Byzantine Reliable Broadcast

Mohsen Lesani

Byzantine reliable broadcast

- Almost the same primitive as the fail-silent model. It needs to reach agreement on delivered messages.
- To simplify the protocol, a Byzantine reliable broadcast instance is used to deliver **one message**.
- A priori declares a **sender process** for the instance

Authenticated communication primitives

- Recall modules in model with crash failures
  - Perfect Links (pl)
  - Best-effort Broadcast (beb)
- Authenticated versions can be defined that tolerate network subject to attacks
  - Authenticated Perfect Links (al)
  - Authenticated Best-effort Broadcast (abeb)
- Implemented using cryptographic authentication (MACs or digital signatures)

Byzantine broadcast variants

- Byzantine consistent broadcast
- Byzantine reliable broadcast

Byzantine Consistent Broadcast (bcb)

Events

- Request <broadcast (m)>

  Broadcasts a message m to all processes
- Indication <deliver (p, m)>

  Delivers a message m from sender p

Byzantine Consistent Broadcast (bcb)

Properties:

- BCB1 (Validity) = BEB1:

  Every message broadcast by a correct process is eventually delivered by every correct process.

- BCB2 (No duplication):

  Every correct process delivers at most one message.

- BCB3 (Integrity):

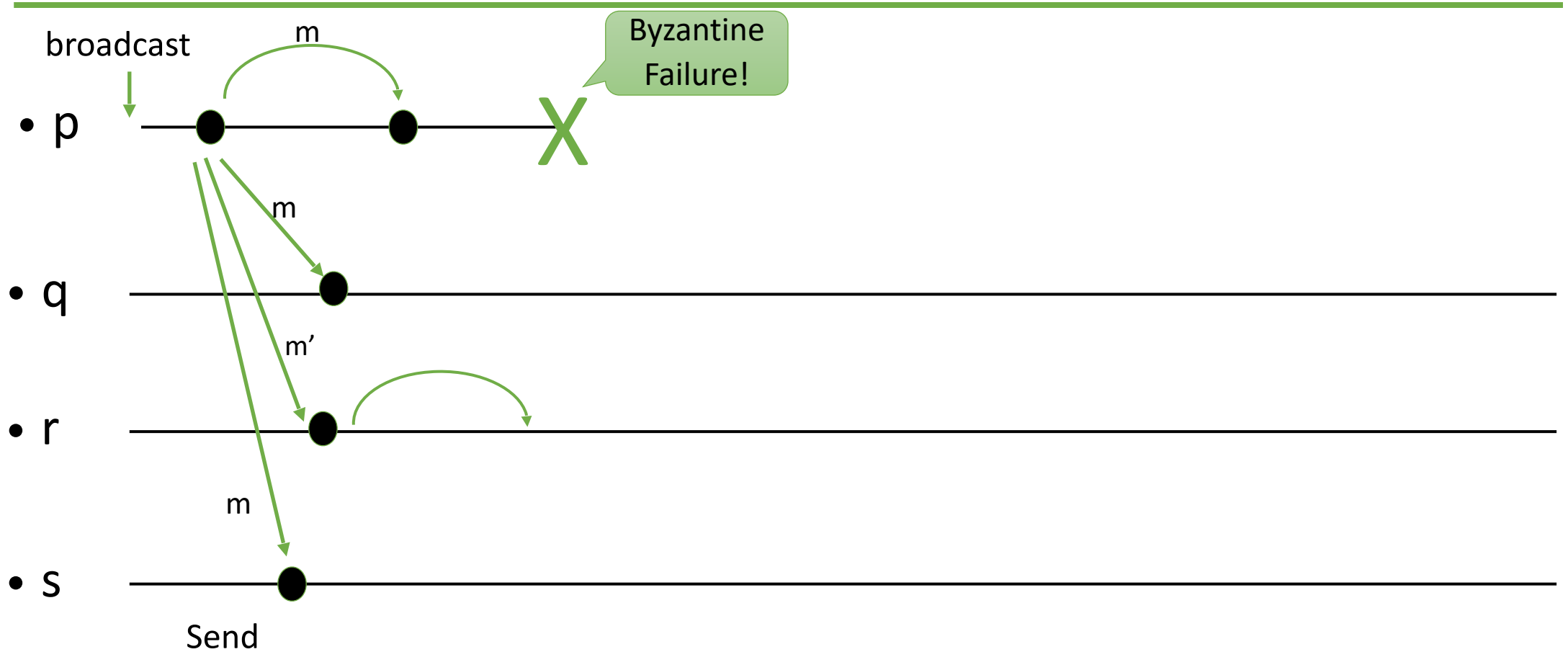
  If a correct process delivers m with sender p, and p is correct, then p has broadcast m.

- BCB4 (**Consistency**):

  If a correct process delivers message m and another correct process delivers message m', then m=m'.

