Global Snapshot

Etienne Renault

2 octobre 2020

https://www.lrde.epita.fr/~renault/teaching/algorep/

Problem Statement 1/2

Why recording the global state of a distributed system is important ?

- Check-pointing and recovery if the system fails, it can start start up from a meaningful state
- Monitoring and Debugging
- Termination or Deadlock detection

Problem Statement 2/2

Problem

- No global clock
- No shared memory
- Unpredictable message delays

Problem Statement 2/2

Problem

- No global clock
- No shared memory
- Unpredictable message delays

How to achieve this snapshot ?

1. Global States and Cuts
2. Snapshot for FIFO channels
3. Snapshot for non-FIFO channels
   - Lai Yang Algorithm
   - Mattern's Algorithm

Global State

The global state of a distributed system is a collection of the local states of the processes and the channels.

Let us denote by :

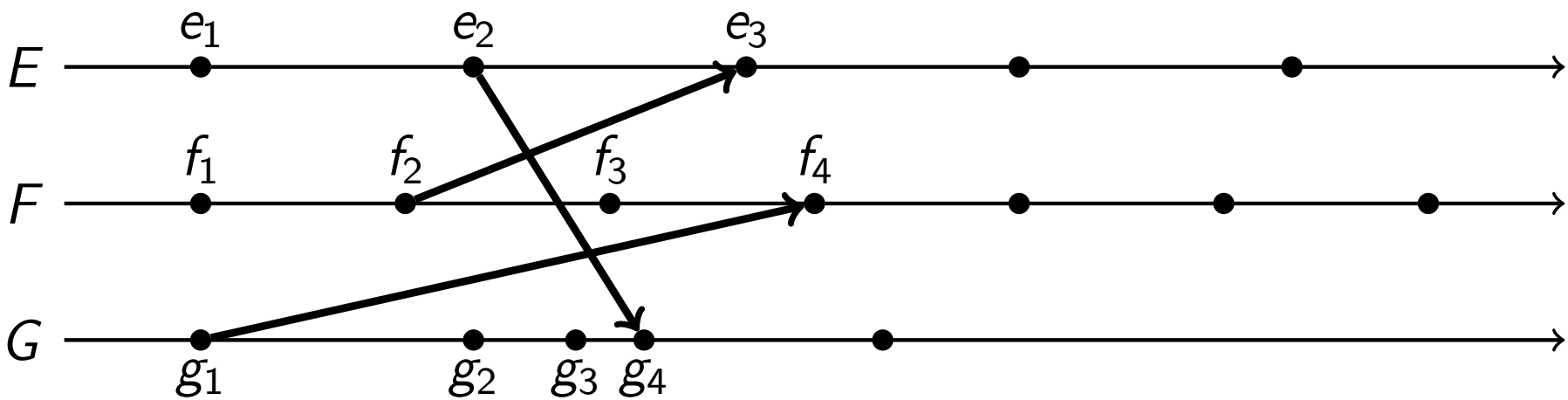
- LS_i the local state of state i
- $SC_{i,j}$ denotes the state of the channel $C_{i,j}$

Global State (formally)

$$GS = \{ \bigcup_{i} LS_i, \bigcup_{i,j} SC_{ij} \}$$

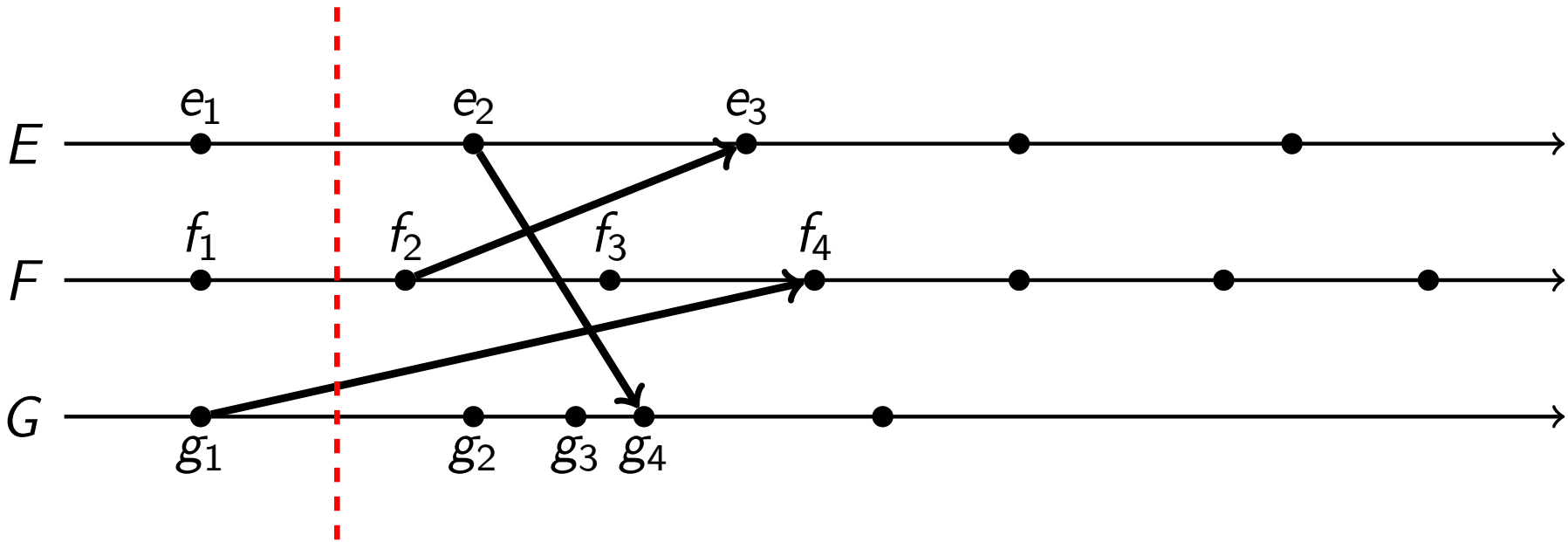
Cuts

A cut in a time diagram is a line joining an arbitrary point on each process line that slices the space-time diagram into a PAST and a FUTURE.



