Distributed Systems
Master of Science in Engineering in Computer Science

AA 2020/2021

LECTURE 17: BROADCAST IN PRESENCE OF BYZANTINE PROCESSES

Recap on Byzantine processes

Byzantine processes may

1. deviate arbitrarily from the instructions that an algorithm assigns to them
   - creating fake messages
   - dropping messages
   - delay the deliveries
   - altering the content of messages
   - …
2. act as if they were deliberately preventing the algorithm from reaching its goals

Basic step to fight Byzantine processes

🔑 Using cryptographic mechanisms to implement the authenticated perfect links abstraction

… But, cryptography alone does not allow to tolerate Byzantine processes

- Considering an arbitrary-faulty sender, asking him/her to digitally sign every broadcast message does not help at all (it may simply sign the two different messages)

Correct and faulty state

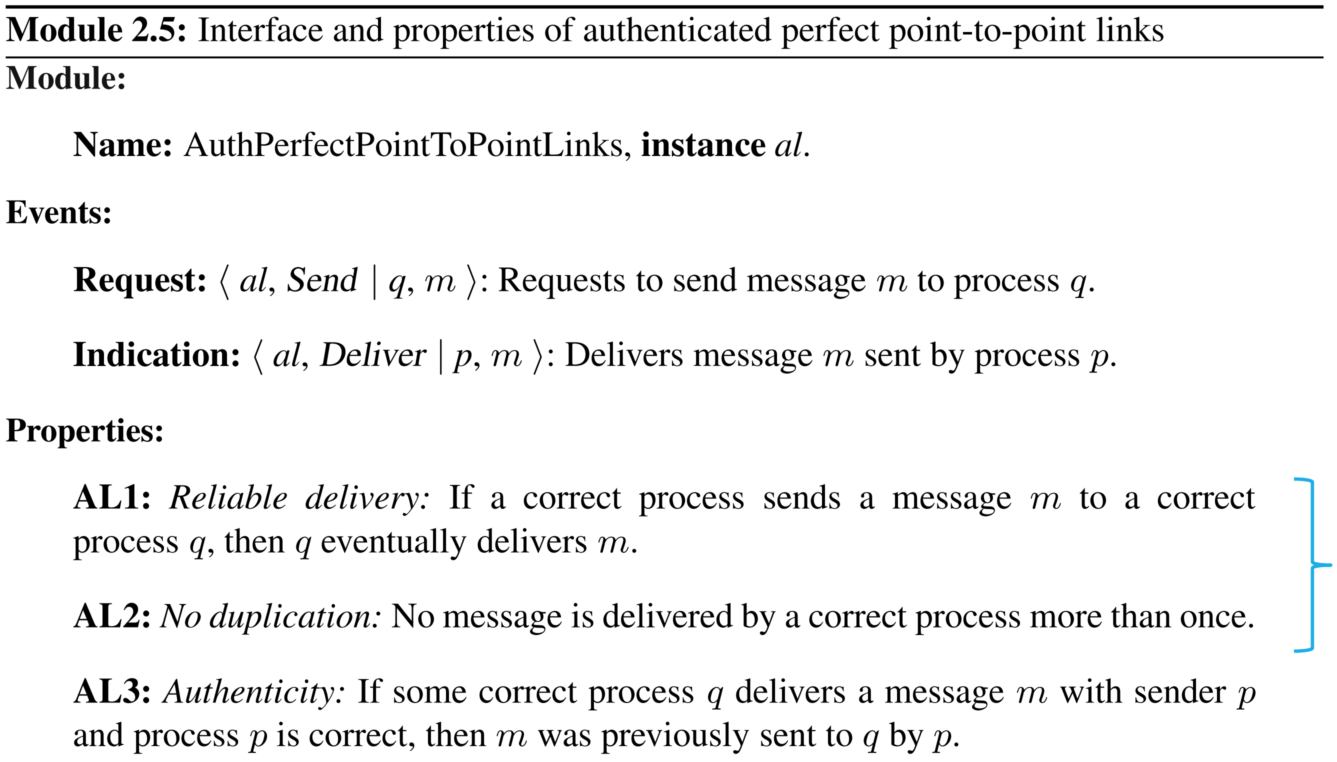
As in the crash failure model, we distinguish between *faulty* and *correct* processes

NOTE: a Byzantine process may act arbitrarily and no mechanism can guarantee anything that relates to its actions.

