Programming assignment 2

Lazy probabilistic broadcast

2013

Mahboobeh Abdal Mahmood Abadi & Mehran Nasseri

KTH

2/14/2013

# Table of Contents

[Table of Contents 2](#_Toc348593939)

[1 Exercise 1 3](#_Toc348593940)

[1.1 Question2 3](#_Toc348593941)

[2 Exercise 2 5](#_Toc348593942)

[2.1 Question 1 5](#_Toc348593943)

[2.2 Question 2 6](#_Toc348593944)

# Exercise 1

## Question2

* Fanout: a process that has not received a particular message from a particular sender selects k processes at random and sends them the retransmission message.
* Store threshold: Every process that delivers a message stores a copy of the message with probability α. The purpose of this approach is to distribute the load of storing messages for future retransmission among all processes.
* Max round: Each step may consist of receiving a retransmission message, from those who missed the message, and it is called a round of gossiping. The algorithm performs up to R rounds of gossiping for each retransmission request. check

The probability of delivering a message to all correct processes depends on fanout and on the reliability of the link. A higher fanout not only increases the probability of having the entire population infected but also decreases the number of rounds required to achieve this. Note also that the algorithm induces a significant amount of redundancy in the message exchanges: any given process may receive the same message many times.

However, increasing the fanout is costly. The higher the fanout, the higher the load imposed on each process and the amount of redundant information exchanged over the network. Note that there are runs of the algorithm where a transmitted message may not be delivered to all correct processes. For instance, a process that have not received the message directly from the sender select the set of k target processes and forward the retransmission message to them, in this case there is a risk of selecting those target processes, whose neither received the message nor store the message. Therefore, the higher store threshold is the higher probability of delivering a message to all correct processes.

This translates into the fact that the probability that some process never delivers the message is nonzero. However, by choosing large enough values of fanout and max rounds in relation to number of nodes and store threshold, this probability can be made arbitrarily small. On the other hand, increasing the fanout and max round will increase the network load, and increasing the store threshold will increase the memory consumption.

1. **No message is lost (all nodes deliver a broadcasted message m):**

Since the module uses delay drop link there is always possibility of losing a message. Even if we set all parameters to maximum value, there could be an execution in which the gossip request or/and replay get lost along the way. Therefore the only way is to set the lose rate to zero.

1. **A broadcasted message is lost in the unreliable broadcast but recovered by gossip for some node *p*:**

In the lazy probabilistic broadcast, we assume that all nodes are fully connected. To force the message being lost the loose rate must be set to a non-zero value. Whenever a node misses a message, it starts to gossip to pull the message. By increasing the fanout the probability of recovering message increases with a lower max round. In addition, probability of recovering lost message can be further improved by increasing the value of store threshold. We can conclude that:

* Highest value of fanout and threshold can guaranty highest probability of recovering lost message and decreases the number of rounds required to achieve this.
  + On the other hand, this increases the probability of redundancy of exchange information.
* The last message may never receives by some nodes, since according to the algorithm, the process will realize missing a massage if and only if it receive another message with higher sequence number than next[sn].

Topology characteristic and parameters value that lead to scenario ‘b’:

* fanout = 3;
* Store threshold = 0.7;
* Max-round = 2;
* Topology

for (int i = 1; i <= 6; i++) {

node(i, "127.0.0.1", 22220 + i);

}

defaultLinks(100, 0.5);

As our experiment shows, as we increase the Store threshold and fanout more nodes was able to retrieve the lost message.

1. **A broadcasted message is lost such that although it is stored on some node(s) in the network, a node *p* missed it in the unreliable broadcast and furthermore, *p* could not retrieve it via gossiping as well.**

To decrease probability of successful message recovery we decrease store-threshold to 0.4 and we decrease fanouts to two to make likelihood of gossip failure higher. In addition increasing the max rounds will decrease chance of failure of message recovery so we kept it unchanged.

* fanout = 2;
* Store threshold = 0.4f;
* Max-round = 2;
* Topology

for (int i = 1; i <= 6; i++) {

node(i, "127.0.0.1", 22220 + i);

}

defaultLinks(100, 0.5);

1. **A broadcasted message, after being delivered by some node(s) and missed by a node *p*, is completely lost such that *p* can never retrieve it through gossiping**.

This could be achieved simply by setting the value of store threshold to zero. However, the description of the assignment asks us to use none zero values for the parameters. Considering this, we can achieve the scenario by setting parameters to the minimum value. However, it is not possible to reduce the probability of retrieving a massage via gossiping to zero. Continuously minimizing the value of these parameters makes probability of successful message pulling closer and closer to zero but not the zero itself. We can reach to zero faster by also increasing the loose rate.

# Exercise 2

## Question 1

The answer is No. the lazy probabilistic broadcast assumes a fully connected topology and it does not work properly in not fully connected topology. The problem is that each node will send a message only to its neighbors and neighbors will not forward it to their neighbors as they used to do in eager probabilistic broadcast. Those nodes, which are not connected directly to the sender, will never be aware of a particular broadcasted message, consequently they will never try to pull the message.

## Question 2

We can forward delivered message to all neighbors except source and self and ignore the already delivered message on deliver event to avoid duplication and broadcasting a message infinite number of times.

Algorithm: Unreliable broadcast

**Implements:**  
UnreliableBroadcast (un).  
**Uses:**  
FairLossPointToPointLinks (flp2p).  
upon event <Init> do  
 delivered := ∅;  
end event  
  
upon event < unBroadcast | m > do

for all pi∈Π do  
 trigger <flp2pSend | pi, m>;  
 end for  
end event  
  
upon event < flp2pDeliver | pi, m > do  
 if m !∈delivered then  
 delivered := delivered ∪ {m};  
 trigger <unDeliver | pi, m >;  
 for all pj∈Π\{pi}\{self} do  
 trigger <flp2pSend | pj, m >;  
 end for  
 endif  
end event