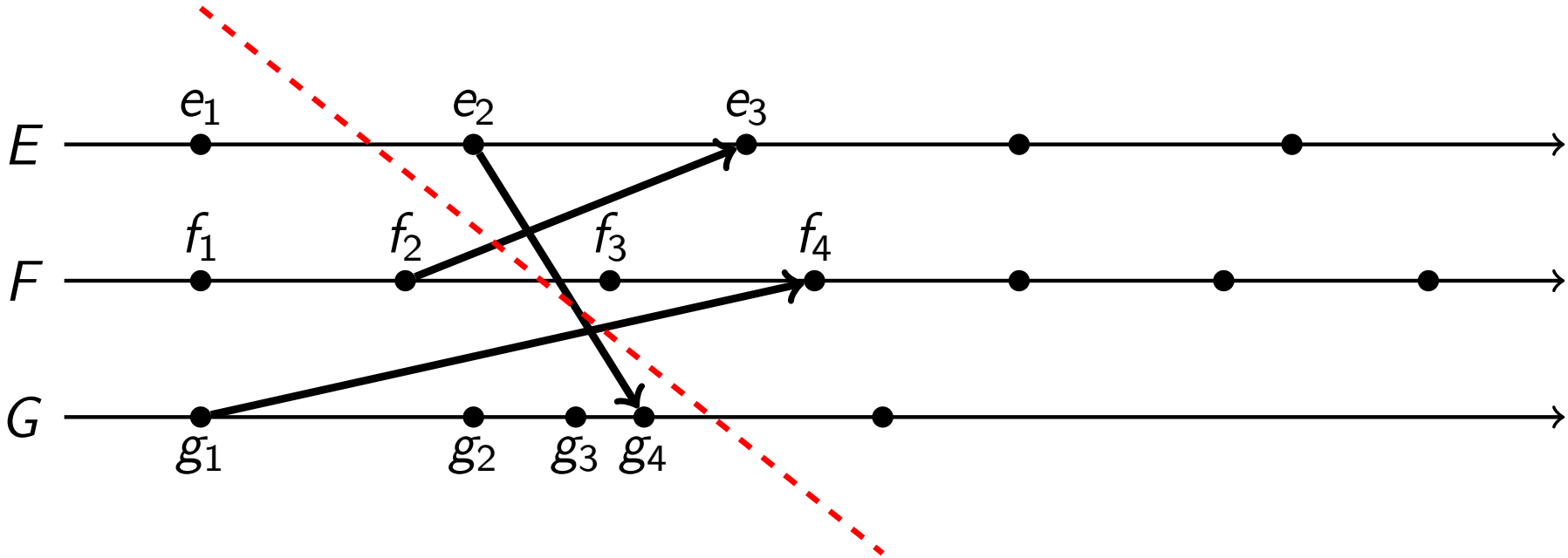
Cuts

A cut in a time diagram is a line joining an arbitrary point on each process line that slices the space-time diagram into a PAST and a FUTURE.



Cuts

A cut in a time diagram is a line joining an arbitrary point on each process line that slices the space-time diagram into a PAST and a FUTURE.



Consistent Global State

A consistent global state corresponds to a cut in which every message received in the PAST of the cut was sent in the PAST of that cut

Consistent Global State

A consistent global state corresponds to a cut in which every message received in the PAST of the cut was sent in the PAST of that cut

A consistent global State must satisfy the two following rules :

1. **C1** : A send implies that the reception is in progress

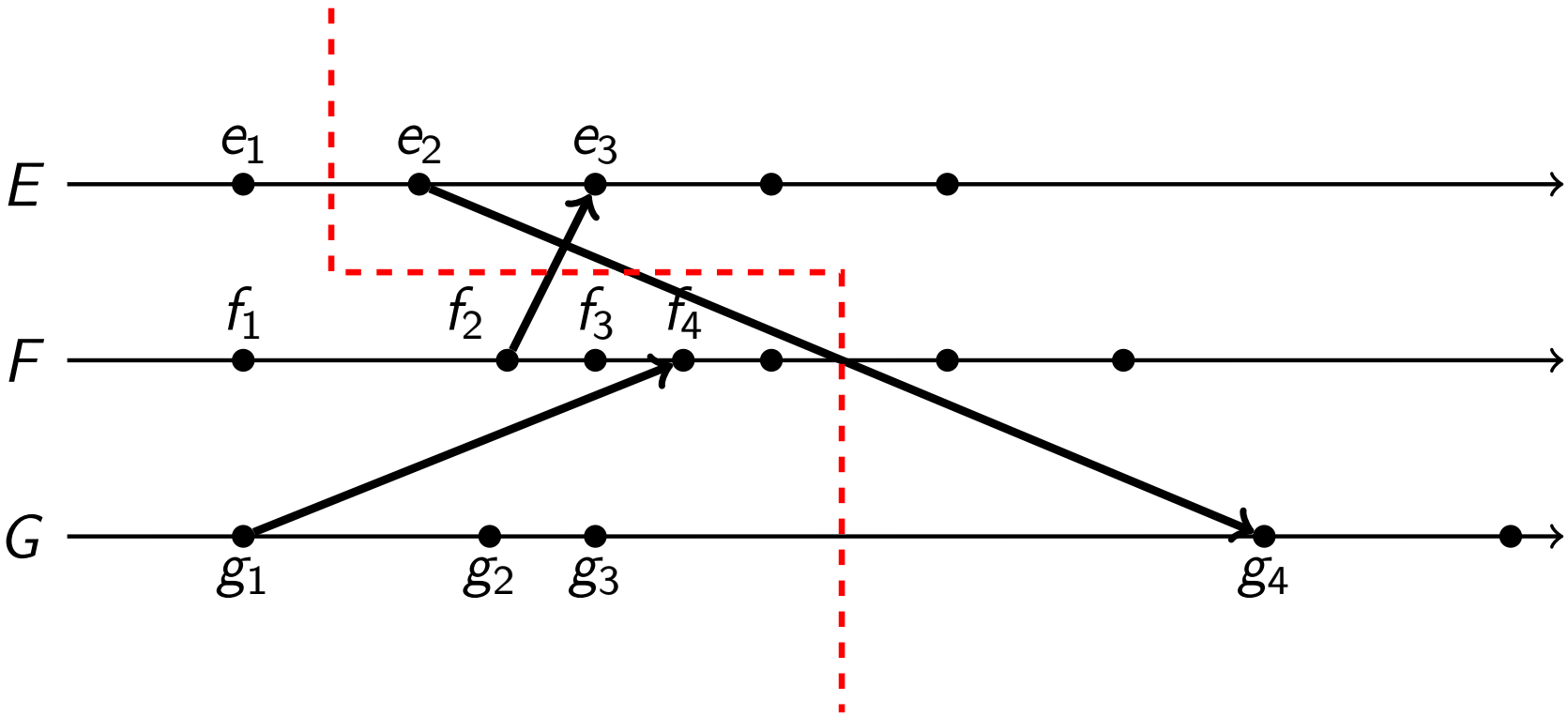
$$send(m_{i,j}) \in LS_i \implies m_{i,j} \in SC_{i,j} \otimes rec(m_{i,j}) \in LS_j$$

