Granular Time Warp Objects

SAPIENZA
UNIVERSITÀ DI ROMA

Nazzareno Marziale
Francesco Nobilia
Alessandro Pellegrini
Francesco Quaglia

High Performance and Dependable
Computing Systems Group
Sapienza, University of Rome

PADS 2016

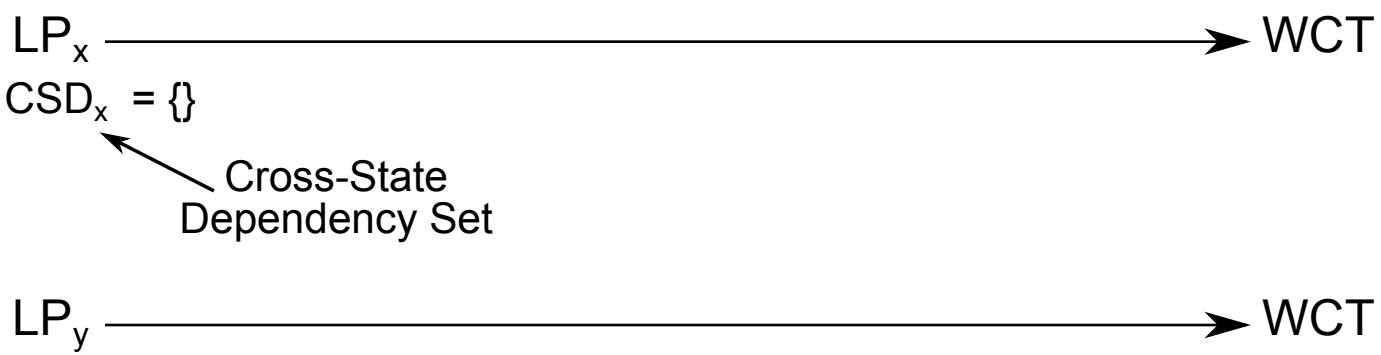
The Problem

- Event-Cross State Synchronization allows multiple LPs to exchange information via in-place memory accesses
- The target is shared-memory multicore systems

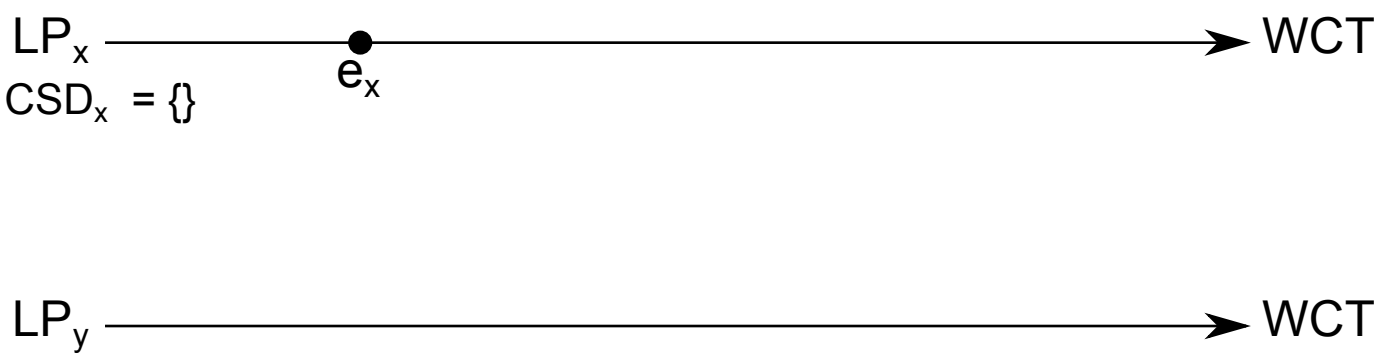
- LPs do not need to be disjoint entities anymore
- Exchange of large amount of data is faster
- Allows for a simpler programming model

- Syncrhonization with ECS can be costly
- This is even more the case when LPs cross-synchronize a lot

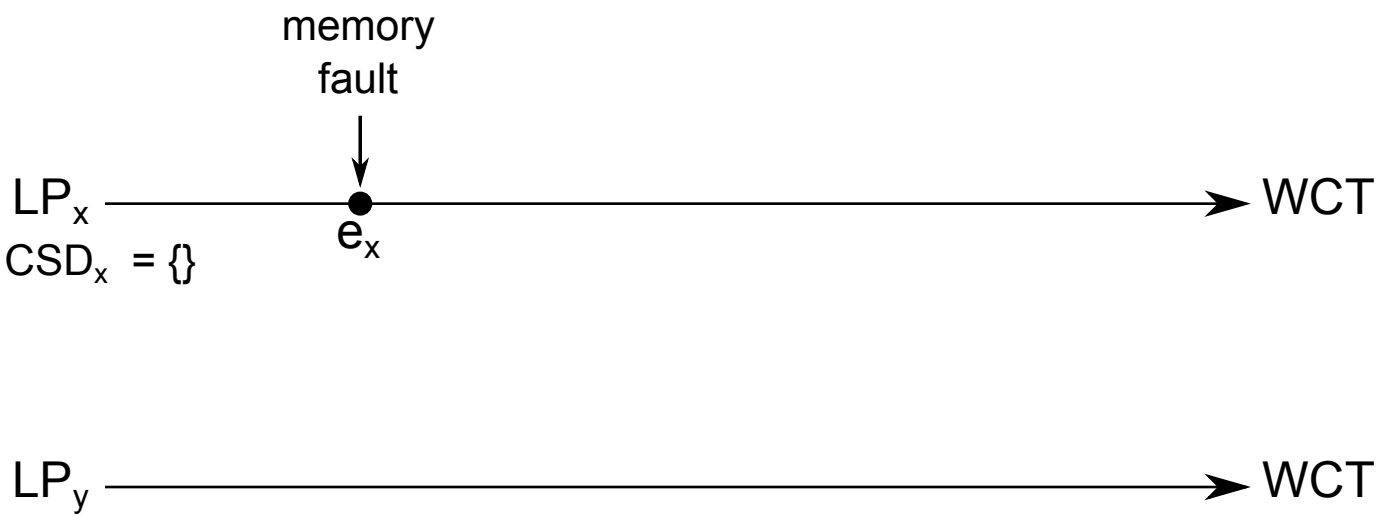
Event Cross-State Synchronization [PADS 2014]



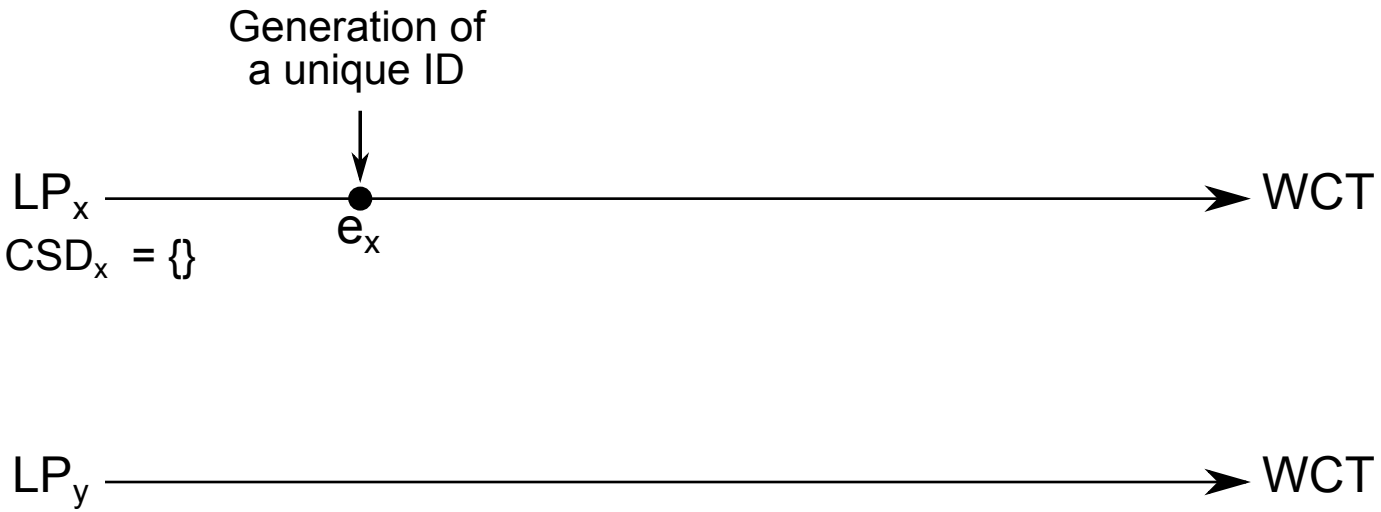
Event Cross-State Synchronization [PADS 2014]



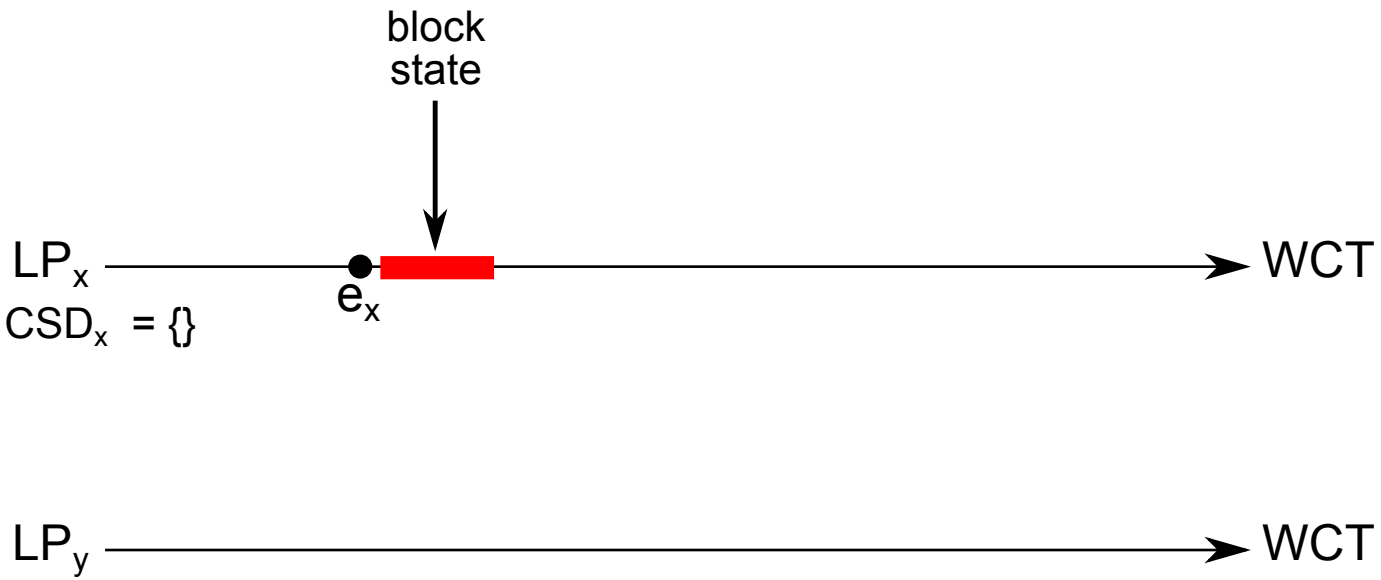
Event Cross-State Synchronization [PADS 2014]



