ID2203 - Tutorial 2

Distributed Systems, Advanced Course

Cosmin Arad icarad(@)kth.se
Tallat M. Shafaat tallat(@)kth.se
Seif Haridi haridi(@)kth.se
KTH - The Royal Institute of Technology

This tutorial

- Broadcast primitives
  - Randomized/Probabilistic Broadcast
- Other flavours of broadcast taught in the class

Randomized algorithms

- Non-deterministic
- Probabilistic claims about guarantees
- Highly scalable and fault-tolerant
- Epidemics/gossiping

Interface of Probabilistic Broadcast

- **Module:**
  - Name: ProbabilisticBroadcast (pb)
- **Events:**
  - **Request:** ⟨pbBroadcast | m⟩
    - Broadcast m to all processes
  - **Indication:** ⟨pbDeliver | src, m⟩
    - Deliver message m from process src
- **Properties:**
  - ***PB1: Probabilistic Validity***
  - ***PB2: No duplication***
  - ***PB3: No creation***

Implementing pb

- Lazy probabilistic broadcast
- Main idea:
  - Use a cheap unreliable broadcast to disseminate a message
  - Some nodes randomly save the message
  - Use gossiping to recover any lost messages
- How to know if a message is lost?
  - Sender tags messages with sequence numbers
  - Gap in sequence numbers of received messages indicates loss of messages

Implementing pb

- **Init**
  - lsn := 0 ►*local sequence number*
  - del[∀p] := 0 ► *sequence number of last message delivered sent by pi*
  - pending := {} ►*set of undelivered messages*

- ⟨pbBroadcast | m⟩
  - lsn := lsn+1
  - ⟨unBroadcast | [m, lsn]⟩ ►*unreliably broadcast m with sequence number*

Implementing pb

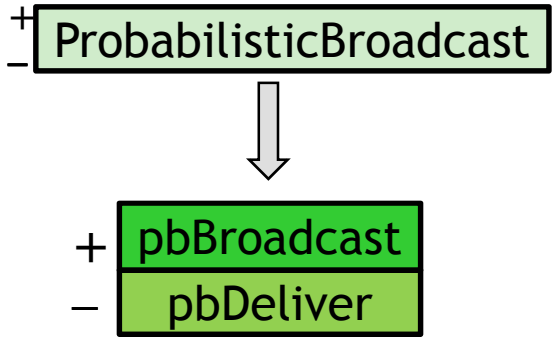
- ⟨unreliableDeliver | p, m, sn⟩
  - Randomly decide if I should store *m*
  - If I have already delivered '*sn*-1' from *p*
    - **trigger** ⟨pbDeliver | m⟩
    - Move delivered pointer to *sn*
  - If I have missed some message(s)
    - Add *m* to the *pending* set
    - For all missed messages, initiate gossip
    - Wait for a 'timeout' for the gossip to succeed

Implementing pb

- ⟨Timeout | p, sn⟩
  - Deliver as many messages as received
  - Move delivered pointer to *sn*