($\otimes$ is Ex-OR operator)

2. **C2** : A reception implies that the sent is in the global state

$$send(m_{i,j}) \notin LS_i \implies m_{i,j} \notin SC_{i,j} \wedge rec(m_{i,j}) \notin LS_j$$

Consistent Cut



Issues in recording a global state

How to distinguish between the messages to be recorded in the snapshot from those not to be recorded ?

- Should respect rules C1 and C2

How to determine the instant when a process takes its snapshot ?

- A process p_j must record its snapshot before processing a message m_{i_j} that was sent by process p_i after recording its snapshot.

1. Global States and Cuts

2. Snapshot for FIFO channels

3. Snapshot for non-FIFO channels
   - Lai Yang Algorithm
   - Mattern's Algorithm

Chandy-Lamport Algorithm : Informal

- Use a control message called *marker* to separate messages in the channels

Chandy-Lamport Algorithm : Informal

- Use a control message called *marker* to separate messages in the channels
- After a site has recorded its snapshot, it sends a marker, along all of its outgoing channels before sending out any more messages

Chandy-Lamport Algorithm : Informal

- Use a control message called *marker* to separate messages in the channels
- After a site has recorded its snapshot, it sends a marker, along all of its outgoing channels before sending out any more messages
- A marker separates the messages in the channel into those to be included in the snapshot from those not to be recorded

Chandy-Lamport Algorithm : Informal

- Use a control message called *marker* to separate messages in the channels
- After a site has recorded its snapshot, it sends a marker, along all of its outgoing channels before sending out any more messages
- A marker separates the messages in the channel into those to be included in the snapshot from those not to be recorded
- A process must record its snapshot no later than when it receives a marker on any of its incoming channels

Chandy-Lamport Algorithm : Informal

- Use a control message called *marker* to separate messages in the channels
- After a site has recorded its snapshot, it sends a marker, along all of its outgoing channels before sending out any more messages
- A marker separates the messages in the channel into those to be included in the snapshot from those not to be recorded
- A process must record its snapshot no later than when it receives a marker on any of its incoming channels
- The algorithm can be initiated by any process

Chandy-Lamport Algorithm : Informal

- Use a control message called *marker* to separate messages in the channels
- After a site has recorded its snapshot, it sends a marker, along all of its outgoing channels before sending out any more messages
- A marker separates the messages in the channel into those to be included in the snapshot from those not to be recorded
- A process must record its snapshot no later than when it receives a marker on any of its incoming channels
- The algorithm can be initiated by any process
- The algorithm terminates after each process has received a marker on all of its incoming channels

Chandy-Lamport Algorithm : Informal

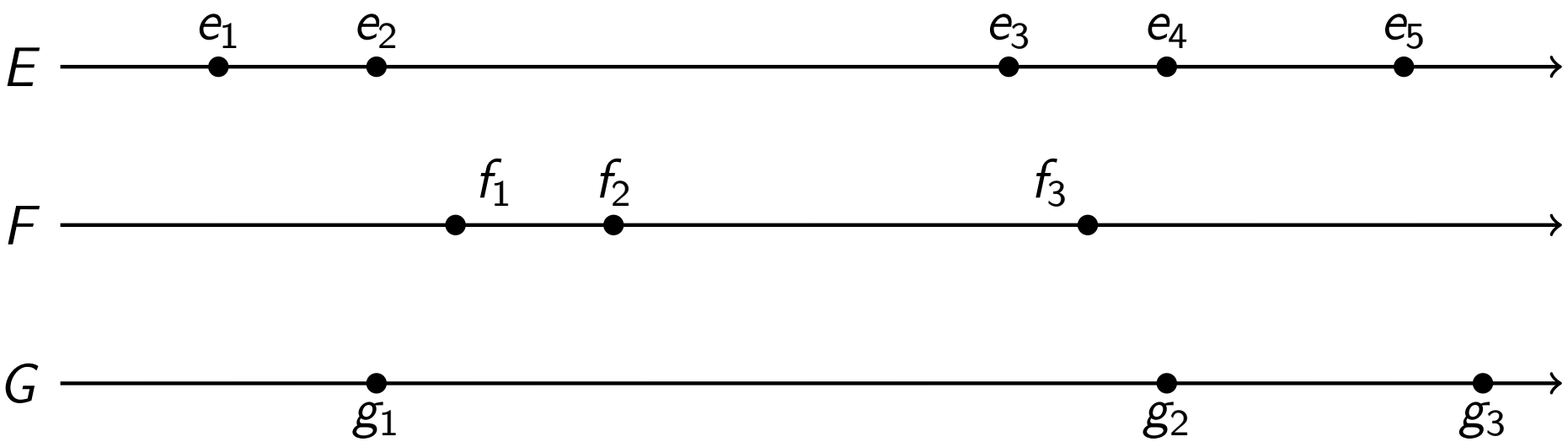
Sending Rule for process i :