⬇

We do not define any “uniform” variants of primitives in the Byzantine failure model.

Authenticated Perfect Link



Module 2.5: Interface and properties of authenticated perfect point-to-point links

Module:

Name: AuthPerfectPointToPointLinks, **instance** *al*.

Events:

Request: ⟨ *al*, *Send* | q, m ⟩: Requests to send message m to process q.

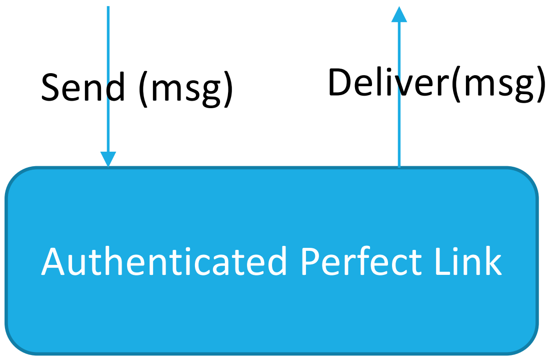
Indication: ⟨ *al*, *Deliver* | p, m ⟩: Delivers message m sent by process p.

Properties:

AL1: *Reliable delivery:* If a correct process sends a message m to a correct process q, then q eventually delivers m.

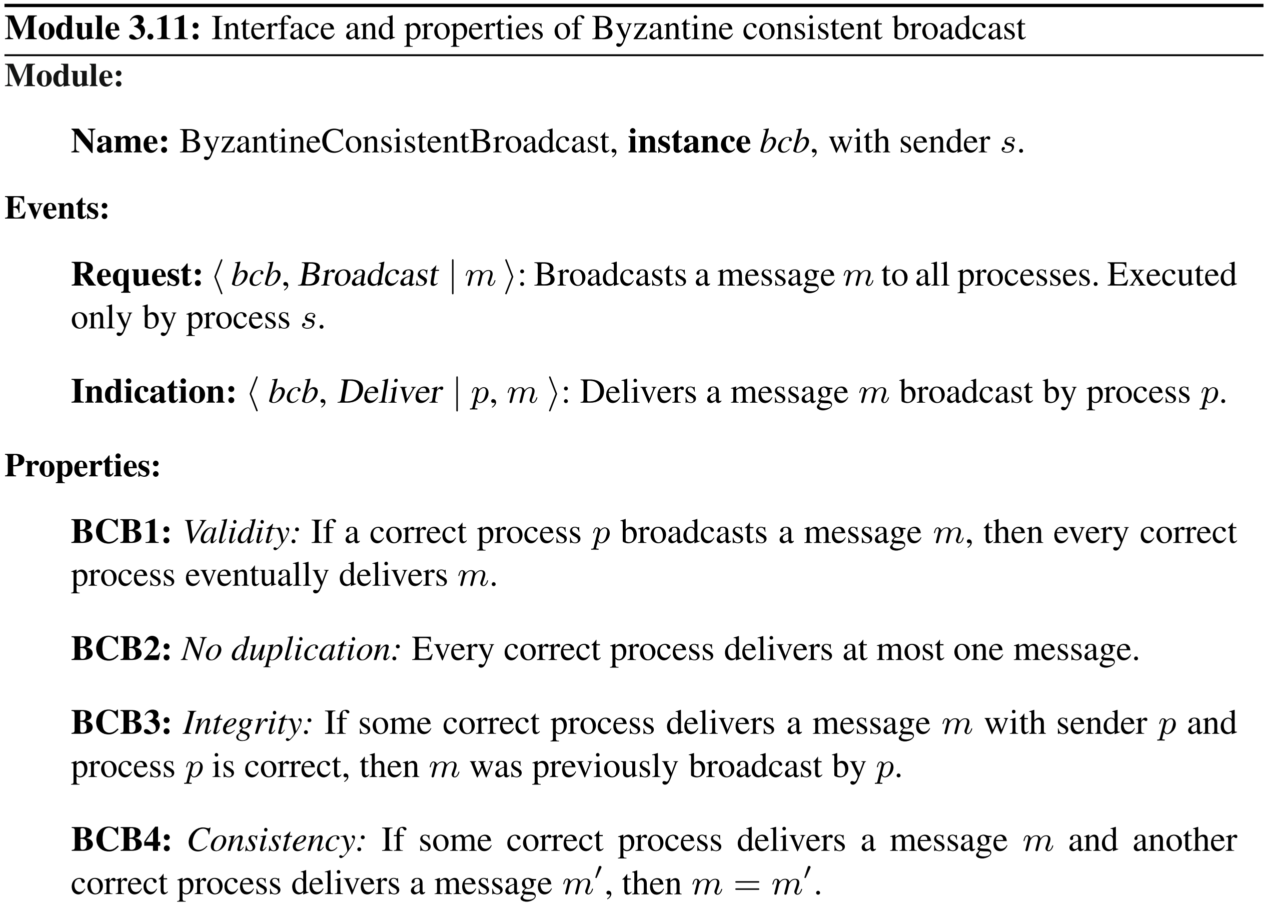
AL2: *No duplication:* No message is delivered by a correct process more than once.

