Application-Level Fault Tolerance for MPI Programs

Kes00av

Solution Space

Checkpointing

Solution Space Detail

- Checkpointing *[Our Choice]*
  - Save application state periodically
  - When a process fails, all processes go back to last consistent saved state.
- Message Logging
  - Processes save outgoing messages
  - If a process goes down it restarts and neighbors resend it old messages
  - Checkpointing used to trim message log

Checkpointing: Two problems

- Saving the state of each process
- Coordination of checkpointing

Saving process state

- System-

Coordinating checkpoints

- Uncoordinated
  - Dependency-tracking, time-coordinated, ...
  - Suffer from exponential rollback
- Coordinated *[Our Choice]*
  - Blocking

    Coordinated

Blocking Co-ordinated Checkpointing

- Many programs are bulk-synchronous programs (BSP model: Valiant).
- At barrier, all processes take their checkpoints.
  - assumption: no message Tc 0 TPj 22.58 TD 0.0771493c 0 Tw (-)

Chandy-Lamport protocol

- Processes
  -

Algorithm explanation

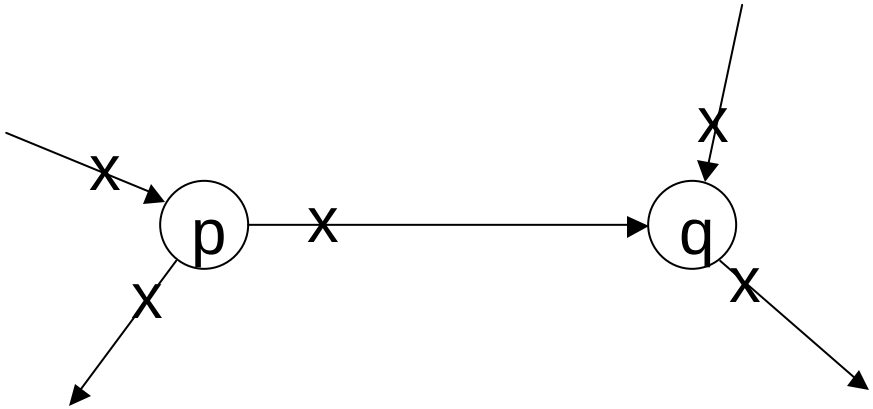
1. Saving process states
   - How do we avoid inconsistent messages?
2. Saving in-flight messages
3. Termination