- Note: some correct process may not deliver any message (agreement is not required yet)

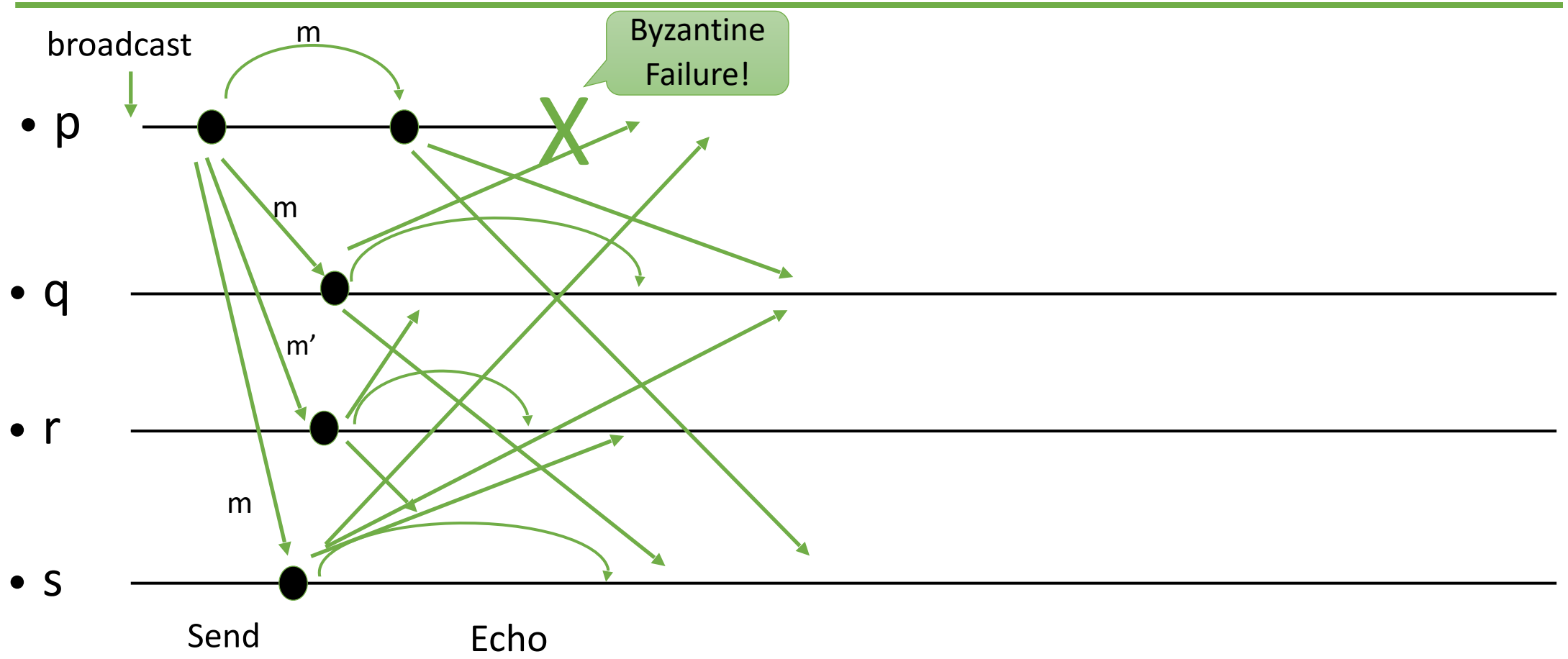
Example



- Faulty sender p sends two different message m and m'.
- Processes q and s deliver the message m.

- Process r does not deliver any message. It does not receive a quorum.
- 2 rounds, $O(n^2)$ messages
  $O(n^2 |m|)$ communicated data