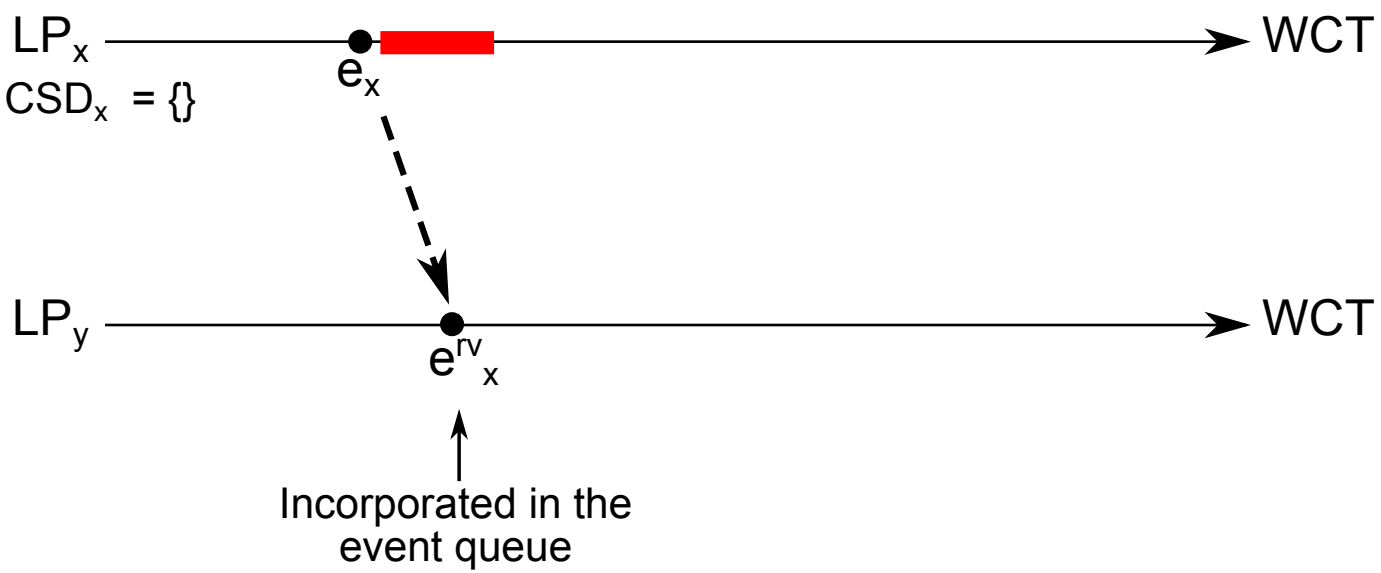
Event Cross-State Synchronization [PADS 2014]



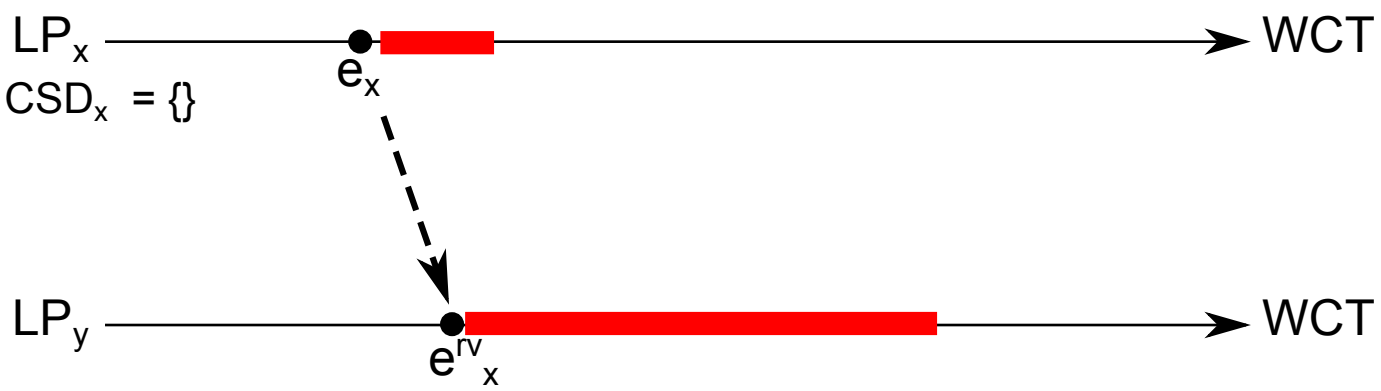
Event Cross-State Synchronization [PADS 2014]



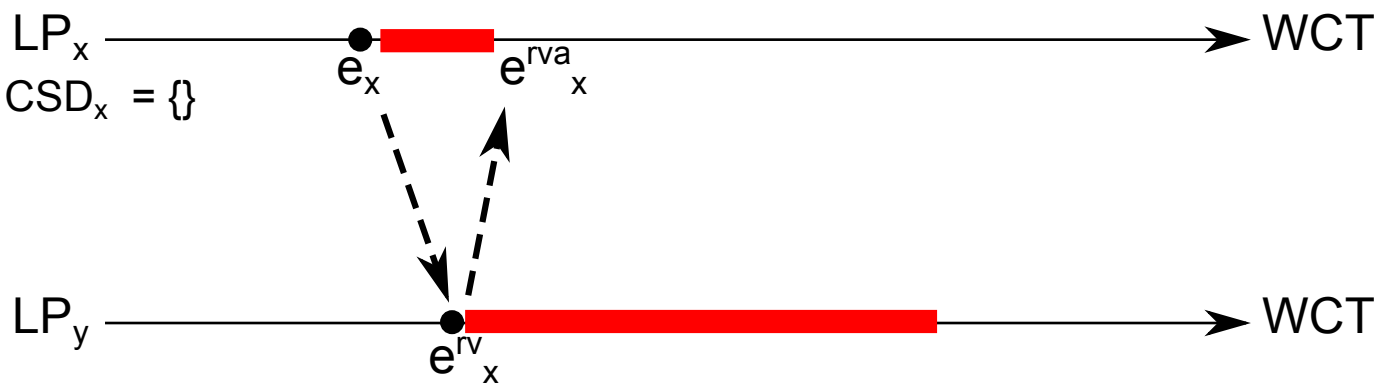
Event Cross-State Synchronization [PADS 2014]



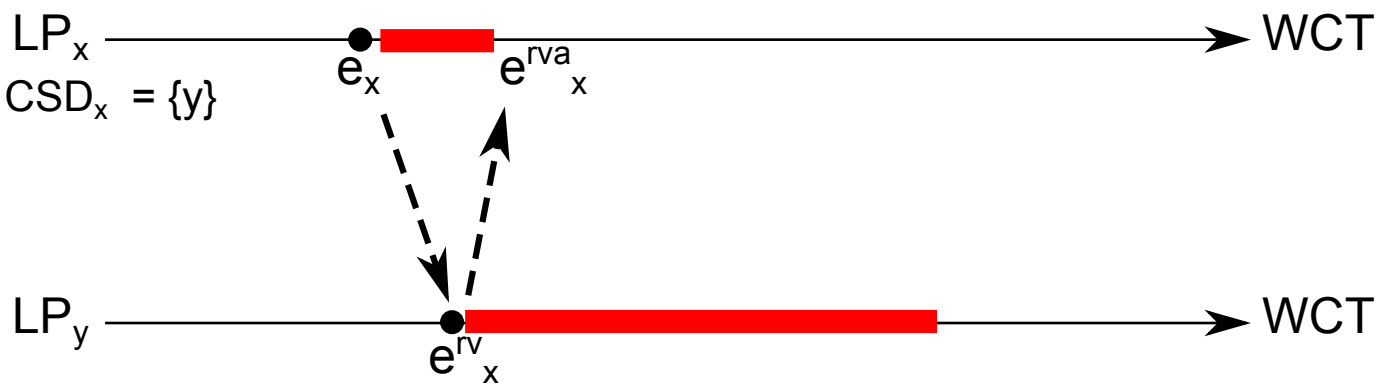
Event Cross-State Synchronization [PADS 2014]



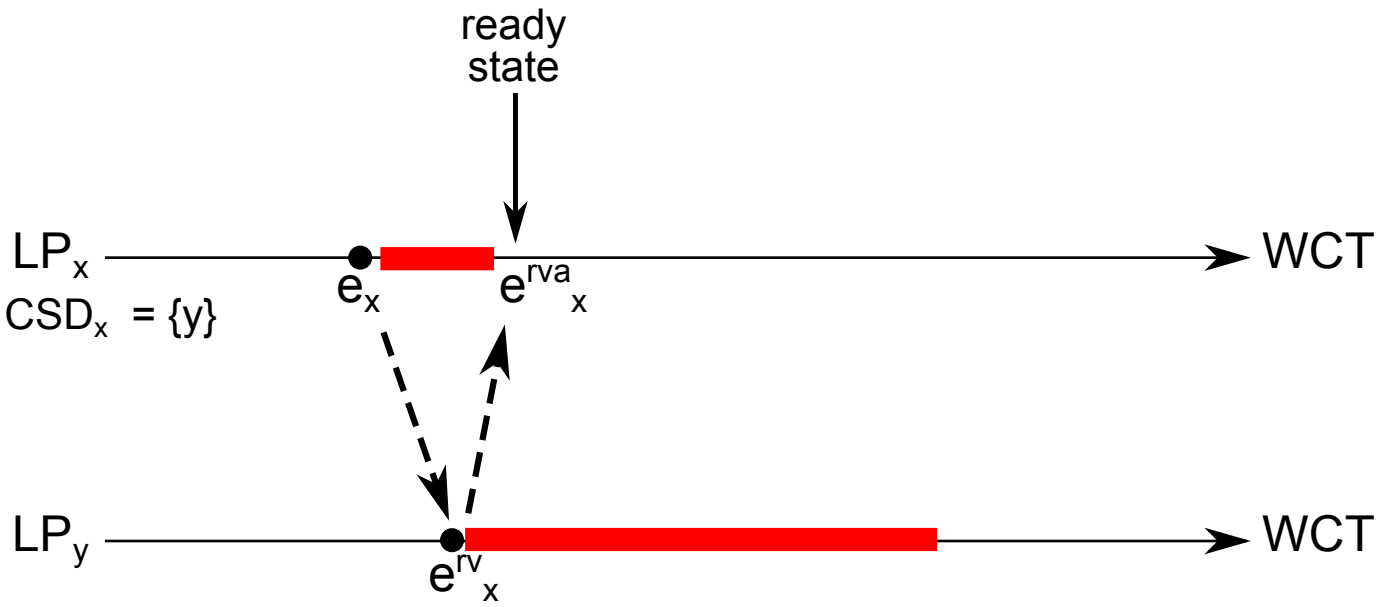
Event Cross-State Synchronization [PADS 2014]



