Berner Fachhochschule - Technik und Informatik

On Public Bulletin Boards

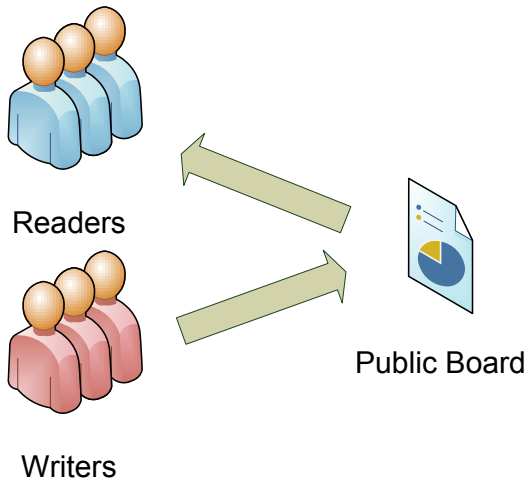
E-Voting Seminar

Eric Dubuis

December 17th, 2010

Problem Statement

Given a set of readers $R = \{r_1, r_2, \ldots\}$, a set of writers $W = \{w_1, w_2, \ldots\}$ posting messages $M = \{m_1, m_2, \ldots\}$ onto a *public bulletin board* B, how can the writers be sure that their messages having been posted appear on the board (and stay thereafter), and how can readers be sure that whenever they read the content of the board at time t_0, and later at time $t_1 \gg t_0$, that the content at t_1 is the same one as at t_0, possibly having some new messages appended?



Solution Strategies

I present two solution strategies:

- single board, certified publishing
- replicated, cooperating boards

Outline

Outline

Single Board, Certified Publishing

Idea:

- accredited writers
- proof of origin

Single Board, Certified Publishing – Properties

Definitions:

Unalterable History A public board has an *unalterable history* if whenever a reader retrieves the contents of the board at time t_0 and again at t_1, it is able to check that the content it read at t_0 is a prefix of the content at t_1.

Certified Publishing A public board has *certified publishing* if whenever a reader retrieves the contents of the board, either he or she can detect corruption of the board, or he or she will have a proof, for each message on the board:

1. of who posted the message;
2. that the writer intended the message to be published with the stated timestamp and at this point in the board's sequence of messages.

Single Board, Certified Publishing – History

The public board stores a sequence of entries $\langle e_1, \ldots, e_n \rangle$, where each $e_i = \langle m_i, t_i, W_i, h_i, \langle h_i \rangle_{K_{W_i}}, \langle \langle h_i \rangle_{K_{W_i}}, t_i' \rangle_{K_B} \rangle$.

$\langle h_i \rangle_{K_{W_i}}$ is the writer's commitment.
$\langle \langle h_i \rangle_{K_{W_i}}, t_i' \rangle_{K_B}$ is the board's commitment.

The following invariants are satisfied:

1. $h_i = h(\langle m_i, t_i, W_i, h_{i-1} \rangle)$, where $h_0 = 0$
2. $t_i \leq t_i' < t_i + \epsilon$

Single Board, Certified Publishing – Reading

Whenever a reader reads from the board, the following message is returned from the board:

1. $B \longrightarrow R : \langle\langle e_1, \ldots, e_n\rangle, \langle h_n, t_B\rangle_{K_B}\rangle$

where $h_n = h(\langle m_n, t_n, W_n, h_{n-1}\rangle)$, $n \geq 1$, is the so-called *state hash*.

Single Board, Certified Publishing – Writing

Whenever a writer writes to the board, the following message exchange occurs:

1. $B \longrightarrow W : \langle h_n, t_B \rangle_{K_B}$
2. $W \longrightarrow B : \langle m, t_W, W, h, \langle h \rangle_{K_W} \rangle$
3. $B \longrightarrow W : \langle \langle h \rangle_{K_W}, t' \rangle_{K_B}$

where $h = h(\langle m, T, W, h_n \rangle)$ the *current state hash*.

If signature and hash are correct, and if t_W is fresh, the board appends

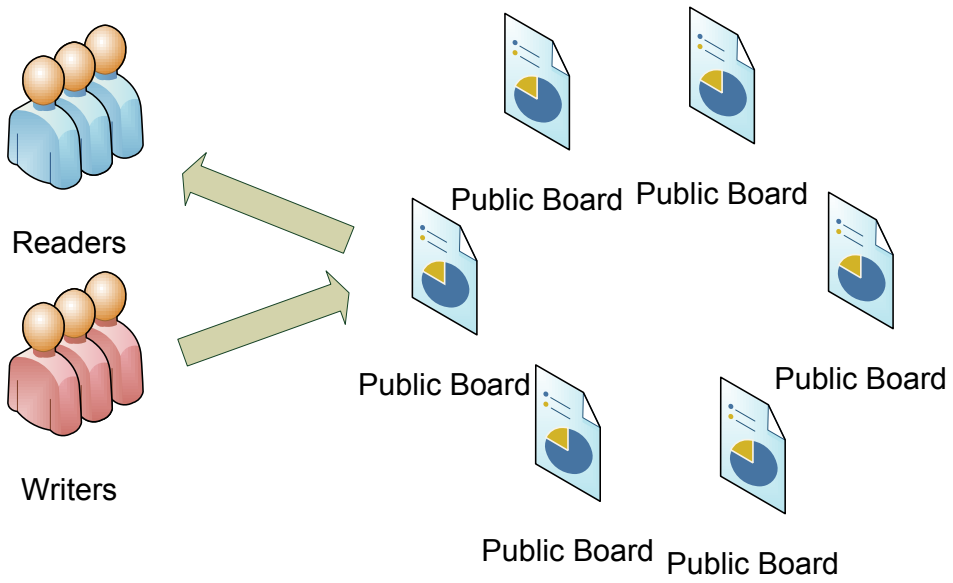
- $\langle m, t_W, h, \langle h \rangle_{K_W}, \langle \langle h \rangle_{K_W}, t' \rangle_{K_B} \rangle$

to the history.

Outline

Replicated Cooperating Boards

In this approach, boards are replicated. They exchange messages such that, in the end, at least $\lceil (2|P|+1)/3 \rceil$ boards have the same content, where P is the set of boards.



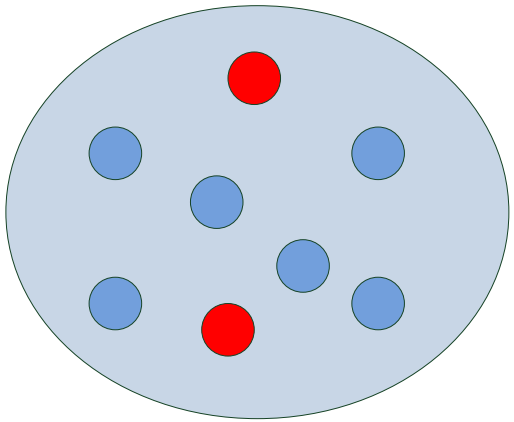
Outline

System Model

processes Set of processes $P = \{p_1, p_2, \ldots\}$, we often use p, q, and r to refer to particular processes

honest process a process that behaves according to its specification

corrupt process a process that behaves in any fashion (byzantine process)

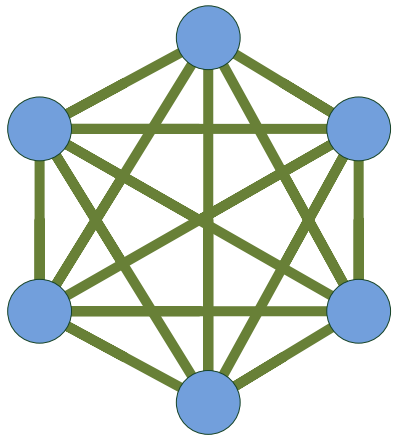


Communication Model

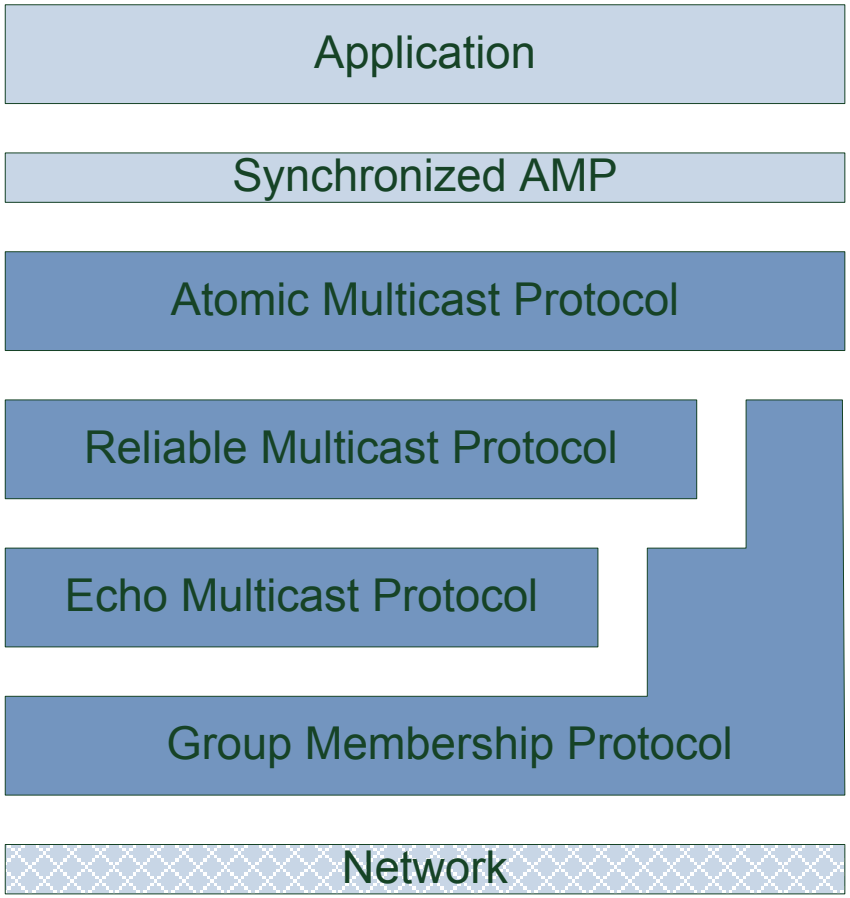
network point-to-point, channels between each pair of processes

channel authenticated, reliable, has integrity protection, FIFO, asynchronous

signed message each process p_i possesses private key K_i; signed message is denoted by $\langle \ldots \rangle_{K_i}$ (each process obtains the public key of other processes as needed)

