Script **generated by TTT**

Title: Distributed_Applications (11.06.2013)

Date: Tue Jun 11 15:46:58 CEST 2013

Duration: 15:37 min

Pages: 11

Taxonomy of multicast

Multicast messages for constructing distributed systems based on group communication;

different multicast communication semantics

Multicast classes

Relationship between multicast classes

Multicasting can be realized by using IP multicast which is built on top of the Internet protocol IP.

Java API provides a datagram interface to IP multicast through the class `MulticastSocket` .

Generated by Targeteam

Multicast classes

Depending on the message delivery guarantee, five classes of multicast services can be distinguished.

1. ***unreliable multicast*** : an attempt is made to transmit the message to all members without acknowledgement; at-most-once semantics with respect to available members; message ordering is not guaranteed.
2. ***reliable multicast*** : the system transmits the messages according to "best-effort", i.e. the "at-least-once" semantics is applied.

   B-multicast primitive: guarantees that a correct process will eventually deliver the message as long as the multicaster does not crash.

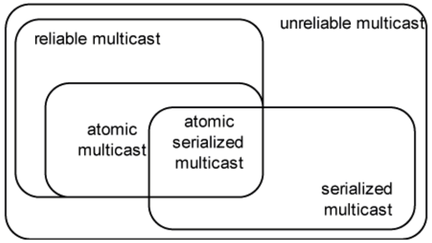
   B-deliver primitive: corresponding primitive when a message is received.
3. ***serialized multicast*** : consistent sequence for message delivery; distinction between

   totally ordered

   causally ordered (i.e. virtually synchronous)
4. ***atomic multicast*** : a reliable multicast which guarantees that either all operational group members receive a message, or none of them do.
5. ***atomic, serialized multicast*** : atomic message delivery with consistent delivery sequence

Generated by Targeteam

Relationship between multicast classes



Generated by Targeteam

Taxonomy of multicast

Multicast messages for constructing distributed systems based on group communication;

- different multicast communication semantics

Multicast classes

Relationship between multicast classes

Multicasting can be realized by using IP multicast which is built on top of the Internet protocol IP.

- Java API provides a datagram interface to IP multicast through the class `MulticastSocket` .

Group communication in ISIS

The ISIS system developed at Cornell University is a framework for reliable distributed computing based upon process groups. It specifically supports group communication. Successor of ISIS was **Horus** .

- ISIS is a toolkit whose basic functions include process group management and ordered multicast primitives for communication with the members of the process group.
  - abcast: totally ordered multicast.
  - cbcast: causally ordered multicast.

abcast protocol

cbcast protocol

Group communication

Introduction

- Group communication facilities the interaction between groups of processes.
  - **Motivation**
  - **Important issues**
  - **Conventional approaches**

Groups of components

Management of groups

Message dissemination

Message delivery

Taxonomy of multicast

Group communication in ISIS

JGroups

Group communication in ISIS

The ISIS system developed at Cornell University is a framework for reliable distributed computing based upon process groups. It specifically supports group communication. Successor of ISIS was **Horus** .

- ISIS is a toolkit whose basic functions include process group management and ordered multicast primitives for communication with the members of the process group.
  - abcast: totally ordered multicast.
  - cbcast: causally ordered multicast.

abcast protocol

cbcast protocol

Generated by Targeteam

atomic broadcast supports a total ordering for message delivery, i.e. all messages to the group G are delivered to all group members of G in the same sequence.

abcast realizes a serialized multicast

abcast is based on a ***2-phase commit*** protocol; message serialization is supported by a distributed algorithm and logical timestamps.

Phase 1

Sender S sends the message N with logical timestamp T_S (N) to all group members of G (e.g. by multicast).

Each g ∈ G determines a new logical timestamp T_g (N) for the received message N and returns it to S.

Phase 2

S determines a new logical timestamp for N; it is derived from all proposed timestamps T_g (N) of the group members g.

$T_{S,new}$ (N) = max (T_g (N)) + j/|G| , with j being a unique identifier of sender S.

S sends a commit to all g ∈ G with $T_{S,new}$ (N).

Each g ∈ G delivers the message according to the logical timestamp to its associated application process.

causal broadcast guarantees the correct sequence of message delivery for causally related messages.

Concurrent messages can be delivered in any sequence; this approach minimizes message delay.

Let n be the number of group members of G. Each g ∈ G has a unique number of {1, ..., n} and a state vector z which stores information about the received group messages.

The state vector represents a vector clock .

Each message N of sender S has a unique number; message numbers are linearly ordered with increasing numbers.

Let j be a group member of the group G.

the state vector $z_j = (z_{ji})_{i \in \{1,...,n\}}$ specifies the number of messages received in sequence from group member i.

Example: z_{ji} = k; k is the number of the last message sent by member i ∈ G and received in correct sequence by the group member j.

at group initialization all state vectors are reset (all components are 0).

Sending a message N; j ∈ G sends a message to all other group members.

$z_{jj} := z_{jj} + 1$; the current state vector is appended to N and sent to all group members.

Receiving a message N sent by member i ∈ G.

Message N contains state vector z_i . There are two conditions for delivery of N to the application process of j

(C 1): $z_{ji} = z_{ii} - 1$.

(C 2): $\forall$ k ≠ i: $z_{ik} \leq z_{jk}$.

Introduction

Group communication facilities the interaction between groups of processes.