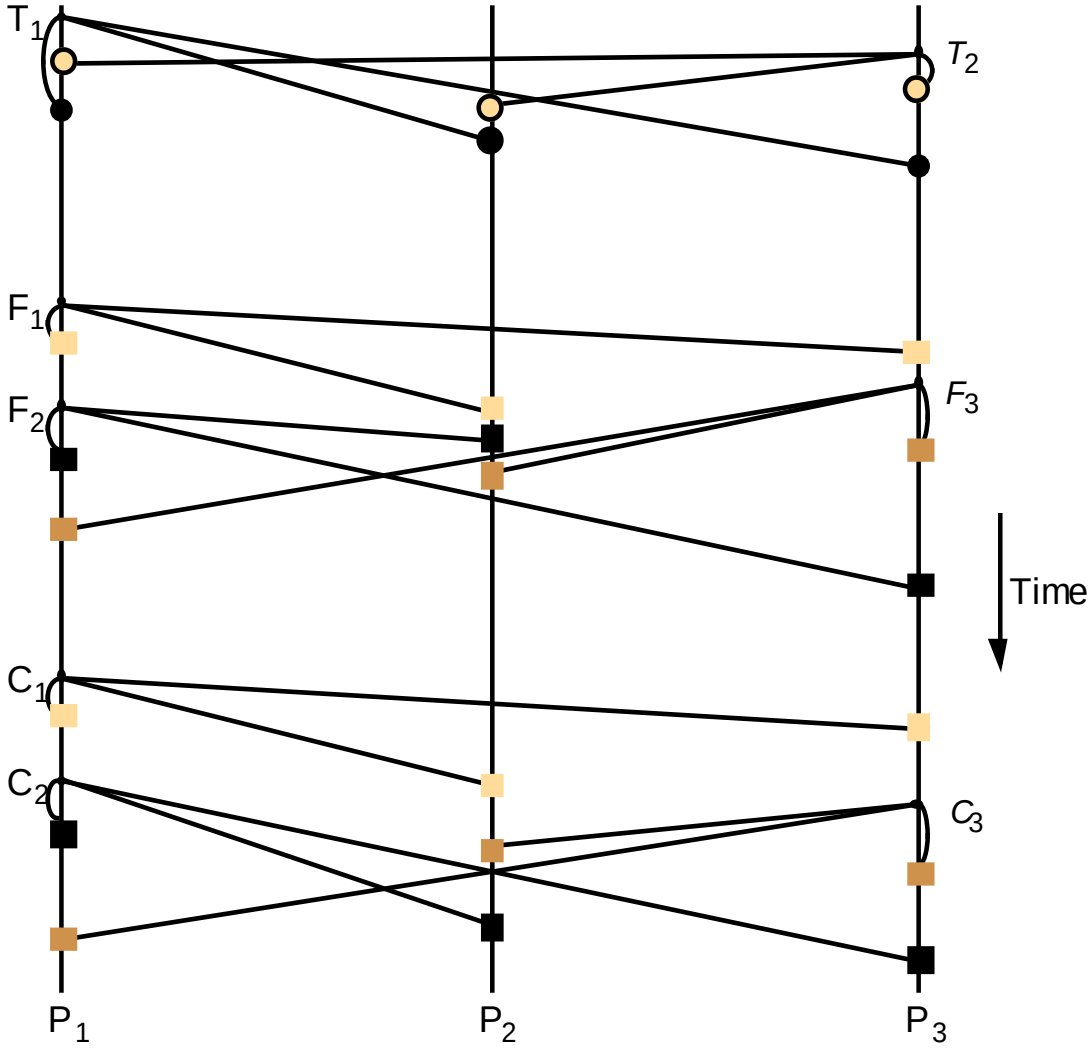
Lecture 30

- Administration

Total, FIFO and causal ordering of multicast messages

Notice the consistent ordering of totally ordered messages T_1 and T_2, the FIFO-related messages F_1 and F_2 and the causally related messages C_1 and C_3 – and the otherwise arbitrary delivery ordering of messages.



FIFO-ordered Multicast

- FIFO-ordered multicast:
  - Assumption:
    - a process belongs to at most one group.
  - Implementation:
    - Local variables at *p*: $S_p = 1$, $R_p[\ |g|\ ]=\{0\}$;
    - ***FO-multicast(g, m)*** *at p:*

      B-multicast(g, <m, S_p>);

      S_p++;
    - *On B-deliver(<m, S>) from q:*

      if(S = $R_p[q]$ + 1)

      FO-deliver(m);

      $R_p[q]$:= S;

      else if(S > $R_p[q]$ + 1)

      place <m, S> in the queue until S = $R_p[q]$ + 1;

      FO-deliver(m);

      $R_p[q]$:= S;

      end if

Total Order Multicast

Total Order Algorithm

1. Algorithm for group member p

On initialization: $r_g := 0$;

To TO-multicast message m to group g
    B-multicast$(g \cup \{sequencer(g)\}, <m, i>)$;

On B-deliver$(<m, i>)$ *with* $g = group(m)$
    Place $<m, i>$ in hold-back queue;

On B-deliver$(m_{order} = <$"order", $i, S>)$ *with* $g = group(m_{order})$
    wait until $<m, i>$ in hold-back queue and $S = r_g$;
    TO-deliver m; // (after deleting it from the hold-back queue)
    $r_g = S + 1$;

2. Algorithm for sequencer of g

On initialization: $s_g := 0$;

On B-deliver$(<m, i>)$ *with* $g = group(m)$
    B-multicast$(g, <$"order", $i, s_g>)$;
    $s_g := s_g + 1$;