AL3: *Authenticity:* If some correct process q delivers a message m with sender p and process p is correct, then m was previously sent to q by p.



Same as Perfect point-to-point links

Byzantine consistent broadcast specification



Module 3.11: Interface and properties of Byzantine consistent broadcast

Module:

Name: ByzantineConsistentBroadcast, **instance** *bcb*, with sender s.

Events:

Request: $\langle$ *bcb*, *Broadcast* | m $\rangle$: Broadcasts a message m to all processes. Executed only by process s.

Indication: $\langle$ *bcb*, *Deliver* | p, m $\rangle$: Delivers a message m broadcast by process p.

Properties:

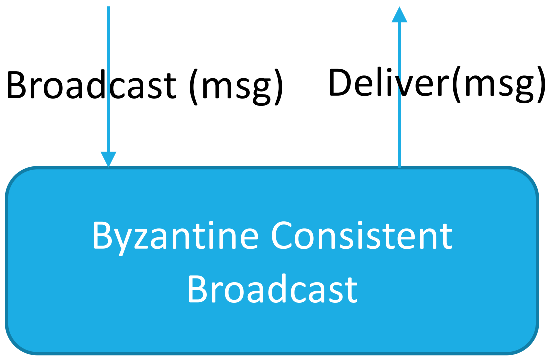
BCB1: *Validity:* If a correct process p broadcasts a message m, then every correct process eventually delivers m.

BCB2: *No duplication:* Every correct process delivers at most one message.

BCB3: *Integrity:* If some correct process delivers a message m with sender p and process p is correct, then m was previously broadcast by p.

BCB4: *Consistency:* If some correct process delivers a message m and another correct process delivers a message m', then $m = m'$.

The specification refers to a single broadcast event!