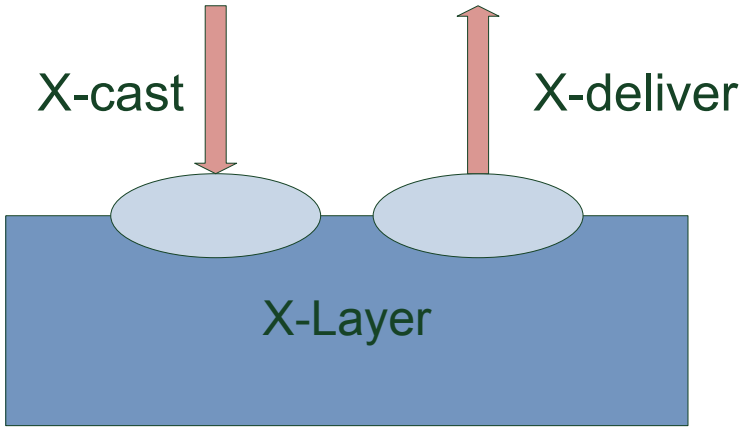


Protocol Layers



Service Primitives at Layers

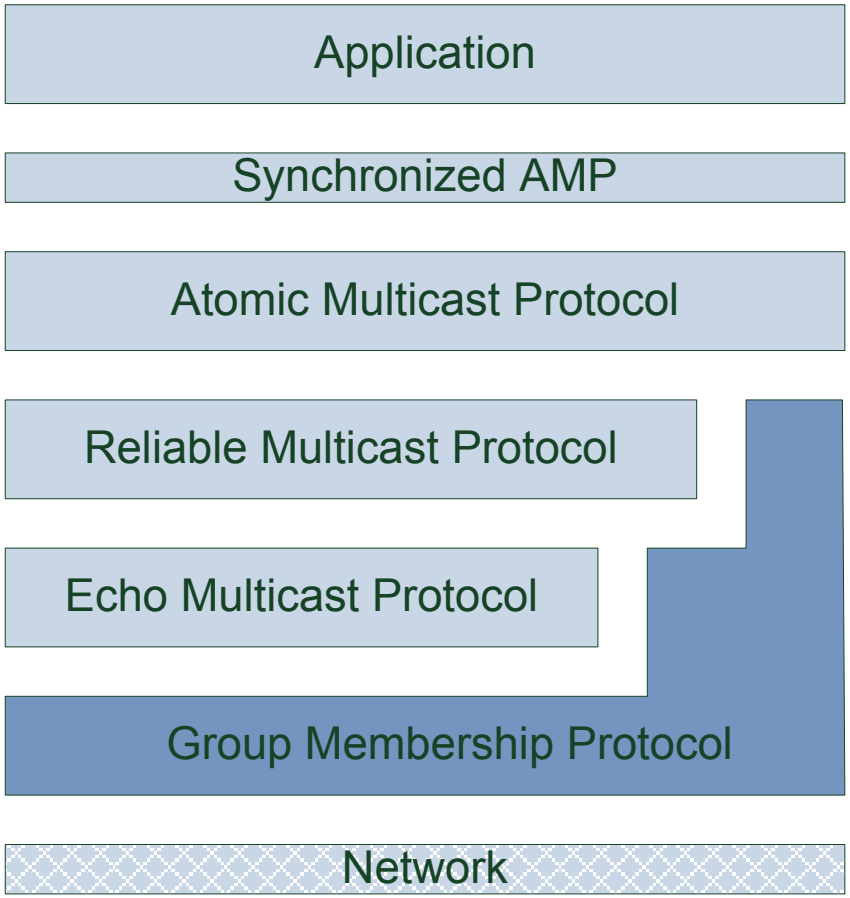
For the layers A (atomic multicast), R (reliable multicast), and E (echo multicast), the following service primitives are available:



Exception: For expressing the services of the Group Membership Protocol, other primitives will be used.

Outline

Group Membership Protocol



Group Membership Protocol

Purpose

- establishes a group view of processes
  (i.e., every process p "sees" the same (presumably) honest members of the group)
- delivery of new group views to upper layer protocols
- adding processes to and removing processes from group views
- perhaps no agreement for common view possible
  (e.g., if process fluctuation occurs to frequently)

Group Membership Protocol (cont'd)

Basic Assumptions

group views	a set of process identifiers V^1, V^2, ...
progress	each V^{x+1} differs from V^x by adding or removing a single process $p \in P$
honest processes	at least $\lceil (2\|V^x\| + 1)/3 \rceil$ processes are honest
corrupt processes	at most $\lfloor (\|V^x\| - 1)/3 \rfloor$ processes are corrupt

Group Membership Protocol (cont'd)

This protocol ensures that

- each honest p receives V^x iff $p \in V^x$
- p receives V^x before V^y if $x < y$ (and p is in both)

Group Membership Protocol (cont'd)

Interfaces (for the rest of this talk) for processes to influence future membership changes

$remove(x, p)$	a member of V^x requests to remove some $p \in V^x$ to form group V^{x+1}
$adds(x)$	a member of V^x enables additions to occur in V^x

Outline

Multicast Semantics

Reliable multicast ensures that all group members deliver the same message. Properties to satisfy are:

Integrity For all p and m, a honest process executes *R-deliver*(p, m) at most once in view x and, if p is honest, only if p executed *R-mcast*(m) in view x.

Agreement If p and q are honest members of V^{x+k} for all $k \geq 0$ and p executes *R-deliver*(r, m) in view x, then q executes *R-deliver*(r, m) in view x.

Validity-1 If p is a honest member of V^{x+k} for all $k \geq 0$, then p executes *R-deliver*$(V^x, .)$

Validity-2 If p and q are honest members of V^{x+k} for all $k \geq 0$ and p executes *R-mcast*(m) in view x, then q executes *R-deliver*(p, m) in view x.

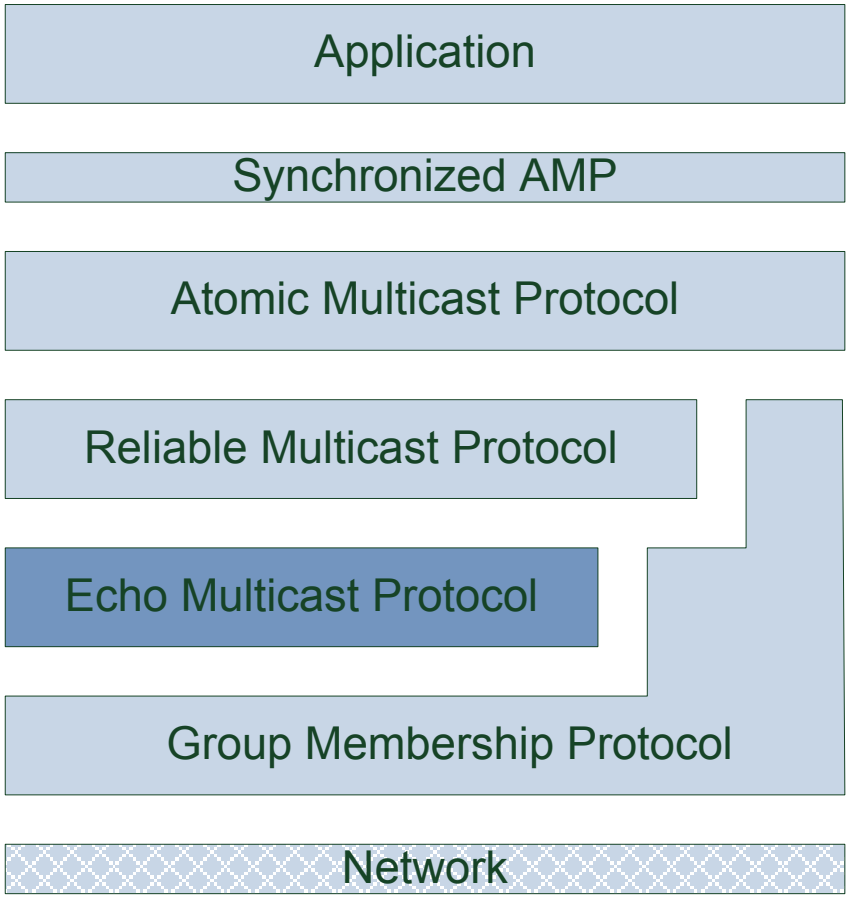
Multicast Semantics (cont'd)

Atomic multicast is built on top of reliable multicast. It adds one additional property:

Order If p and q are honest members of V^{x+k} for all $k > 0$ and p executes *A-deliver*(r, m) before *A-deliver*(r', m') in view x, then q executes *A-deliver*(r, m) before *A-deliver*(r', m') in view x.

Outline

Echo Protocol



Echo Multicast Protocol

Purpose:

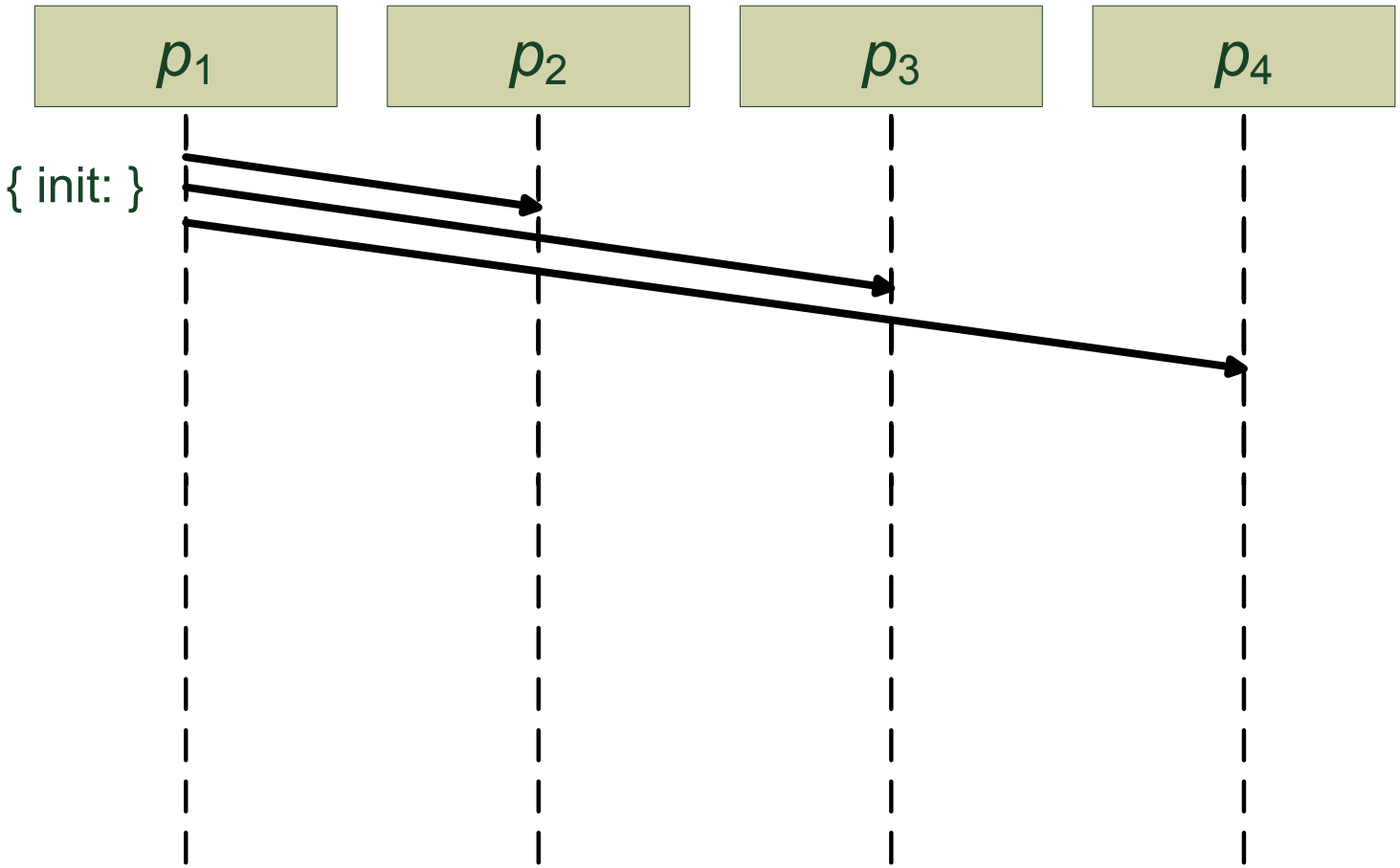
- Enables a $p \in V^x$ to multicast message m to V^x.
- Ensures that every honest process p of the current group view V^x delivers the same message m to the upper layer.

