CS4231
Parallel and Distributed Algorithms

Lecture 5

Instructor: Haifeng YU

Review of Last Lecture

- Chapter 7 “Models and Clocks”
  - Goal: Define “time” in a distributed system
- Logical clock
  - “happened before” $\Rightarrow$ smaller clock value
- Vector clock
  - “happened before” $\Leftrightarrow$ smaller clock value
- Matrix clock
  - Gives a process knowledge about what other processes know

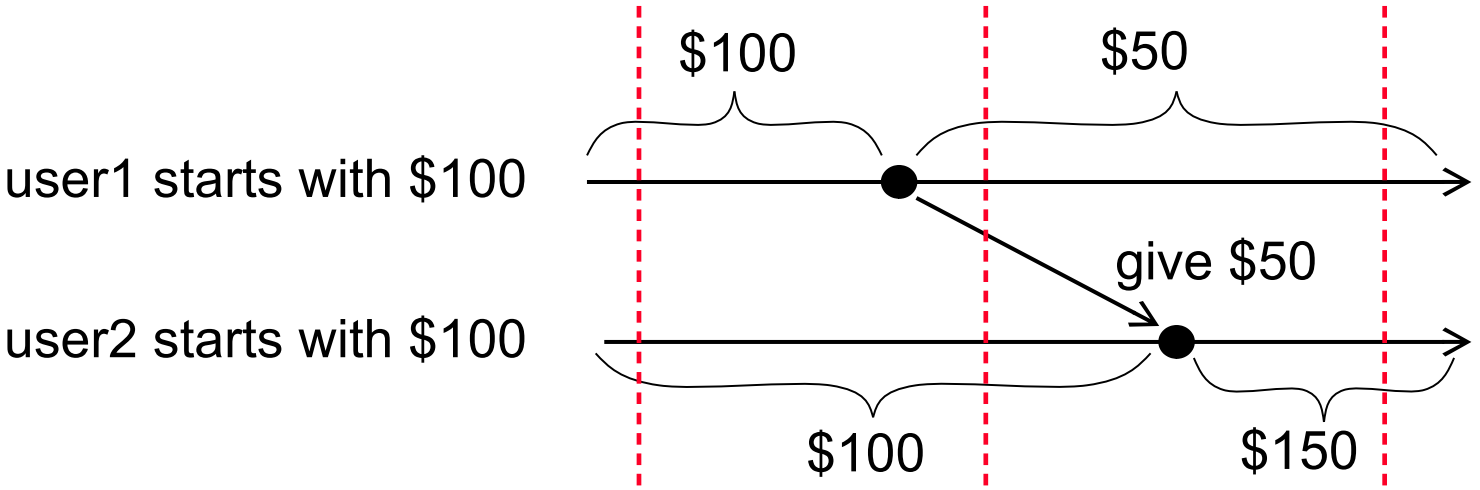
Today's Roadmap

- Chapter 9 "Global Snapshot"
- What is our goal?
- Formalizing consistent global snapshot
- Protocol for capturing a consistent global snapshot

Motivation

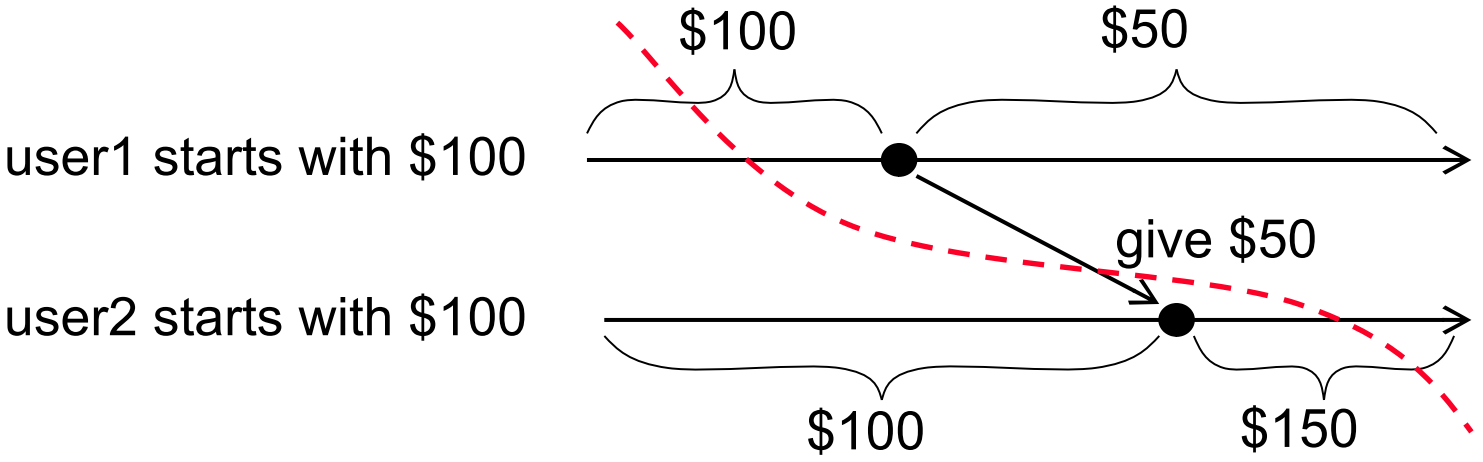
- Goal: Take a picture of the global computation
  - A snapshot of local states on n processes, together all messages on the fly, taken at exactly the same time
  - Note: The book uses two terms “global state” and “global snapshot” – we will stick to “global snapshot”
- Useful for debugging
- Useful for backup/check-pointing
- Useful for calculating global predicate
  - E.g., Exactly how much currency do we have in the country (notice that money flows among people constantly)?