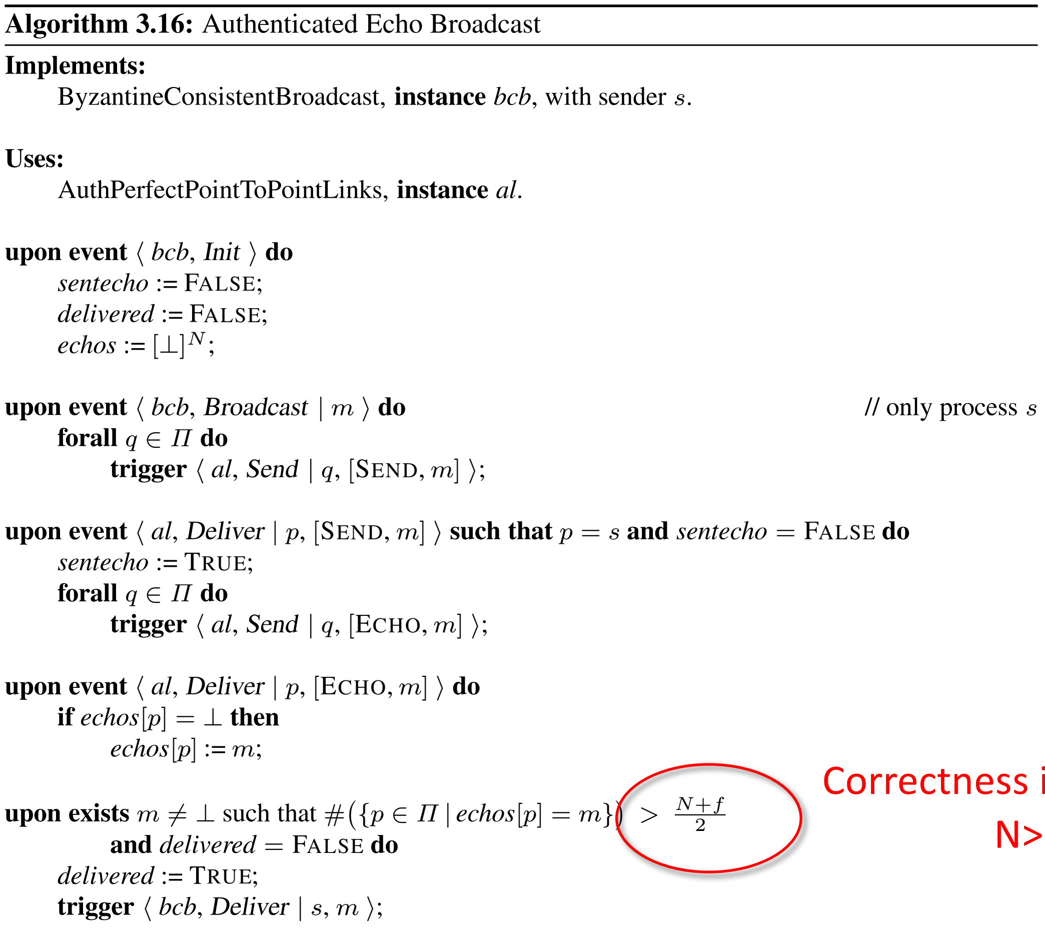


Broadcast (msg)

Deliver(msg)

Byzantine Consistent Broadcast

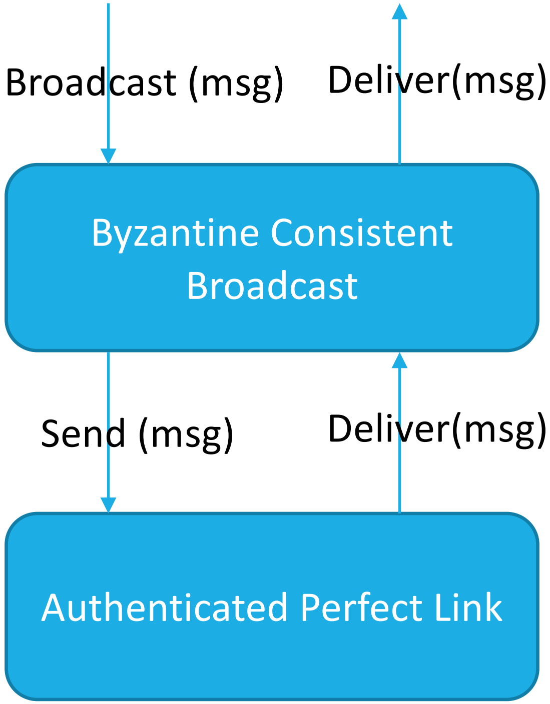
Byzantine consistent broadcast implementation