Echo Multicast Protocol – Course of Events

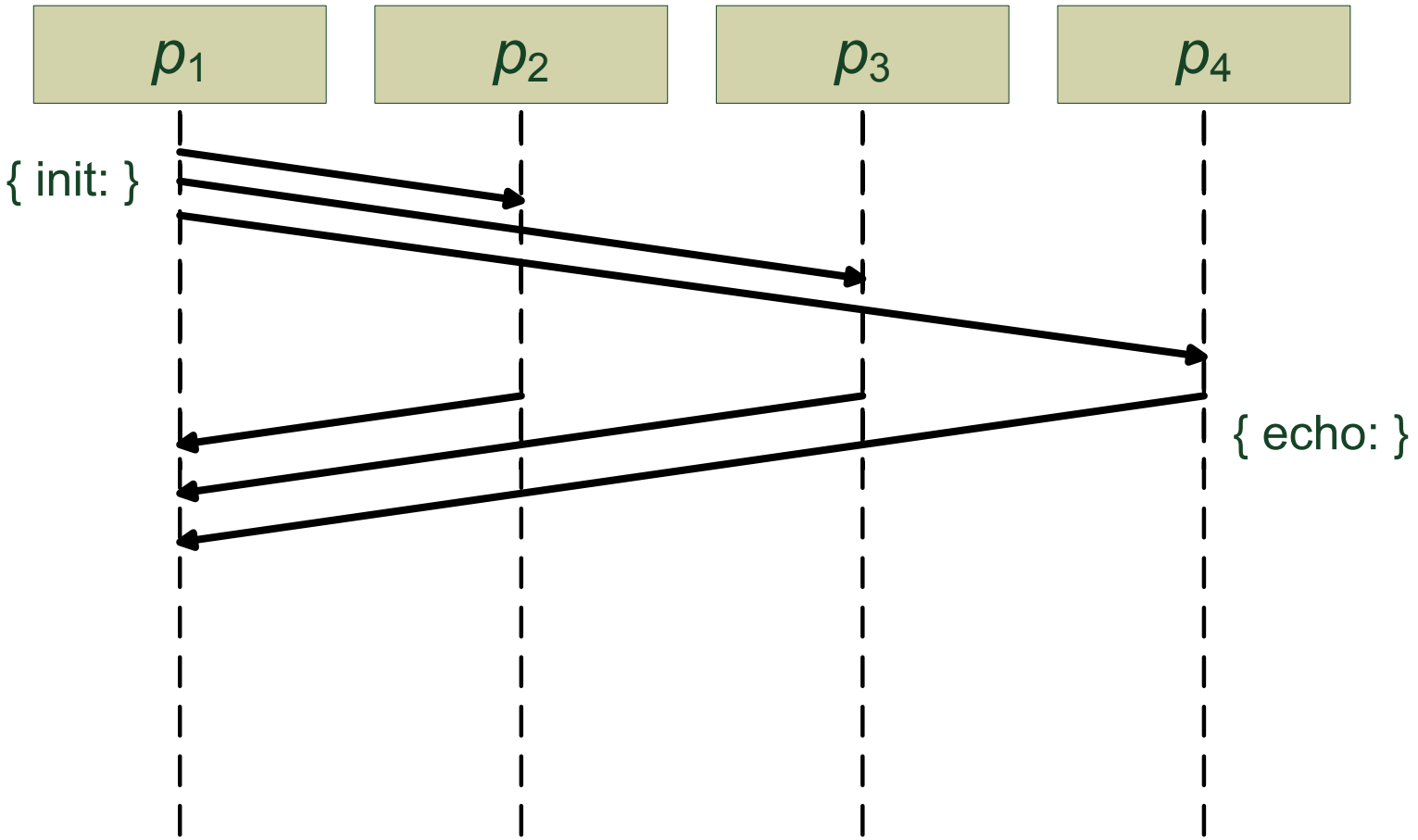
Informal description:

1. p initiates a multicast of a message by sending an "init" message to all p_j
2. each honest p_j returns a signed message, the "echo"
3. after having received enough echoes, p sends them as one message, the "commit"
4. each honest p_j
   - → checks the signatures
   - → checks the view number

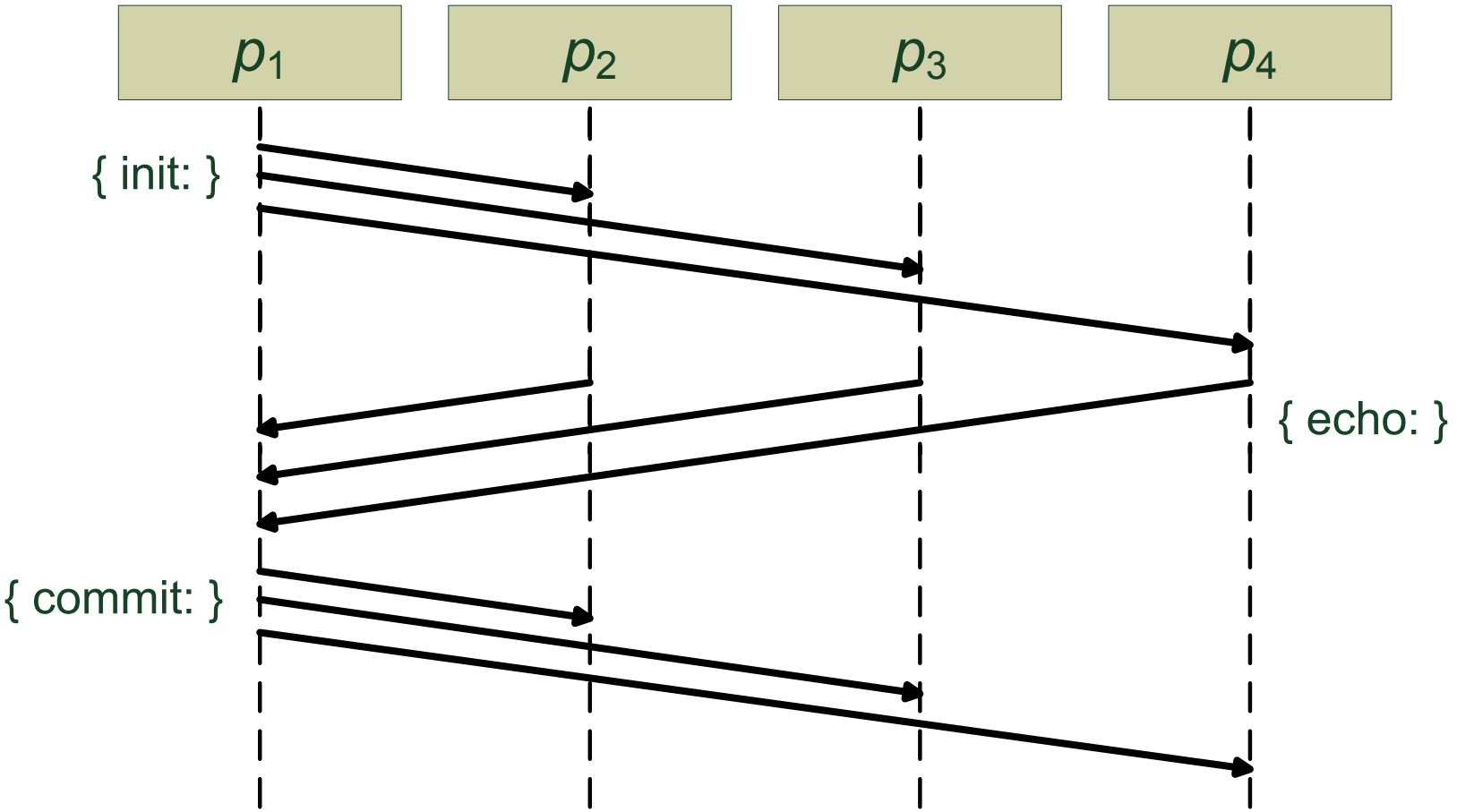
Echo Multicast Protocol – Course of Events



Echo Multicast Protocol – Course of Events



Echo Multicast Protocol – Course of Events



Echo Multicast Protocol – Description

1. $p \longrightarrow p_j \in V^x$: send $\langle \text{init: } x, h(m) \rangle$ where $h(m)$ is a message digest of message m.
2. $p_j \in V^x \longrightarrow p$: send $\langle \text{echo: } p, x, l, h(m) \rangle_{K_j}$ to p where l is the l-th message received from p.
3. $p \longrightarrow p_j \in V^x$: if $P \subseteq V^x$ and $|P| = \lceil (2|V^x| + 1)/3 \rceil$ then send $m_c =$ $\langle \text{commit: } p, x, m, \{\langle \text{echo: } p, x, l, h(m) \rangle_{K_j}\}_{p_j \in P} \rangle$ to p.
4. p_j: if $l > c_i^x$, $P \subseteq V^x$ and $|P| = \lceil (2|V^x| + 1)/3 \rceil$ then $commits_x \leftarrow m_c \cup commits_x$ where $commits_x$ is a set of messages and c_i^x is an element of a set of counters maintained by p_j.
5. p_j: $\forall m_c \in commits_x$ such that $c_i^x + 1 = l$, execute $E\text{-}deliver(p_i, x, m)$ and $c_i^x \leftarrow c_i^x + 1$.