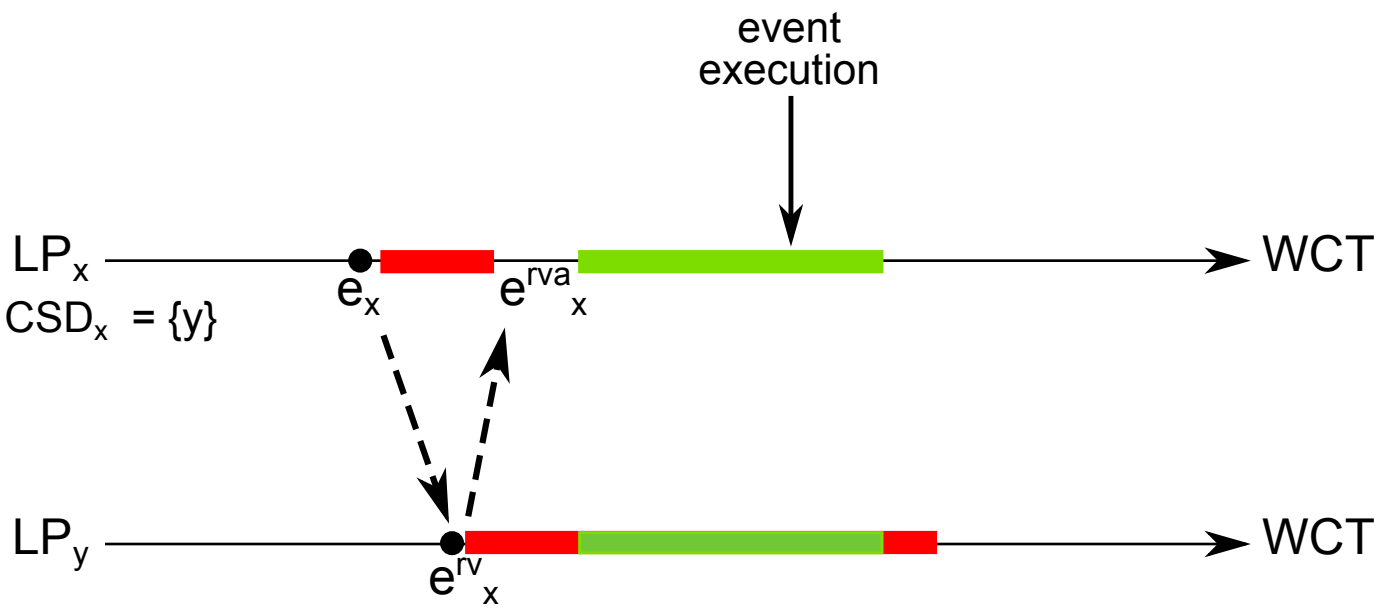
Event Cross-State Synchronization [PADS 2014]



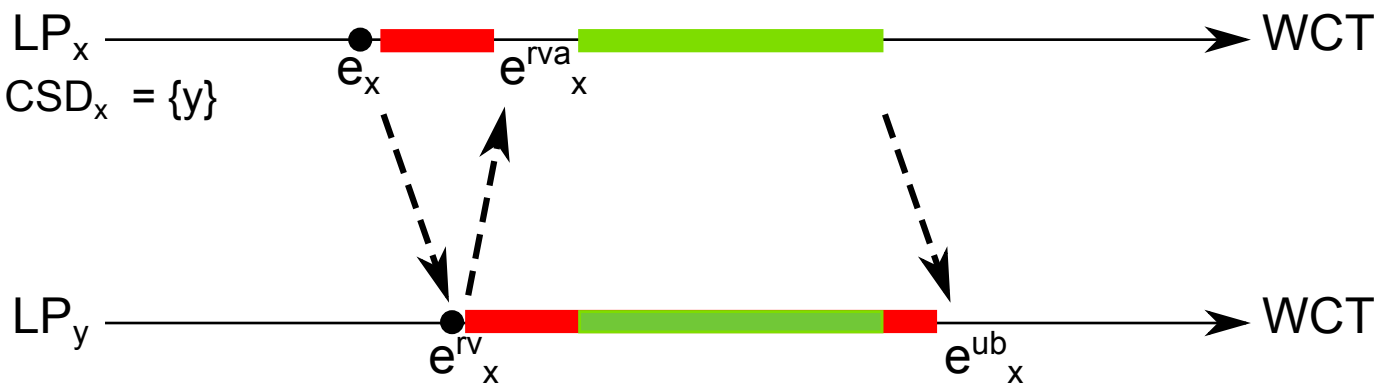
Event Cross-State Synchronization [PADS 2014]



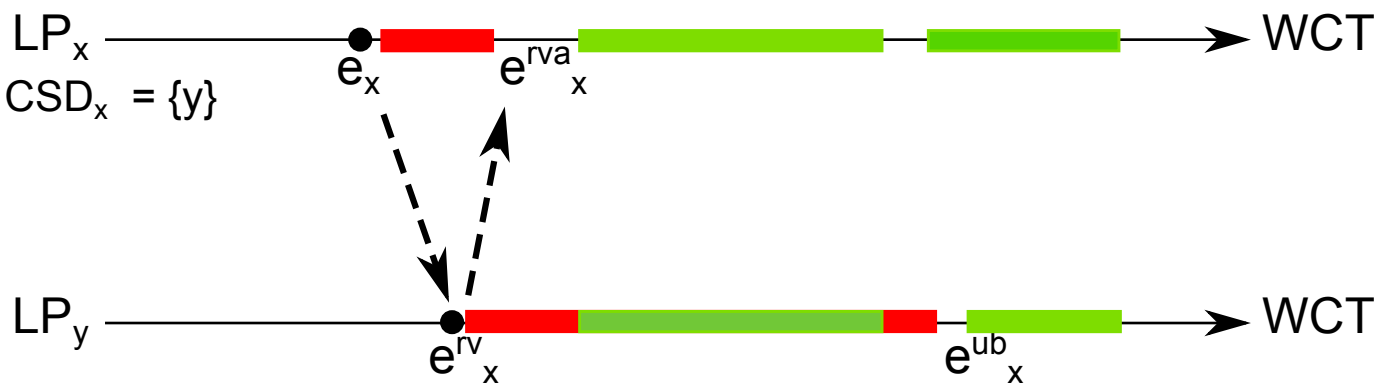
Event Cross-State Synchronization [PADS 2014]



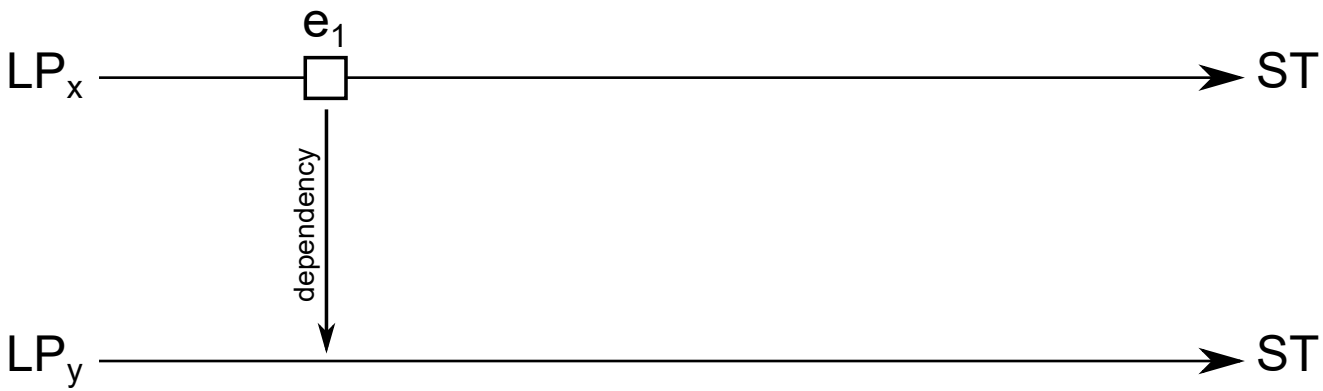
Event Cross-State Synchronization [PADS 2014]



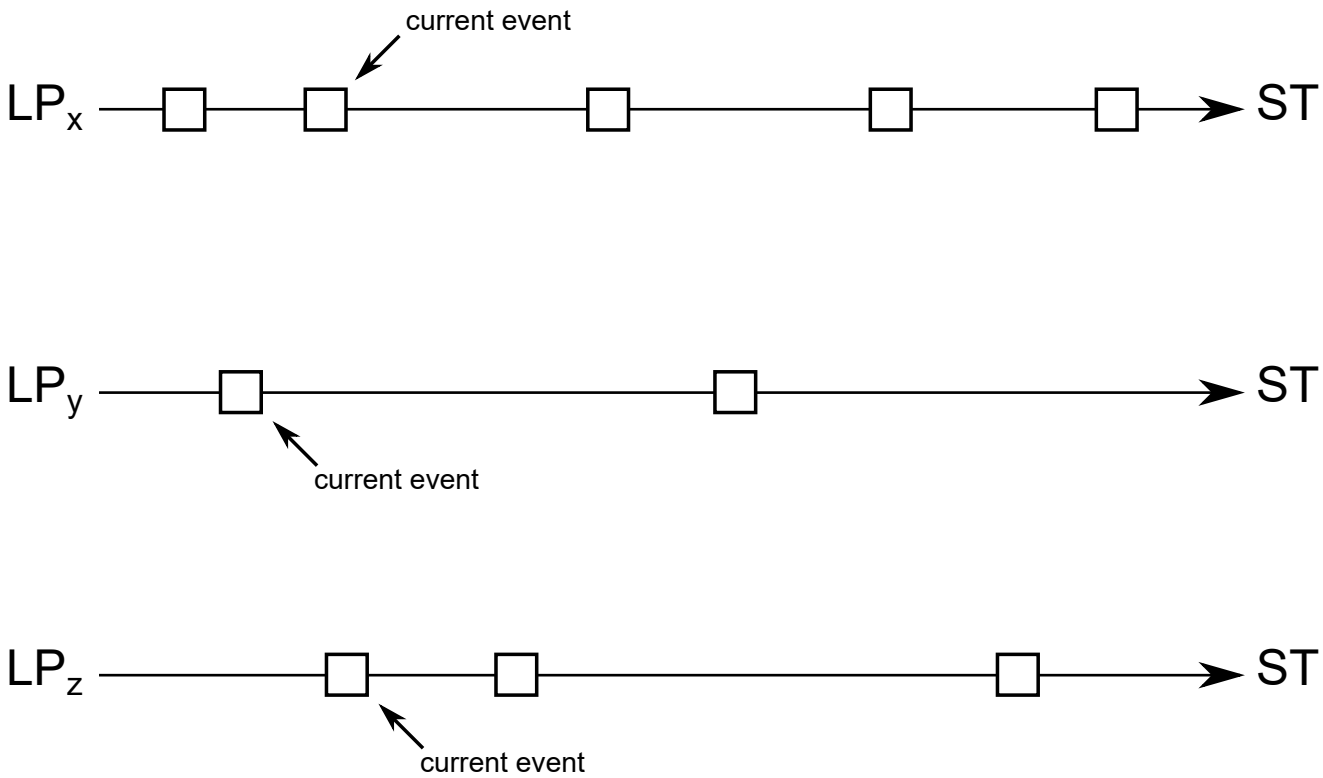
Event Cross-State Synchronization [PADS 2014]



