

Lab 2 Group Communication

Farnaz Moradi

Based on slides by Andreas Larsson

Overview

- Introduction to group communication
- Desired group communication
- Multicast communication
- Group membership service



Coordination in distributed systems

- Coordination is needed by distributed systems but hard to achieve:
 - Events happen concurrently
 - Communication links are not reliable
 - Computers can crash
 - New nodes can join the systems
 - Asynchronous environments
- ⇒ Need of an efficient way to coordinate a group of processes



A Distributed System in war: Synchronous Example



A Distributed System in war: Reality



Group communication

- What is a group?
 - A number of processes which cooperate to provide a service.
 - An abstract identity to name a collection of processes.
- Group Communication
 - For coordination among processes of a group.



Who Needs Group Communication?

- Highly available servers (client-server)
- Database Replication
- Multimedia Conferencing
- Online Games
- Cluster management



...



High availability



DSII: Group Comm.



Online Game



• Fault-tolerance, Order

Distributed Computing and Systems Chalmers university of technology



Different Communication Methods

- Unicast
 - Point-to-Point Communication
 - Multiple copies are sent.
- Broadcast
 - One-to-All Communication
 - Abuse of Network Bandwidth
- Multicast
 - One-to-multiple Communication



DSII: Group Comm.

Overview

- Introduction to group communication
- Desired group communication
- Multicast communication
- Group membership service

Distributed Computing and Systems Chalmers university of technology

Desired Group Communication

- Name Abstraction
- Efficiency ⇒Multicast
- Delivery Guarantees ⇒Reliability, Ordering
- Dynamic Membership ⇒Group membership service



Properties of Communication

- Ordering
 - Total ordering, causal ordering
- Failure behavior
- Reliability
 - Validity, integrity, agreement



Properties of Group

- Name of group
- Addresses of group members
- Dynamic group membership
- Options:
 - Peer group or client-server group
 - Closed or Open Group





Peer Group

- All the members are equal.
- All the members send messages to the group.
- All the members receive all the messages.



Client-Server Group

- Replicated servers.
- Clients do not care which server answers.





Overview

- Introduction to group communication
- Desired group communication
- Multicast communication
- Group membership service



Multicast communication

- Use network hardware support for broadcast or multicast when it is available.
- Send messages over a distribution tree.
- Minimize the time and bandwidth utilization



Reliability

Correct processes: those that never fail.

Integrity

A correct process delivers a message at most once.

Validity

A message from a *correct* process will be *delivered* by the process eventually.

Agreement

A message *delivered* by a *correct* process will be *delivered* by all other *correct* processes in the group.

\Rightarrow Validity + Agreement = Liveness



DSII: Group Comm.

Ordering

Assumptions: a process belongs to at most one group.

- **FIFO**
 - if $m_p \rightarrow m'_p$, all correct processes that deliver m'_p will deliver m_p (that is from the same sender) before *m*'_{*p*}.
- Causal
 - if $m \rightarrow m'$, all correct processes that deliver m' will deliver m before *m*'.
- Total
 - if a correct process delivers *m* before *m'*, all other correct processes that deliver m' will deliver *m* before *m*'.



Examples

- Assumption:
 - Reliable one-to-one *send* operation (e.g. TCP)
- Basic multicast
 - Requirement:
 - All correct processes will eventually deliver the message from a *correct* sender.
 - Implementation:
 - **B-multicast(g, m)**: ∀p ∈ g: **send**(p, m);
 - On receive(m) at p: B-deliver(m) at p.

 \Rightarrow Properties: integrity, validity.



Basic multicast: Agreement?





Examples (cont.)

- Reliable multicast
 - Requirements: integrity, validity, agreement
 - Implementation:
 - *Received := {};*
 - R-multicast(g, m) at process p: B-multicast(g, m);
 - On B-deliver(m) at process p from process q if(m ∉ Received)

```
Received := Received \cup \{m\};
if( q \neq p) B-multicast(g, m);
R-deliver(m);
```

end if

⇒ Inefficient: each message is sent O(|g|) times to each process



Reliable multicast



Distributed Computing and Systems Chalmers university of technology

Examples (cont.)