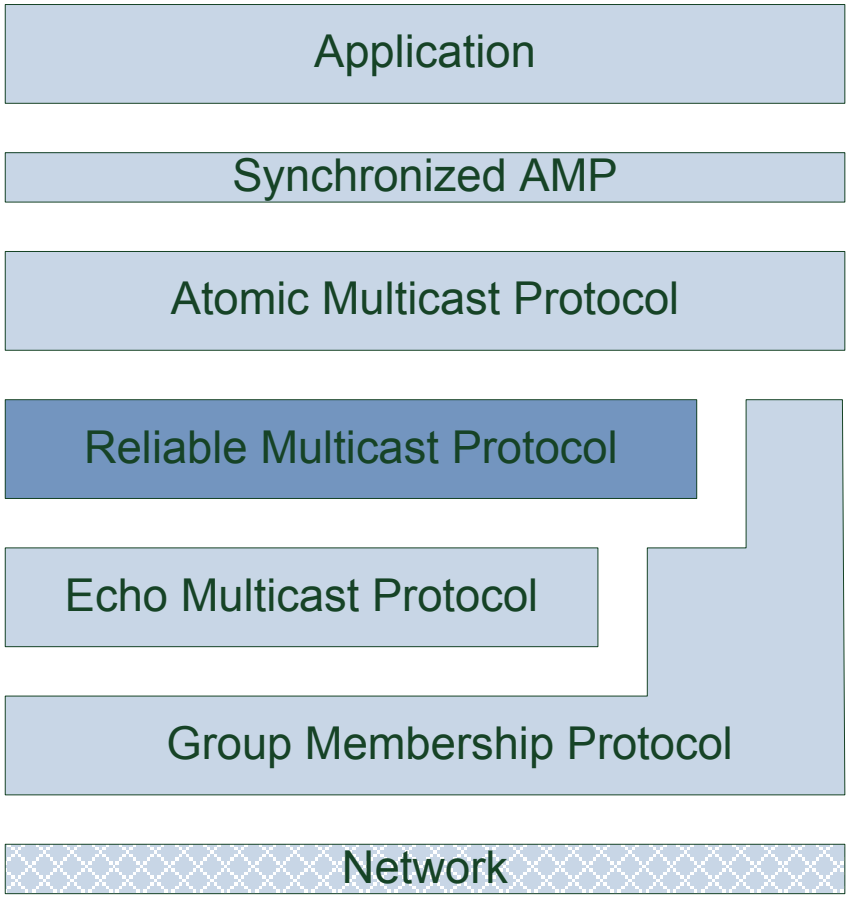
Echo Multicast Protocol – Stability

Informal presentation:

- each process q periodically echo multicasts its set of counters $\{c_i^x\}_{p_i \in V^x}$ to each other process p_j
- a process removes (and delivers) a message from $commits_x$ if and only if $c_i^x \geq l$ at all honest members of V^x.
- a process does not permit to a multicast from process r to remain unstable for longer than a specified timeout duration
  - → it tries to make it stable, or
  - → it executes $remove(x, r)$

Outline

Reliable Multicast

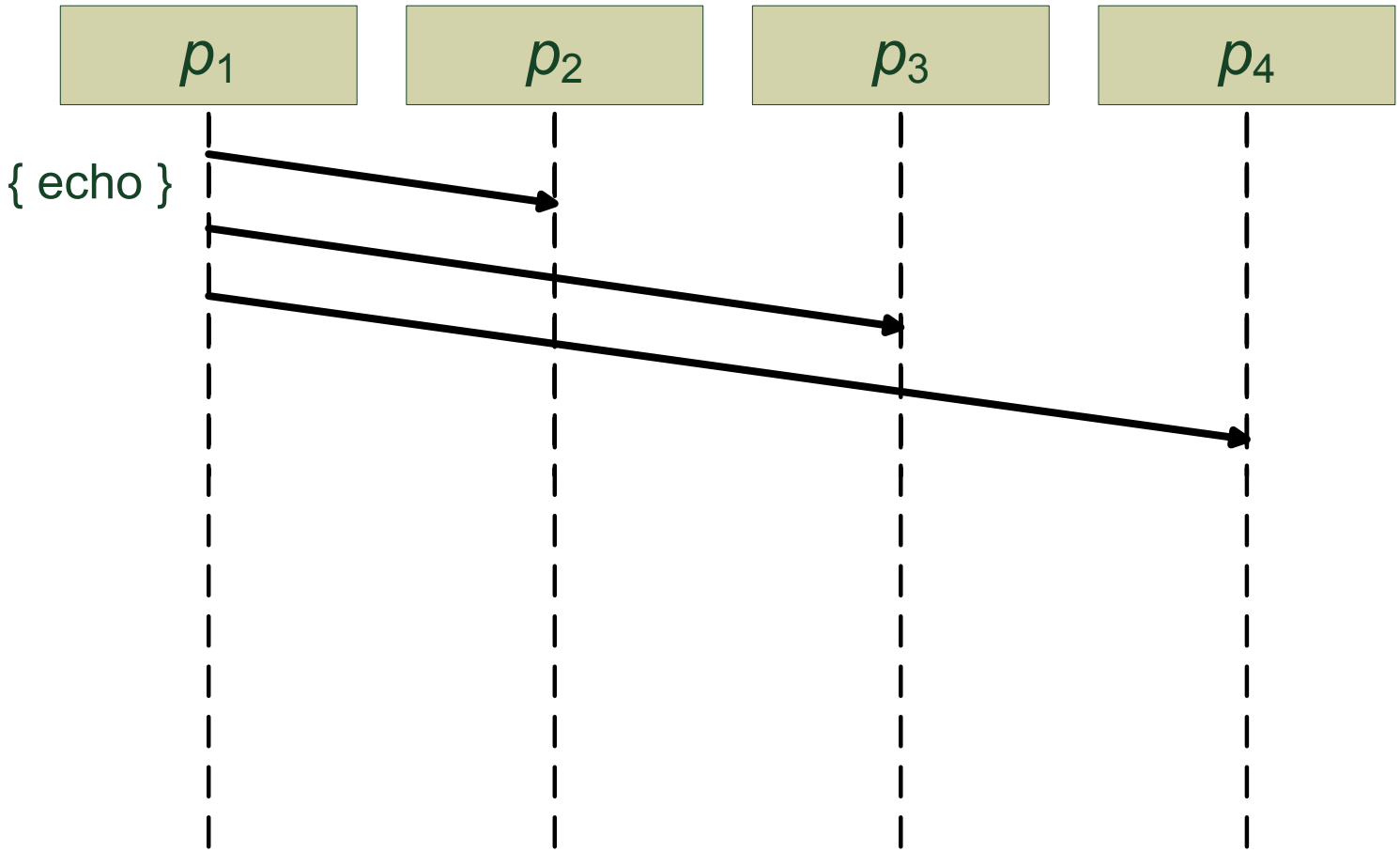


Reliable Multicast Protocol

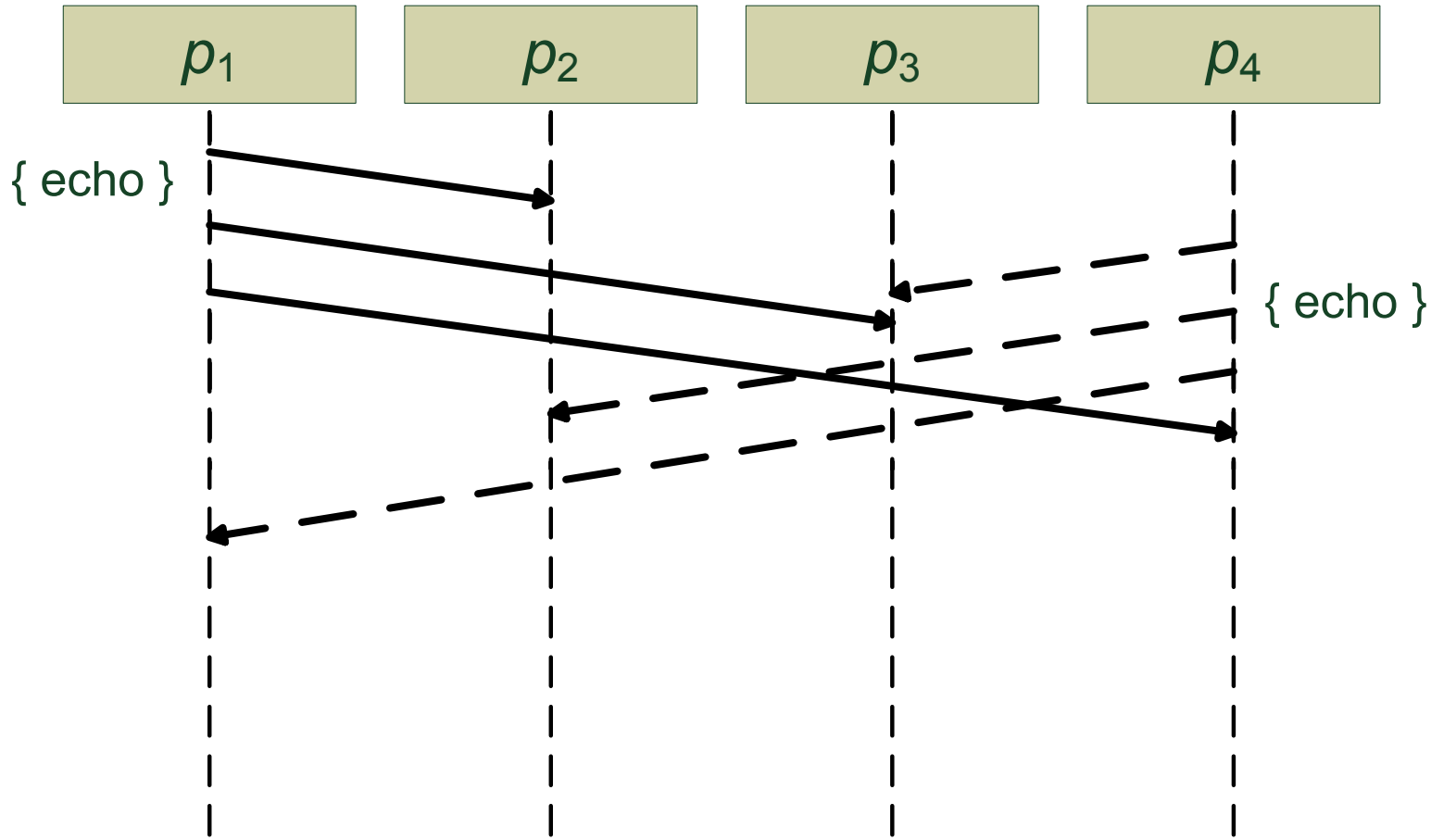
Purpose:

- All members $p \in V^x$ receive the same messages (not necessarily having the same order) despite the fact that there are multicast messages of malicious processes.
- Ensures common group views V^x.