- Process i records its state
- For each outgoing channel $C_{i,j}$ on which a marker has not been sent, i sends a marker along $C_{i,j}$ before i sends further messages along $C_{i,j}$

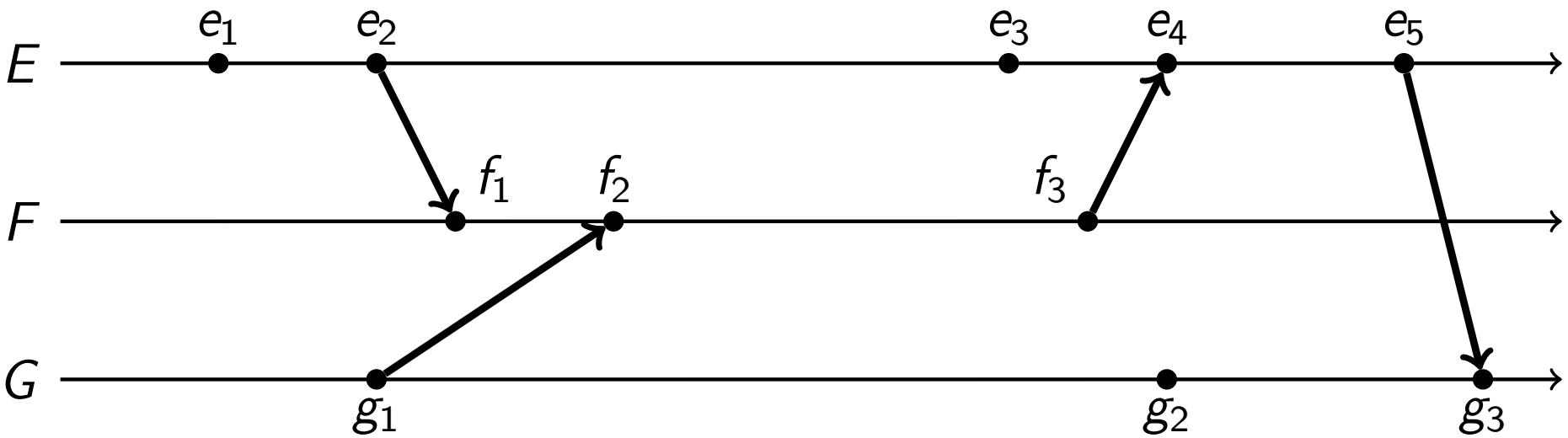
Receiving Rule for process j :

- On receiving a marker along channel $C_{i,j}$:
- if j has not recorded its state then
  - Record the state of C_i, j as the empty set
  - Follow the "Marker Sending Rule"
- else
  - Record the state of $C_{i,j}$ as the set of messages received along $C_{i,j}$ after j's state was recorded and before j received the marker along $C_{i,j}$.