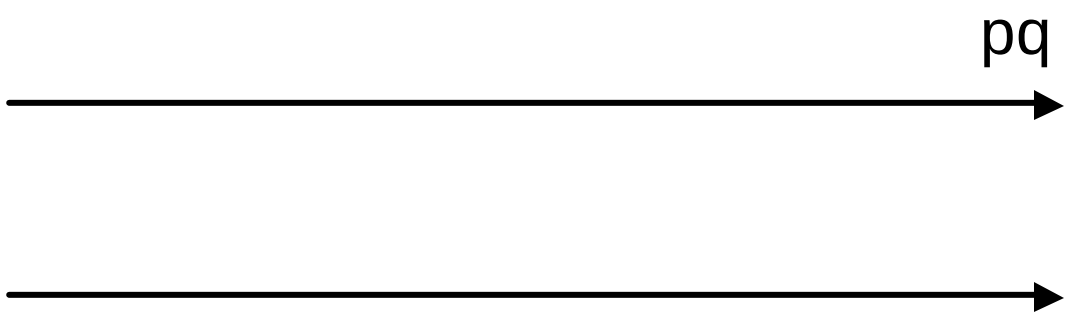
Next: Model of Distributed System

-6320 .75 43a saving process states630.72

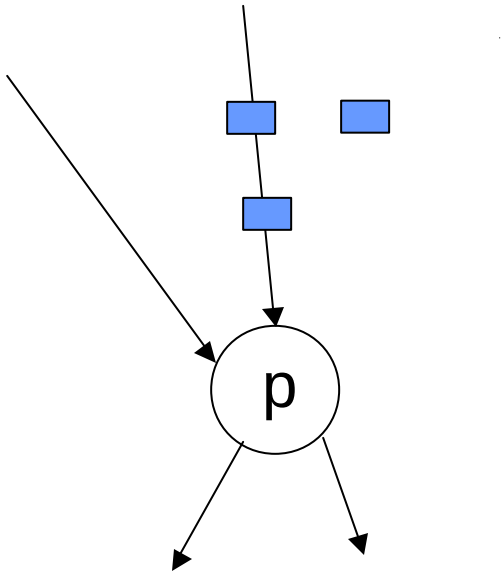
2-

Theorem: Saved states form consistent cut





Example



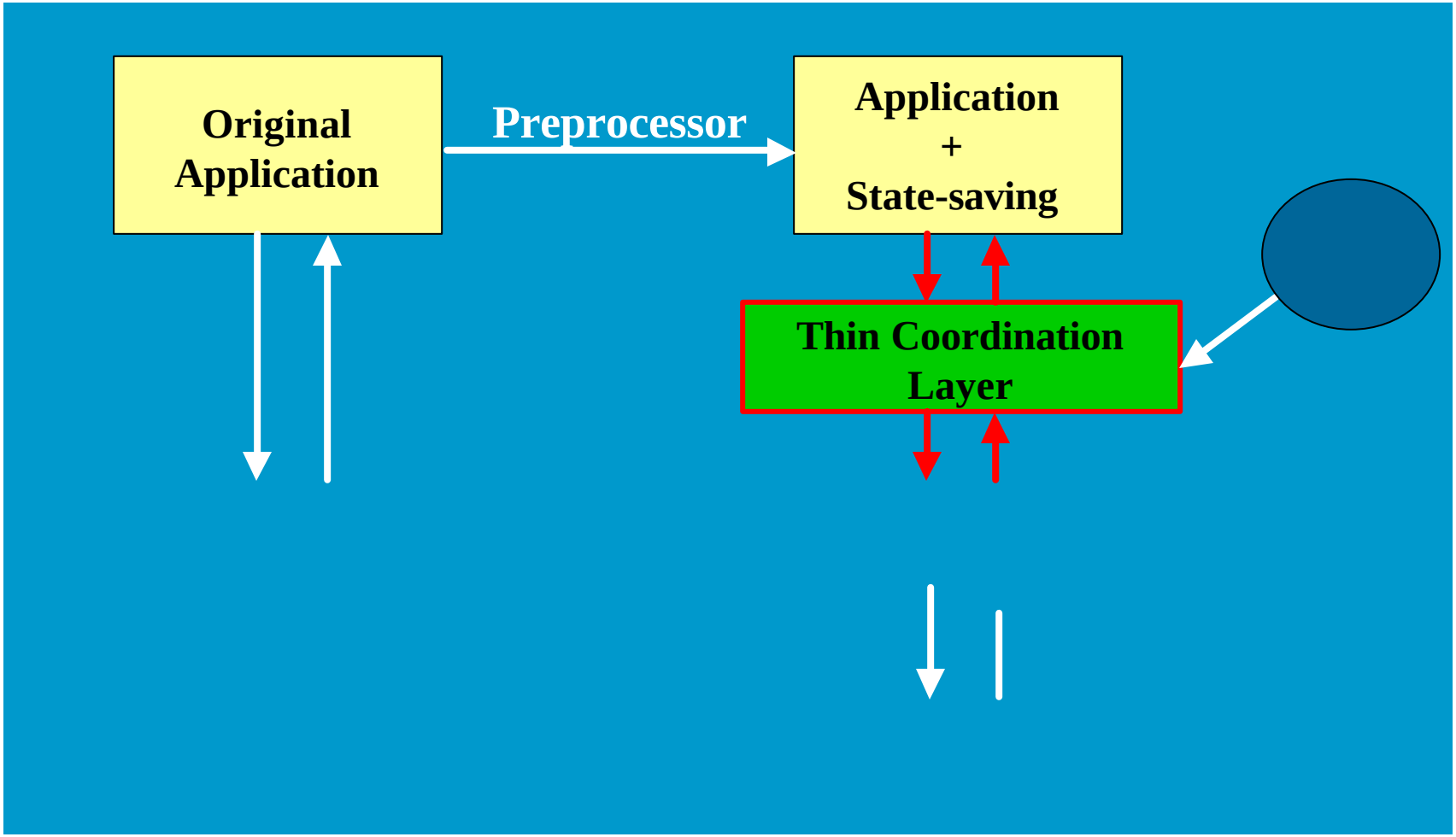
Example(cont’)