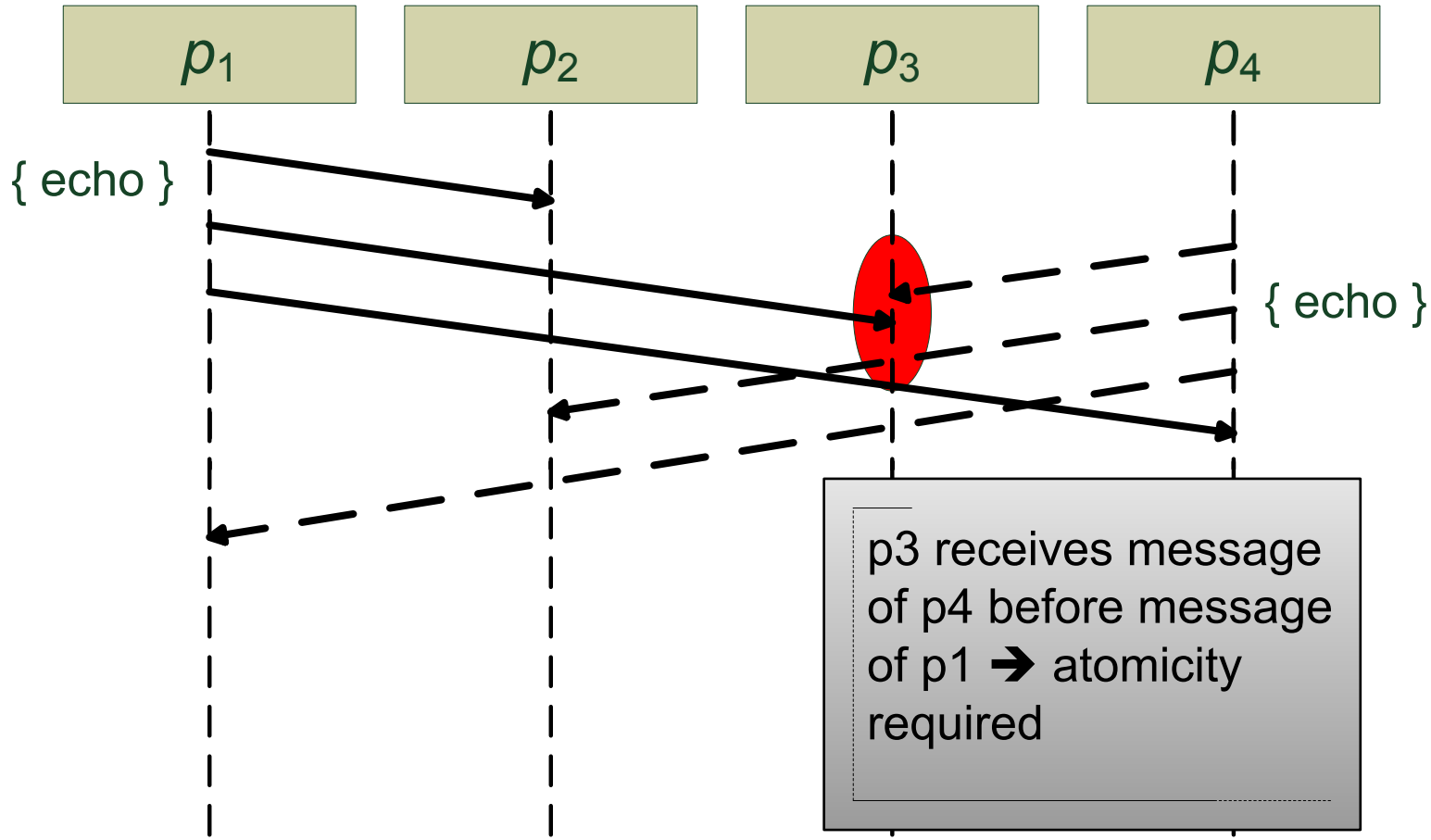
Reliable Multicast Protocol – Course of Events



Reliable Multicast Protocol – Course of Events



Reliable Multicast Protocol – Course of Events



Reliable Multicast Protocol – Messages

- The message is either
  - → a multicast message m of a $p_j \in V^x$, or
  - → a group view V^x
- Progress may depend on the distance of a group member

Reliable Multicast Protocol – Group View

Two cases:

1. No changes of the group view:
   - → reliable multicast executes the echo protocol
   - → delivers messages that are delivered by the echo protocol
2. Group view is changed.

Reliable Multicast Protocol – Group Membership Changes

Again, two cases:

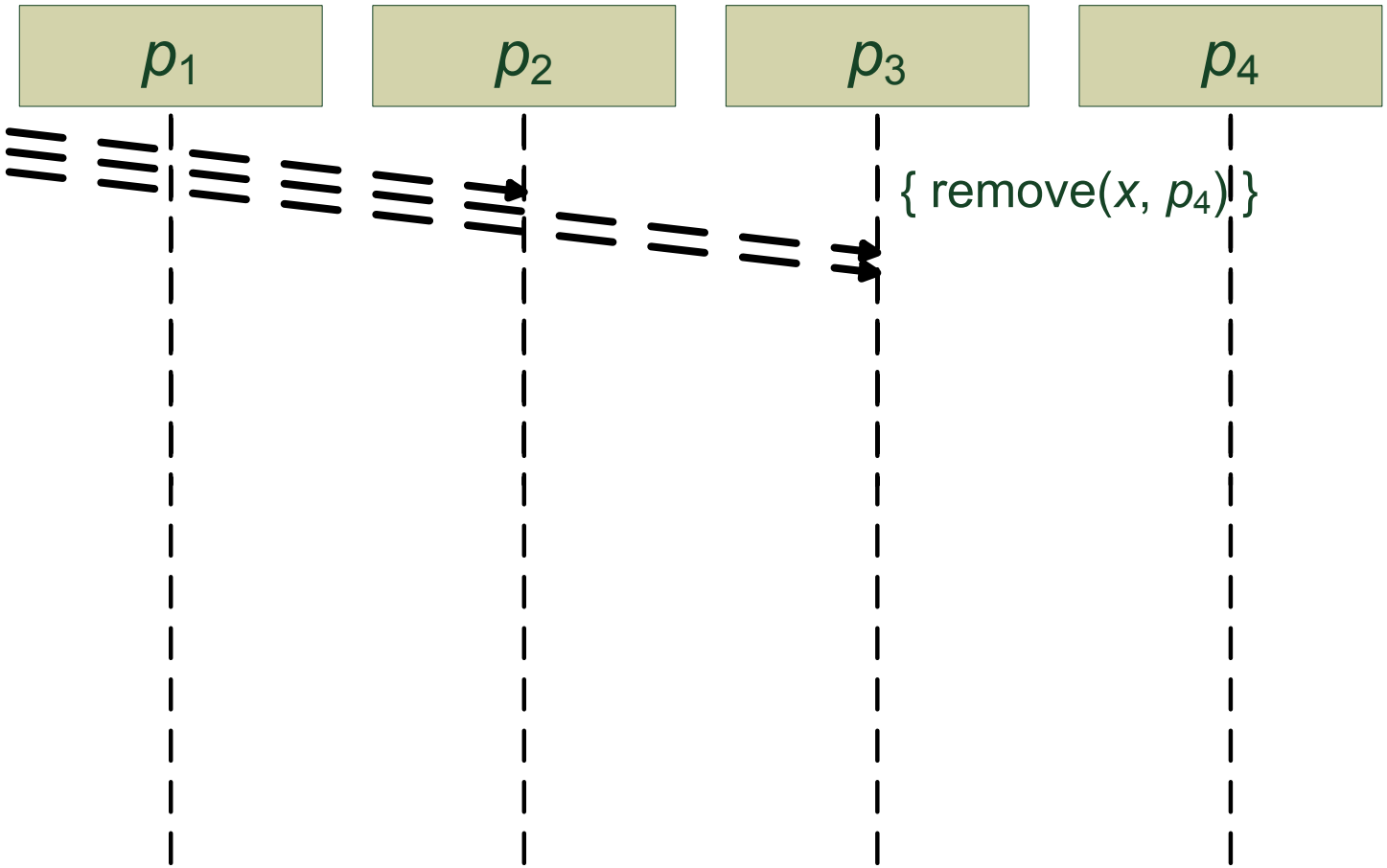
1. No further change during the protocol course
2. Further changes are required if “end” or “flush” messages never received by one or more processes