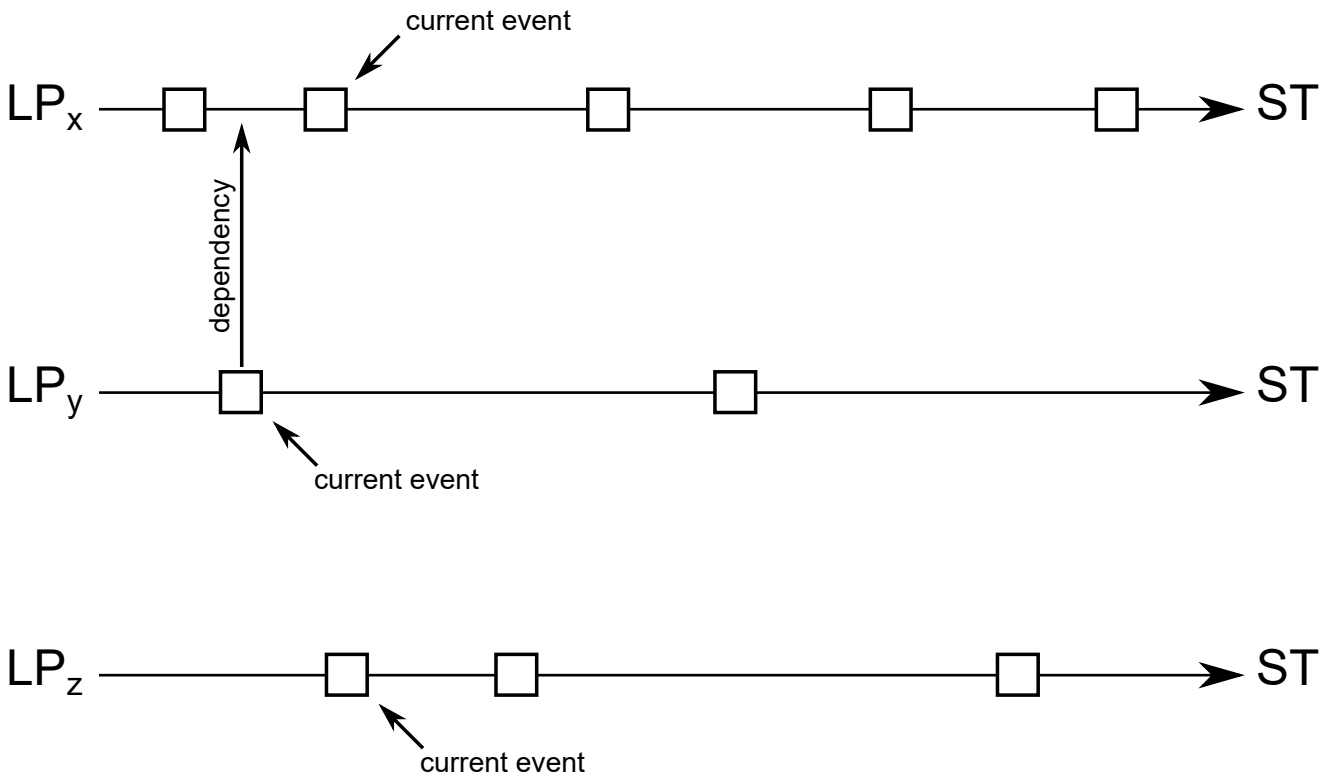
Event Cross-State Synchronization [PADS 2014]