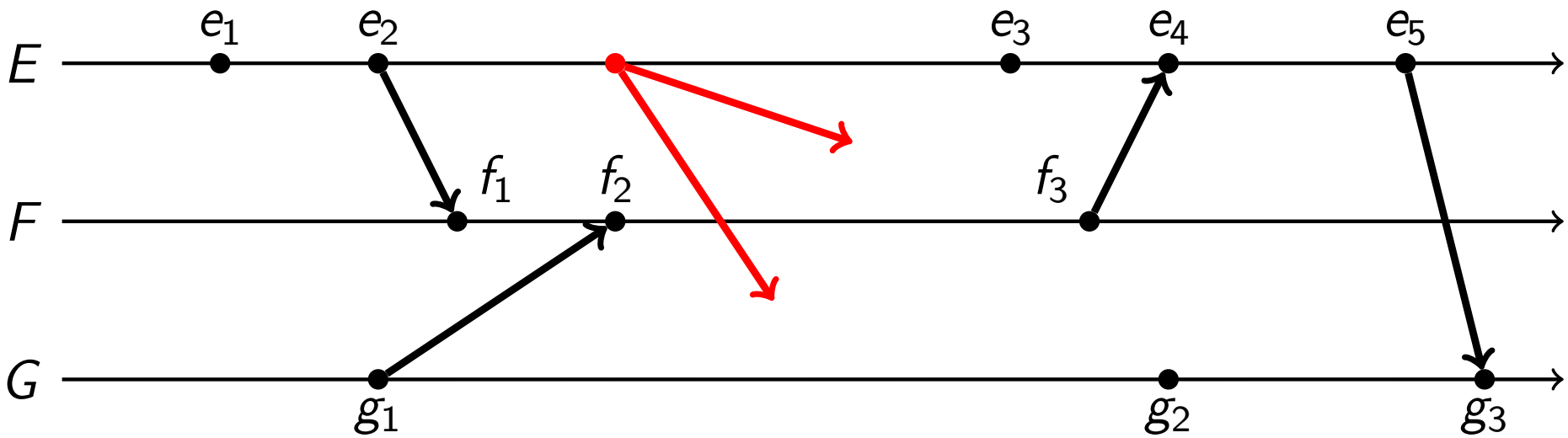
Example



Example

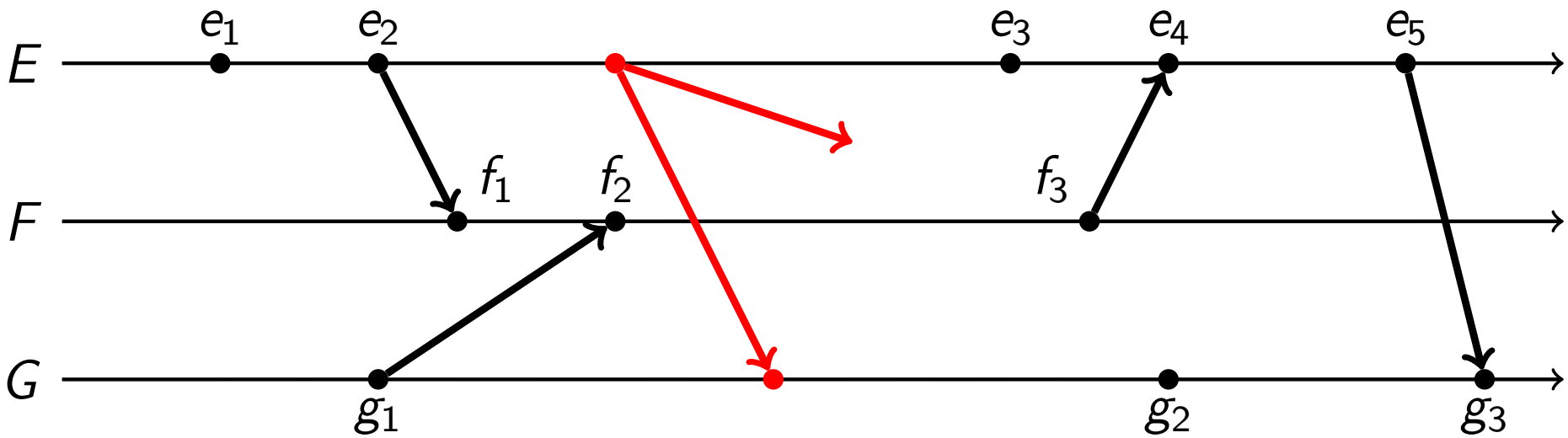


Example



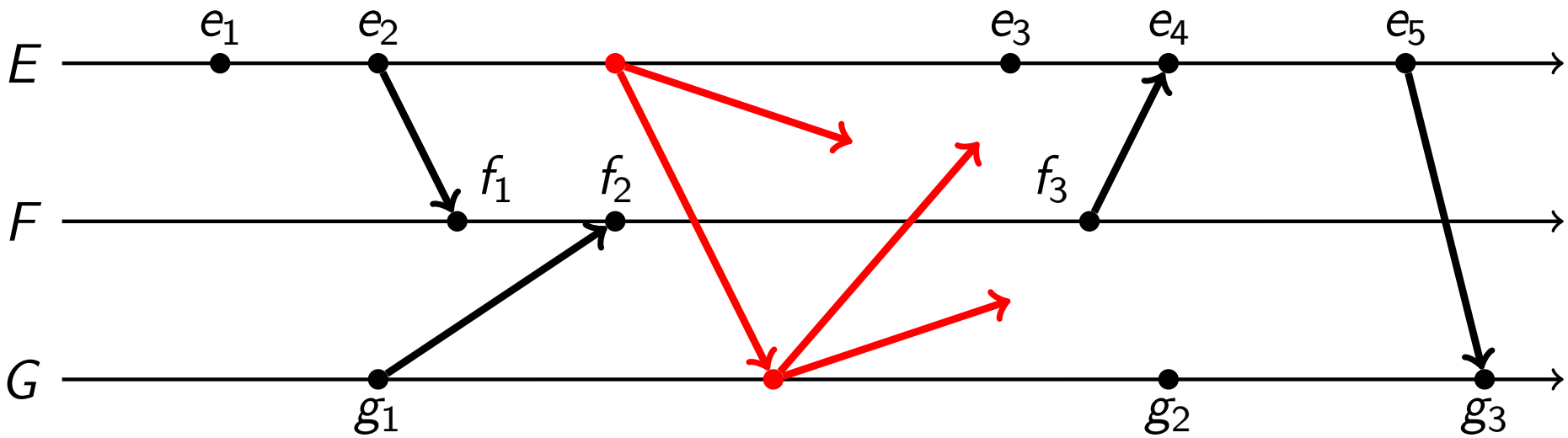
Afer e_2 E decide to take a snapshot.
E record local state S_E
Send marker through channels C_{E-F} and C_{E-G} (start recording channels status)

Example



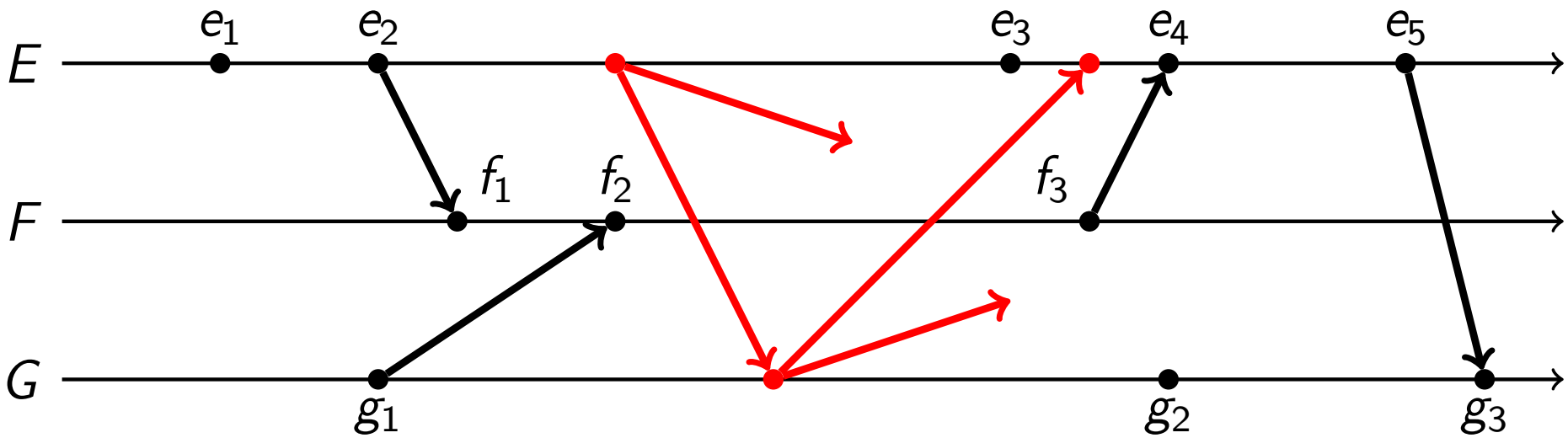
G receive marker from E
G record local state S_G
G record channel C_{E-G} as empty
Message to E-F is still in transit.