2-

Comments on C-L protocol

-

Our approach:System Architecture

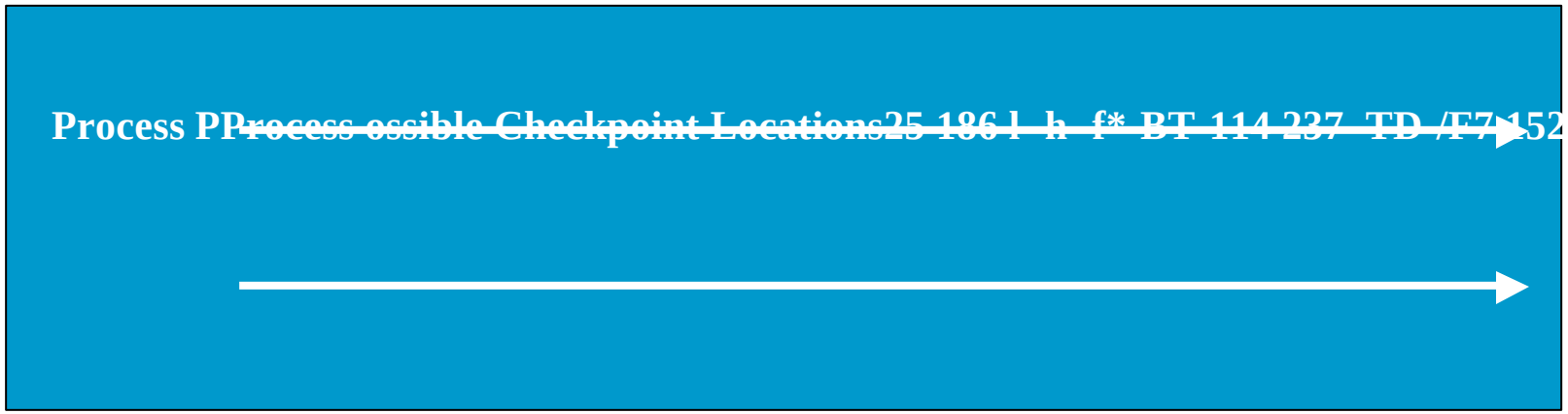


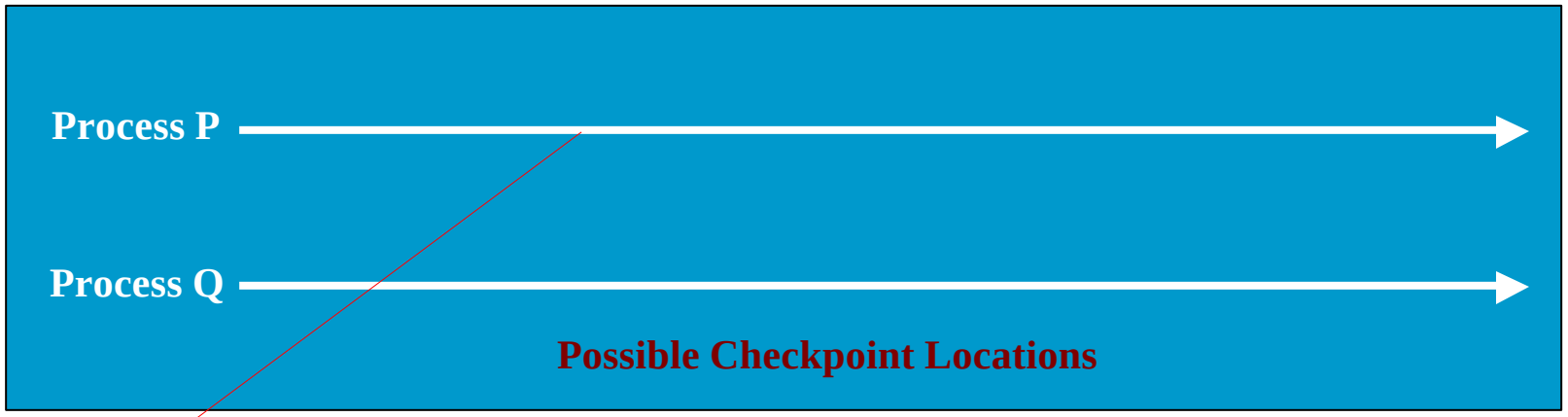
2-

Saving Application State

2-


Process P
Process Q


Process PProcess ossible Checkpoint Locations25 186 l h f* BT 114 237 TD /F7 152


Process P
Process Q
Possible Checkpoint Locations

Early Messages

- To reco 0 9 (10) Tj 1u316st either: TD 9 -350.280

The Protocol

High-level view of our protocol: (I)

-

High-level view of our protocol: (II)

- After taking a checkpoint each process
  -


Initiator
Chkpt
Process P
Process Q


Initiator
Chkpt_Ok
Process P
Process Q

The Global View

- A program's execution is divideTc /F0 325 T