Motivation



- If each process have access to completely accurate physical time – then trivial
  - All processes record local state at 1:00:00pm
  - We always get $200 total regardless when we take the snapshot

Motivation



- But we don't have completely accurate physical time
- user1 and user2 may record states in slightly different time
- How much inaccuracy can we tolerate?
  - Inaccuracy needs to be below minimum message propagation delay
- Hard!

Weaken Our Goal

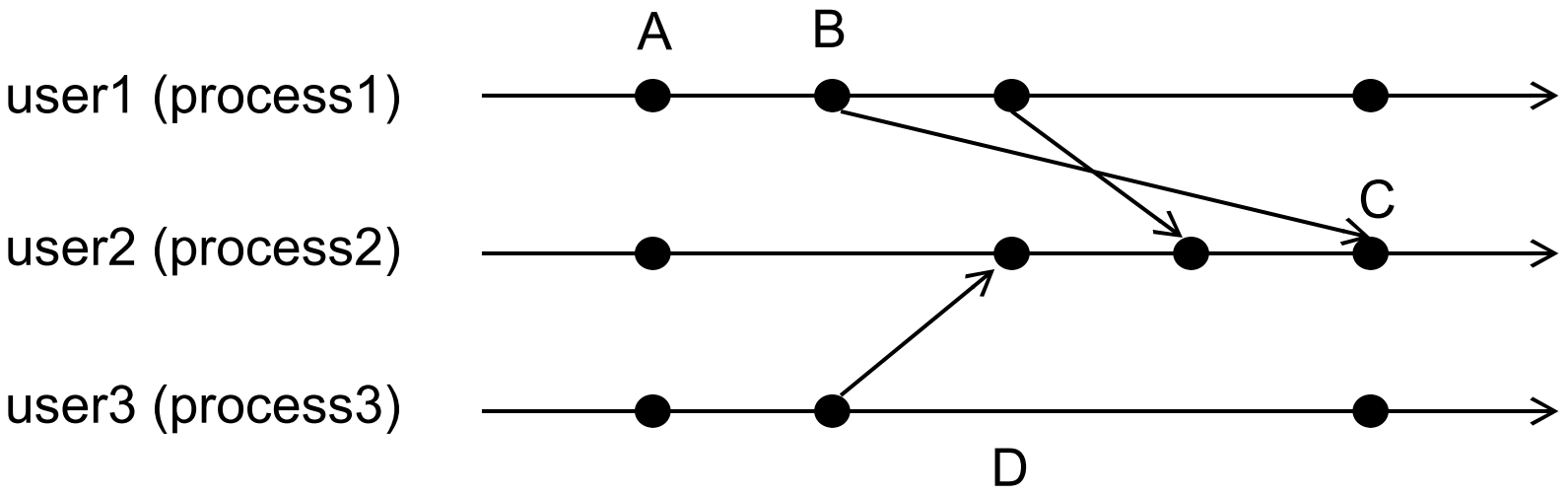
- Goal: Take a snapshot of the global computation
  - ~~A snapshot of local states on n processes taken at exactly the same time~~
  - A snapshot of local states on n processes such that the global snapshot could have happened sometime in the past

- "In the past"
  - We don't know when it occurred
- "Could have happened" – User cannot tell the difference
  - But in reality, may not have happened

- Somewhat surprisingly, such a weaker goal is still useful (e.g., debugging, backup, computing global predicates)

