Programming assignment 2 – Broadcast

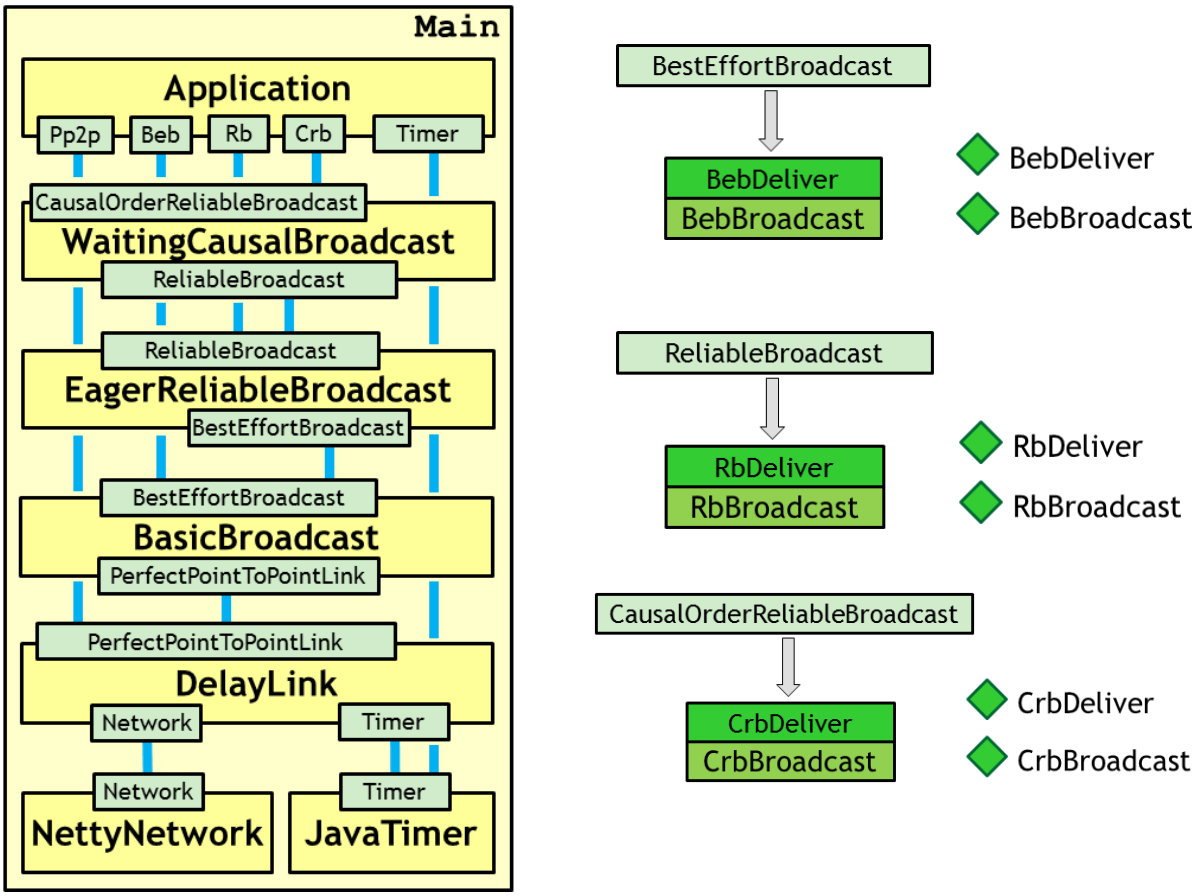
Introduction

In this programming assignment you shall implement three components: Basic Broadcast (provides the Best Effort Broadcast service), Eager Reliable Broadcast (provides Reliable Broadcast) and Waiting Causal Broadcast (provides Causal Order Reliable Broadcast). You shall use algorithm 3.1, 3.3 and 3.15 from the textbook. These algorithms are also reproduced at the end of this document, but please note that the Eager Reliable Broadcast algorithm in this document is more detailed.

Installation

Download id2203-ass2-broadcast.zip from the course website, unpack and import into Eclipse in the same way as for previous assignments.

Architecture



Code to write

There are three files where you should implement the components: `BasicBroadcast.java`, `EagerRb.java` and `WaitingCrb.java`. You will also have to add files for messages and events as you see fit.

Various notes

- You need to add the following messages (events):
  - In BasicBroadcast component: `BebDataMessage` ⊆ `Pp2pDeliver`

    - In EagerReliableBroadcast component: `RbDataMessage` ⊆ `BebDeliver`
    - In WaitingCausalBroadcast component: `CrbDataMessage` ⊆ `RbDeliver`
- Remember that events that are to be sent over the network must be serializable and have a unique `serialVersionUID` field.
- If you want to add an object of some class `C` to a `HashSet<C>` then `C` must implement the `equals()` and `hashCode()` methods. If `o1.equals(o2)` for two objects `o1` and `o2`, then `o1.hashCode()==o2.hashCode()` must hold.
- If you instead use `TreeSet<C>`, then `C` must implement the `Comparable<C>` interface and the `compareTo()` method must be consistent with `equals()`.