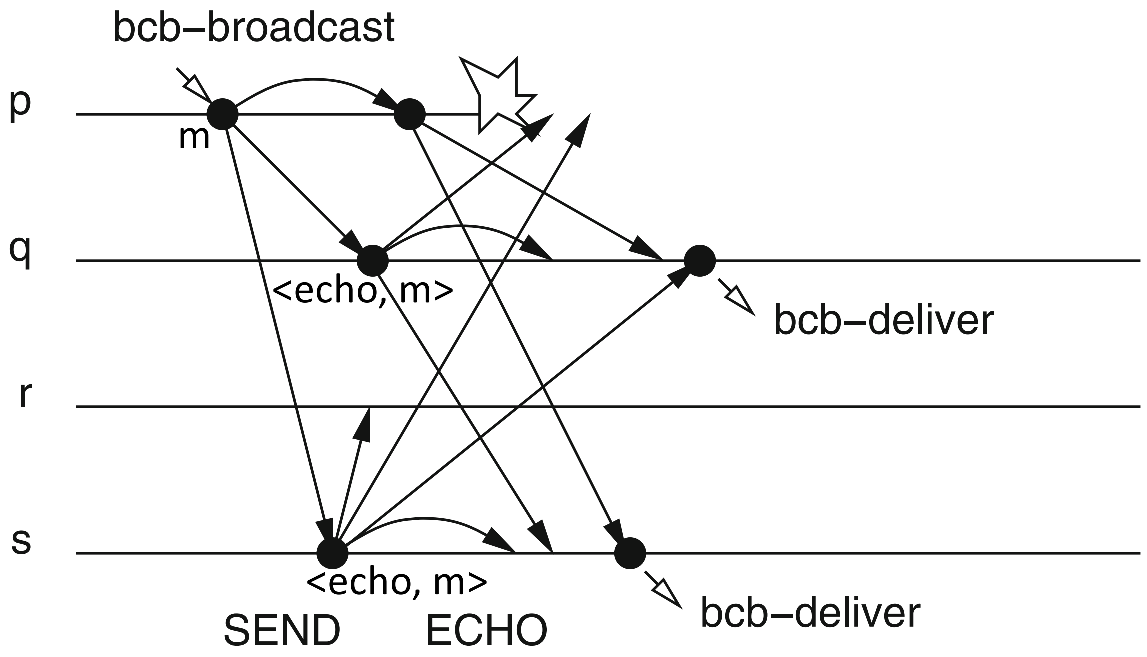
Algorithm 3.16: Authenticated Echo Broadcast

```
Implements:
    ByzantineConsistentBroadcast, instance bcb, with sender $s$.

Uses:
    AuthPerfectPointToPointLinks, instance al.

upon event ⟨ bcb, Init ⟩ do
    sentecho := FALSE;
    delivered := FALSE;
    echos := $[\perp]^N$;

upon event ⟨ bcb, Broadcast | $m$ ⟩ do                              // only process $s$
    forall $q \in \Pi$ do
        trigger ⟨ al, Send | $q$, [SEND, $m$] ⟩;

upon event ⟨ al, Deliver | $p$, [SEND, $m$] ⟩ such that $p = s$ and sentecho = FALSE do
    sentecho := TRUE;
    forall $q \in \Pi$ do
        trigger ⟨ al, Send | $q$, [ECHO, $m$] ⟩;

upon event ⟨ al, Deliver | $p$, [ECHO, $m$] ⟩ do
    if echos[$p$] = $\perp$ then
        echos[$p$] := $m$;

upon exists $m \neq \perp$ such that $\#(\{p \in \Pi \mid echos[p] = m\}) > \frac{N+f}{2}$
        and delivered = FALSE do
    delivered := TRUE;
    trigger ⟨ bcb, Deliver | $s$, $m$ ⟩;
```

