Distributed Algorithms

Reliable Broadcast

Prof R. Guerraoui
Lpdwww.epfl.ch
Broadcast
Broadcast abstractions

Best-effort broadcast
Reliable broadcast
Uniform broadcast
Modules of a process

Applications

indication

request

(deliver)

Failure detector

indication

(deliver)

(request)

Channels

(B-U) Reliable broadcast

(indication)

(request)

(deliver)

(deliver)
Intuition

Broadcast is useful for instance in applications where some processes subscribe to events published by other processes (e.g., stocks)

The subscribers might require some reliability guarantees from the broadcast service (we say sometimes quality of service – QoS) that the underlying network does not provide
Overview

We shall consider three forms of reliability for a broadcast primitive

1. Best-effort broadcast
2. (Regular) reliable broadcast
3. Uniform (reliable) broadcast

We shall give first specifications and then algorithms
Best-effort broadcast (beb)

*Events*

- Request: `<bebBroadcast, m>`
- Indication: `<bebDeliver, src, m>`

- **Properties:** BEB1, BEB2, BEB3
Best-effort broadcast (beb)

Properties

**BEB1. Validity:** If \( p_i \) and \( p_j \) are correct, then every message broadcast by \( p_i \) is eventually delivered by \( p_j \)

**BEB2. No duplication:** No message is delivered more than once

**BEB3. No creation:** No message is delivered unless it was broadcast
Best-effort broadcast
Best-effort broadcast

delivery

m1

m2

p1

p2

p3

delivery

m1

m2

delivery

crash
Reliable broadcast (rb)

Events

- Request: <rbBroadcast, m>
- Indication: <rbDeliver, src, m>

- Properties: RB1, RB2, RB3, RB4
Reliable broadcast (rb)

Properties

- $RB1 = BEB1$.  
- $RB2 = BEB2$.  
- $RB3 = BEB3$.  
- $RB4. Agreement$: For any message $m$, if any correct process delivers $m$, then every correct process delivers $m$. 
Reliable broadcast

p1

m1

delivery

m2

delivery

p2

p3

m1

delivery

m2

delivery

13
Reliable broadcast

p1
m1 delivery
m2

p2

p3
delivery
m1

crash
Reliable broadcast

p1: delivery → delivery
   m1 → m2
   crash

p2: delivery → delivery
   m2
   crash

p3: delivery
Uniform broadcast (urb)

**Events**

- Request: `<urbBroadcast, m>`
- Indication: `<urbDeliver, src, m>`

- **Properties:** URB1, URB2, URB3, URB4
Uniform broadcast (urb)

Properties

\[ \text{URB1} = \text{BEB1}. \]

\[ \text{URB2} = \text{BEB2}. \]

\[ \text{URB3} = \text{BEB3}. \]

\[ \text{URB4. Uniform Agreement:} \quad \text{For any message } m, \text{ if any process delivers } m, \text{ then every correct process delivers } m \]
Uniform reliable broadcast

\begin{itemize}
\item p1
\item delivery
\item delivery
\item crash
\item m1
\item delivery
\item crash
\item m2
\item m1
\item delivery
\item delivery
\item m2
\item delivery
\end{itemize}
Uniform reliable broadcast

p1

p2

p3

delivery

m1

m2

crash

delivery

crash

delivery

m1
Overview

We consider three forms of reliability for a broadcast primitive

(1) Best-effort broadcast
(2) (Regular) reliable broadcast
(3) Uniform (reliable) broadcast

We give first specifications and then algorithms
Algorithm (beb)

- **Implements:** BestEffortBroadcast (beb).
- **Uses:** PerfectLinks (pp2p).
- **upon event** `< bebBroadcast, m> do
 forall pi ∈ S do
  trigger `< pp2pSend, pi, m>;
- **upon event** `< pp2pDeliver, pi, m> do
  trigger `< bebDeliver, pi, m>;
Algorithm (beb)

p1

m

delivery

p2

m

delivery

p3
Algorithm (beb)

Proof (sketch)

**BEB1. Validity:** By the validity property of perfect links and the very facts that (1) the sender sends the message to all and (2) every correct process that \( pp2pDelivers \) a message bebDelivers it.