Example



G send marker through channels C_{G-E} and C_{G-F}

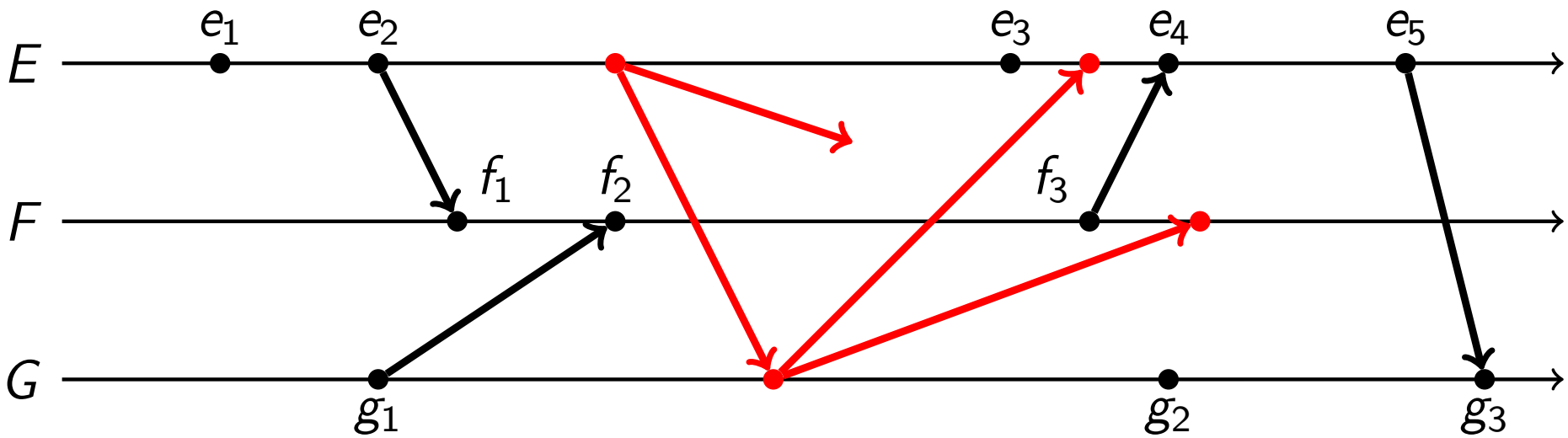
Example



Marker received by E

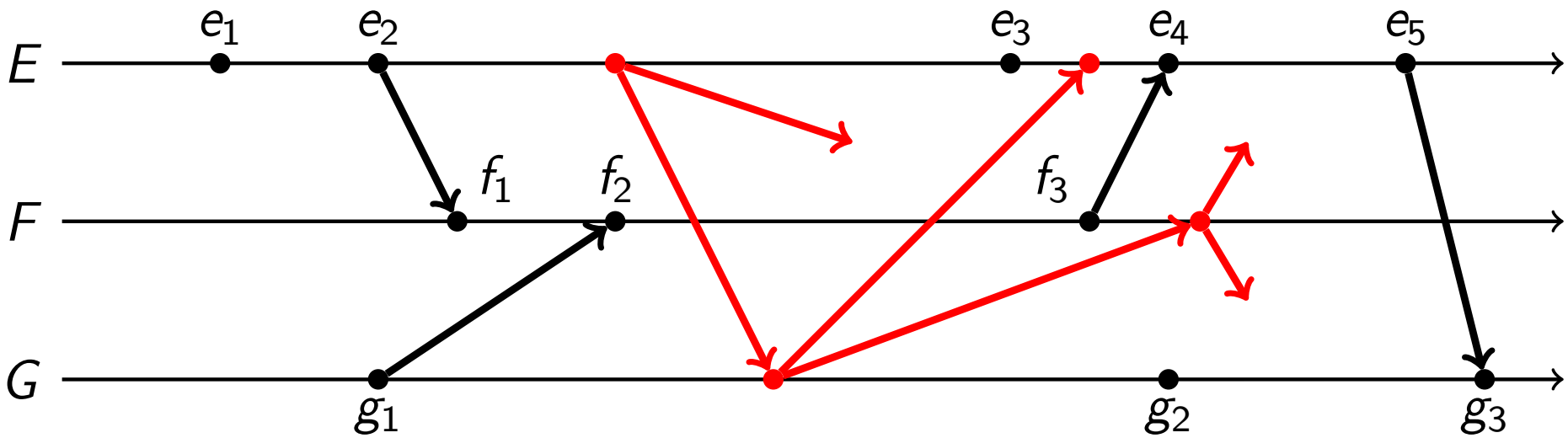
Duplicate marker : stop recording state of channel C_{E-G} and record it as empty.

Example



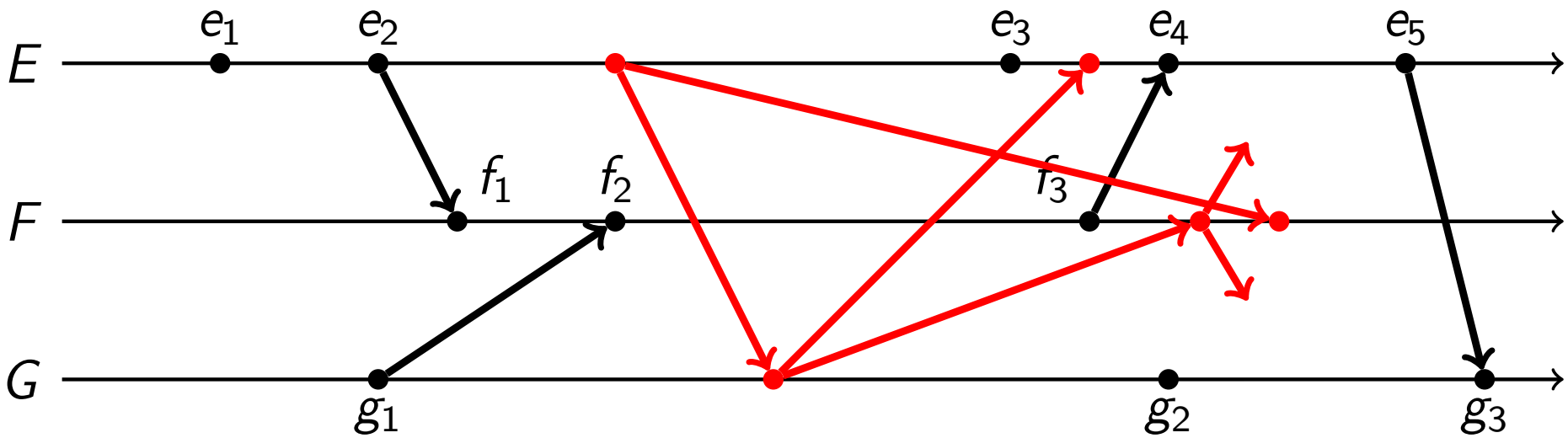
Marker received by F
F record local state S_F
F record channel C_{G-F} as empty