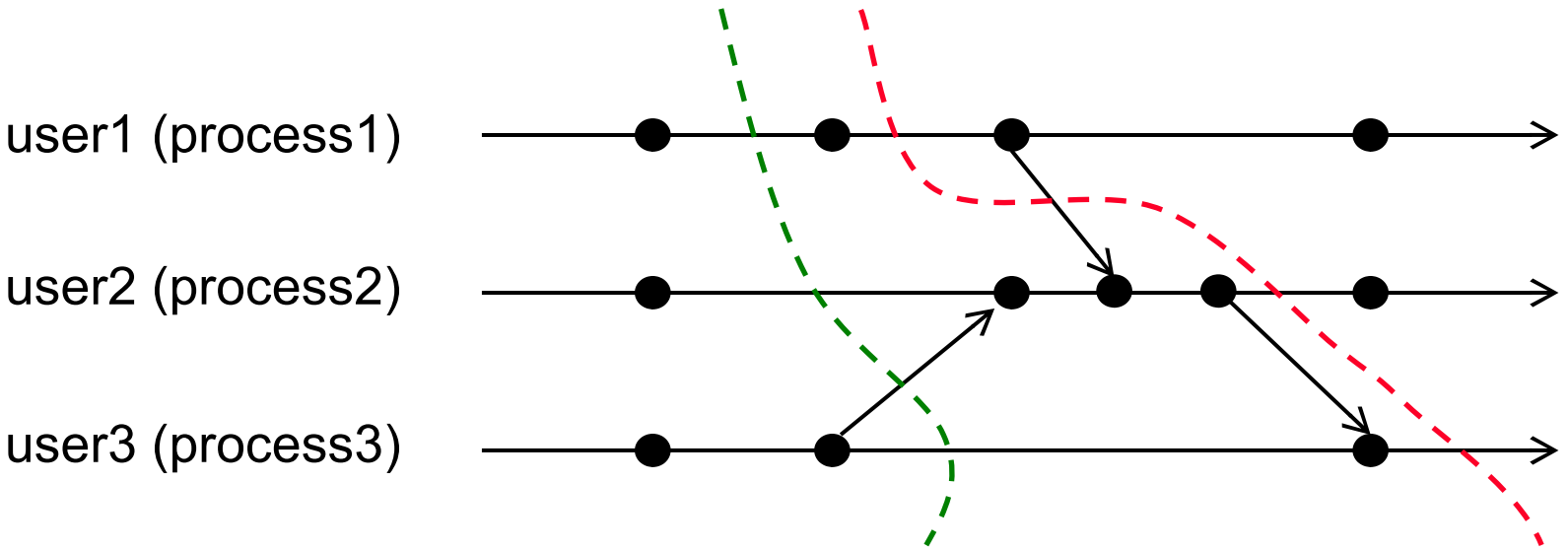
Review of "Happened-Before" Relation

- A partial order among events (i.e., local computation, send, receive)
  - Process order, send-receive order, transitivity



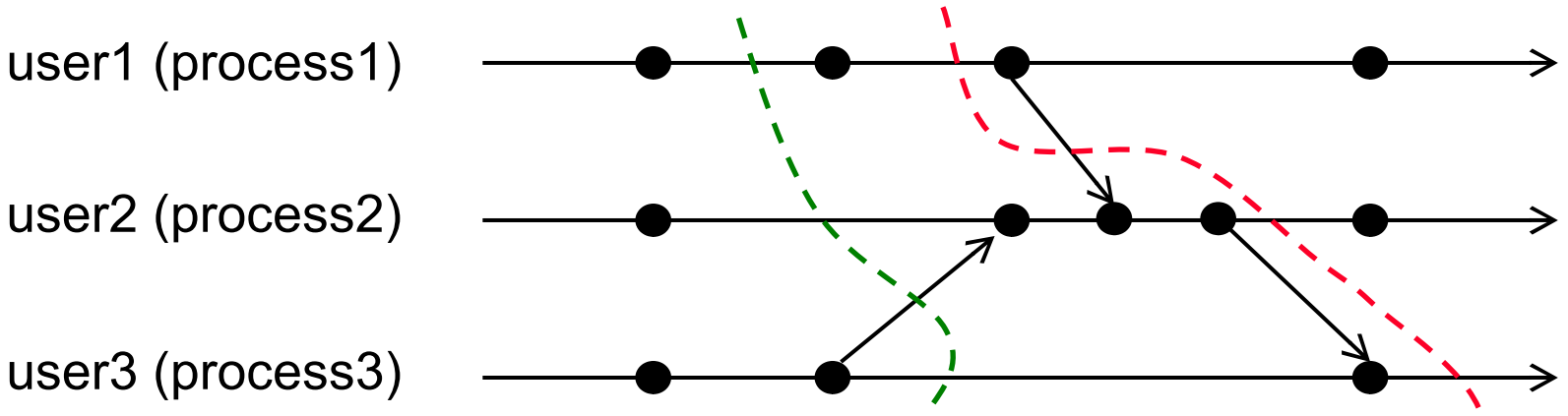
Consistent Snapshot Examples

- Consistent snapshot:
  - A snapshot of local states on n processes such that the global snapshot could have happened sometime in the past
  - The green cut could have happened in the past
  - The red cut could never happen in the past – why?



Formalizing Consistent Snapshot

- Global snapshot:
  - A set of events such that if e2 is in the set and e1 is before e2 in process order, then e1 must be in the set
  - How does this compare to your intuition?
- Consistent global snapshot:
  - A global snapshot such that if e2 is in the set and e1 is before e2 in send-receive order, then e1 must be in the set
  - Do we need to include transitive relations as in “Happened-before”?