Mechanism: Control Information

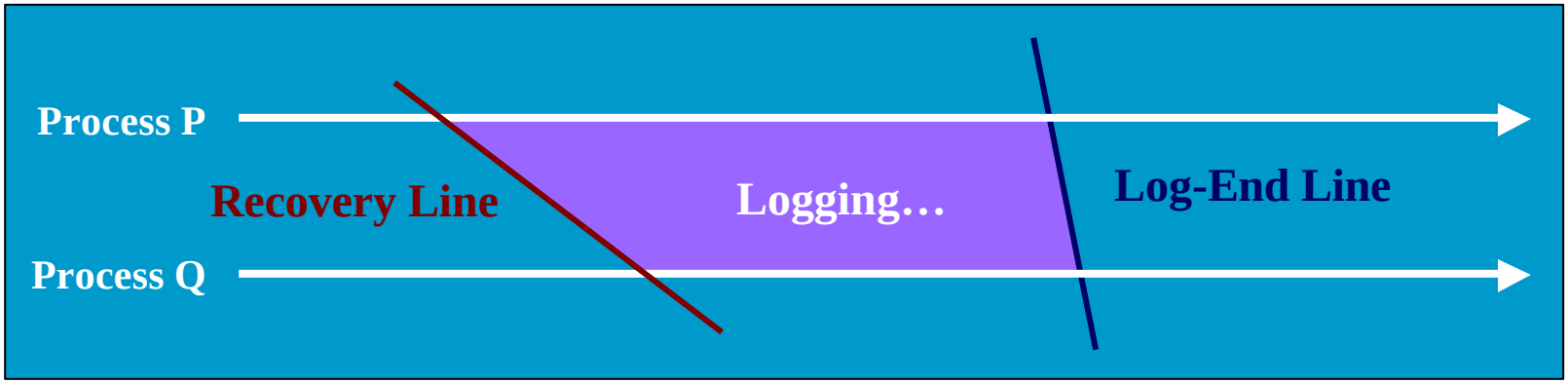
- •46is e suf0D38

Mechanism: The Log

- Keep a log after taking a checkpoint
- During Logging phase
  - Record late messages at receiver
  - Log all non-deterministic events

    ex: rand(), MPI_Test(), MPI_Recv(ANY_SOURCE)

Log-End Line



- Terminate log to preserve these semantics:
  - No message may cross Log-End line backwards
  - No late message may cross Log-End line
- Solution:
  - Send Ready_to_stop_logging message after receiving all late messages
  - Process stops logging when it receives Stop_log message from initiator or when it receives a message from a process that has itself stopped logging