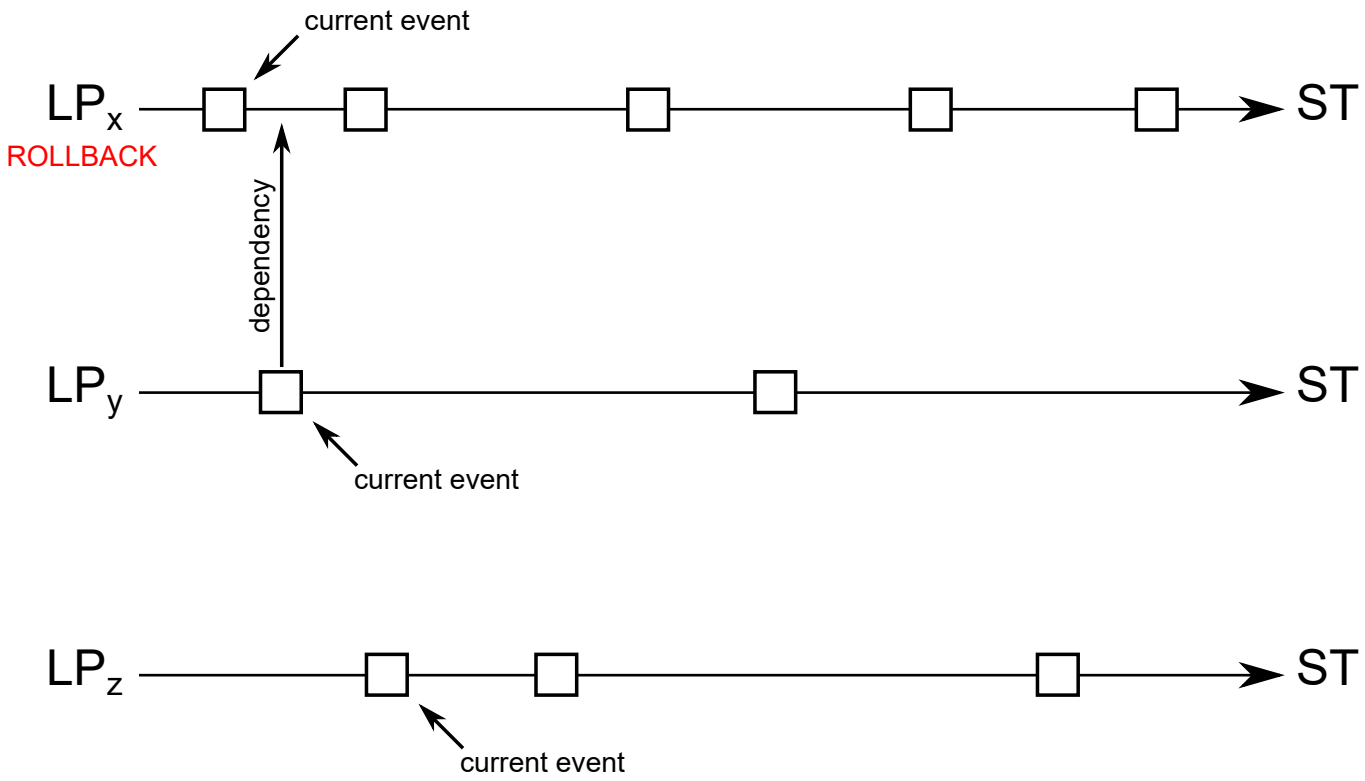
A Performance Problem with ECS



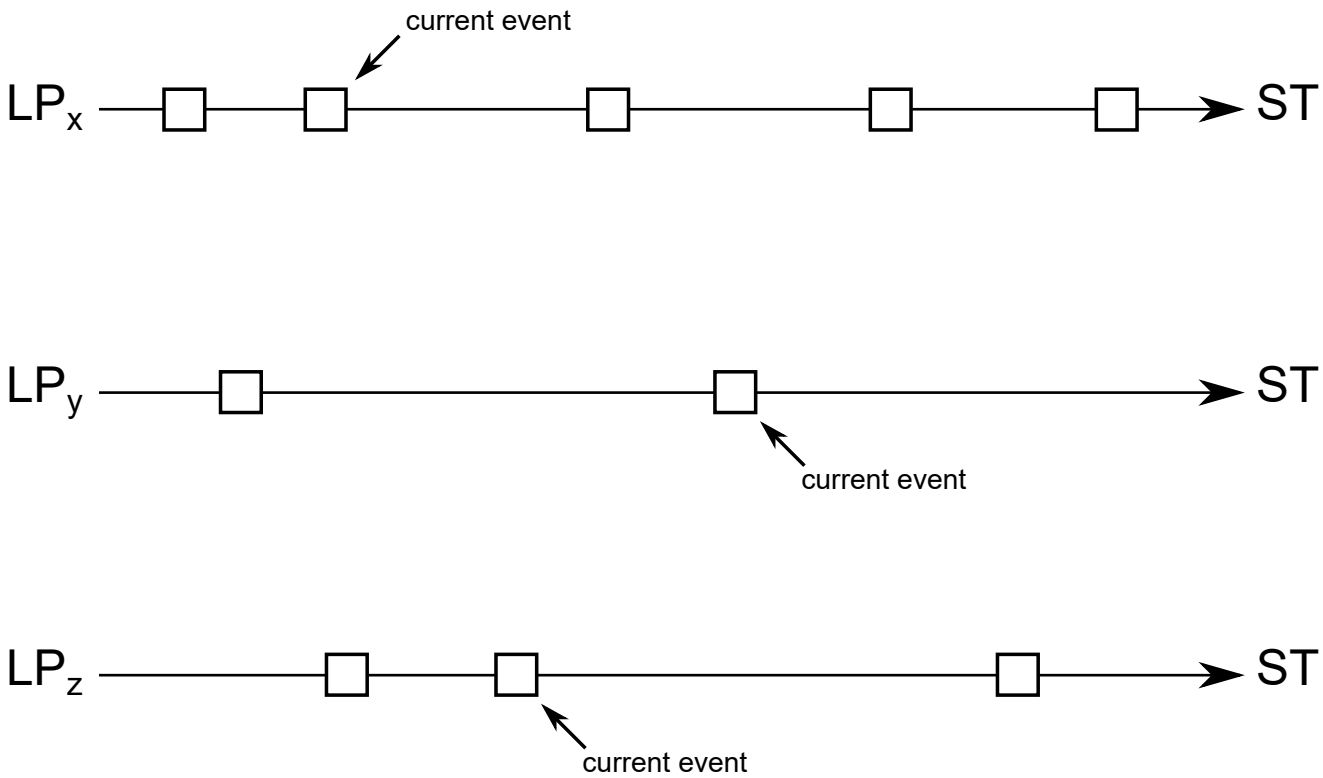
A Performance Problem with ECS



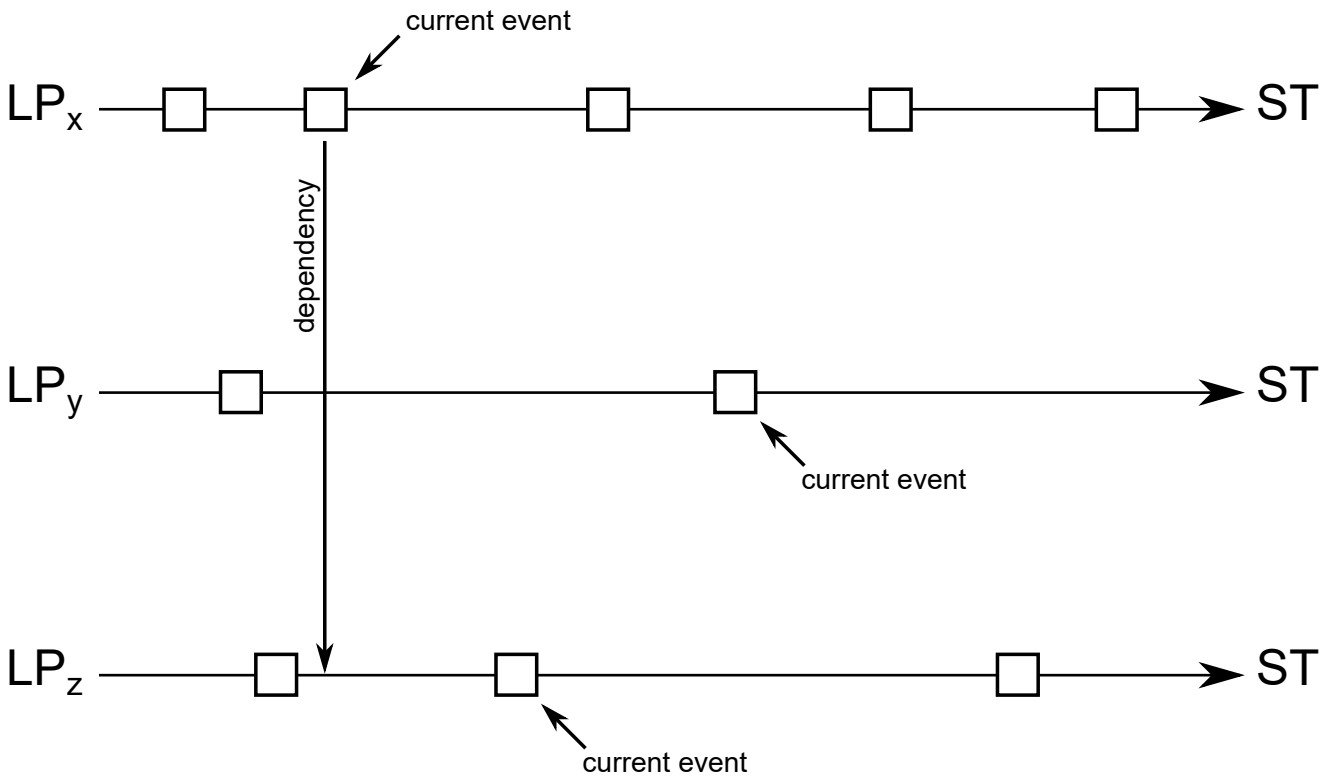
A Performance Problem with ECS



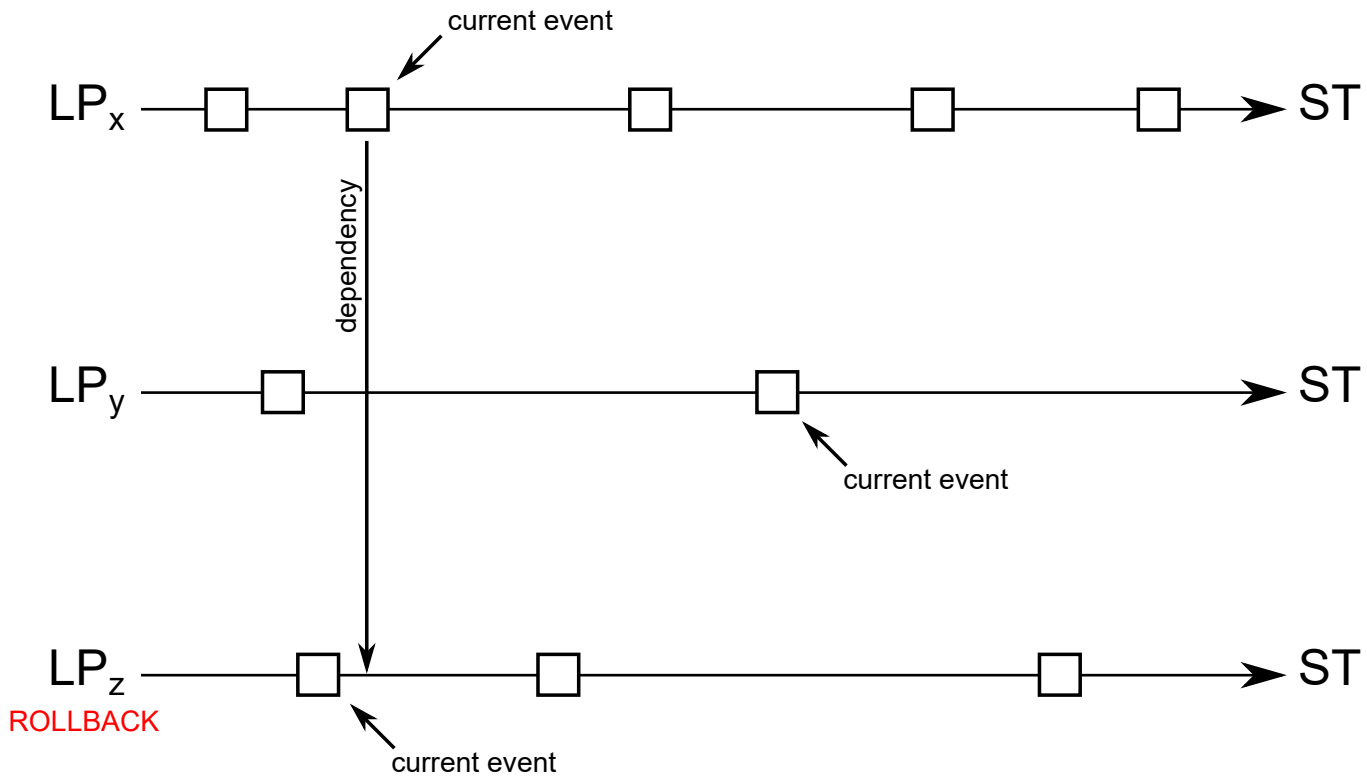
A Performance Problem with ECS



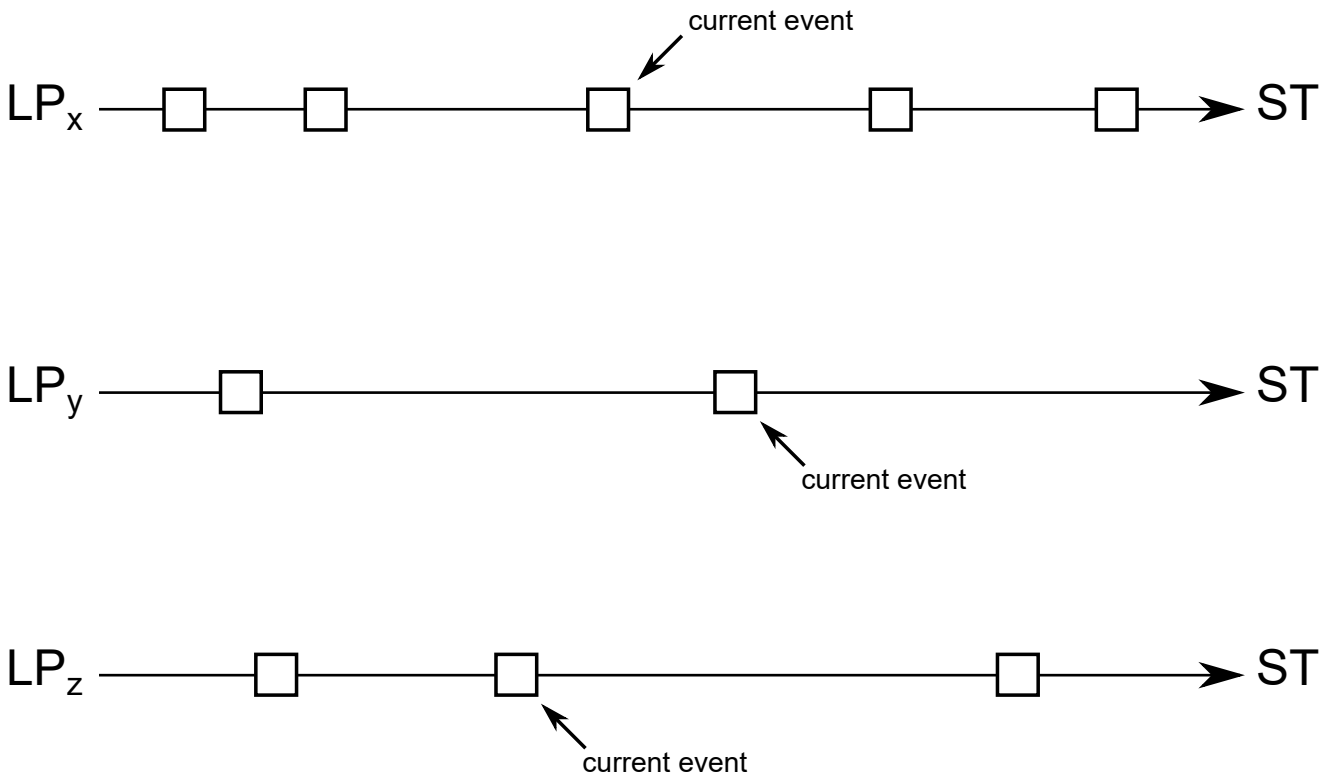
A Performance Problem with ECS



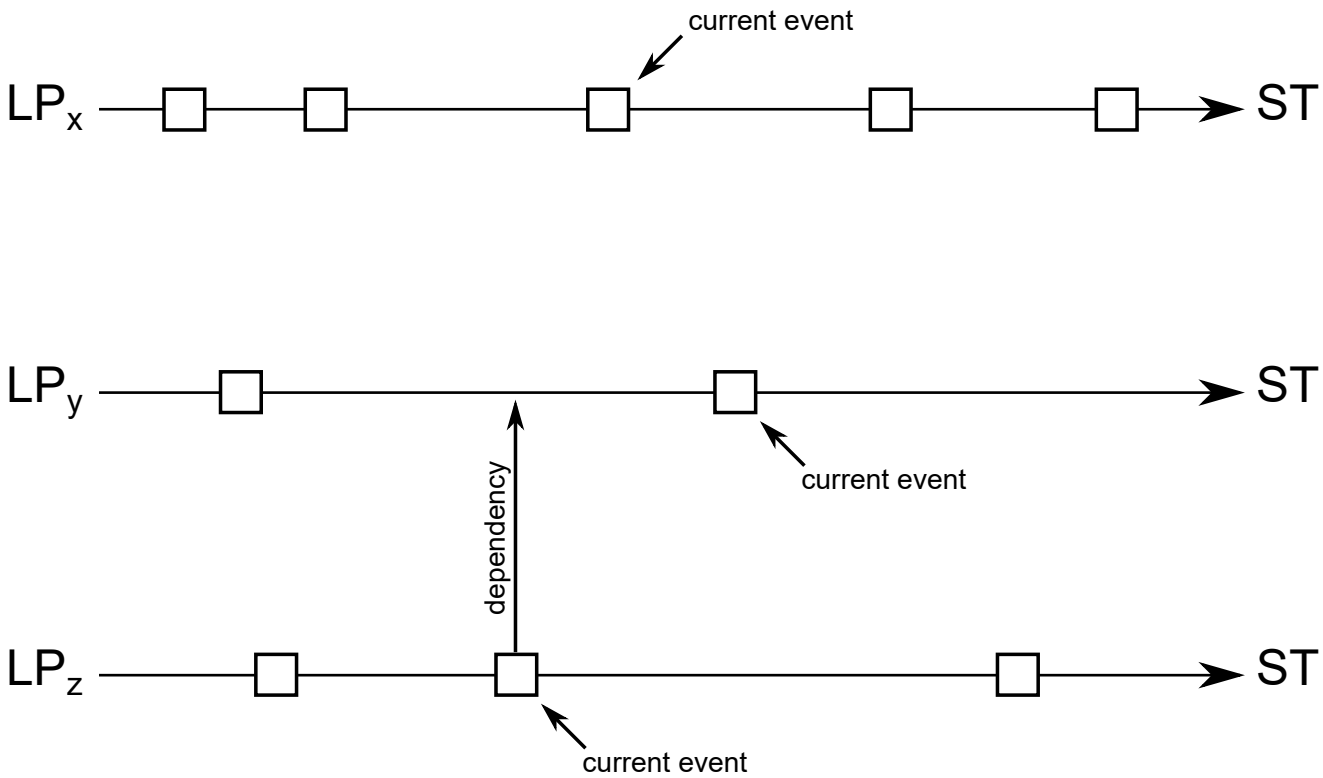
A Performance Problem with ECS



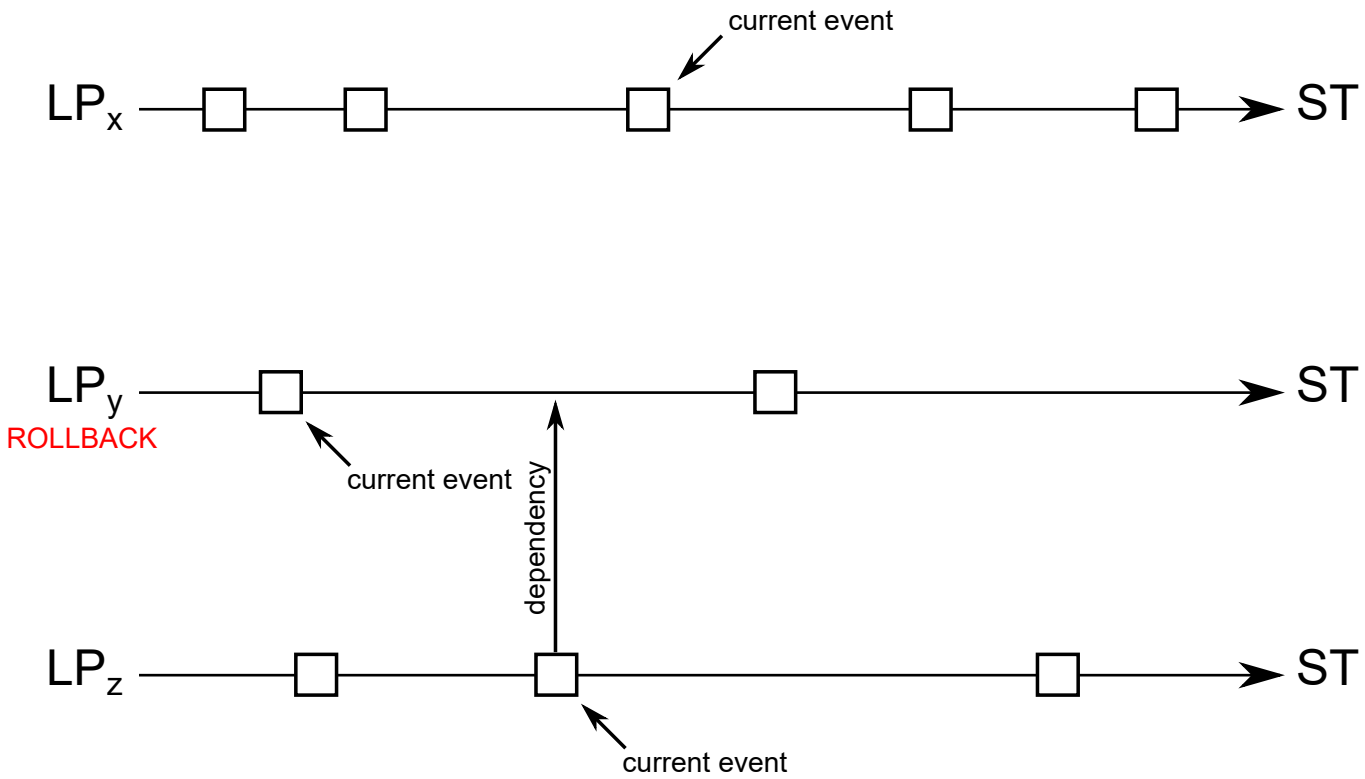
A Performance Problem with ECS



A Performance Problem with ECS



A Performance Problem with ECS



Granular Time Warp Objects

- ECS dependencies are always considered ephemeral
- They could represent some property of the model to capture

- Granular LPs (GLP) are dynamic clusters of LPs which execute events in timestamp order
  - This limits the optimism
  - Tries to capture a synergistic execution phase of the model

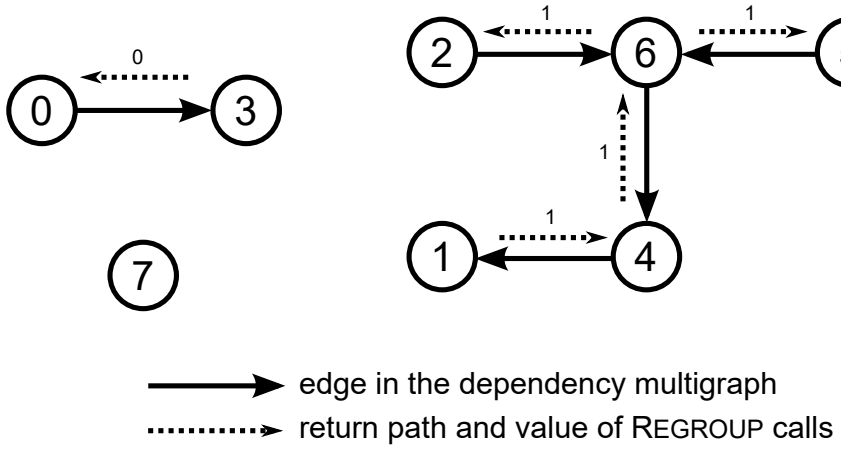
- Granulation is re-evaluated during the simulation, to account for different phases