Reliable Multicast Protocol – Group Membership Changes

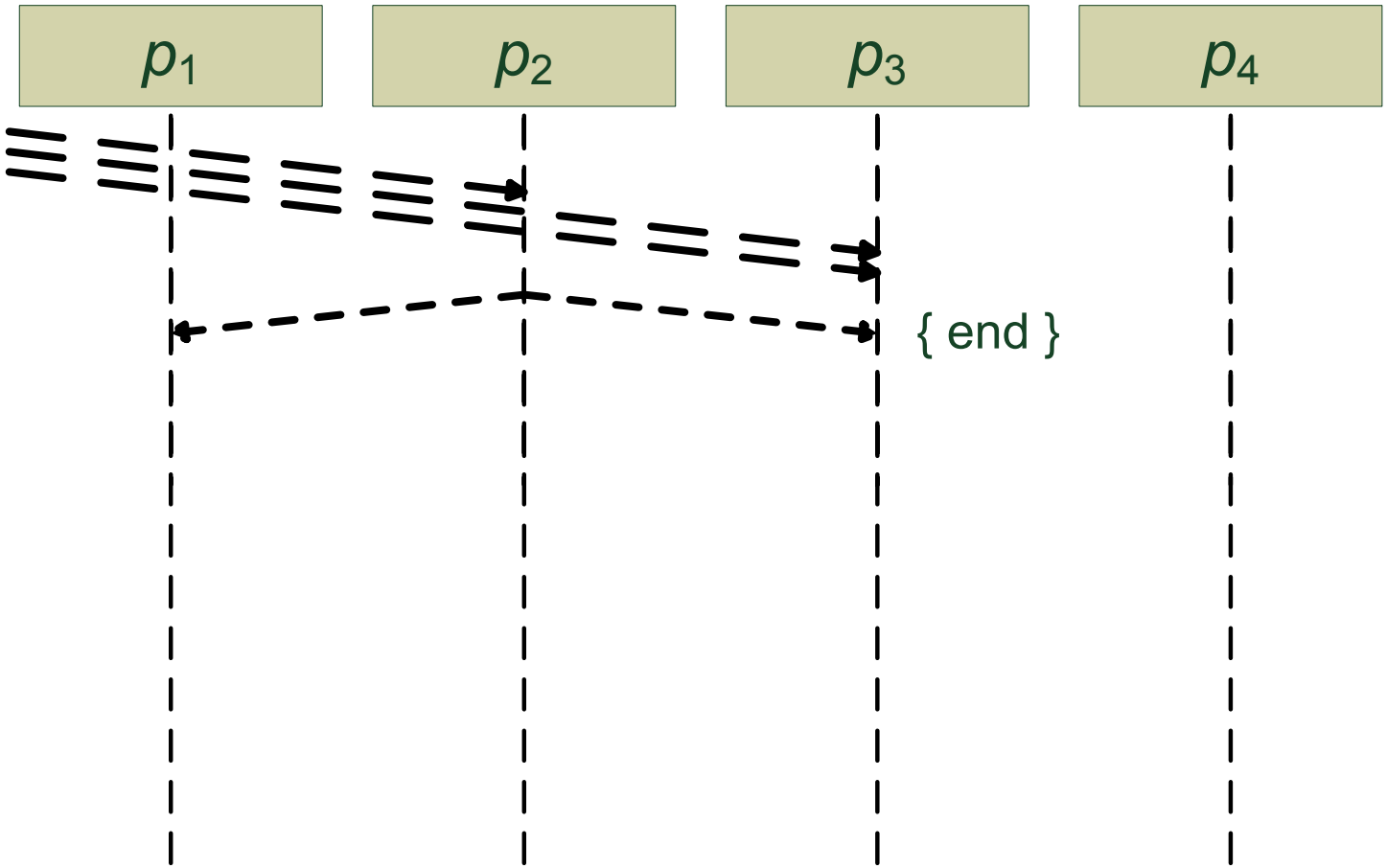
Change of a group membership, *no* change during the protocol course:

1. Multicast of a new group view V^{x+1}.
2. Upon reception, each honest process p:
   - → stops sending multicast messages
   - → sends an "end" message
   - → awaits the "end" messages of other processes
3. After having received the "end" messages:
   - → sends a "flush" message
   - → awaits the "flush" messages of other processes
4. Delivers the new group view V^{x+1} to the upper layer.
5. Resumes the reliable multicast.

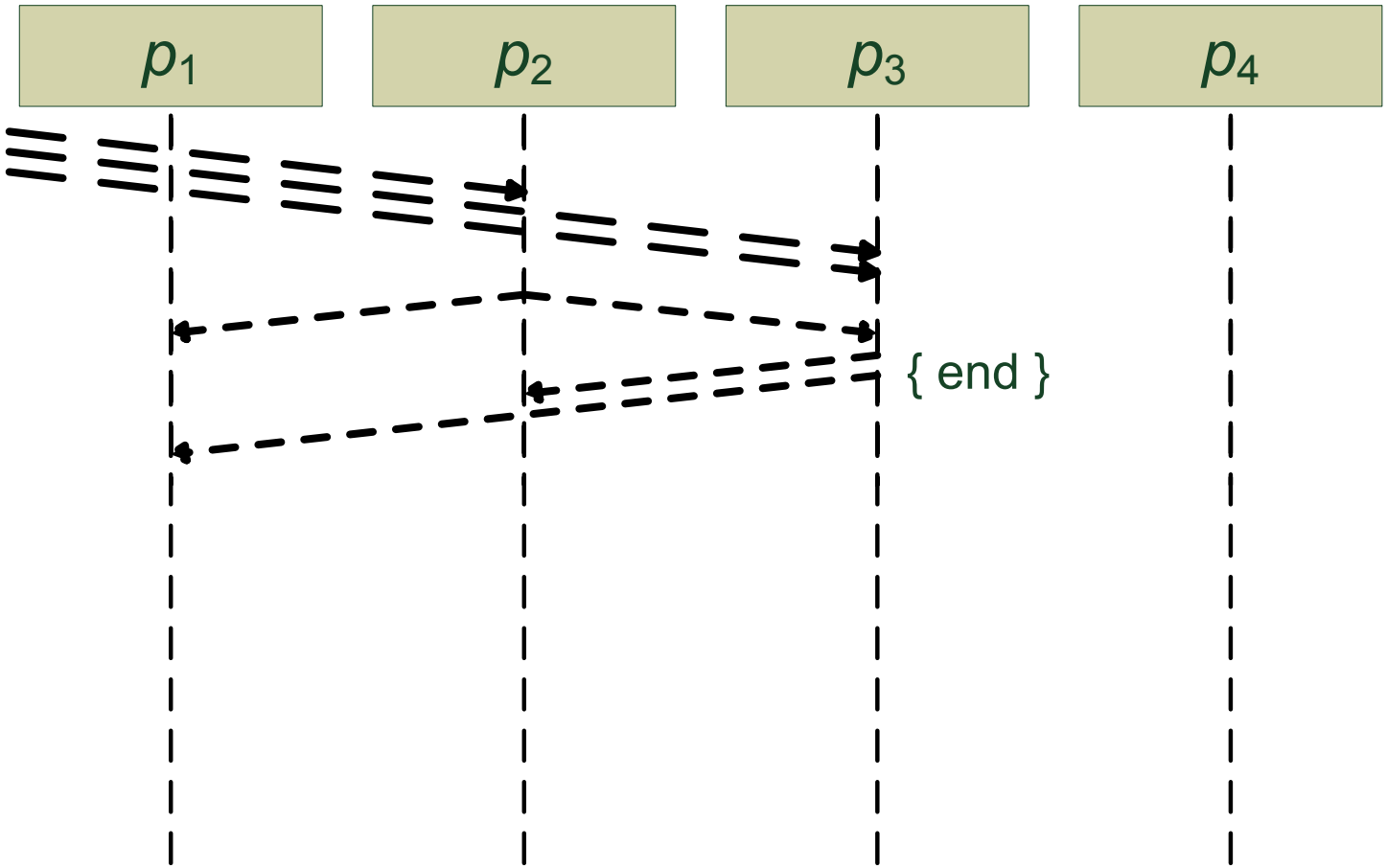
Reliable Multicast Protocol – Group Membership Change



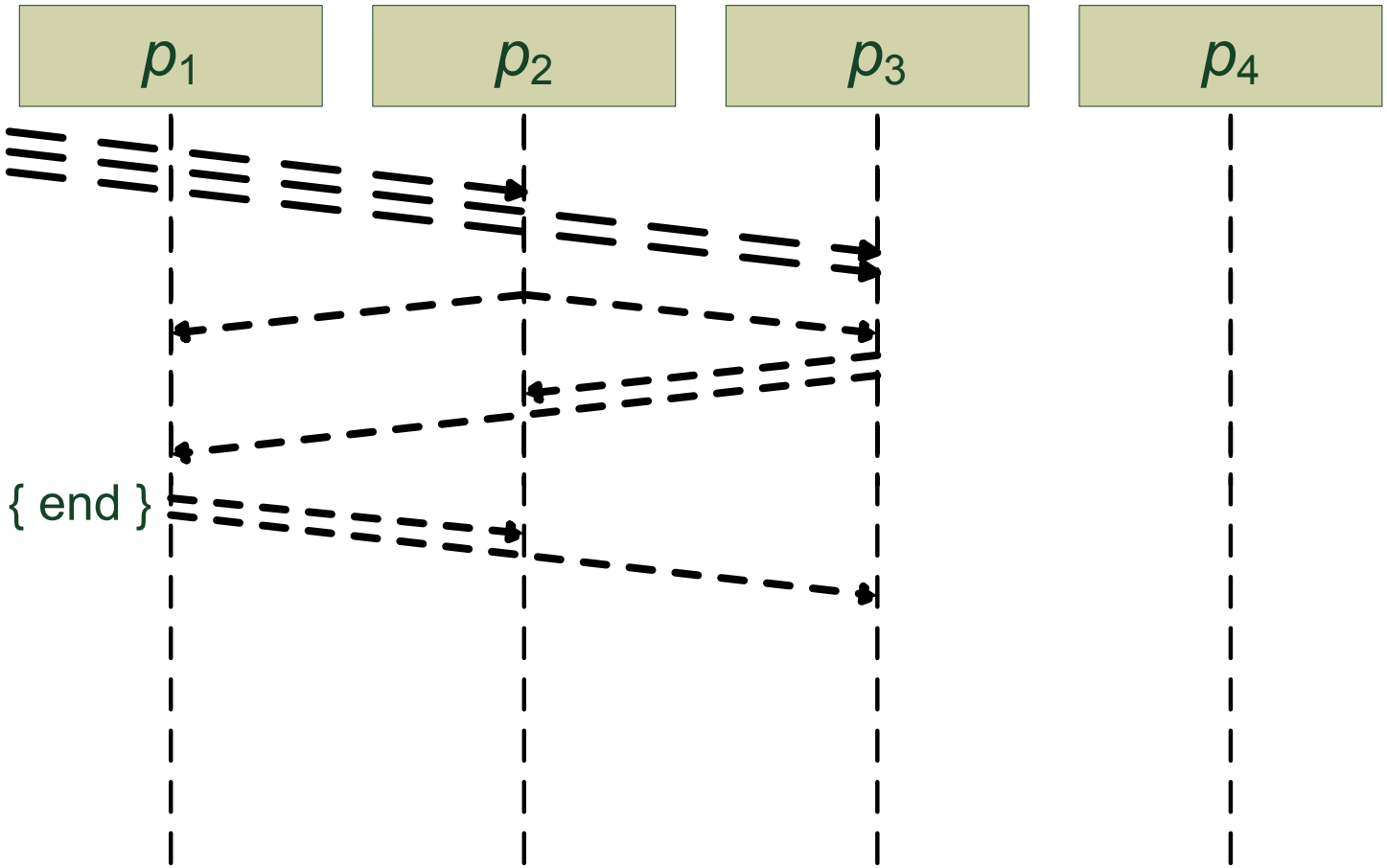
Reliable Multicast Protocol – Group Membership Change