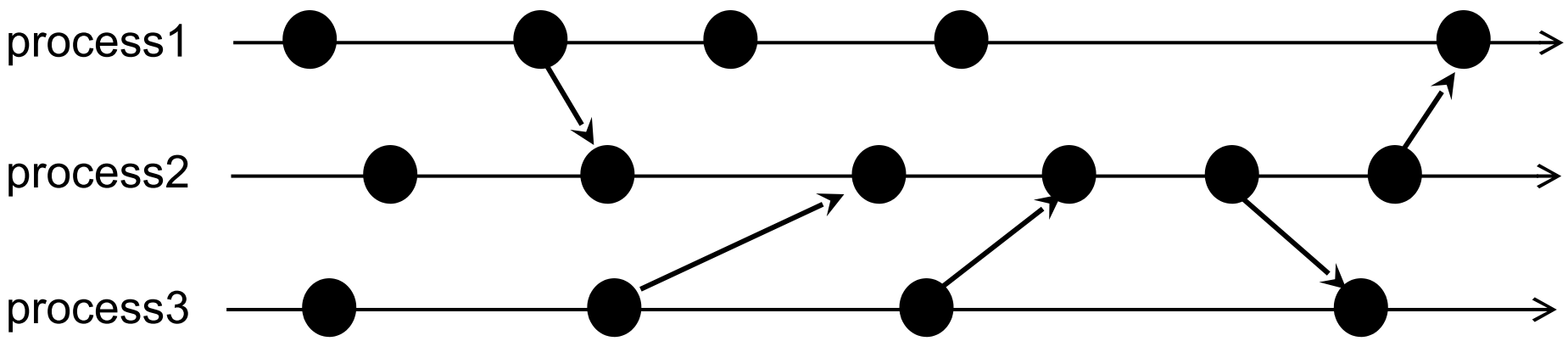


Existence of Consistent Global Snapshot

- It is a good habit to prove the existence of something newly defined (why?)
  - The proof will also later lead to an important characterization in our case…
- Consider the ordered events e1, e2, … on any given process
- Theorem: For any positive integer m, there exists a consistent global snapshot S such that

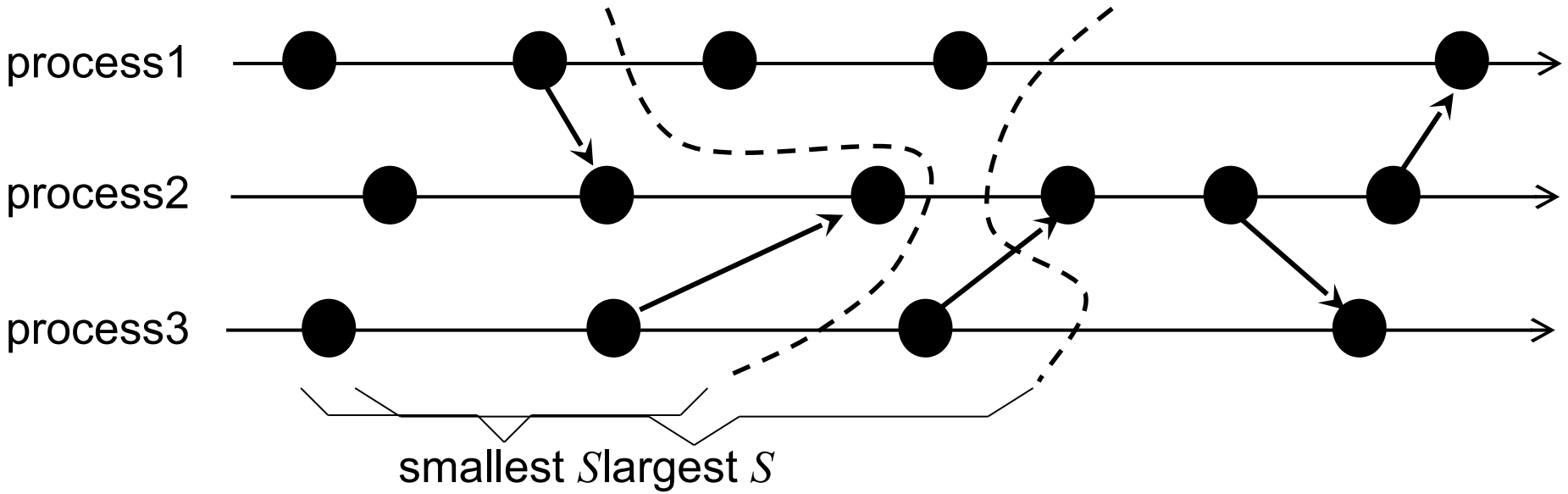
$$\begin{cases} e_i \in S \text{ for } i \leq m \\ e_i \notin S \text{ for } i > m \end{cases}$$

Proof Intuition



- Example: On process2, we want all blue events to be in S, and all red events not to be in S
  - Find all events that happened before ***any of*** the blue events and include them in S
  - Claim: S must be a consistent global snapshot