Grouping LPs

- The *materialization* of a cross-state access should be used to build a *relation* among LPs
- We use the *LpDependencies* matrix to count ECS interactions
  - $LpDependencies[i, j] = LpDependencies[j, i] =$ number of ECS interactions
  - Small values are filtered out—τ_{dep} threshold
- Periodically, this matrix is used to build a Directed Multigraph over the LPs
- This considers, for each LP_k, the LP_i with the highest dependency count—$MaxDep_k$
- A graph visiting algorithm is used to build a GLP

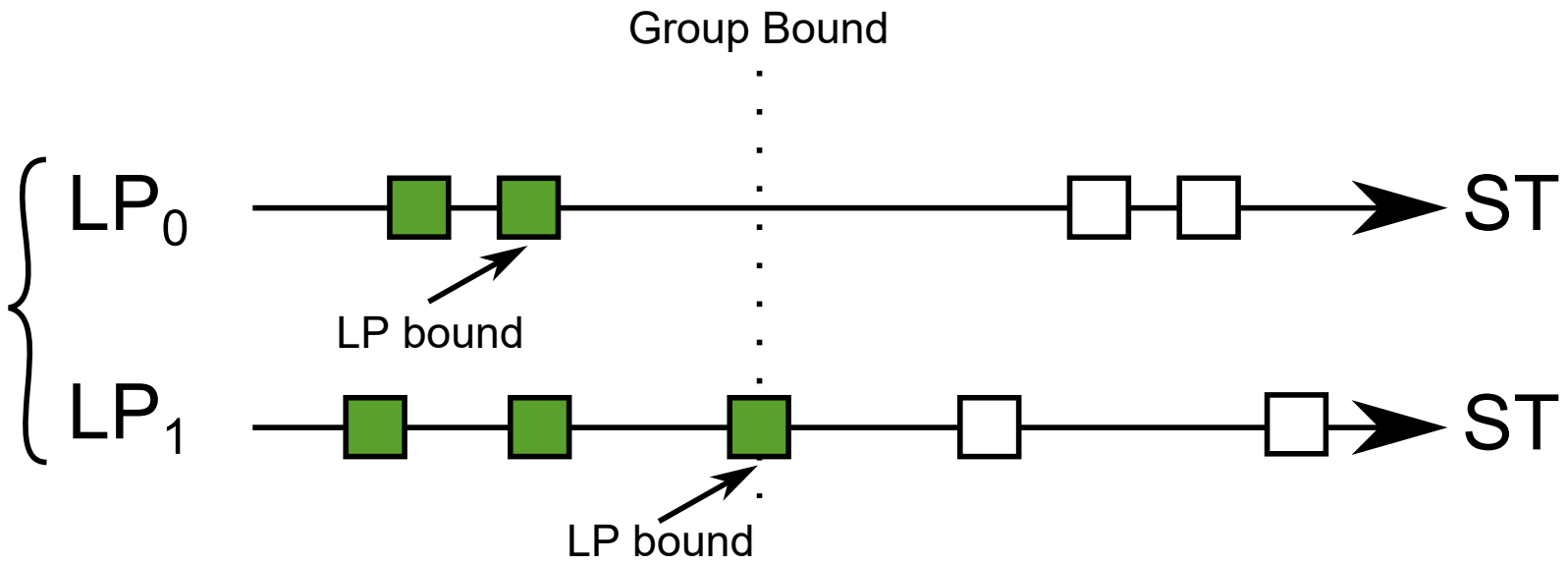
Grouping LPs

```
 1: procedure REGROUP(LpGranulation GLP, int LPid, int group)
 2:     if GLP[LPid].group ≠⊥ then
 3:         return GLP[LPid].group
 4:     if group ≠⊥ then
 5:         GLP[LPid].group ← group
 6:     else
 7:         GLP[LPid].group ← LPid
 8:     if GLP[LPid].MaxDep ≠⊥ then
 9:         GLP[LPid].group = REGROUP(GLP, GLP[LPid].MaxDep, GLP[LPid].group)
10:     return GLP[LPid].group
```