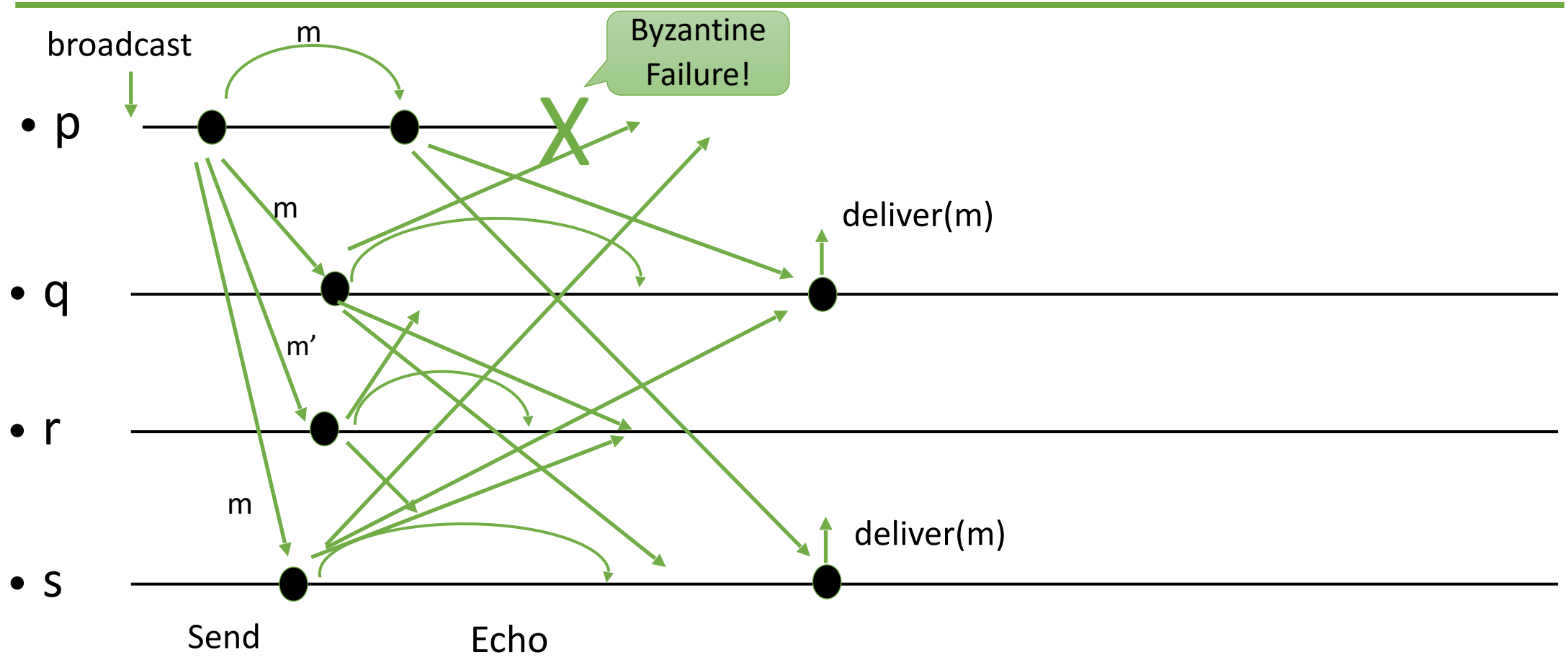
Example



- Faulty sender p sends two different message m and m'.
- Processes q and s deliver the message m.

- Process r does not deliver any message. It does not receive a quorum.
- 2 rounds, $O(n^2)$ messages
  $O(n^2 |m|)$ communicated data

Example



- Faulty sender p sends two different message m and m'.
- Processes q and s deliver the message m.

- Process r does not deliver any message. It does not receive a quorum.
- 2 rounds, $O(n^2)$ messages
  $O(n^2 |m|)$ communicated data

Authenticated Echo Broadcast

Implements bcb, **uses** abeb, with sender s (where N = 3f + 1)

upon <bcb, broadcast (m)> **do**
    trigger <abeb, broadcast(Send(m))>

upon <abeb, deliver(s, Send(m))> **do**
    trigger <abeb, broadcast(Echo(m))>

upon <abeb, deliver(p, Echo(m))> **do**
    echo[p] := m
    if ∃m : #{p | echo[p]=m} ≥ 2f + 1 **then**
        trigger <deliver(s, m)>

Note: The code to prevent duplicate execution is omitted.

Using Byzantine Quorums

- System of N > 3f processes, f are faulty.
- Let’s have N = 3f + 1 for simplicity.
- Every subset with size larger than or equal to 2f+1 processes is a quorum.
- The collection of all quorums is a quorum system.

- Two distinct quorums together contain more than 4f+2 processes.
- Thus, they overlap in at least f + 1 processes.
- At least one of them is a correct process.

Proof of the consistency property:

- This correct process has beb broadcast the same message Echo[m] to all processes.
- Therefore, the two processes deliver the same message m.

Byzantine Reliable Broadcast (brb)

Events

- Request <broadcast(m)>
- Indication <deliver(p, m)>

Properties

- BRB1 (Validity) = BCB1
- BRB2 (No duplication) = BCB2
- BRB3 (Integrity) = BCB3
- BRB4 (Consistency) = BCB4
- BRB5 (**Totality**):

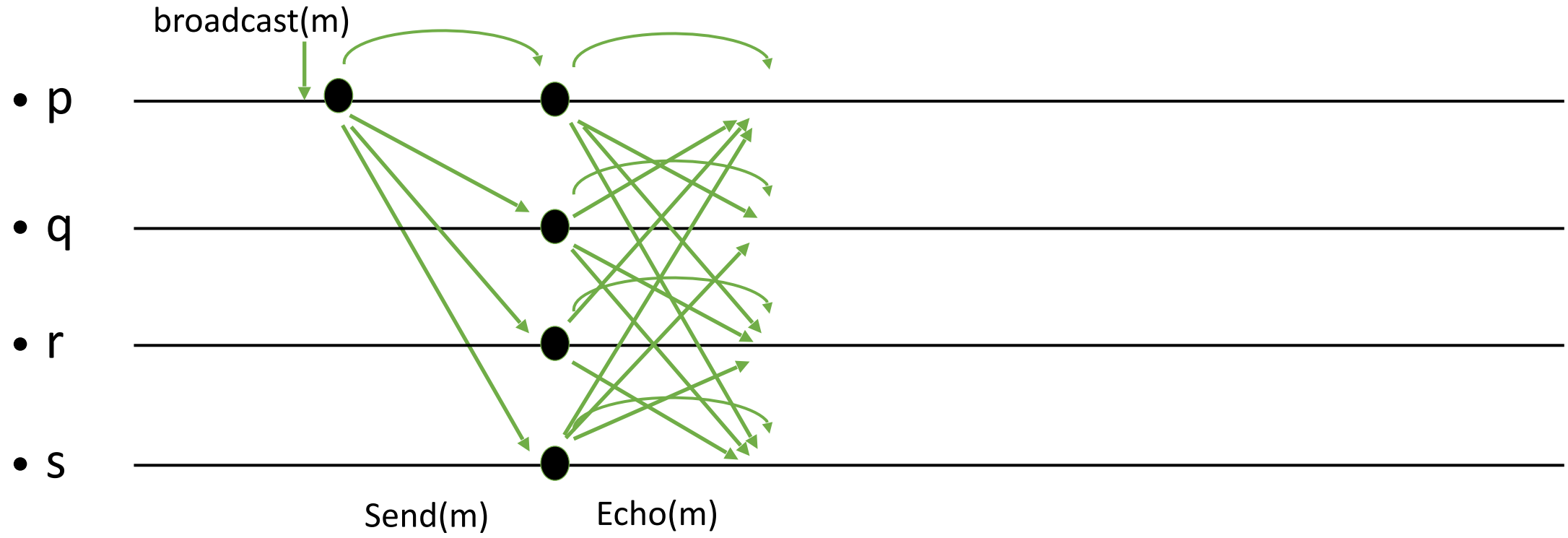
  If some correct process delivers a message, then every correct process eventually delivers a message.