Example



Send marker through channels C_{F-E} and C_{F-G}

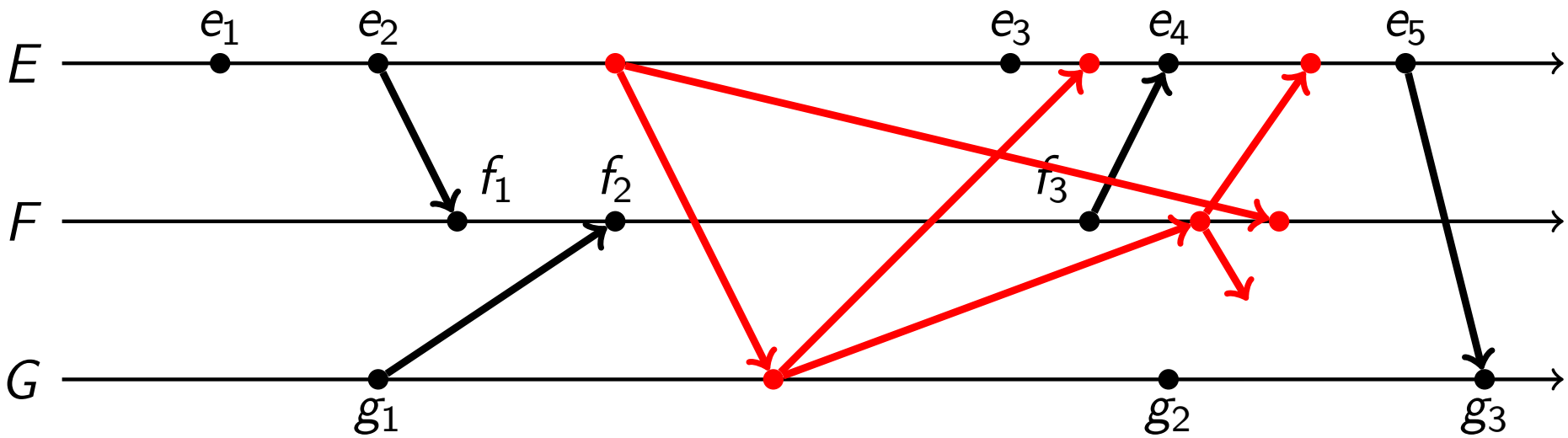
Example



Marker received by F

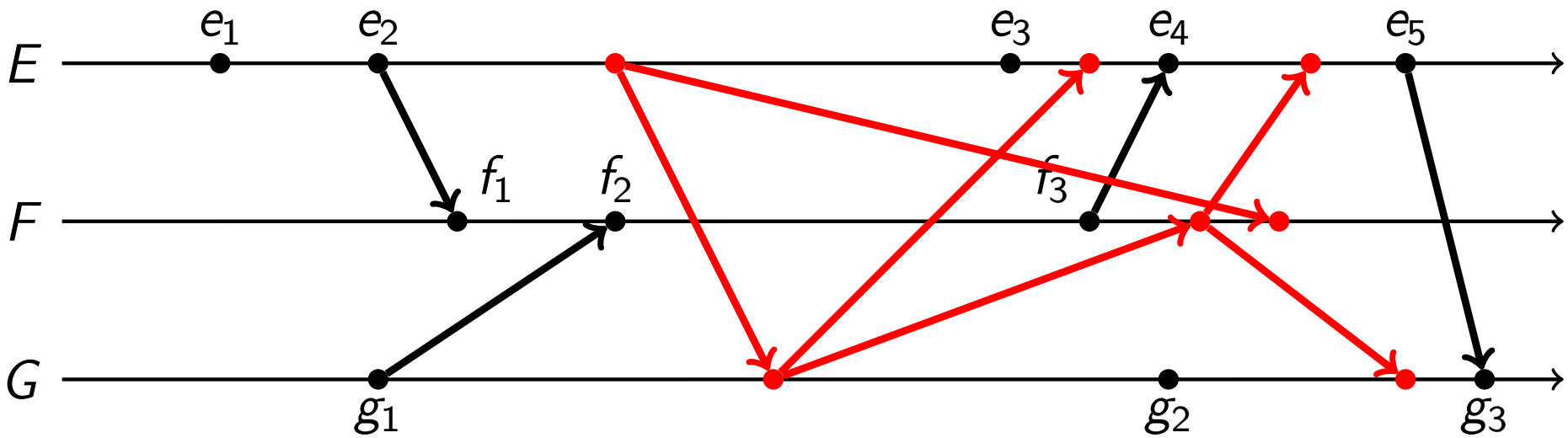
Duplicate marker : stop recording state of channel C_{E-F} and record it as empty.

Example



Marker received by E
Duplicate marker : stop recording state of channel C_{F-E}
Record it with message (f_3-e_4).

Example



Marker received by G

Duplicate marker : stop recording state of channel C_{F-G}

Correctness

- When a process j receives message $m_{i,j}$ that precedes the marker on channel $C_{i,j}$: If j has not taken its snapshot yet, then it includes $m_{i,j}$ in its recorded snapshot. Otherwise, it records $m_{i,j}$ in the state of the channel $C_{i,j}$. Thus, condition C1 is satisfied.