Characterization of Consistent Global Snapsht



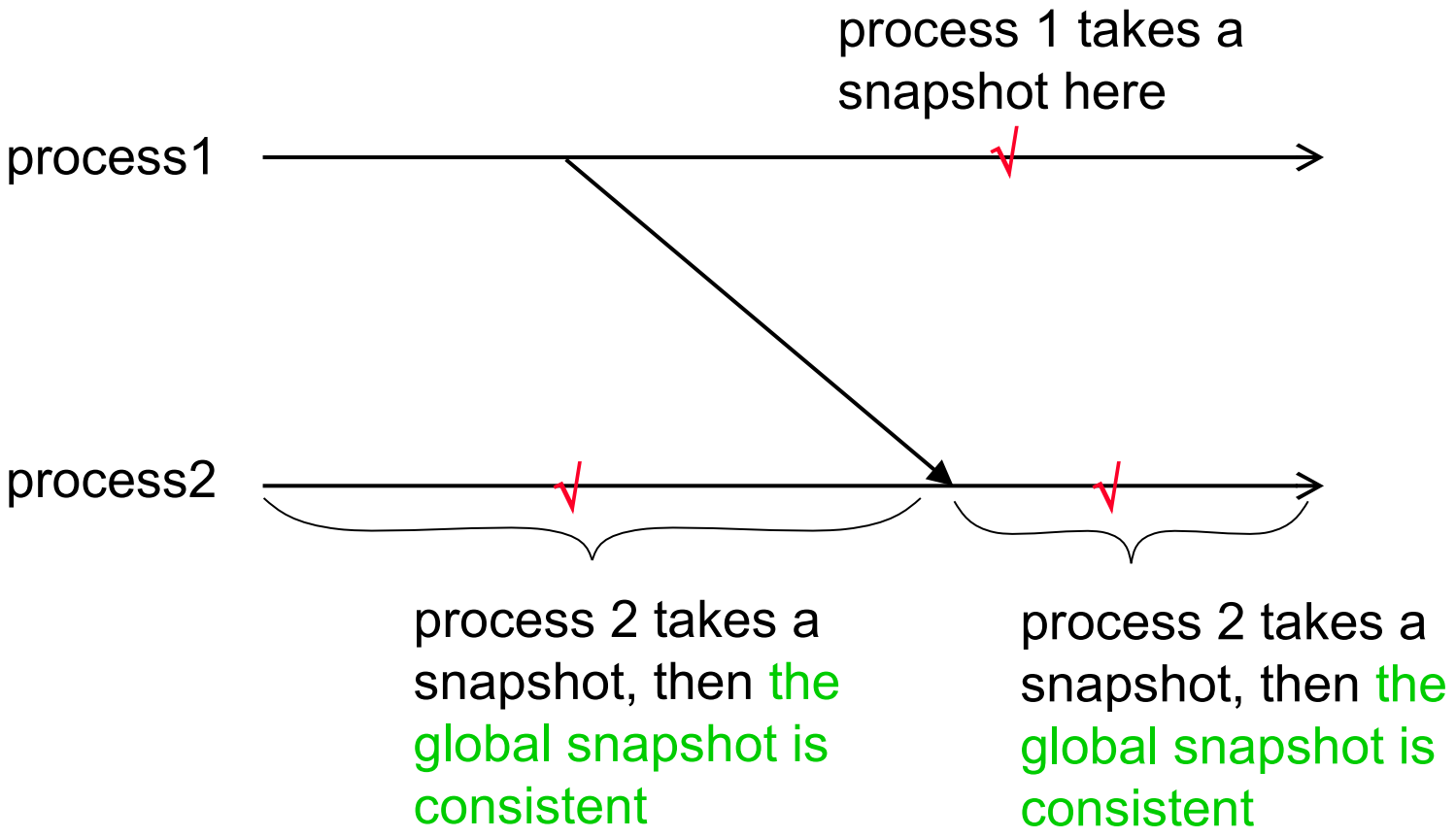
- There are other valid choices for S
- A key characterization: 3 kinds of events
  - Must be in S: Those happened before blue
  - Cannot be S: Those happened after red
  - May or may not be S: Other events