By totality, either all or none of the correct processes deliver a message. By consistency, they deliver the same message.

Example

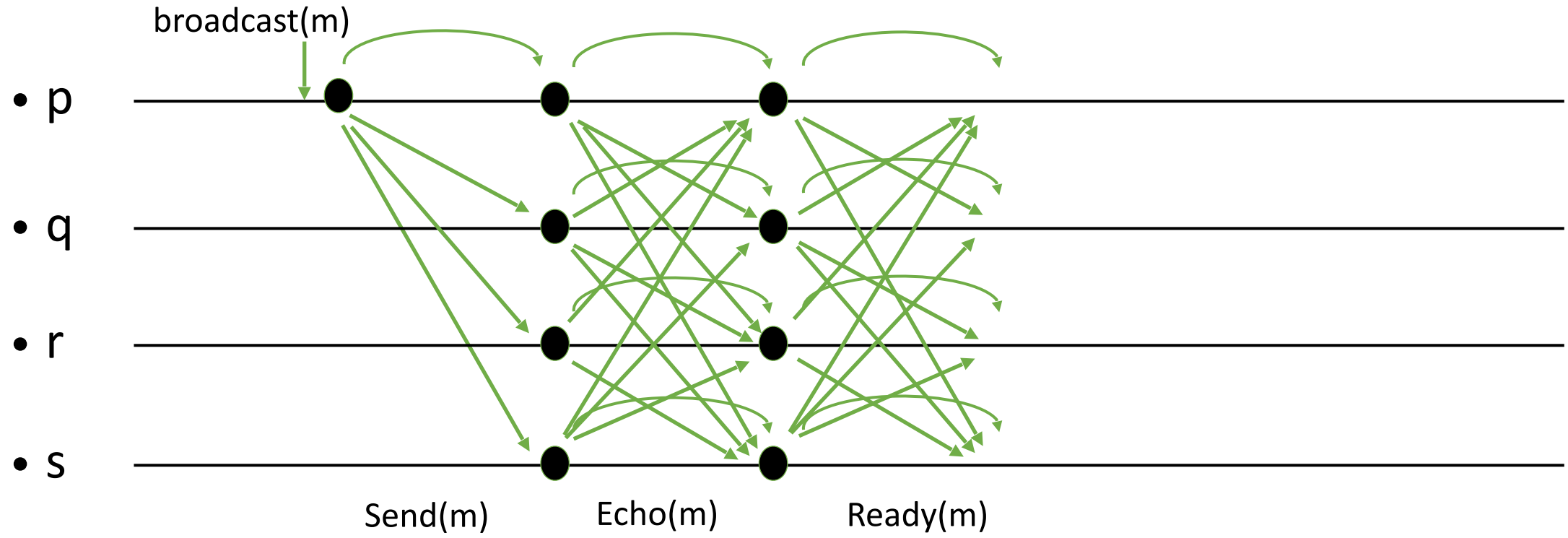
3 rounds; $O(n^2)$ messages; $O(n^2 |m|)$ communication

Example



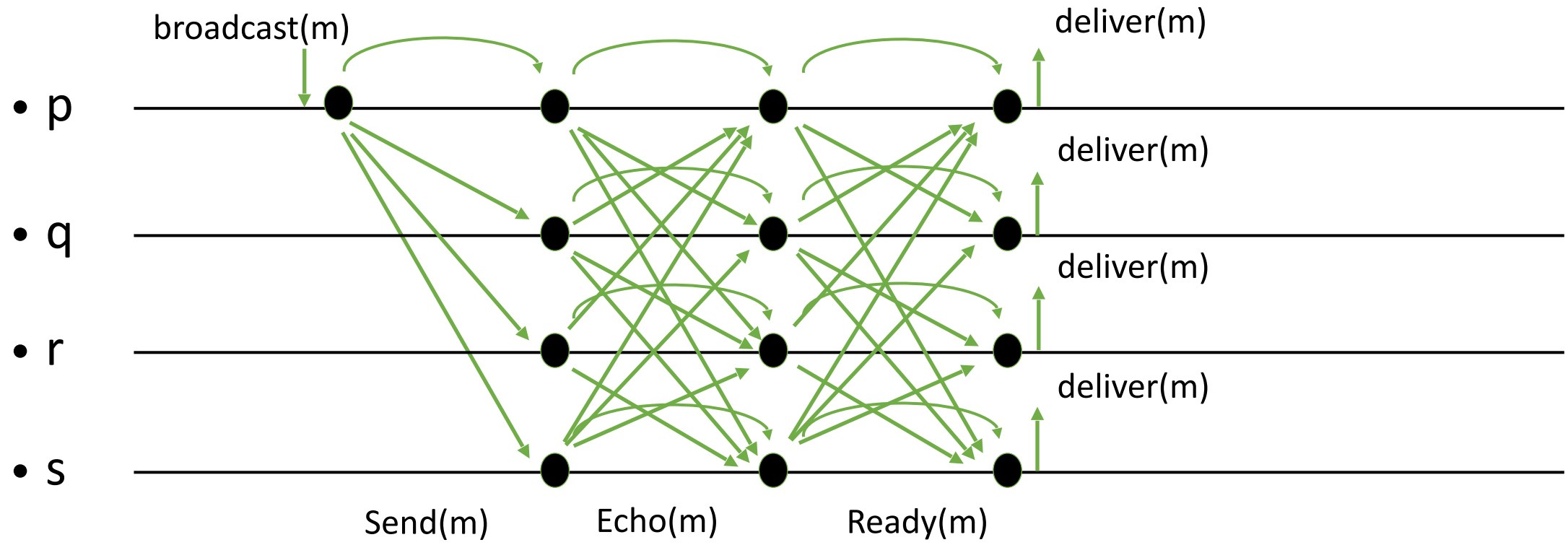
3 rounds; $O(n^2)$ messages; $O(n^2 |m|)$ communication