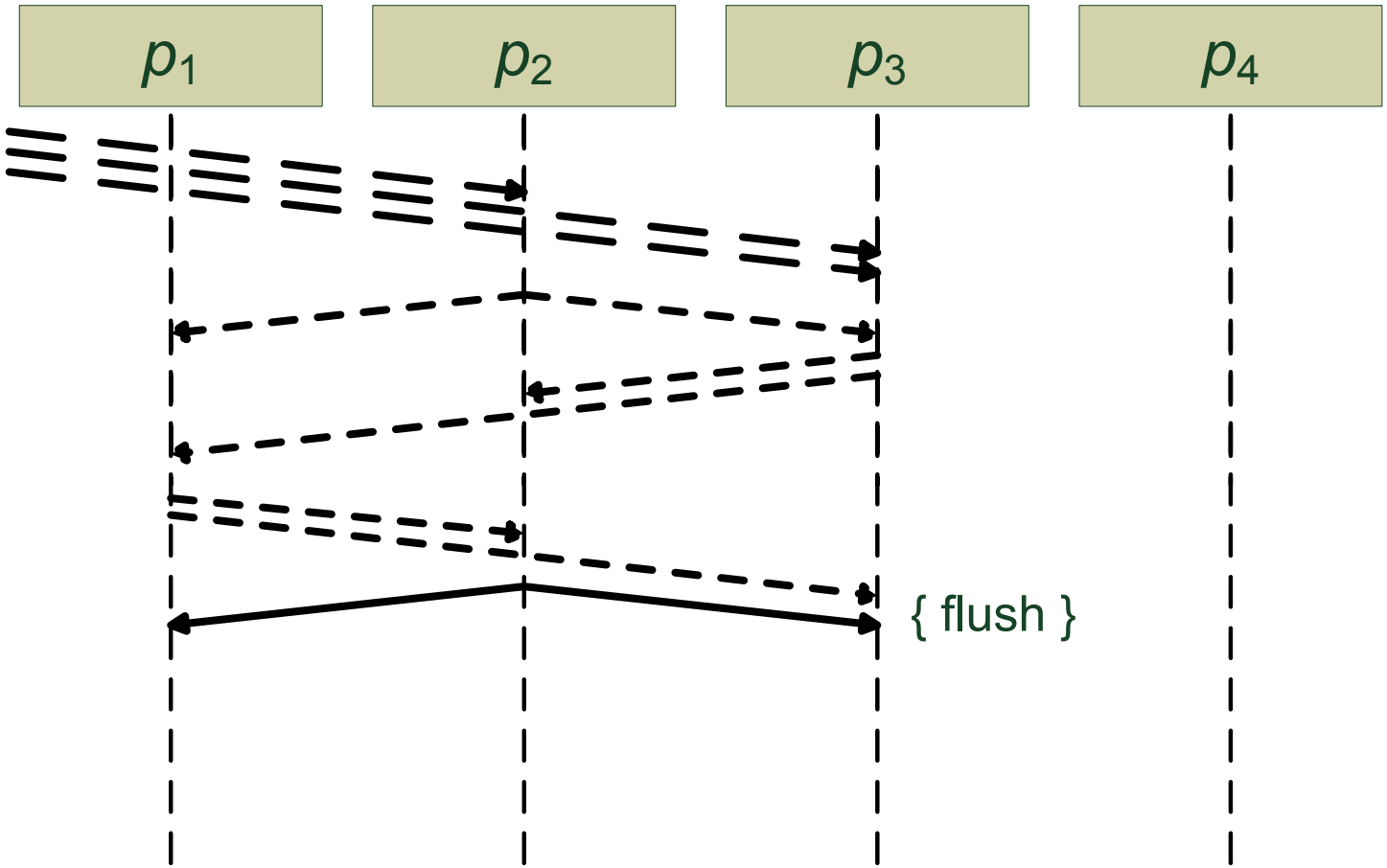
Reliable Multicast Protocol – Group Membership Change



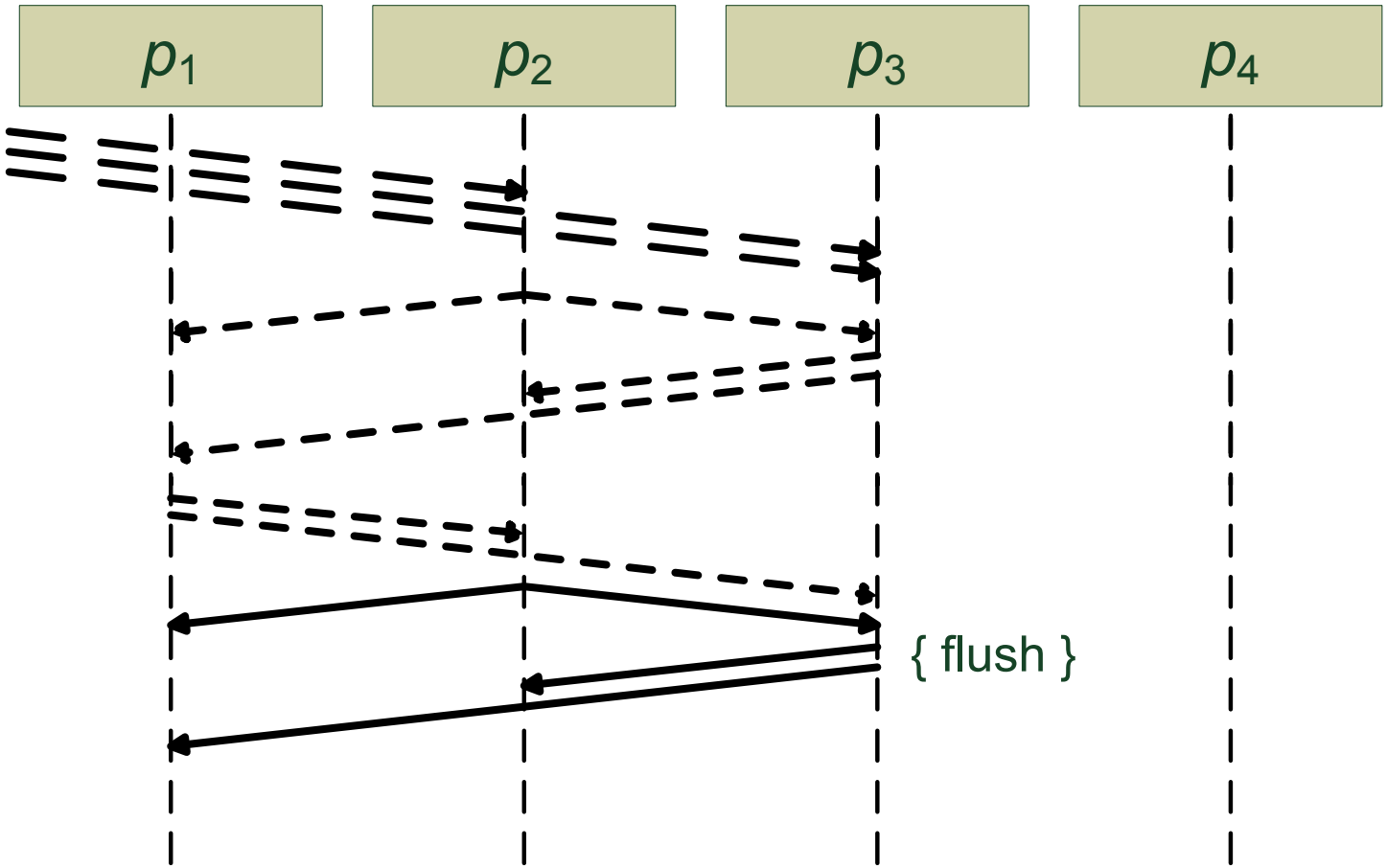
Reliable Multicast Protocol – Group Membership Change



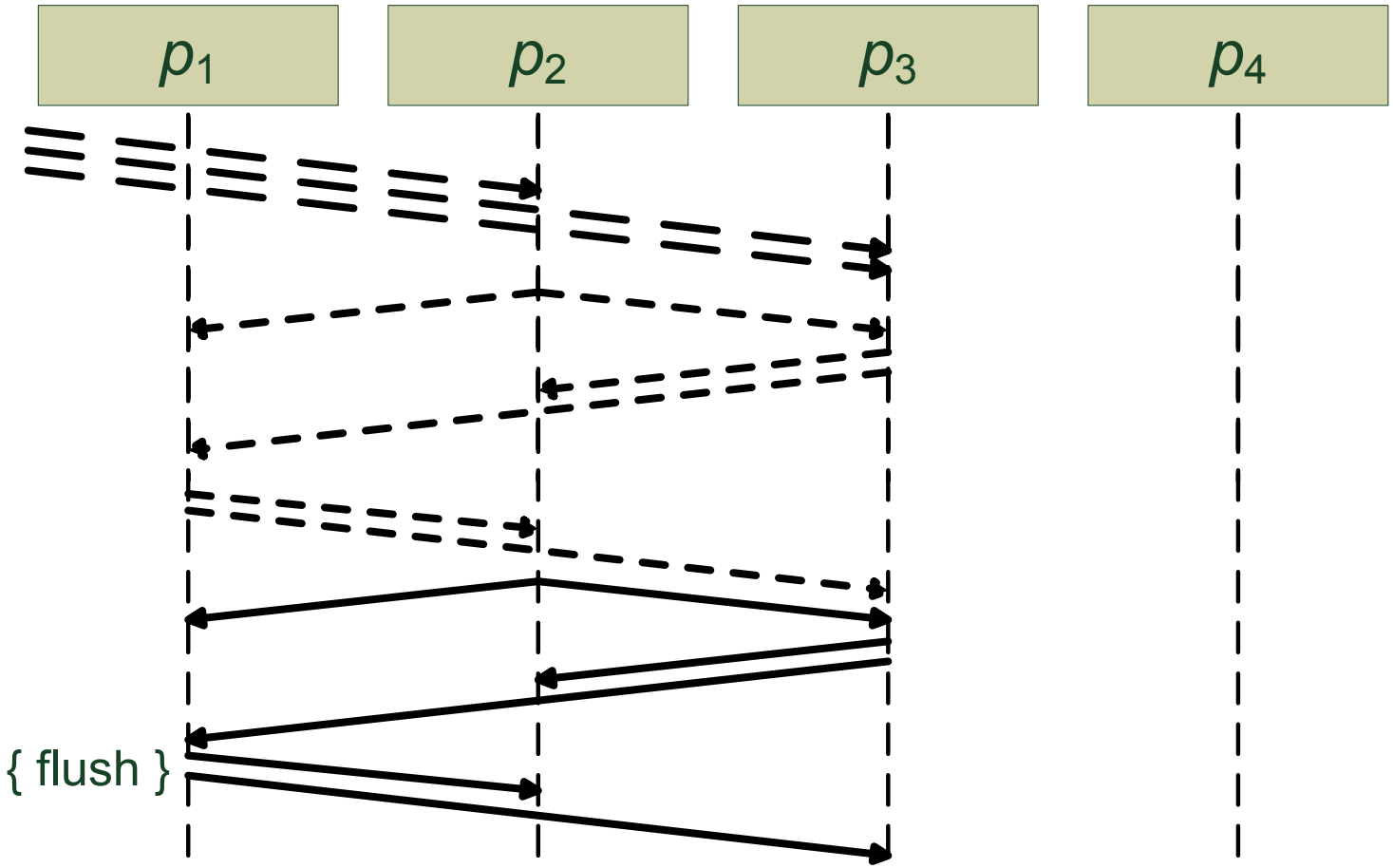
Reliable Multicast Protocol – Group Membership Change