Capturing a Consistent Global Snapshot

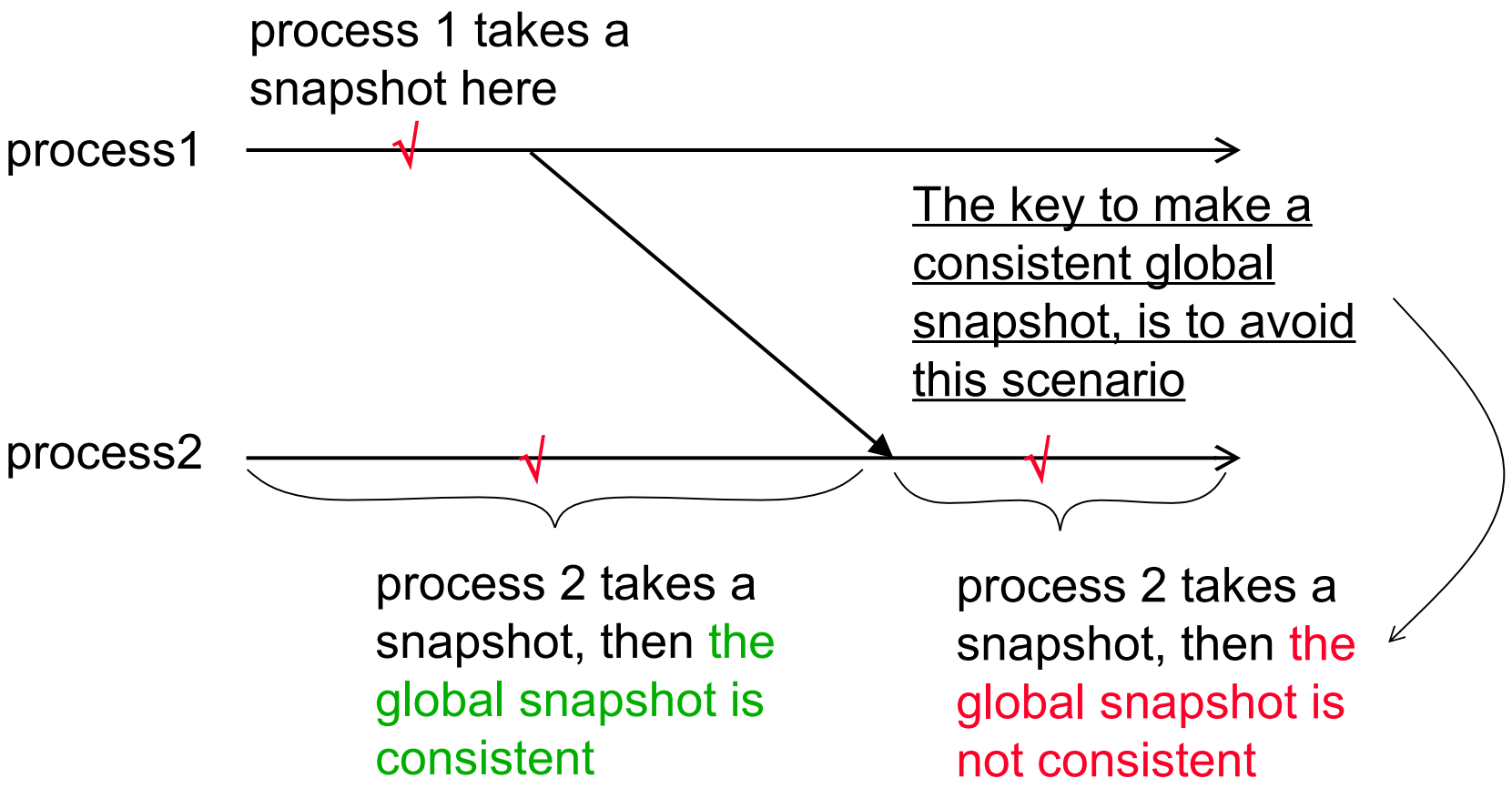
- Communication model
  - No message loss
  - Communication channels are unidirectional
  - FIFO delivery on each channel

- Ensuring FIFO:
  - Each process maintains a message number counter for each channel and stamps each message sent
  - Receiver will only deliver messages in order