Example



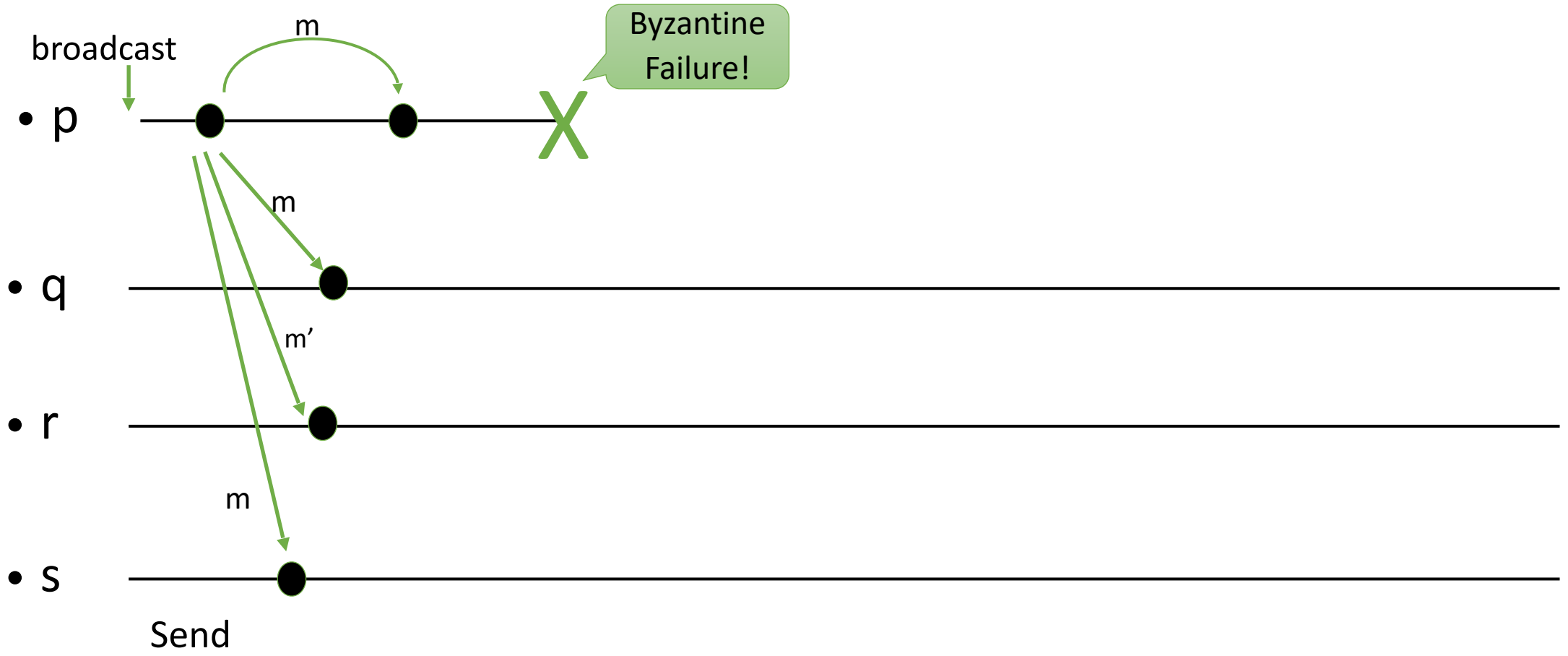
3 rounds; $O(n^2)$ messages; $O(n^2 |m|)$ communication