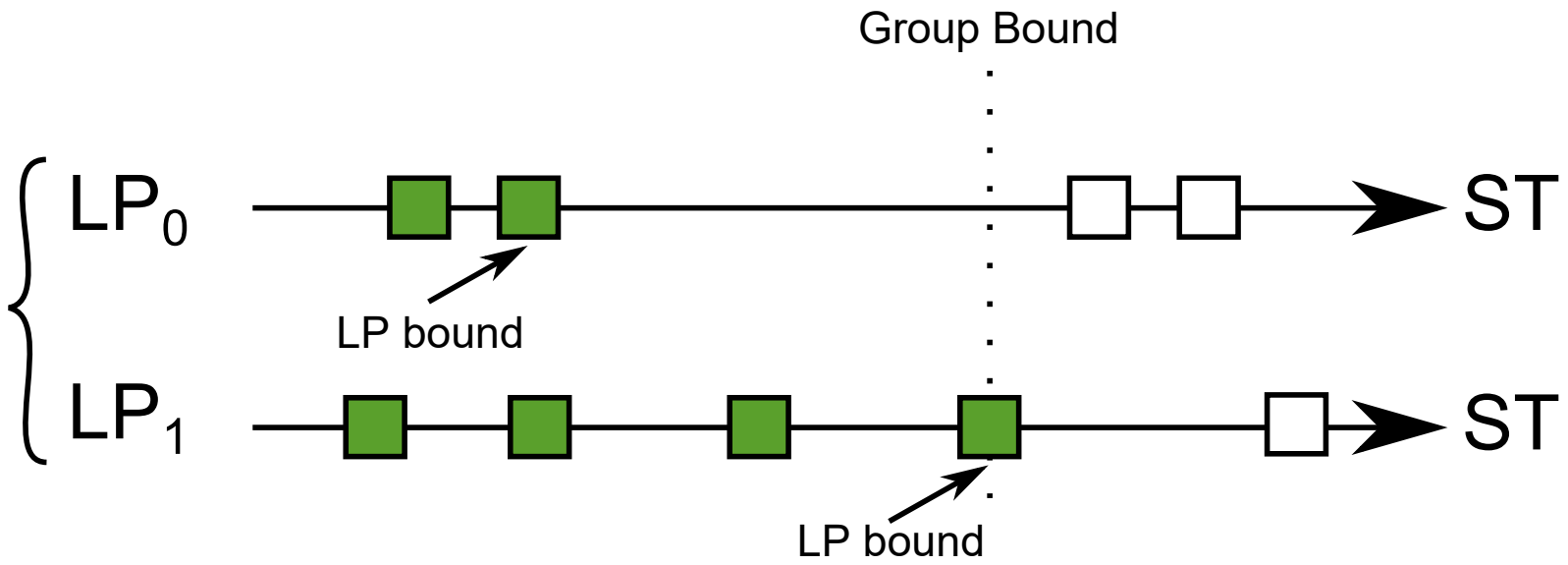
Revelation of a GLP

- All LPs in a GLP are *bound* to the same worker thread and can access any LP state in the GLP
- At this point, a group is *determined*, but not yet *revealed*
  - LPs were executing independently: we must avoid anomalies in case of speculative execution



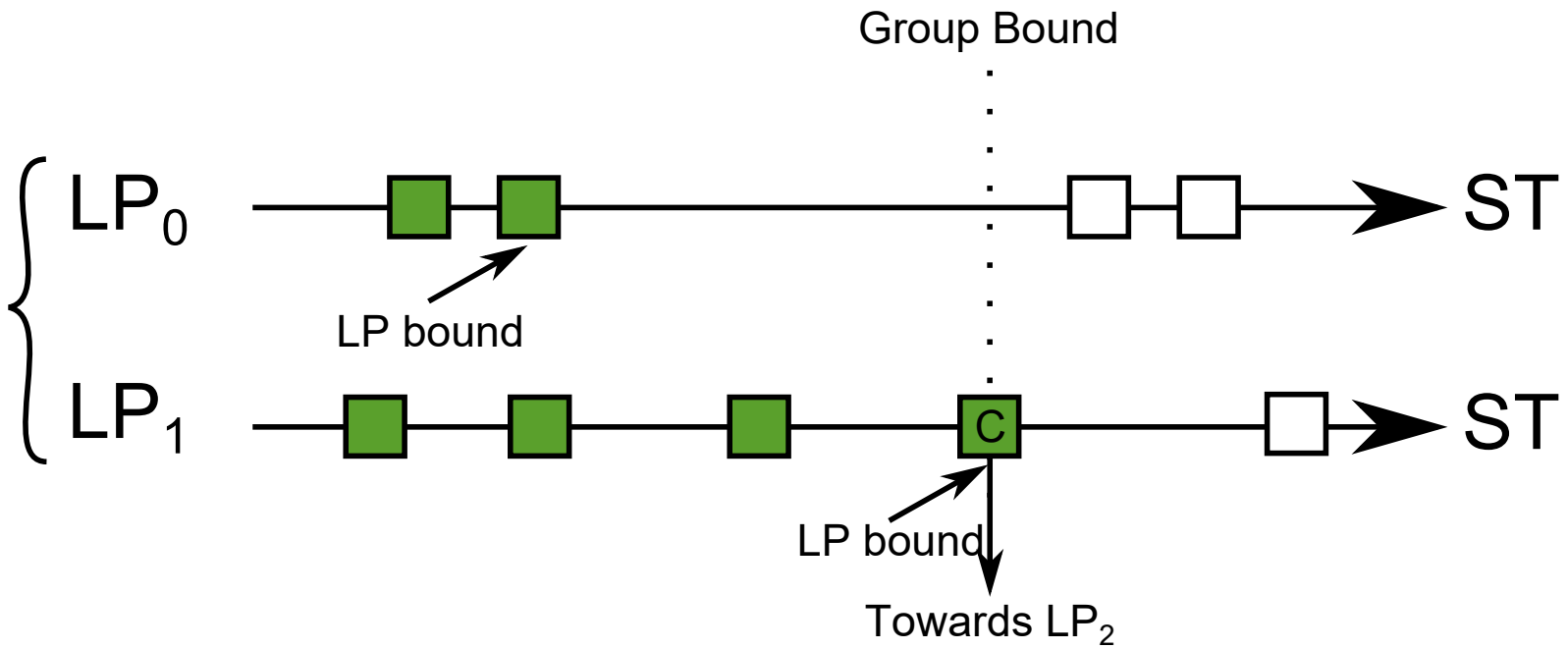
Revelation of a GLP

- All LPs in a GLP are *bound* to the same worker thread and can access any LP state in the GLP
- At this point, a group is *determined*, but not yet *revealed*
  - LPs were executing independently: we must avoid anomalies in case of speculative execution



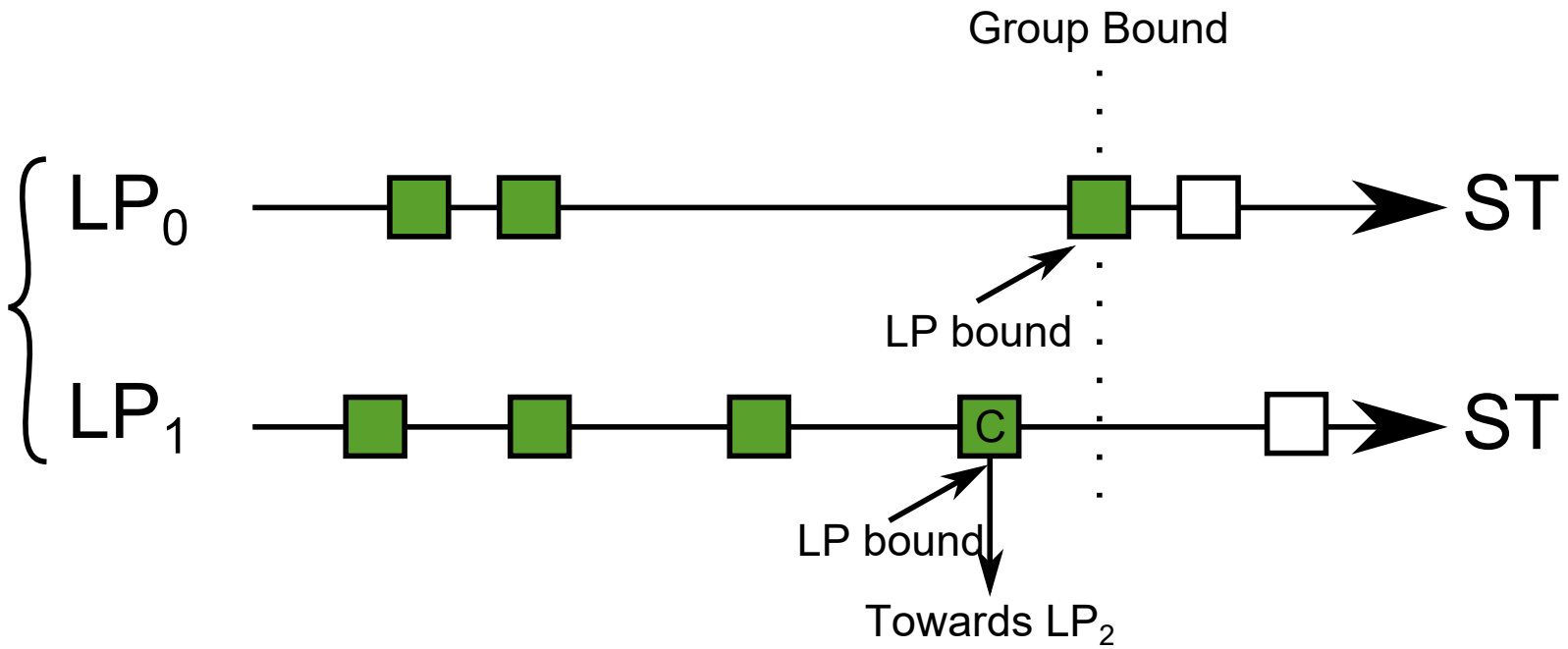
Revelation of a GLP

- All LPs in a GLP are *bound* to the same worker thread and can access any LP state in the GLP
- At this point, a group is *determined*, but not yet *revealed*
  - LPs were executing independently: we must avoid anomalies in case of speculative execution