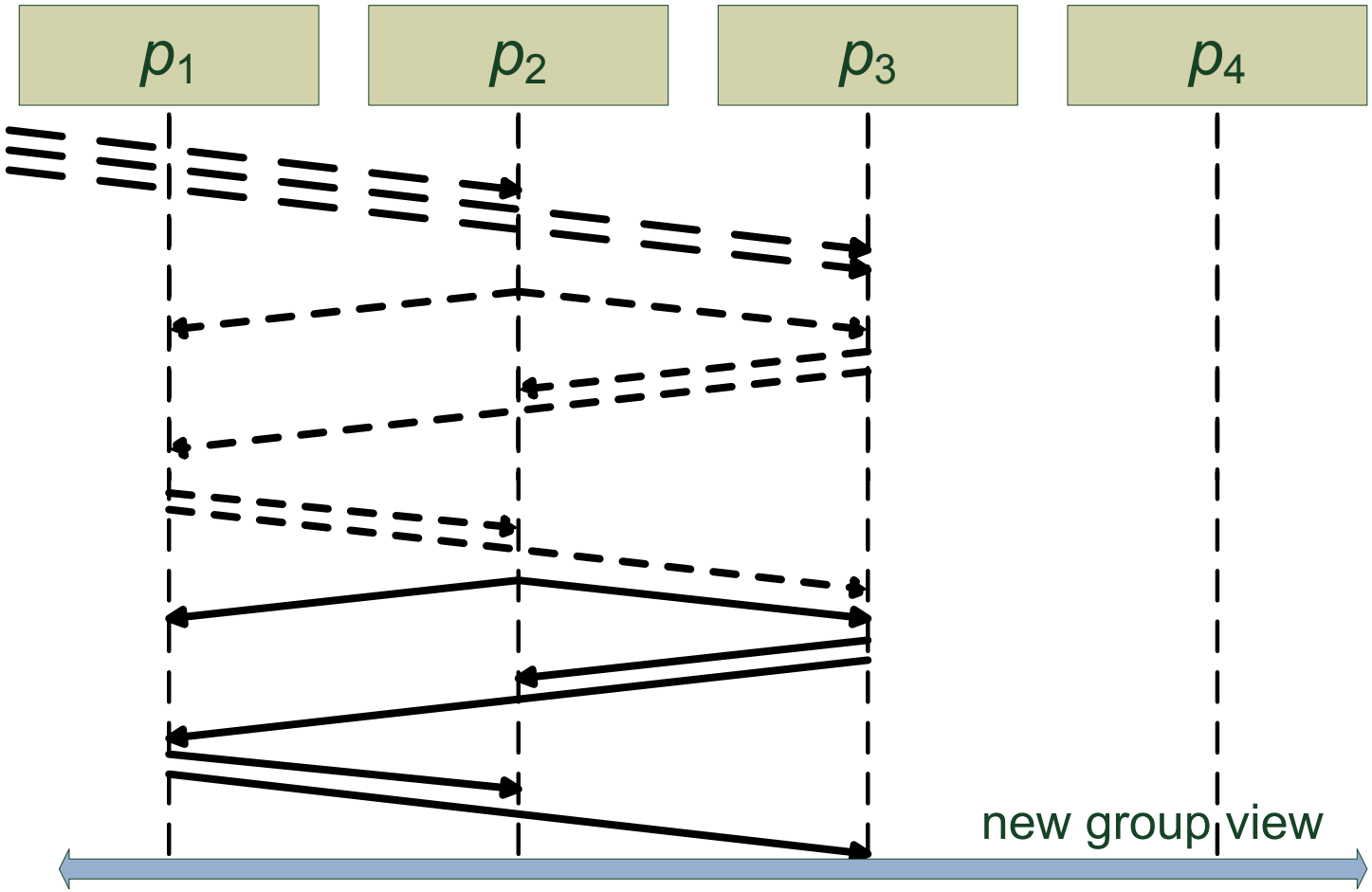
Reliable Multicast Protocol – Group Membership Change



Reliable Multicast Protocol – Group Membership Change



Reliable Multicast Protocol – Group Membership Change



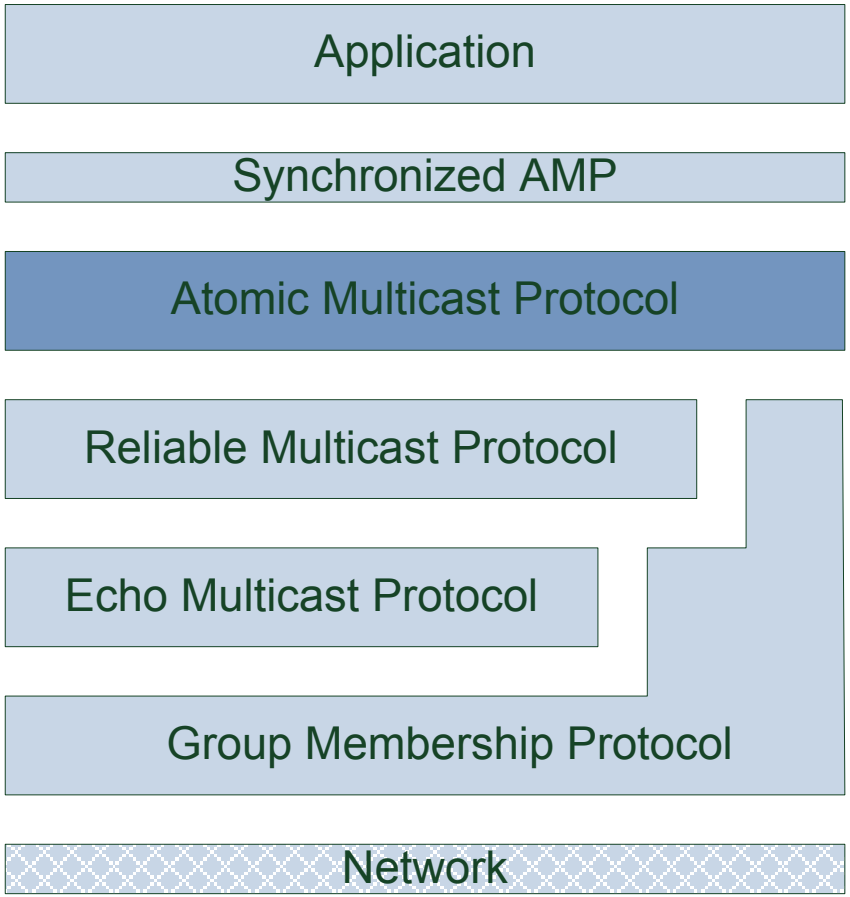
Reliable Multicast Protocol – Group Membership Changes

Change of a group membership *and* change during the protocol course:

1. Multicast of a new group view V^{x+1}.
2. Accept no new members.
3. Remove non-acting and non-reaching members, establish new view V^{x+k}.
4. Repeat until stable group view can be delivered to upper layer.

Outline

Atomic Multicast

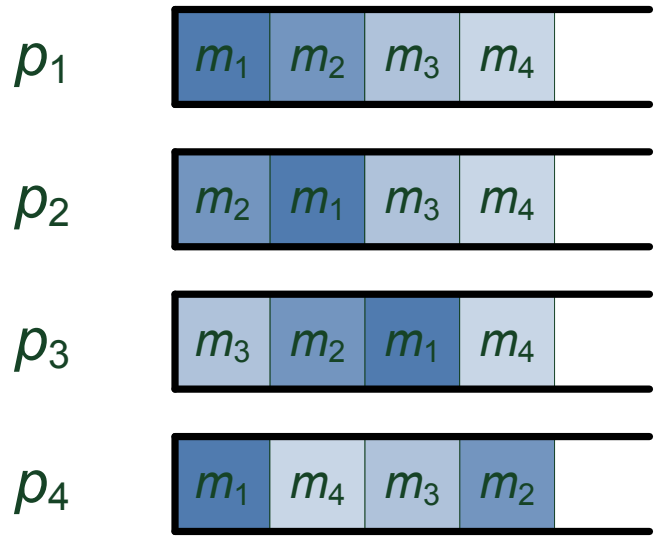


Atomic Multicast

Purpose:

- All honest members of group view V^x deliver the same message having the same order.

Reliable multicast yields:



Atomic Multicast – Sequencer

- A designated member of V^x, the *sequencer*, is deterministically chosen.
- The sequencer determines the order of the messages delivered to the upper layer.

Atomic Multicast – Course of Events

1. To A-mcast a message, a process simply R-mcasts it.
2. The sequencer chooses A-delivery order, and periodically R-mcasts a special "order" message.
3. A process R-delivering an "order" message then A-delivers the messages, given the indicated order.
4. If a new view V^{x+1} is delivered:
   - 4.1 each p_j A-delivers all remaining messages (no corresponding "order" message received) in any deterministic order
   - 4.2 view V^{x+1} is A-delivered

Atomic Multicast – Corrupt Sequencer

A corrupt sequencer might:

- refuse to send "order" messages
- tamper with "order" messages

If a honest process p does not A-deliver a message within some predetermined timeout period after R-delivering it then p requests the sequencer to be removed from the group.

Outline

Summary

- two approaches
  - → certified publishing
  - → cooperating boards
- both approaches can perhaps be combined
- certified publishing seems difficult if anonymous channel needs to be used
- not discussed how clients interact with cooperating boards for writing, reading

Bibliography

- J. Heather, D. Lundin: The Append-Only Web Bulletin Board. In: P. Degano et al. (editors) FAST 2008, LNCS 5491, Springer, 2009.
- M. K. Reiter: Secure Agreement Protocols: Reliable and Atomic Multicast in Rampart. In: Proceedings of the 2nd ACM Conference on Computer and Communications Security, ACM, 1994.
- M. K. Reiter: A Secure Membership Protocol. In: IEEE Transactions on Software Engieering, Vol. 22, No. 1, January 1996.
- M. Pease, R. Shostak, L. Lamport: Reaching Agreement in the Presence of Faults. In: Journal of the ACM, Vol. 27, No. 2, April 1980.
- L. Lamport, R. Shostak, M. Pease: The Byzantine Generals Problem. In: ACM Transactions on Programming Languages and Systems, Vol. 4, No. 3, July 1982.
- R. A. Peters: A Secure Bulletin Board. Master thesis, L. A. M. Schoenmakers, Technical University of Eindhoven, June 2005.