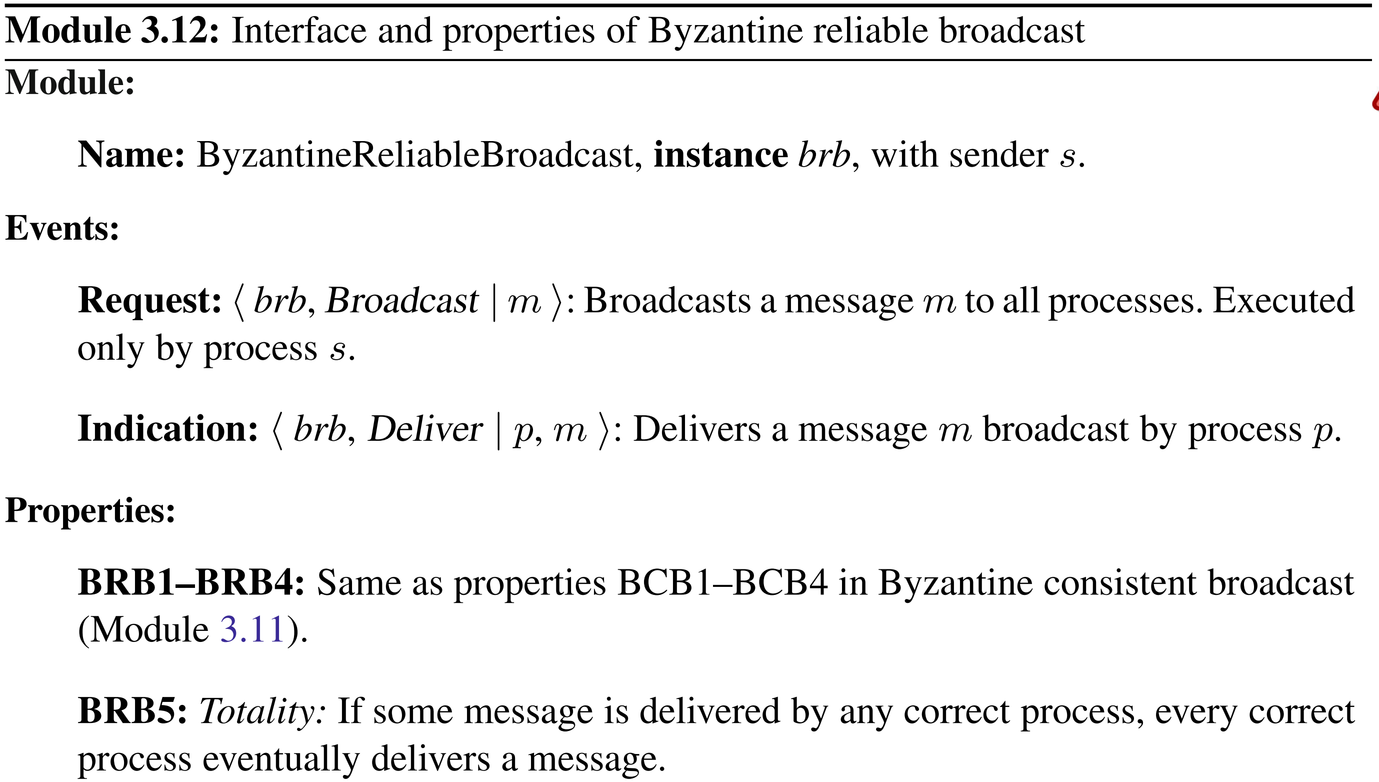
Correctness is ensured if N>3f



Byzantine consistent broadcast example



Byzantine Reliable Broadcast specification



Module 3.12: Interface and properties of Byzantine reliable broadcast

Module:

Name: ByzantineReliableBroadcast, **instance** *brb*, with sender s.

Events:

Request: $\langle$ *brb*, *Broadcast* $\mid m \rangle$: Broadcasts a message m to all processes. Executed only by process s.

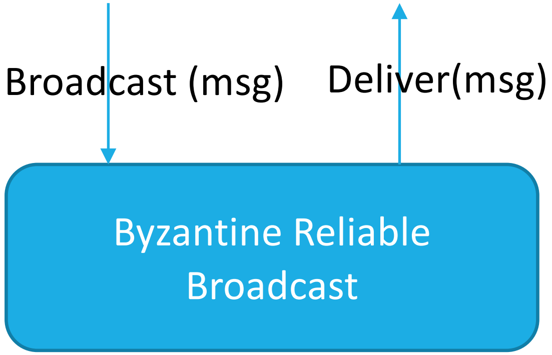
Indication: $\langle$ *brb*, *Deliver* $\mid p, m \rangle$: Delivers a message m broadcast by process p.

Properties:

