
    - compose the protocol layers
    - connect to the network

Schedule

- The Ensemble system
  - Introduction
  - Architecture & Protocols
  - How does Ensemble achieve the group communication properties ?
- The Bulletin Board System

The basic stack

- Each group has a *leader* for the membership protocol.

Layers	Functionality
Gmp	Membership algorithm (7 layers)
Slander	Failure suspicion sharing
Synch	Block during membership change
Top_appl	Interface to the application
Sequencer	Total ordering
Suspect	Failure detector
Stable	Stability detection
Mnak	Reliable fifo
Bottom	Interface to the network

Failure detector

- Suspect layer:
  - Regularly ping other members to check for suspected failures
  - Protocol:
    - If (#unacknowledged Ping messages for a member > threshold)
      send a Suspect event down
- Slander layer:
  - Share suspicions between members of a partition
    - The leader is informed so that faulty members are removed, even if the leader does not detect the failures.
  - Protocol:
    - The protocol multicasts slander messages to other members whenever receiving a new Suspect event

Stability

- Stable layer:
  - Track the stability of multicast messages
  - Protocol:
    - Maintain Acks[N][N] by unreliable multicast:
      - Acks[s][t]: #(*s*' messages) that *t* has acknowledged
      - Stability vector
        StblVct = {(minimum of row s): ∀s}
      - *NumCast* vector
        NumCast = {(maximum of row s): ∀s}
    - Occasionally, recompute *StblVct and NumCast*, then send them down in a Stable event.

Reliable multicast

- Mnak layer:
  - Implements a reliable FIFO-ordered multicast protocol
    - Messages from live members are delivered reliably
    - Messages from faulty members are retransmitted by live members
  - Protocol:
    - Keep a record of all multicast messages to retransmit on demand
    - Use Stable event from Stable layer:
      - *StblVct* vector is used for garbage collection
      - *NumCast* vector gives an indication to lost messages ⇒ recover them

Ordering property

- Sequencer layer:
  - Provide total ordering
  - Protocol:
    - Members buffer all messages received from below in a local buffer
    - The leader periodically multicasts an *ordering message*
    - Members deliver the buffered messages according to the leader's instructions
- See Causal layer for causal ordering

Maintaining membership (1)

- Handle Failure by splitting a group into several subgroups: 1 primary and many non-primary (partitionable)
- Protocol:
  - Each member keeps a list of suspected members via Suspect layer
  - A member shares its suspicions via Slander layer
  - View leader *l:*
    - collect all suspicions
    - reliably multicast a *fail($p_{i0},\ldots,p_{ik}$)* message
    - synchronize the view via Synch layer
    - Install a new view without $p_{i0},\ldots,p_{ik}$
  - A new leader is elected for the view without leader
    - If p_k in view V_1 suspects that all lower ranked members are faulty, it elects itself as leader and does like *l.*
    - A member that agrees with p_k, continues with p_k to the new view V_2 with p_k as the leader.
    - A member that disagrees with p_k, suspects p_k.

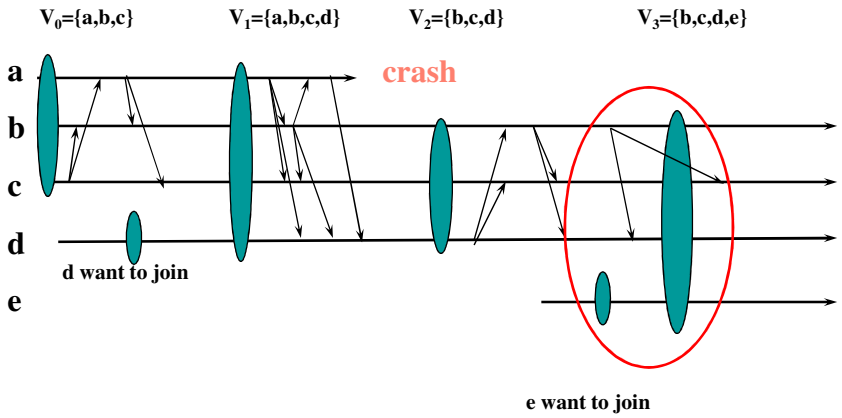
Maintaining membership (2)

- Recover failure by merging non-primary subgroups to the primary subgroup
- Protocol:

  l: local leader, *r*: remote leader

  1. *l* synchronizes its view
  2. *l* sends a merge request to r
  3. *r* synchronizes its view
  4. *r* installs a new view with its mergers and sends the view to *l*
  5. *l* installs the new view in its subgroup