**BEB2. No duplication:** By the no duplication property of perfect links.

**BEB3. No creation:** By the no creation property of perfect links.
Algorithm (beb)
Algorithm (rb)

**Implements:** ReliableBroadcast (rb).

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

**upon event** < Init > do

- delivered := ∅;
- correct := S;
- **forall** pi ∈ S do from[pi] := ∅;
Algorithm (rb – cont’d)

upon event $<\text{rbBroadcast}, m>$ do
  delivered := delivered $\cup \{m\}$;
  trigger $<\text{rbDeliver}, \text{self}, m>$;
  trigger $<\text{bebBroadcast}, [\text{Data, self, m}]>;$
Algorithm (rb – cont’d)

Upon event < crash, pi > do
  correct := correct \ {pi};
  forall \([pj,m] \in \text{from}[pi]\) do
    trigger < bebBroadcast,\([\text{Data},pj,m]\)>;
upon event < bebDeliver, pi, [Data,pj,m]> do
  if m \notin \text{delivered} then
    \text{delivered} := \text{delivered} \cup \{m\};
  trigger < rbDeliver, pj, m>;
  if pi \notin \text{correct} then
    trigger < bebBroadcast,[Data,pj,m]>;
  else
    from[pi] := from[pi] \cup \{(pj,m)\};
Algorithm (rb)
Algorithm (rb)

p1

m

crash

delivery

m

p2

m

delivery

m

p3

delivery
Algorithm (rb)

Proof (sketch)

- **RB1. RB2. RB3:** as for the 1st algorithm

- **RB4. Agreement:** Assume some correct process pi rbDelivers a message m rbBroadcast by some process pk. If pk is correct, then by property BEB1, all correct processes bebDeliver and then rebDeliver m. If pk crashes, then by the completeness property of P, pi detects the crash and bebBroadcasts m to all. Since pi is correct, then by property BEB1, all correct processes bebDeliver and then rebDeliver m.
Algorithm (urb)

**Implements:** uniformBroadcast (urb).

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

**upon event < Init > do**
- correct := S;
- delivered := forward := Ø;
- ack[Message] := Ø;
Algorithm (urb – cont’d)

upon event < crash, pi > do
  correct := correct \ {pi};

upon event < urbBroadcast, m> do
  forward := forward U {[self,m]};
  trigger < bebBroadcast, [Data,self,m]>;
Algorithm (urb – cont’d)

upon event < bebDeliver, pi, [Data,pj,m]> do

ack[m] := ack[m] U {pi};

if [pj,m] \not\in forward then

forward := forward U {[pj,m]};

trigger < bebBroadcast,[Data,pj,m]>;
Algorithm (urb – cont’d)

upon event (for any \([pj,m] \in \text{forward}\) \(\text{<correct} \subseteq \text{ack}[m]\) and \(\text{<m} \notin \text{delivered}\) do

- \(\text{delivered} := \text{delivered} \cup \{m\}\);
- \(\text{trigger} \prec \text{urbDeliver}, pj, m\);
Algorithm (urb)
Algorithm (urb)
Algorithm (urb)

Proof (sketch)

UBR2. URB3: follow from BEB2 and BEB3

Lemma: If a correct process \( pi \) bebDelivers a message \( m \), then \( pi \) eventually urbDelivers \( m \).

Any process that bebDelivers \( m \) bebBroadcasts \( m \). By the completeness property of the failure detector and property BEB1, there is a time at which \( pi \) bebDelivers \( m \) from every correct process and hence urbDelivers \( m \).
Algorithm (urb)

Proof (sketch)

URB1. Validity: If a correct process \( p_i \) urbBroadcasts a message \( m \), then \( p_i \) eventually bebBroadcasts and bebDelivers \( m \): by our lemma, \( p_i \) urbDelivers \( m \).

URB4. Agreement: Assume some process \( p_i \) urbDelivers a message \( m \). By the algorithm and the completeness and accuracy properties of the failure detector, every correct process bebDelivers \( m \). By our lemma, every correct process will urbDeliver \( m \).