- Due to FIFO property of channels, no message sent after the marker on that channel is recorded in the channel state. Thus, condition C2 is satisfied

Complexity

Message Complexity

$O(e)$ for the record of a single instance of the algorithm, with e the number of edges in the graph

Time Complexity

$O(d)$ time with d the diameter of the graph

Remarks

- The recorded global state may not correspond to any of the global states that occurred during the computation

BUT...

- The recorded global state may not correspond to any of the global states that occurred during the computation

- The recorded global state is a valid state in an equivalent execution

1. Global States and Cuts

2. Snapshot for FIFO channels

3. Snapshot for non-FIFO channels
   - Lai Yang Algorithm
   - Mattern's Algorithm

Problem

A marker cannot be used to delineate messages into those to be recorded in the global state from those not to be recorded in the global state

Either some degree of inhibition or piggybacking of control information on computation messages to capture out-of-sequence messages.

1. Global States and Cuts

2. Snapshot for FIFO channels

3. Snapshot for non-FIFO channels
   - Lai Yang Algorithm
   - Mattern's Algorithm

Lai Yang Algorithm : Informal 1/2

- Use a coloring scheme

Lai Yang Algorithm : Informal 1/2

- Use a coloring scheme
- Every process is initially white and turns red while taking a snapshot.

Lai Yang Algorithm : Informal 1/2

- Use a coloring scheme
- Every process is initially white and turns red while taking a snapshot.
- Every message sent by a white/red process is colored white/red

Lai Yang Algorithm : Informal 1/2

- Use a coloring scheme
- Every process is initially white and turns red while taking a snapshot.
- Every message sent by a white/red process is colored white/red
  - White : a message that was sent before the sender of that message recorded its local snapshot

Lai Yang Algorithm : Informal 1/2

- Use a coloring scheme
- Every process is initially white and turns red while taking a snapshot.
- Every message sent by a white/red process is colored white/red
  - White : a message that was sent before the sender of that message recorded its local snapshot
  - Red : a message that was sent after the sender of that message recorded its local snapshot

Lai Yang Algorithm : Informal 1/2

- Use a coloring scheme
- Every process is initially white and turns red while taking a snapshot.
- Every message sent by a white/red process is colored white/red
  - White : a message that was sent before the sender of that message recorded its local snapshot
  - Red : a message that was sent after the sender of that message recorded its local snapshot
- Every white process takes its snapshot at its convenience, but no later than the instant it receives a red message.

Lai Yang Algorithm : Informal 2/2

- Every white process records a history of all white messages sent or received by it along each channel

Lai Yang Algorithm : Informal 2/2

- Every white process records a history of all white messages sent or received by it along each channel
- When a process turns red, it sends these histories along with its snapshot to the initiator process that collects the global snapshot

Lai Yang Algorithm : Informal 2/2

- Every white process records a history of all white messages sent or received by it along each channel
- When a process turns red, it sends these histories along with its snapshot to the initiator process that collects the global snapshot
- The initiator process evaluates $transit(LS_i, LS_j)$ to compute the state of a channel $C_{i,j}$

$$SC_{i,j} = \{send(m_{i,j}) \mid send(m_{i,j}) \in LS_i\} - \{rec(m_{i,j}) \mid rec(m_{i,j}) \in LS_j\}.$$

Example

1. Global States and Cuts

2. Snapshot for FIFO channels

3. Snapshot for non-FIFO channels
   - Lai Yang Algorithm
   - Mattern's Algorithm

Mattern's Algorithm : Informal 1/2

Based on vector clocks and assumes a single initiator process.

1. The initiator "ticks" its local clock and selects a future vector time T at which it would like a global snapshot to be recorded.

Mattern's Algorithm : Informal 1/2

Based on vector clocks and assumes a single initiator process.

1. The initiator "ticks" its local clock and selects a future vector time T at which it would like a global snapshot to be recorded.
2. The initiator then broadcasts this time s and freezes all activity until it receives all acknowledgements of the receipt of this broadcast

Mattern's Algorithm : Informal 1/2

Based on vector clocks and assumes a single initiator process.

1. The initiator "ticks" its local clock and selects a future vector time T at which it would like a global snapshot to be recorded.
2. The initiator then broadcasts this time s and freezes all activity until it receives all acknowledgements of the receipt of this broadcast
3. When a process receives the broadcast, it remembers the value T and returns an acknowledgement to the initiator.

Mattern's Algorithm : Informal 2/2

4. After having received an acknowledgement from every process, the initiator increases its vector clock to T and broadcasts a dummy message to all processes

Mattern's Algorithm : Informal 2/2

4. After having received an acknowledgement from every process, the initiator increases its vector clock to T and broadcasts a dummy message to all processes
5. The receipt of this dummy message forces each recipient to increase its clock to a value $\geq T$ if not already $\geq T$.

Mattern's Algorithm : Informal 2/2

4. After having received an acknowledgement from every process, the initiator increases its vector clock to T and broadcasts a dummy message to all processes
5. The receipt of this dummy message forces each recipient to increase its clock to a value $\geq T$ if not already $\geq T$.
6. Each process takes a local snapshot and sends it to the initiator when (just before) its clock increases from a value less than T to a value $\geq T$

Mattern's Algorithm : Informal 2/2

4. After having received an acknowledgement from every process, the initiator increases its vector clock to T and broadcasts a dummy message to all processes
5. The receipt of this dummy message forces each recipient to increase its clock to a value $\geq T$ if not already $\geq T$.
6. Each process takes a local snapshot and sends it to the initiator when (just before) its clock increases from a value less than T to a value $\geq T$
7. The state of $C_{i,j}$ is all messages sent along $C_{i,j}$ whose timestamp is smaller than T and which are received by j after recording LS_j.

Mattern's Algorithm for non-FIFO channels

Messages sent before the snapshot are white, red otherwise.

- Each process i keeps a counter cpt_i that indicates the difference between the number of white messages it has sent and received before recording its snapshot
- It reports this value to the initiator process along with its snapshot and forwards all white messages, it receives henceforth, to the initiator
- Snapshot collection terminates when the initiator has received $\sum cpti$ number of forwarded white messages

Example