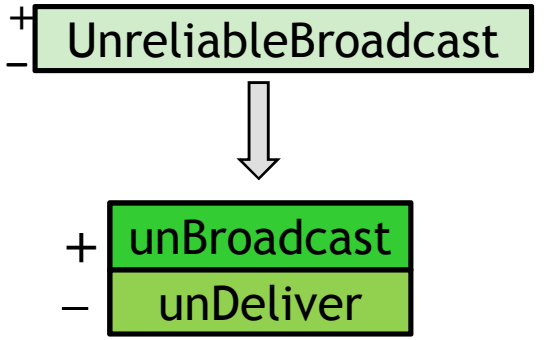
Suggestions for lpb

- SUBInit ⊆ Init
- LazyPB ⊆ Init

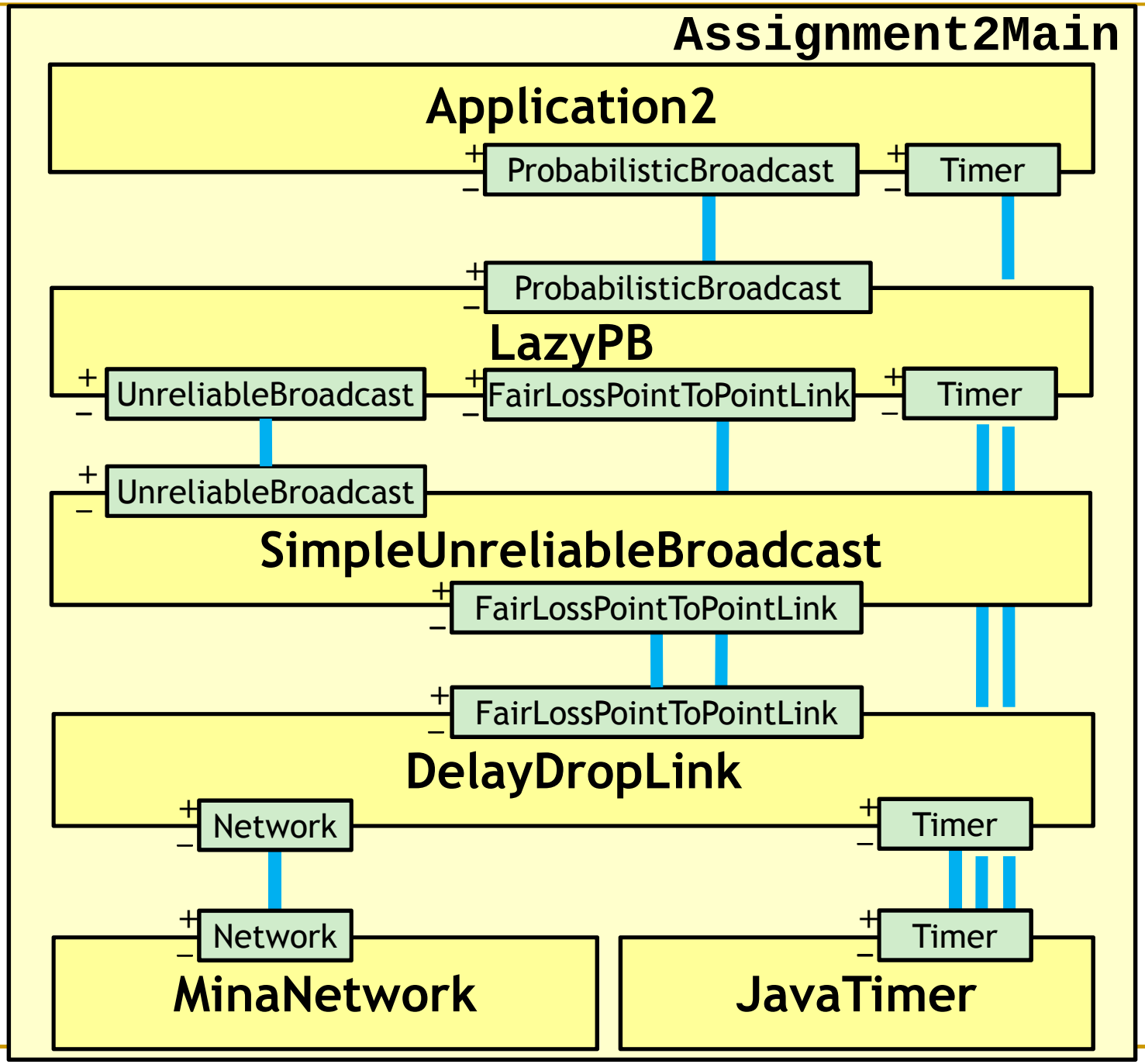


- pbBroadcast
- pbDeliver
- unBroadcast
- unDeliver

- LPBTimeout ⊆ Timeout



- RequestMessage ⊆ Flp2pDeliver
- DataMessage ⊆ Flp2pDeliver


Assignment2Main
Application2
ProbabilisticBroadcast
Timer
ProbabilisticBroadcast
LazyPB
UnreliableBroadcast
FairLossPointToPointLink
Timer
UnreliableBroadcast
SimpleUnreliableBroadcast
FairLossPointToPointLink
FairLossPointToPointLink
DelayDropLink
Network
Timer
Network
Timer
MinaNetwork
JavaTimer

Exercises

- Exercise 1
  - Correct the algorithm.
  - Give your new algorithm in your report along with discussion on the errors fixed.
  - Implement your algorithm in Kompics.
  - The algorithm depends on *fanout*, *store-threshold* and *maxrounds*. Discuss their effect on the broadcast.

Exercises

- Exercise 2
  - Lazy probabilistic broadcast assumes you have a fully connected topology. Discuss what will happen if the topology is not fully connected. Discuss any modifications needed.
    - You do not need to implement this!
- As always, questions on the forum are welcome!