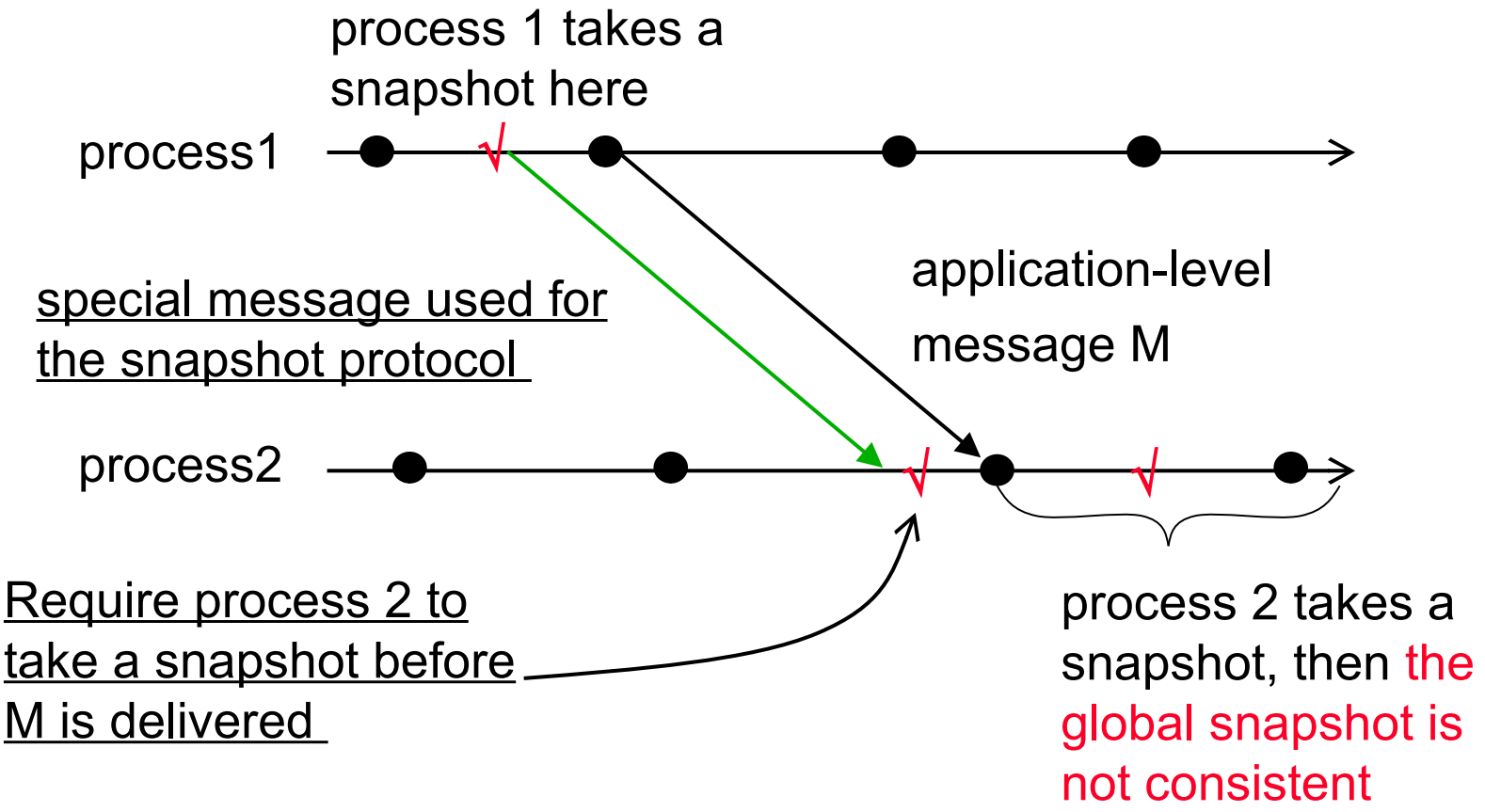
Protocol Intuition



Protocol Intuition



Protocol Intuition

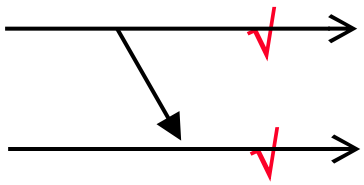


Chandy&Lamport's Protocol: Taking local snapshots

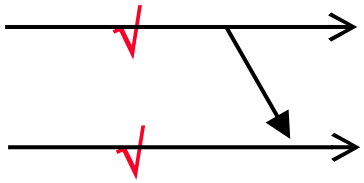
- Each process is either
  - Red (it has taken the local snapshot), OR
  - White (it has not taken the local snapshot)
- Protocol initiated by a single process by turning itself from White to Red
  - Once a process turns to Red, it immediately send out Marker messages to all other processes
  - Upon receiving Marker, a process turns Red
- Next, we would like also to capture messages that are "on-the-fly" when the snapshot is taken

Characterization of Application Messages

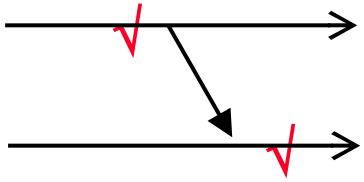
Given a message M, there are 4 posssibilities



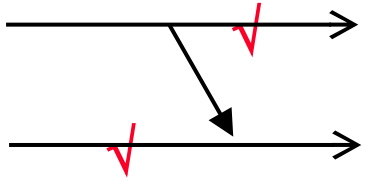
- M is sent before the local snapshot (on the sender) and received before the local snapshot (on the receiver): Not “on-the-fly” (why?)