Additional Issues

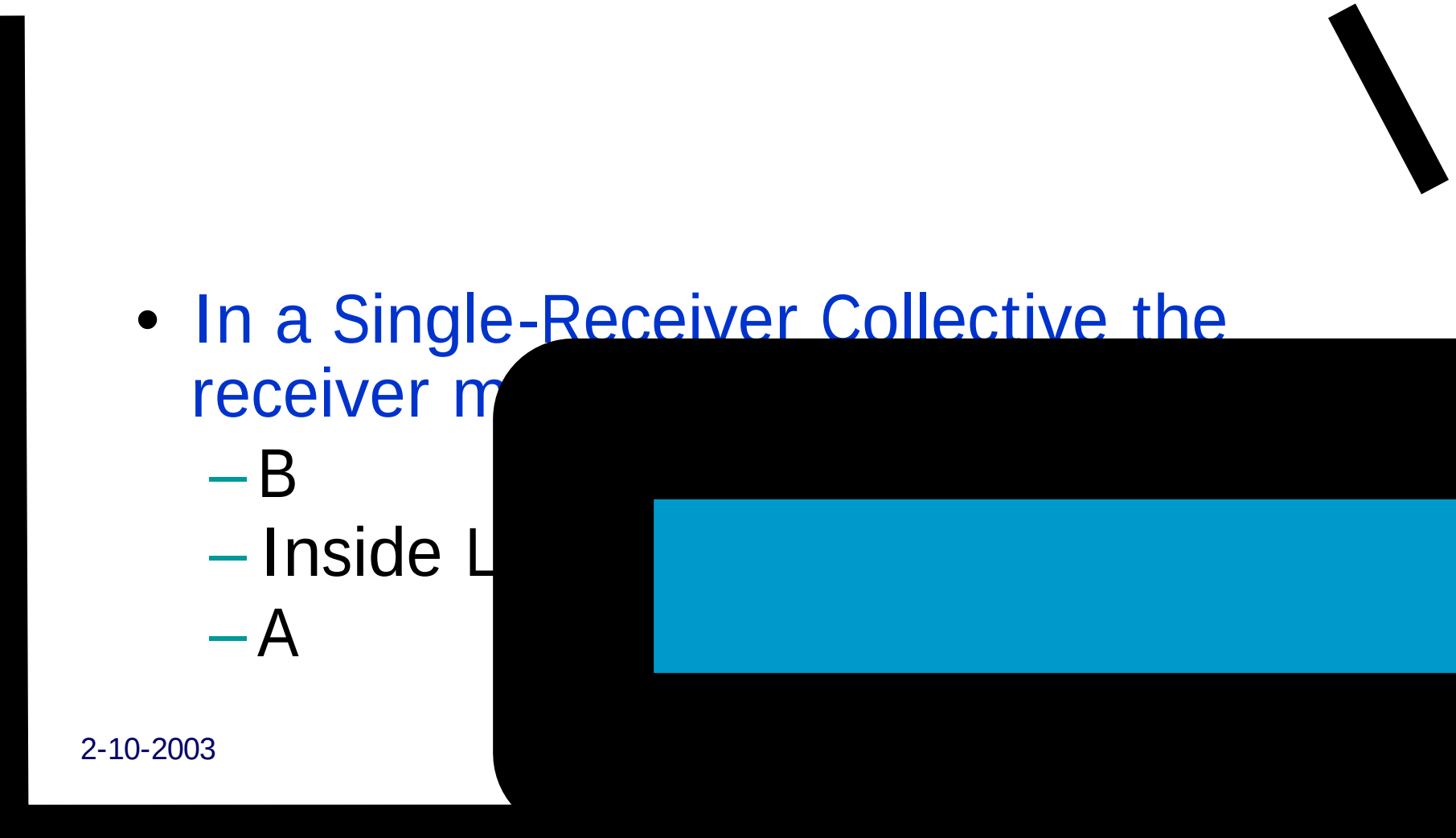
- How do we
  - Deal with non-

Possible Solutions

- We have a protocol for point-to-p

Single-Receiver Collectives

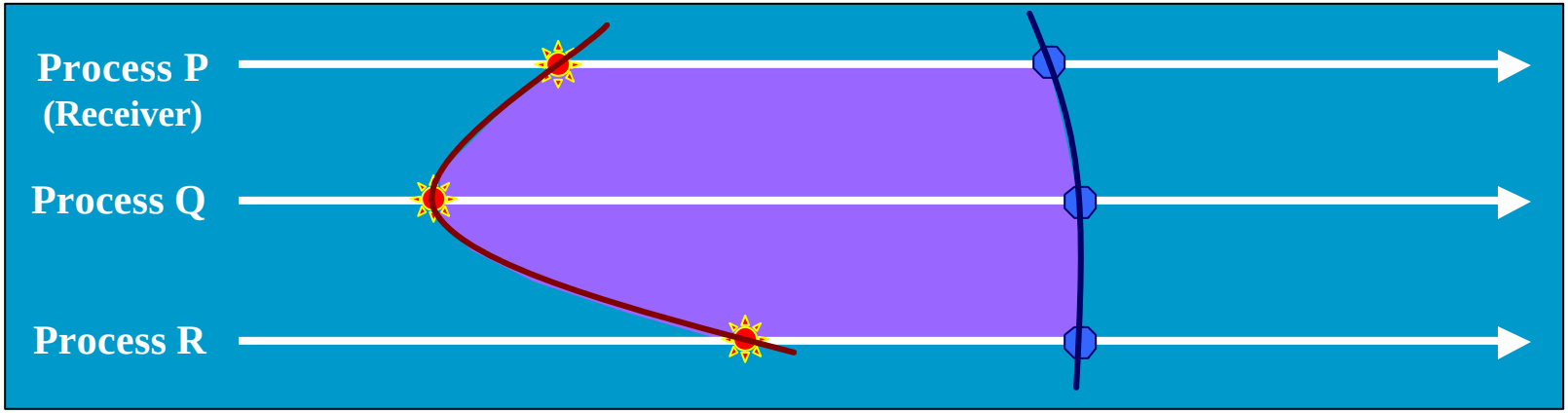
MPI_Gather(), MPI_Reduce()