ISIS Algorithm

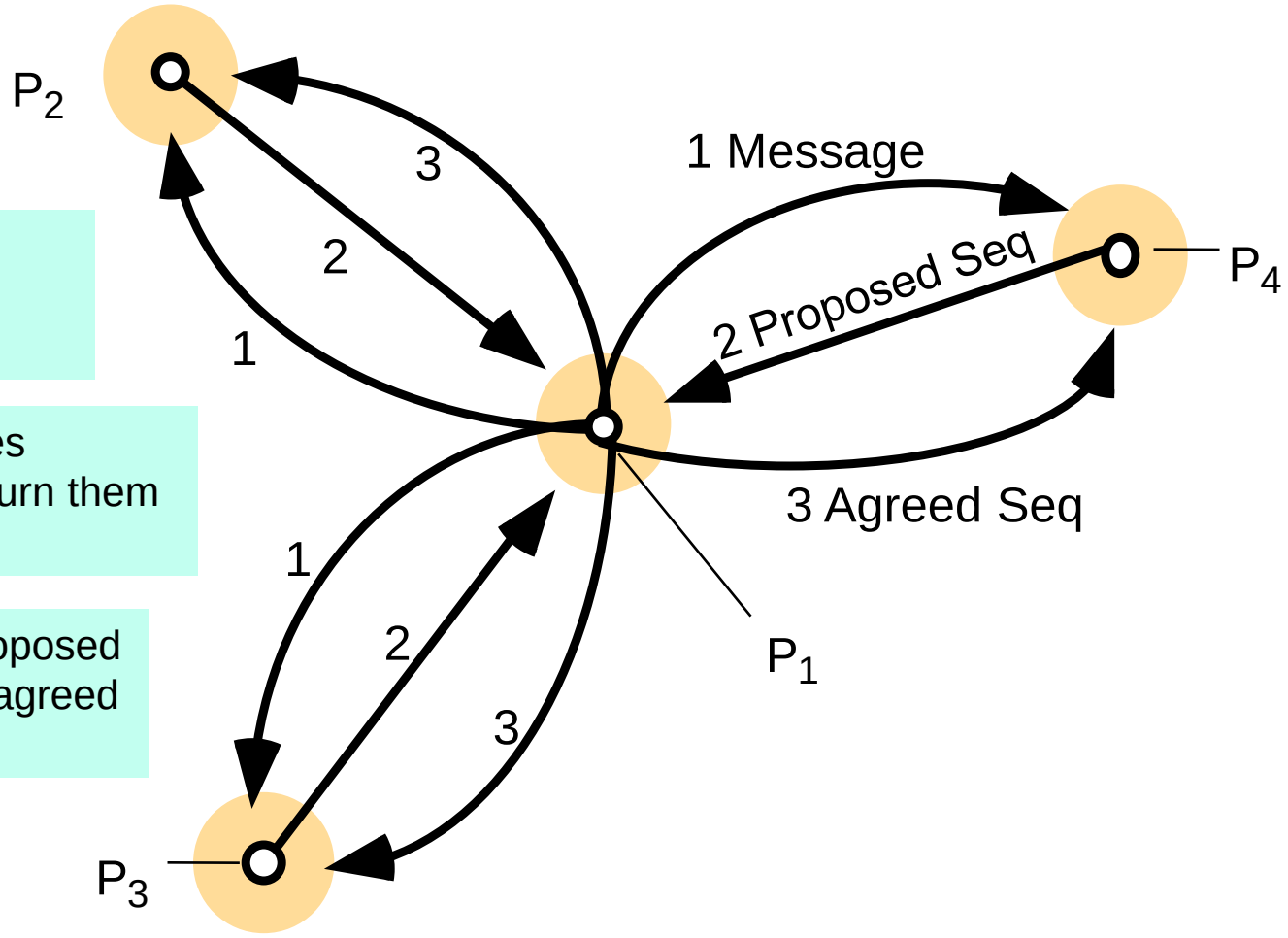
- Each process, *q* keeps:
  - A^q_g the largest agreed sequence number it has seen and
  - P^q_g its own largest proposed sequence number
- 1. Process *p* *B-multicasts* <*m*, *i*> to *g*, where *i* is a unique identifier for *m*.
- 2. Each process *q* replies to the sender *p* with a proposal for the message's agreed sequence number of
  - $P^q_g := \text{Max}(A^q_g, P^q_g) + 1$.
  - assigns the proposed sequence number to the message and places it in its hold-back queue
- 3. *p* collects all the proposed sequence numbers and selects the largest as the next agreed sequence number, *a*.
  It *B-multicasts* <*i*, *a*> to *g*. Recipients set $A^q_g := \text{Max}(A^q_g, a)$, attach *a* to the message and re-order hold-back queue.

ISIS Algorithm

1. the process *P1 B-multicasts* a message to members of the group

2. the receiving processes propose numbers and return them to the sender

3. the sender uses the proposed numbers to generate an agreed number



Discussion of ordering in ISIS protocol

- Hold-back queue
- ordered with the message with the smallest sequence number at the front of the queue
- when the agreed number is added to a message, the queue is re-ordered
- when the message at the front has an agreed id, it is transferred to the delivery queue
  - even if agreed, those not at the front of the queue are not transferred
- every process agrees on the same order and delivers messages in that order, therefore we have total ordering.
- Latency
  - 3 messages are sent in sequence, therefore it has a higher latency than sequencer method
  - this ordering may not be causal or FIFO