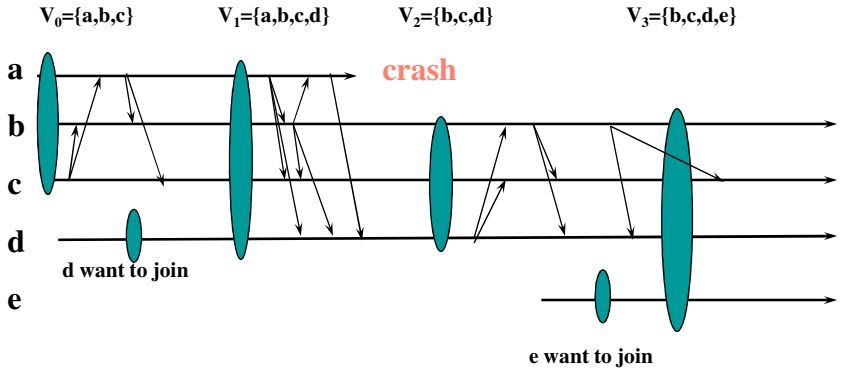
Join Group



Virtual synchrony

- Achieved by a simple leader-based protocol:
  - Idea:
    - Before a membership change from V_1 to V_2 all messages in V_1 must become stable
  - Protocol: before any membership change
    - The leader activates the Synch protocol $\Rightarrow$ the set, M_{V1}, of messages needed to deliver in V_1 is bounded.
    - The leader waits until live members agree on M_{V1} via sending negative acknowledgements and recovering lost messages (i.e. *StblVct = NumCast*)

Virtual Synchrony



Schedule

- The Ensemble system
  - Introduction
  - Architecture and Protocols
  - How does Ensemble achieve the group communication properties ?
- The Bulletin Board System

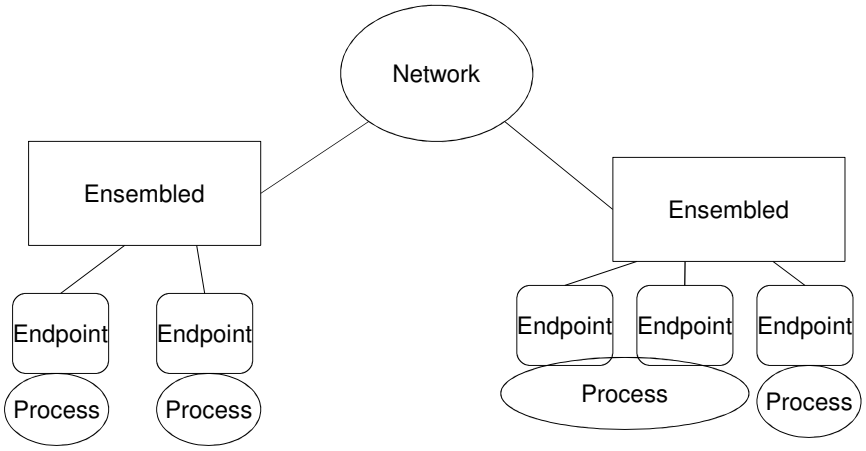
The Bulletin Board System

- Bulletin board with messages
  - Subject, ID, sender, time
  - Arriving messages shall be displayed immediately
  - No agreed order of messages is needed
  - **But:** Replies should always be after their parent
    - Take advantage of ensemble to do this
- Peer to peer application
  - The bulletin board is shared
  - No server that keeps the messages
  - Stability while endpoints join and leaves
- The application should be able to stand the loss of any client
  - Except last one
  - Warn client when it is the only one left

Ensembl*ed*

- An **`ensembled`** process needs to run at each computer
  - If none is running at your computer run /chalmers/users/larandr/ensemble/ensembled
  - Already runs on: remote{1,2,3,4,5}.student.chalmers.se
- Ensembled is providing the ensemble service
  - Ensemble servers are not centralized servers
  - The server serves one host
  - The servers connect to each other and form the network

Map over the network



Get to know the system

- Look at the documentation:
  - Ensemble tutorial
    - Chapter 8 for the java interface
    - 5.8 for quick view of properties/layers
  - Ensemble reference manual
    - Chapter 11 for more details on the layers
    - For additional information
  - client/java/ensemble/JoinOps.java
    - Under /chalmers/users/larandr/ensemble/
    - To see how to select layers
    - Do not change JoinOps itself.
- Understand the example program
  - Get the Talk app from the course page
  - Get it to run and figure out how it works