Distributed Systems

Terminating Reliable Broadcast

Prof R. Guerraoui

Distributed Programming Laboratory
Terminating Reliable Broadcast
Terminating Reliable Broadcast

- Like reliable broadcast, terminating reliable broadcast (TRB) is a communication primitive used to disseminate a message among a set of processes in a reliable way.

- TRB is however strictly stronger than (uniform) reliable broadcast.
(Uniform) Reliable Broadcast

- **p1**: broadcast(m)
- **p2**: broadcast(m)
- **p3**: broadcast(m)

- **crash**
- **deliver(m)**

- **deliver(m)**
(Uniform) Reliable Broadcast

broadcast(m)

\( p_1 \)

\( p_2 \)

\( p_3 \)

\( crash \)
Terminating Reliable Broadcast

broadcast(m)

deliver(m)
crash
deliver(m)
Terminating Reliable Broadcast

\[ \text{broadcast}(m) \]

\[ \text{crash} \]

\[ \text{deliver}(\varphi) \]

\[ \text{deliver}(\varphi) \]
Terminating Reliable Broadcast

• **Like** with reliable broadcast, correct processes in TRB agree on the set of messages they deliver

• **Like** with (uniform) reliable broadcast, every correct process in TRB delivers every message delivered by any process

• **Unlike** with reliable broadcast, every correct process delivers a message, even if the broadcaster crashes
Terminating Reliable Broadcast

- The problem is defined for a specific broadcaster process \( p_i = \text{src} \) (known by all processes).
- Process \( \text{src} \) is supposed to broadcast a message \( m \) (distinct from \( \varphi \)).
- The other processes need to deliver \( m \) if \( \text{src} \) is correct but may deliver \( \varphi \) if \( \text{src} \) crashes.
Terminating Reliable Broadcast (pi)

**TRB1. Integrity:** If a process delivers a message m, then either m is φ or m was broadcast by src

**TRB2. Validity:** If the sender src is correct and broadcasts a message m, then src eventually delivers m

**TRB3. (Uniform) Agreement:** For any message m, if a correct (any) process delivers m, then every correct process delivers m

**TRB4. Termination:** Every correct process eventually delivers exactly one message
Terminating Reliable Broadcast

**Events**

- Request: <trbBroadcast, m>
- Indication: <trbDeliver, p, m>

- **Properties:**
  - TRB1, TRB2, TRB3, TRB4
Algorithm (trb)

**Implements:** trbBroadcast (trb).

**Uses:**
- BestEffortBroadcast (beb).
- PerfectFailureDetector (P).
- Consensus(cons).

**upon event** < Init > **do**
- prop := ⊥;
- correct := S;
Algorithm (trb – cont’d)

upon event < trbBroadcast, m> do
   trigger < bebBroadcast, m> ;

• upon event < crash, src > and (prop = ⊥) do
  • prop := φ ;
Algorithm (trb – cont’d)

upon event <bebDeliver, src, m> and (prop = ⊥) do

prop := m;

• upon event (prop ≠ ⊥) do
  • trigger < Propose, prop >;

• upon event < Decide, decision> do
  • trigger < trbDeliver, src, decision>;
Algorithm (trb); src = p2

p1

p2

p3

broadcast(m)
crash

UCons(φ,φ-m) deliver(φ -m)

UCons(φ,φ-m) deliver(φ -m)

UCons(m,φ-m)
Terminating Reliable Broadcast

- The TRB algorithm uses the perfect failure detector P (i.e., P is sufficient)
- Is P also necessary?
  - Is there an algorithm that implements TRB with a failure detector that is strictly weaker than P? (this would mean that P is not necessary)
  - Is there an algorithm that uses TRB to implement P (this would mean that P is necessary)
Terminating Reliable Broadcast

- We give an algorithm that implements $P$ using $\text{TRB}$; more precisely, we assume that every process $p_i$ can use an infinite number of instances of TRB where $p_i$ is the sender $\text{src}$
  - 1. Every process $p_i$ keeps on $\text{trbBroadcasting}$ messages $m_{i1}$, $m_{i2}$, etc
  - 2. If a process $p_k$ delivers $\varphi_i$, $p_k$ suspects $p_i$
  - NB. The algorithm uses (non-uniform) TRB