Example

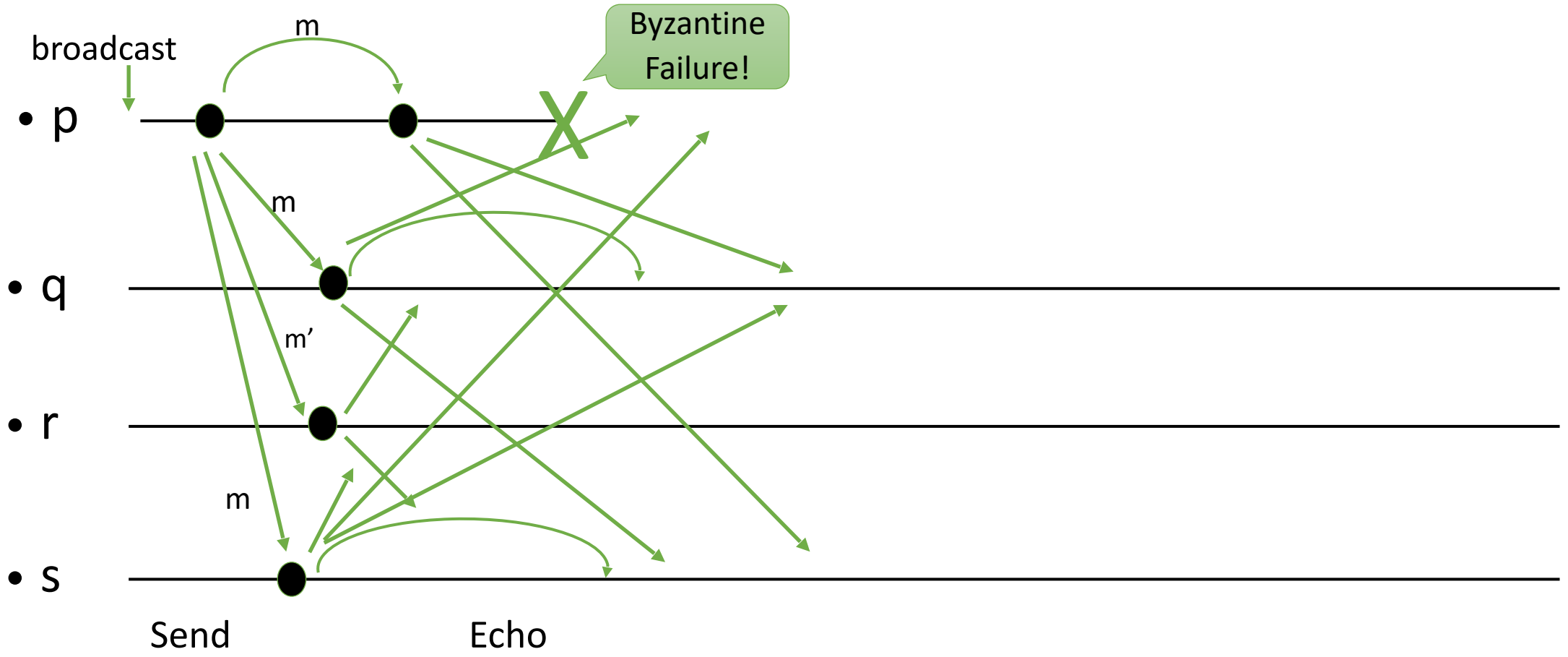


3 rounds; $O(n^2)$ messages; $O(n^2 |m|)$ communication

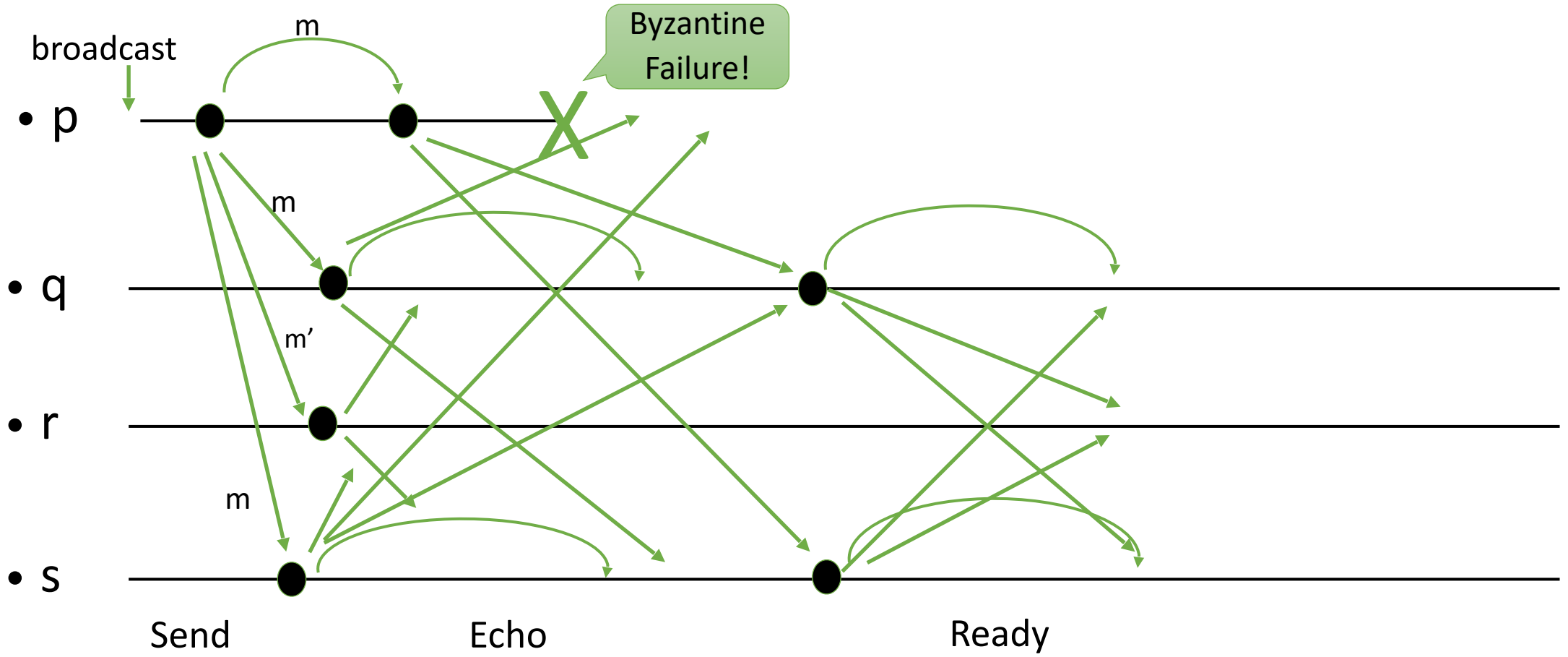
Example



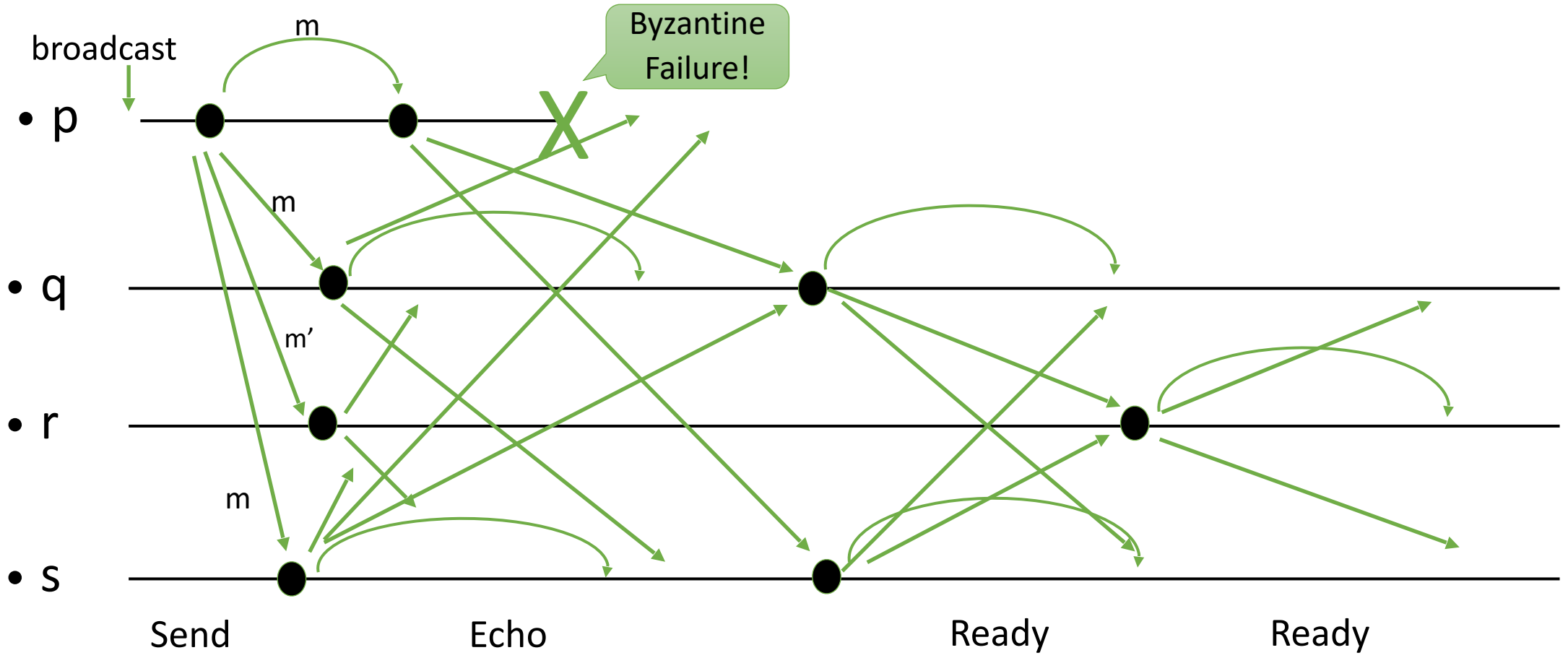
Example



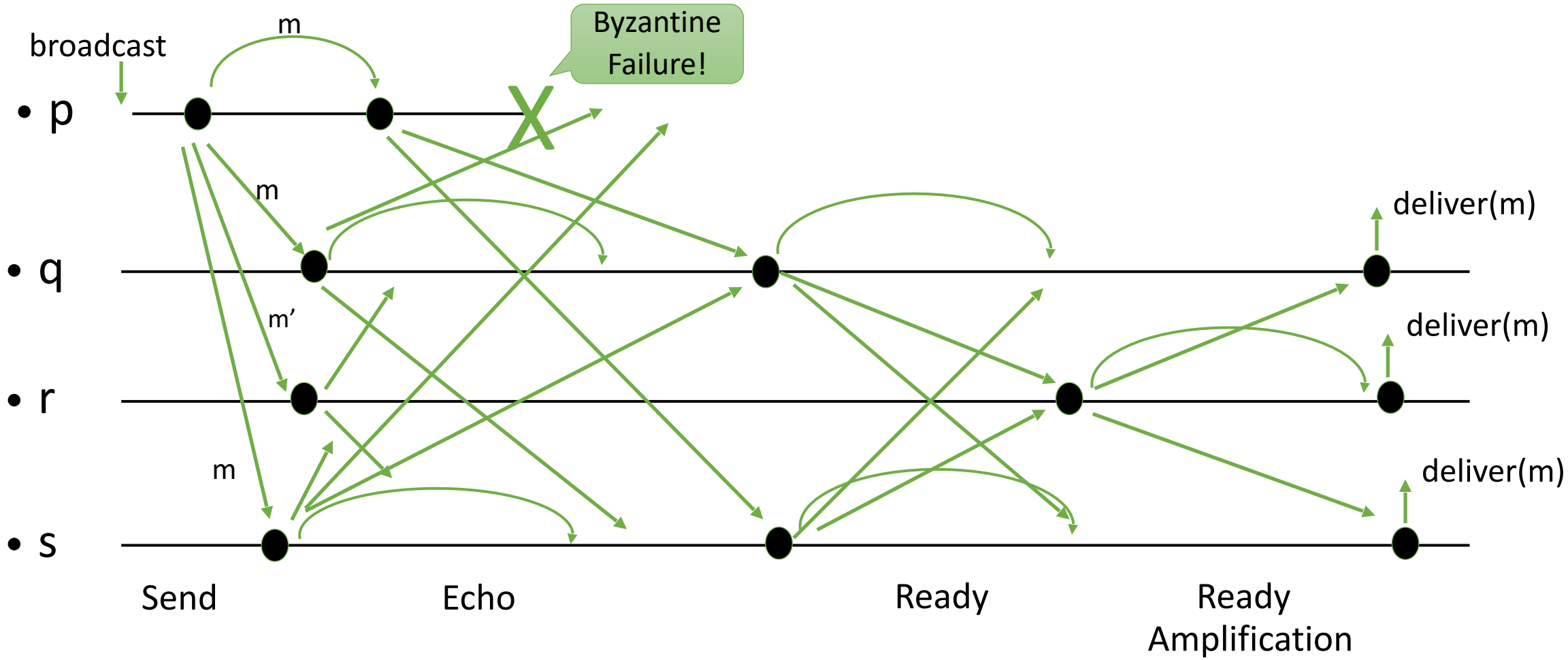
Example



Example


broadcast
m
Byzantine Failure!
• p
m
• q
m'
• r
m
• s
Send
Echo
Ready
Ready

Example


broadcast
m
Byzantine Failure!
• p
m
• q
m'
• r
m
• s
deliver(m)
deliver(m)
deliver(m)
Send
Echo
Ready
Ready Amplification

Authenticated Double-Echo Broadcast (Bracha)

Implements brb, **uses** abeb, with sender s (N>3f)

sent-ready := **false**

upon <brb, broadcast(m)> **do**
    trigger <abeb, broadcast(Send[m])>

upon <abeb, deliver(s, Send(m))> **do**
    trigger <abeb, broadcast(Echo[m])>

upon <abeb, deliver(p, Echo(m))> **do**
    echo[p] := m
    if ∃m : #{p | echo[p]=m} > 2f+1 ∧ ¬sent-ready **then**
        sent-ready := **true**
        trigger <abeb, broadcast(Ready(m))>

Authenticated Double-Echo Broadcast (Bracha)

upon <abeb, deliver(p, Ready(m))> **do**
    ready[p] := m
    if ∃m : #{p | ready[p]=m} > f ∧ ¬sent-ready **then**
        ▷ amplification of READY messages
        sent-ready := **true**
        trigger <abeb, broadcast(Ready(m))>
    else if ∃m : #{p | ready[p]=m} > 2f+1 **then**
        trigger <Deliver(s, m)>

