|  |
| --- |
| Software Engineering of Distributed Systems, KTH |
| Distributed Systems Advanced Homework 3 |
| Implementation of Reliable Broadcast Component, Unreliable Broadcast Component and Lazy Probabilistic Broadcast Component |

|  |
| --- |
| Shanbo Li and Sike Huang  2/19/2008 |

### Exercise 1. Modify Algorithm 3 such that it garbage collects the delivered set. Messages that no longer need to be maintained in the delivered set should be removed. Update your implementation of RB and describe the new algorithm in the report.

Answer:

We record the number of the same message which we received. If ***this number is greater than current correct nodes’ number*** then we affirmative that the initial source of this message has already crashed and all other nodes has forward this message to me. So I will never get this message again. And there is no need to store this message in deliver list to filter.

Principle:

A node will get a same message by no more than the number of current correct nodes plus one.

**Scenario 1:**

A node (p1) gets a message *m* from *m’*s initial node (p0).

When it detects p0 crashed. It will broadcast m to all lived nodes include himself. And others will eventually do the same thing. Then p1 gets m by the number of current correct nodes since p0 crashed. And p1 get the first m from p0, m’s initial node. So the number of m which p1 get is the number of current correct nodes plus one. P1 will never get m again, so it can do the garbage collection to remove m from delivered list.

|  |  |
| --- | --- |
| Scenario 2: (focus on p1) | |
| 1. Set of all node = {p0, p1, p2, p3, p4} | |
| 1. p0 sends m to all | |
| 1. p0 crashes and failed to send m to p1 | |
| 1. p2, p3, p4 get m | |
| 1. p2 detects p0 crashed and broadcast m to all nodes. | |
| 1. p1 gets m for the first time from p2 | (counter(m) = 1, correct.size =4 ) |
| 1. p3, p4 detect p0 crash and broadcast m to all nodes. | |
| 1. p1 gets m from p3, p4 | (counter(m) = 3, correct.size = 4) |
| 1. p3, p4 crash. | |
| 1. p2 crashes | |
| 1. p1 detects p2 crashed and broadcast m to all nodes. | |
| 1. p1 gets m from himself. | (counter(m) = 4, correct.size = 1) |
| 1. counter(m) > correct.size() do garbage collection(m) | |

The code for this algorithm is illustrated below:

|  |
| --- |
| Data structure: |
| private Map<SourceMessagePair, Integer> delivered; |

|  |
| --- |
| Algorithm |
| public void handleBebDeliverEvent(BebDeliverEvent bebDeliverEvent) {  …  delivered.put(sourceMessagePair, new Integer(delivered.get(sourceMessagePair) + 1)); |
|  |
| if (delivered.get(sourceMessagePair)>correct.size()) { |
| delivered.remove(sourceMessagePair); |
| System.out.println(""); |
| System.out.println("Garbage collect delivered list, remove message \"" |
| + sourceMessagePair.getMessage() + "\" from Node " |
| + sourceMessagePair.getSource().getId()); |
| }  …  } |

The output logs located in *\Logs\exercise1*

*Operation:*

* node 0:
  + send “a”
  + send “b”
  + send “c”
  + crash
* node 4:
  + crash ( before d(0) )
* node 3:
  + crash ( after d(4) )