- In a Single-Receiver Collective the receiver m
  - B
  - Inside L
  - A


Process P

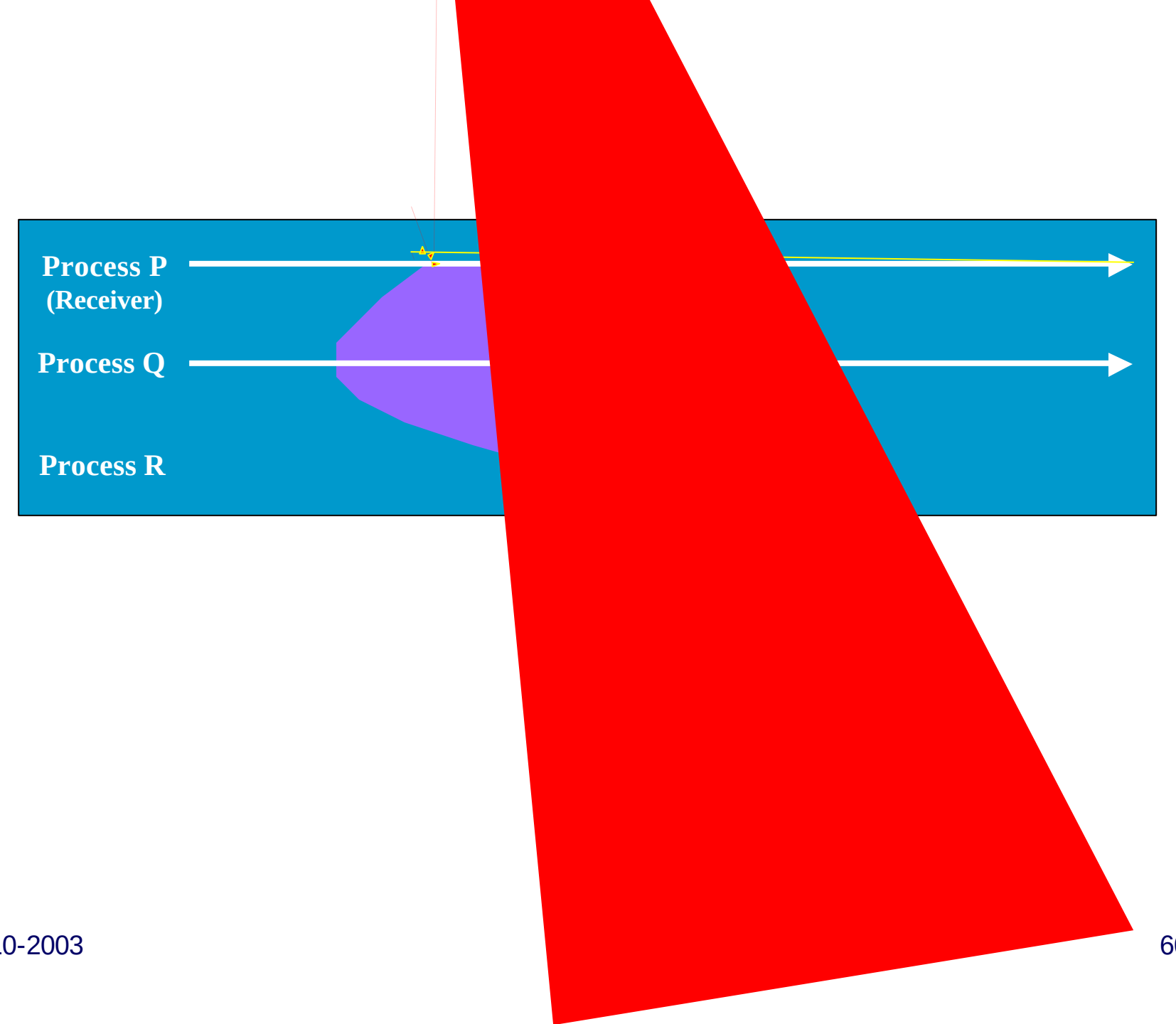
Single-Receiver Collectives



Single-Receiver Collectives

Receive is before the checkpoint

- Therefore, since neither Q or R will resend, we don't need to re-


Process P
(Receiver)
Process Q
Process R

Shared Memory

- Symmetric Multiprocessors – nodes of several (2-64) processors connected by a fast network.
- Different nodes are connected by a slower network.
- Typical communication style:
  - Hardware shared memory inside the node
  - MPI-type message passing between nodes

OpenMP

- An industry standard shared memory API.
- Goal: create a thin layer on top of OpenMP

Issues with checkpointing OpenMP

- Parallel for