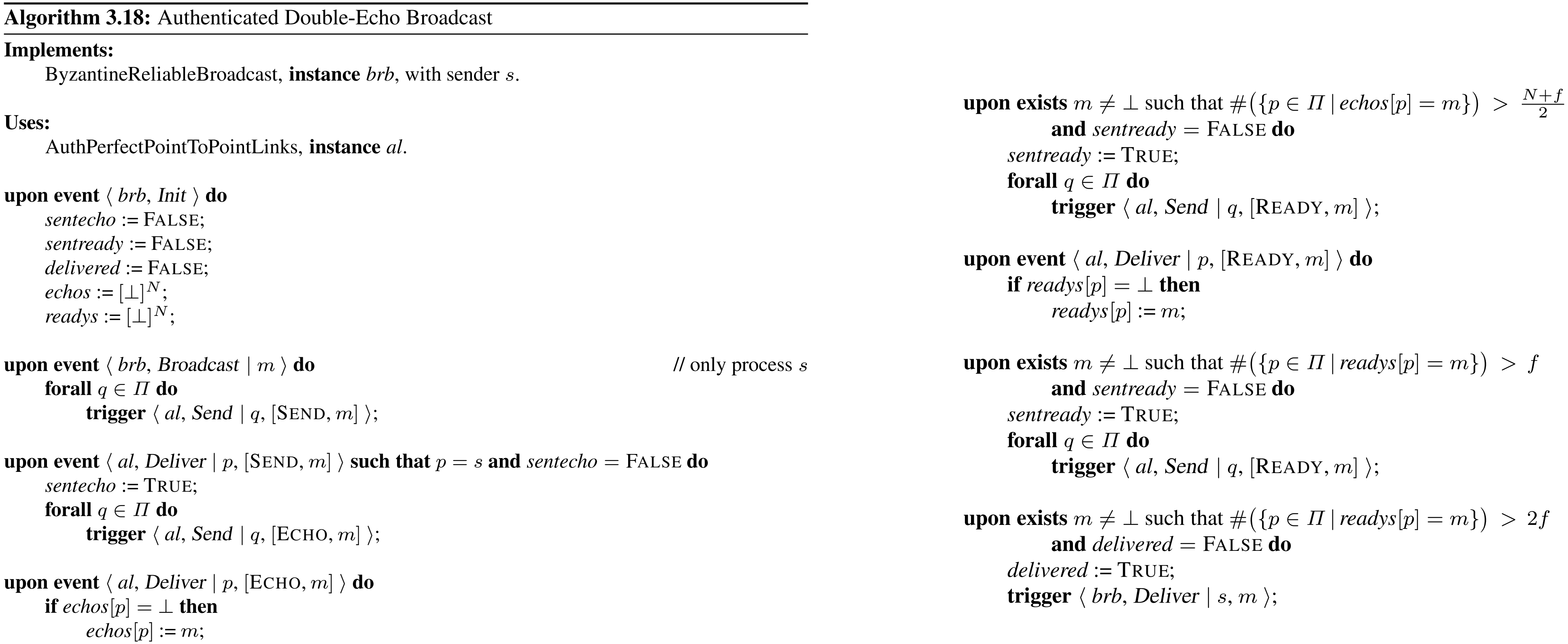
BRB1–BRB4: Same as properties BCB1–BCB4 in Byzantine consistent broadcast (Module 3.11).

BRB5: *Totality:* If some message is delivered by any correct process, every correct process eventually delivers a message.

⚠ The specification refers to a single broadcast event!