Link delay

The link delay from one process to another is given by a normal distribution, with the mean value specified in the topology in `Executor.java`. The standard deviation of the normal distribution is specified by the value sigma in the `DelayLinkInit` object provided when the `DelayLink` component is created. At line 76 of `Main.java` the sigma value is set to 1000 ms. The variance in link delay implies that two messages that are sent close together in time may arrive in reverse order.

Exploration

As can be seen at lines 120, 131 and 141 of `Application.java`, when the Application component in process 2 delivers a message broadcasted by process 1 then process 2 immediately broadcasts a message composed of the delivered string concatenated with the string "-res".

At lines 42-44 of `Executor.java` there are different strings of commands for the first process. Only one of these lines should be uncommented at any time that the program is run.

You shall do the following:

- Make lines 42 and 43 commented, and line 44 uncommented (in `Executor.java`), and run the program. The following commands will be executed by process 1: causal reliable broadcast message "1", sleep 10 ms, causal reliable broadcast message "2", sleep 10 ms, causal reliable broadcast message "3", sleep for 3000 ms, and then exit.
    - You should verify that the properties of CausalOrderReliableBroadcast are satisfied (Validity, No duplication, No creation, Agreement, Causal delivery).
- Run instead with the commands at line 42 (comment out line 43 and 44). A similar sequence of commands will be executed, but with best effort broadcast instead of causal reliable broadcast.
    - Verify that the properties of BestEffortBroadcast are satisfied.
- Run with the commands on line 43, which use reliable broadcast.
    - Verify that the properties of ReliableBroadcast are satisfied.

Automatic correction

When everything is working you run the `AutomaticCorrection.java` file to test the component and submit the assignment to the http://cloud7.sics.se:11700/ server. Remember to change the email and password strings before running.

Algorithm 1 Basic Broadcast

Implements:

BestEffortBroadcast, **instance** *beb*.

Uses:

PerfectPointToPointLinks, **instance** *pp2p*.

1: **upon event** $\langle\ beb, Broadcast \mid m\ \rangle$ **do**
2: **for all** $q \in \Pi$ **do**
3: **trigger** $\langle\ pp2p, Send \mid q, m\ \rangle$;

4: **upon event** $\langle\ pp2p, Deliver \mid p, m\ \rangle$ **do**
5: **trigger** $\langle\ beb, Deliver \mid p, m\ \rangle$;

Algorithm 2 Eager Reliable Broadcast

Implements:

ReliableBroadcast, **instance** *rb*.

Uses:

BestEffortBroadcast, **instance** *beb*.

1: **upon event** $\langle\ rb, Init\ \rangle$ **do**
2: $delivered := \emptyset$;
3: $seqnum := 0$;

4: **upon event** $\langle\ rb, Broadcast \mid m\ \rangle$ **do**
5: $seqnum := seqnum + 1$;
6: **trigger** $\langle\ beb, Broadcast \mid [\text{DATA}, seqnum, self, m]\ \rangle$;

7: **upon event** $\langle\ beb, Deliver \mid p, [\text{DATA}, sn, s, m]\ \rangle$ **do**
8: **if** $(sn, rank(s)) \notin delivered$ **then**
9: $delivered := delivered \cup \{(sn, rank(s))\}$;
10: **trigger** $\langle\ rb, Deliver \mid s, m\ \rangle$;
11: **trigger** $\langle\ beb, Broadcast \mid [\text{DATA}, sn, s, m]\ \rangle$;

Algorithm 3 Waiting Causal Broadcast

Implements:

CausalOrderReliableBroadcast, **instance** *crb*.

Uses:

ReliableBroadcast, **instance** *rb*.

1: **upon event** $\langle\ crb, Init\ \rangle$ **do**
2: $V := [0]^N$;
3: $lsn := 0$;
4: $pending := \emptyset$;

5: **upon event** $\langle\ crb, Broadcast \mid m\ \rangle$ **do**
6: $W := V$;
7: $W[rank(self)] := lsn$;
8: $lsn := lsn + 1$;
9: **trigger** $\langle\ rb, Broadcast \mid [\text{DATA}, W, m]\ \rangle$;

10: **upon event** $\langle\ rb, Deliver \mid p, [\text{DATA}, W, m]\ \rangle$ **do**
11: $pending := pending \cup \{(p, W, m)\}$;
12: **while exists** $(p', W', m') \in pending$ such that $W' \leq V$ **do**
13: $pending := pending \setminus \{(p', W', m')\}$;
14: $V[rank(p')] := V[rank(p')] + 1$;
15: **trigger** $\langle\ crb, Deliver \mid p', m'\ \rangle$;