  - different threads execute different iterations in parallel
  - iteration assignment is non-deterministic
- Flush
  - shared data that has been locally updated by different threads is redistributed globally
- Locks
  - carry only synchronization, no data

OpenMP – parallel for

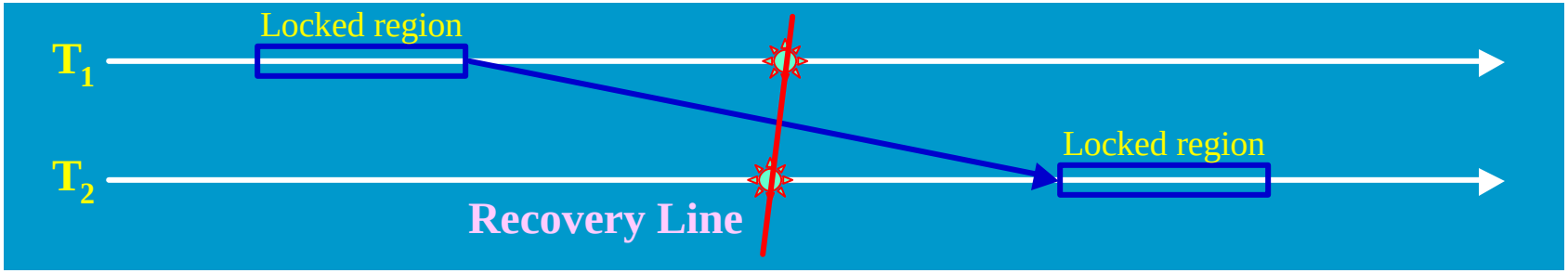
- While executing a parallel for we keep track of which iterations we've completed.

Above: [0,1,2,5] are completed
[7] is in progress

OpenMP – parallel for

- If any thread in a recovery line checkpoints inside a parallel for, we must

OpenMP – Locks



- Locks are data flows that carry no data.
- This lock flow is trivial to enforce.
- Backwards lock flows are more complex.
- W6 annot guarantee true synchronization wrt outside world.