- FIFO-ordered multicast:
 - Assumption:
 - a process belongs to at most one group.
 - Implementation:
 - Local variables at $p: S_p = 1, R_p[|g|] = \{0\};$
 - FO-multicast(g, m) at p: B-multicast(g, <m, S_p>); S_p++;
 On B-deliver(<m, S>) at p from q: if(S = R_p[q] + 1) FO-deliver(m); R_p[q] := S; else if(S > R_p[q] + 1) place <m, S> in the queue until S = R_p[q] + 1; FO-deliver(m); R_p[q] := S; end if

Your task: totally ordered multicasts.



Overview

- Introduction to group communication
- Desired group communication
- Multicast communication
- Group membership service



Group membership service

- Four tasks:
 - Interface for group membership changes
 - Failure detector
 - Membership change notification
 - Group address expansion



- Group partition:
 - Primary-partition one partition only
 - Partitionable many partiations at once

Group views

- Group views:
 - Lists of the current ordered group members
 - A new one is generated when processes join or leave/fail.
- View delivery
 - When a member is notified of a membership change
 - Requirements
 - Order
 - if p delivers $v(g) \rightarrow v'(g)$, no other process delivers $v'(g) \rightarrow v(g)$.
 - Integrity
 - if p delivers v(g), $p \in v(g)$.
 - Non-triviality
 - if q joins a group and becomes indefinitely reachable from p, eventually q is always in the view p delivers.



View-synchronous group comm.

- Extend the reliable multicast semantics with group views.
 - Agreement
 - Correct processes deliver the same set of messages *in any given view*
 - Validity
 - Correct processes always deliver the messages they send.
 - $p \in v_0(g)$ does not deliver m in $v_0(g) \Rightarrow p \notin v_1(g)$ for processes that deliver m.
 - Integrity



Example

- Ensemble: reliable group communication toolkit
 - Previous talk



IP-multicast

- IP: 224.0.0.1 239.255.255.255
- Multicast:

• Yes:

efficiency

- No:
 - Reliability
 - Ordering

• Group membership service:

• Yes:

- Interface for group membership change
- Group address expansion

• No:

- Failure detector
- Membership change notification

References

- *Distributed Systems: Concepts and Design* by G. Coulouris et al., ISBN 0-201-61918-0
 - Section 4.5 Group Communication
 - Section 11.4 Multicast Communication
 - Section 14.2.2 Group Communication

Lab 2: Construct a reliable and ordered multicast

- Reliable
 - Integrity, Validity, Agreement
- Ordered
 - All processes should agree on the order of all received messages.
 - Total and causal order.
- <u>No</u> membership service
 - Processes can still crash though!
- Use the *mcgui* package





Distributed Computing and Systems Chalmers university of technology

DSII: Group Comm.

The interface

- Provide:
 - cast

- Use:
 - deliver
 - debug

- basicreceive
- basicpeerdown

basicsend



Implementation

- Cope with process crashs, not just normal leave
- Don't change the mcgui package
 - Your code will be checked using the original package
- Use the stress test to test your program
- Don't add waits to your code to mask problems in the code
 - The logic of your program will be checked
- Don't start threads
 - Java synchronization is handled in the package
- Don't use IP multicast
 - extend the *Multicaster* class



Lab Report

- How does the protocol satisfy Integrity
 - including when processes can crash
- How does the protocol satisfy Validity
 - including when processes can crash
- How does the protocol satisfy Agreement
 - including when processes can crash
- How does the protocol satisfy the ordering requirements?
 - Describe any used algorithms.
- How does the protocol deal with crashing processes?
 - Describe any used algorithms.