- M is sent after the local snapshot and received after the local snapshot: Not “on-the-fly”

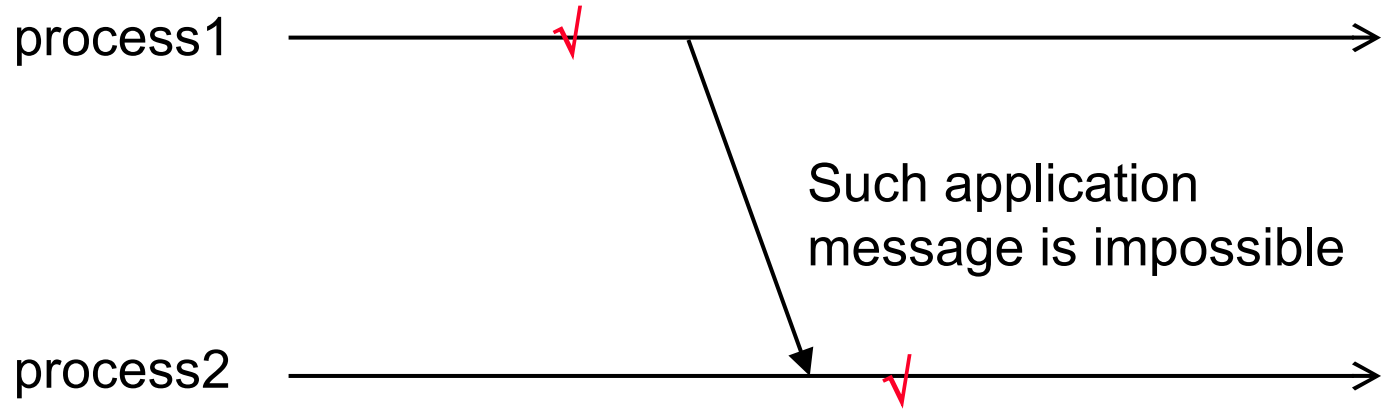


- M is sent after the local snapshot and received before the local snapshot: “on-the-fly”



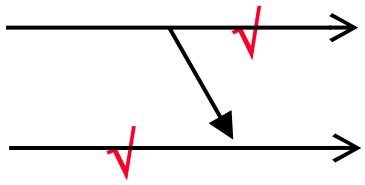
- M is sent before the local snapshot and received after the local snapshot: “on-the-fly”

Chandy&Lamport's Protocol: Taking snapshots for messages on the fly



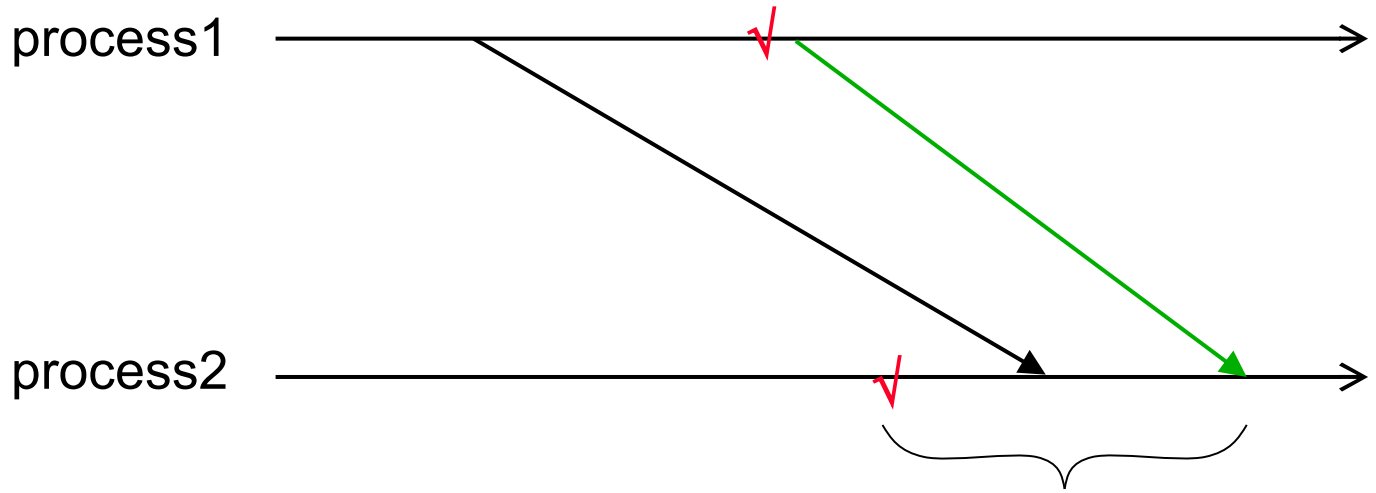
(Note: Figure 9.2 on the textbook cannot really happen with FIFO channel – ignore the figure please.)

So the only remaining case is:



- M is sent before the local snapshot and received after the local snapshot: "on-the-fly"

Chandy&Lamport's Protocol: Taking snapshots for messages on the fly



process2 records all messages received in this window – These are the exact set of messages that are only the "fly"

Summary

- Chapter 9 “Global Snapshot”
- What is our goal?
- Formalizing consistent global snapshot
- Protocol for capturing a consistent global snapshot

Homework Assignment

- Page 161
  - Problem 9.5

    How do you use Lamport's logical clock to compute a consistent global snapshot?

    (You should assume that you are given the set of ALL events. You should prove that the global snapshot you get is consistent.)
- Prove the following
  - If G and H are both consistent global snapshot, then G∩H and G∪H are also consistent global snapshots
- Homework due a week from today
- Read Chapter 12