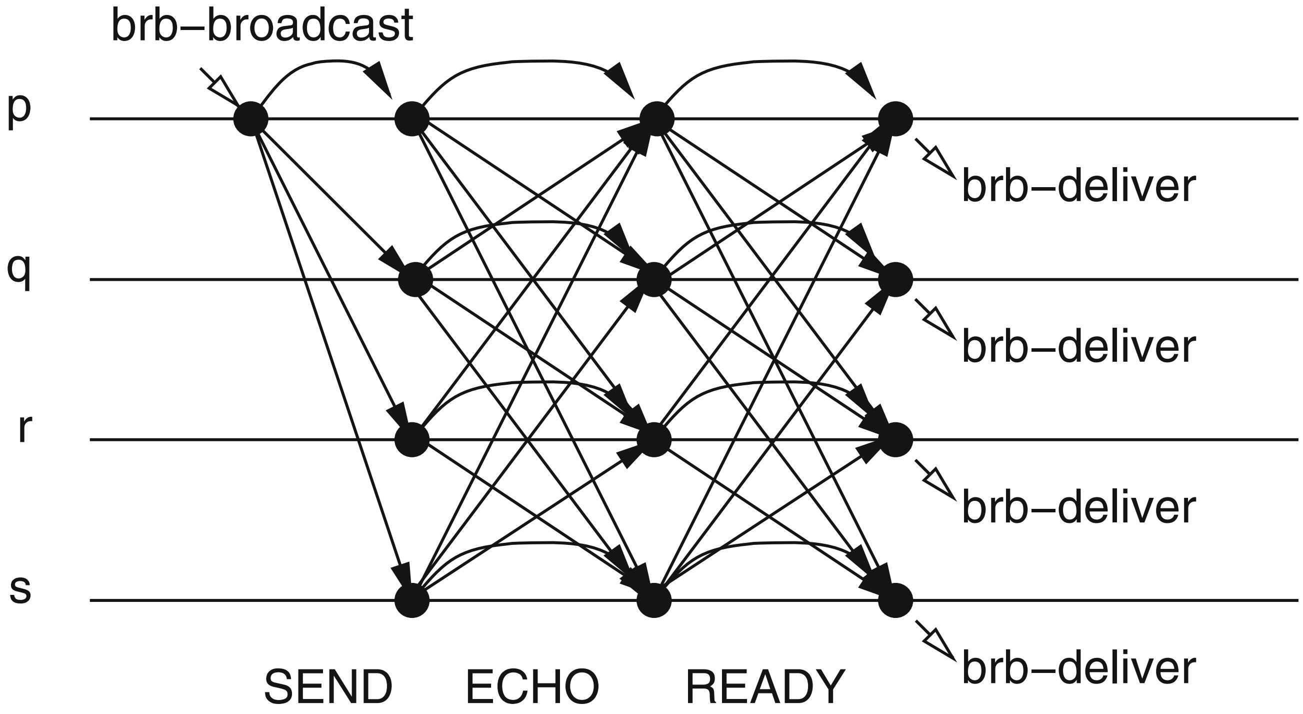
Byzantine Reliable Broadcast implementation



Algorithm 3.18: Authenticated Double-Echo Broadcast

```
Implements:
    ByzantineReliableBroadcast, instance $brb$, with sender $s$.

Uses:
    AuthPerfectPointToPointLinks, instance $al$.

upon event ⟨ brb, Init ⟩ do
    $sentecho$ := FALSE;
    $sentready$ := FALSE;
    $delivered$ := FALSE;
    $echos$ := $[\bot]^N$;
    $readys$ := $[\bot]^N$;

upon event ⟨ brb, Broadcast | $m$ ⟩ do                    // only process $s$
    forall $q \in \Pi$ do
        trigger ⟨ al, Send | $q$, [SEND, $m$] ⟩;

upon event ⟨ al, Deliver | $p$, [SEND, $m$] ⟩ such that $p = s$ and $sentecho$ = FALSE do
    $sentecho$ := TRUE;
    forall $q \in \Pi$ do
        trigger ⟨ al, Send | $q$, [ECHO, $m$] ⟩;

upon event ⟨ al, Deliver | $p$, [ECHO, $m$] ⟩ do
    if $echos[p] = \bot$ then
        $echos[p] := m$;

upon exists $m \neq \bot$ such that $\#(\{p \in \Pi \mid echos[p] = m\}) > \frac{N+f}{2}$
        and $sentready$ = FALSE do
    $sentready$ := TRUE;
    forall $q \in \Pi$ do
        trigger ⟨ al, Send | $q$, [READY, $m$] ⟩;

upon event ⟨ al, Deliver | $p$, [READY, $m$] ⟩ do
    if $readys[p] = \bot$ then
        $readys[p] := m$;

upon exists $m \neq \bot$ such that $\#(\{p \in \Pi \mid readys[p] = m\}) > f$
        and $sentready$ = FALSE do
    $sentready$ := TRUE;
    forall $q \in \Pi$ do
        trigger ⟨ al, Send | $q$, [READY, $m$] ⟩;

upon exists $m \neq \bot$ such that $\#(\{p \in \Pi \mid readys[p] = m\}) > 2f$
        and $delivered$ = FALSE do
    $delivered$ := TRUE;
    trigger ⟨ brb, Deliver | $s$, $m$ ⟩;
```

Byzantine Reliable Broadcast example



References

C. Cachin, R. Guerraoui and L. Rodrigues. Introduction to Reliable and Secure Distributed Programming, Springer, 2011

- Chapter 2 – Section 2.4.6
- Chapter 3 – Section 3.10 (except 3.10.4), Section 3.11