Note: some code to prevent duplicate execution is omitted.

Amplification

Totality by Amplification:

- A correct process has delivered the message. In the 2f+1 processes that it received Ready from, there are at least f+1 correct processes. These correct processes send Ready to all processes.
- After receiving f+1 Ready messages, all correct processes amplify the Ready message if they have not already sent it. There are at least 2f+1 correct processes, and they all send the Ready message.
- Therefore, each one of them receives Ready from 2f+1 processes and delivers the message.

Byzantine Broadcast Channel

- Implemented by a sequence of one-message instances of Byzantine broadcasts for each process
- Every message is delivered together with a unique label
  - Consistency and totality hold for each label
- Two variants
  - Consistent Channel
  - Reliable Channel

Interface and properties of Byzantine consistent channel

Module:

Name: ByzantineConsistentBroadcastChannel, instance bcch.

Events:

- **Request**: ⟨ bcch , broadcast (m) ⟩: Broadcasts a message m to all processes.
- **Indication**: ⟨ bcch , deliver(p, ℓ, m)⟩: Delivers a message m with label ℓ broadcast by process p.

Properties:

- **BCCH1**: Validity: If a correct process p broadcasts a message m, then every correct process eventually delivers m.
- **BCCH2**: No duplication: For every process p and label ℓ, every correct process delivers at most one message with label ℓ and sender p.
- **BCCH3**: Integrity: If some correct process delivers a message m with sender p, and process p is correct, then m was previously broadcast by p.
- **BCCH4**: Consistency: If some correct process delivers a message m with label ℓ and sender s, and another correct process delivers a message m′ with label ℓ and sender s, then m = m′.

Byzantine Consistent Channel

Implements:

ByzantineConsistentBroadcastChannel, instance bcch.

Uses:

ByzantineConsistentBroadcast (multiple instances).

upon event ⟨ init ⟩ **do**

n := $[0]^N$

ready := **true**

forall p ∈ Π do

Initialize a new instance bcb[p, 0] of ByzantineConsistentBroadcast with sender p

n: It keeps the number of the current broadcast instance for each process.
ready: It is used to wait for the previous broadcast instance to finish.

Byzantine Consistent Channel

upon event ⟨ broadcast(m) ⟩ **such that** ready = **true do**
    trigger ⟨ bcb[**self**, n[**self**]], broadcast(m) ⟩
    ready := **false**

upon event ⟨ bcb[p, n[p]], deliver(p, m) ⟩ **do**
    trigger ⟨ deliver(p, n[p], m) ⟩
    n[p] := n[p] + 1
    Initialize a new instance bcb[p, n[p]] of ByzantineConsistentBroadcast with sender p
    if p = self **then**
        ready := **true**