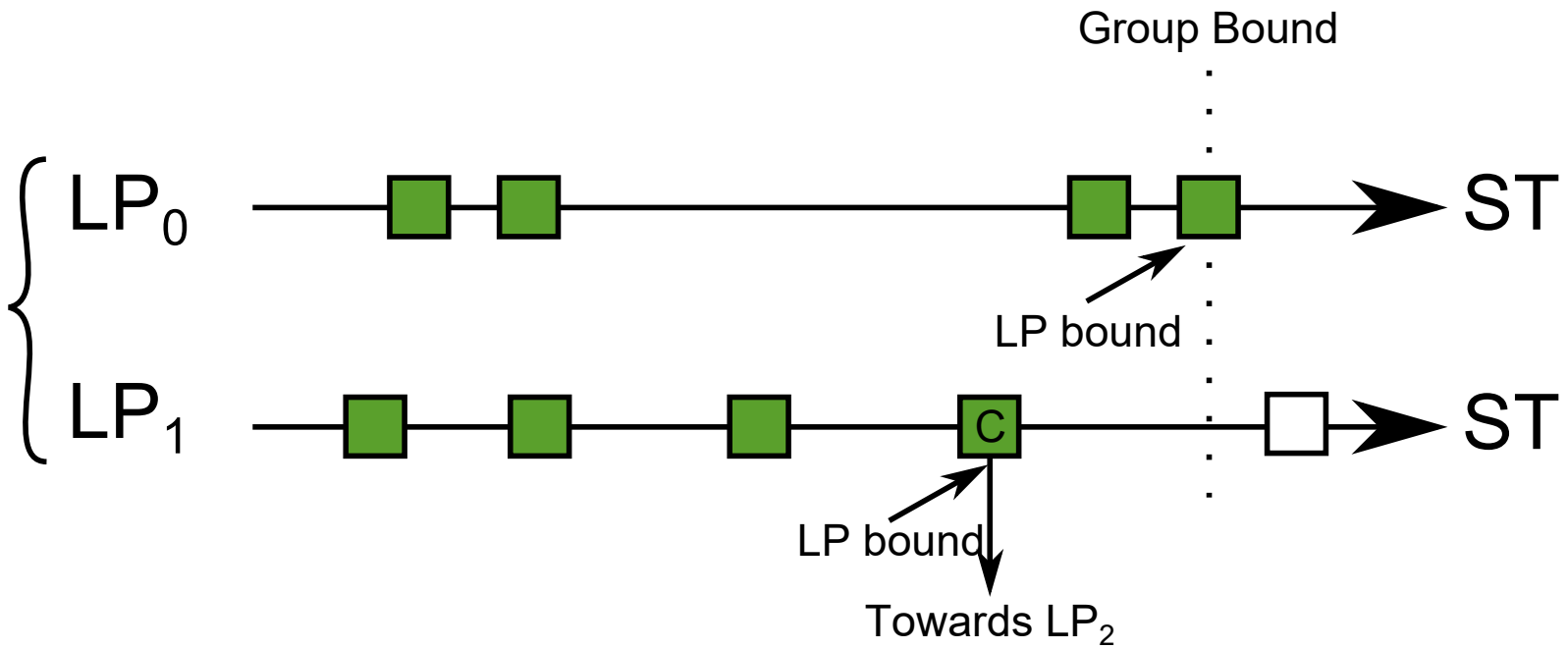
Revelation of a GLP

- All LPs in a GLP are *bound* to the same worker thread and can access any LP state in the GLP
- At this point, a group is *determined*, but not yet *revealed*
  - LPs were executing independently: we must avoid anomalies in case of speculative execution



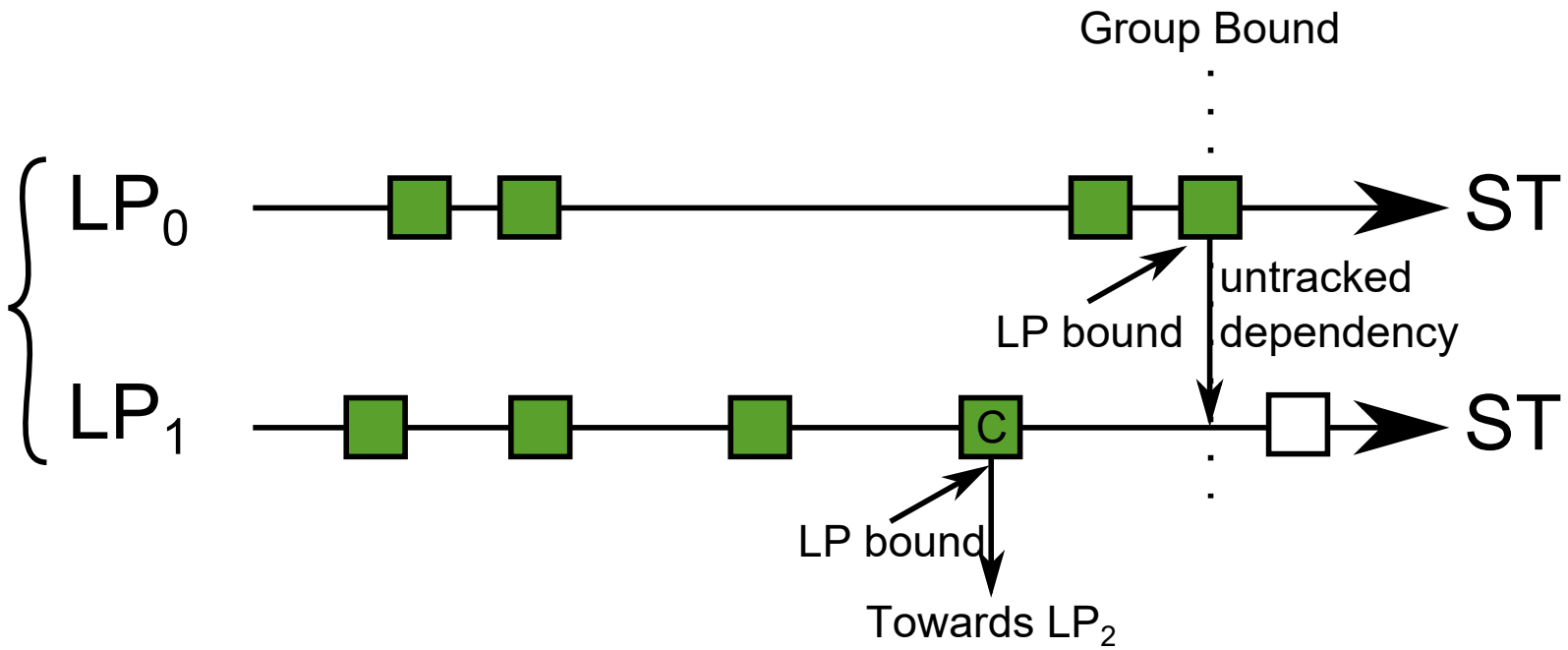
Revelation of a GLP

- All LPs in a GLP are *bound* to the same worker thread and can access any LP state in the GLP
- At this point, a group is *determined*, but not yet *revealed*
  - LPs were executing independently: we must avoid anomalies in case of speculative execution



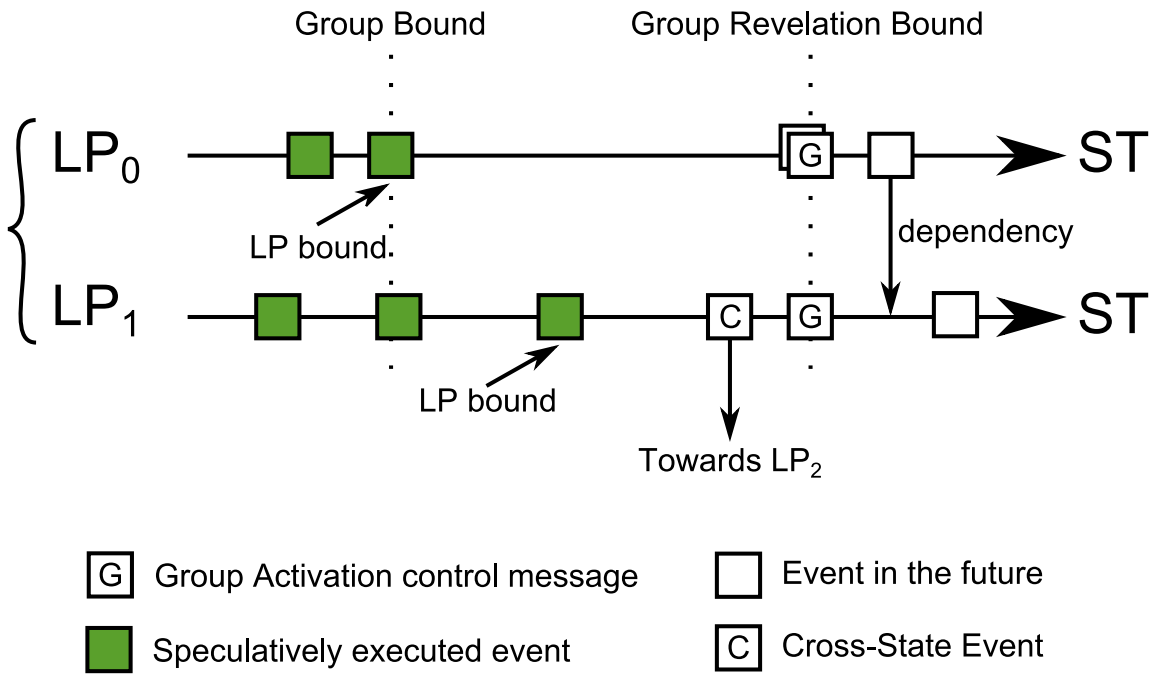
Revelation of a GLP

- All LPs in a GLP are *bound* to the same worker thread and can access any LP state in the GLP
- At this point, a group is *determined*, but not yet *revealed*
  - LPs were executing independently: we must avoid anomalies in case of speculative execution



Revelation of a GLP

- The anomaly is due to some LP still behaving as if the group were not set up
- We thus introduce the *group revelation control message*
- The group becomes revealed once all LPs have reached it



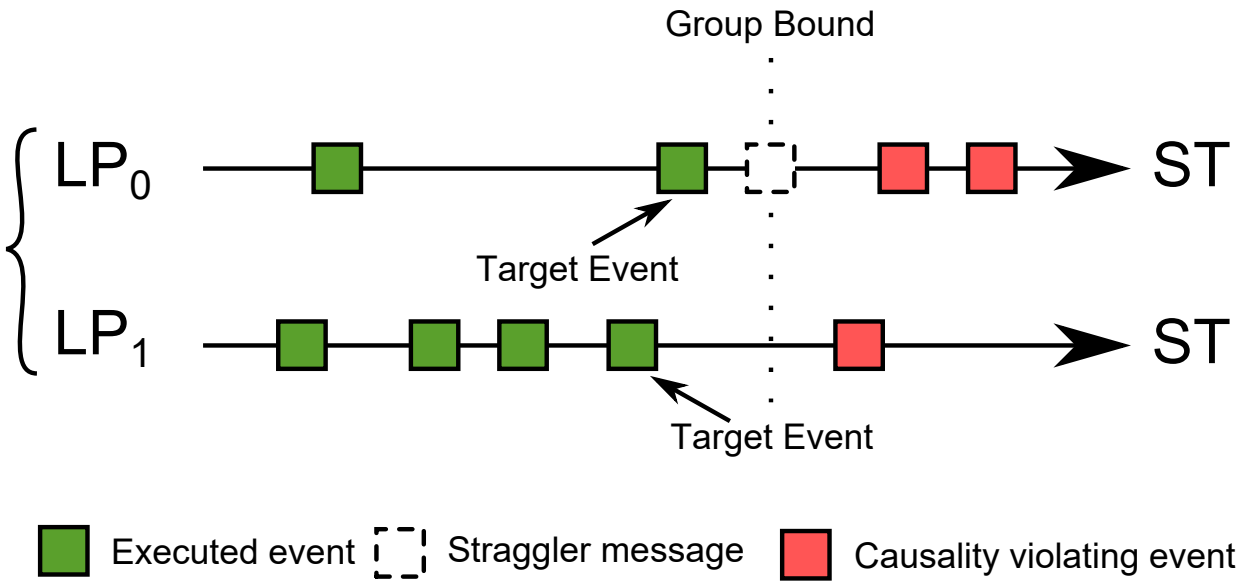
Who does what to setup a GLP?

- We do not want to stop processing events while recomputing groups
- Only one worker thread runs the graph visiting algorithm
- The new grouping and binding is posted in a shared variable
- An atomic counter is used to signal a new era
- All other worker threads eventually notice the new binding and install it

- This is the only actual synchronization point

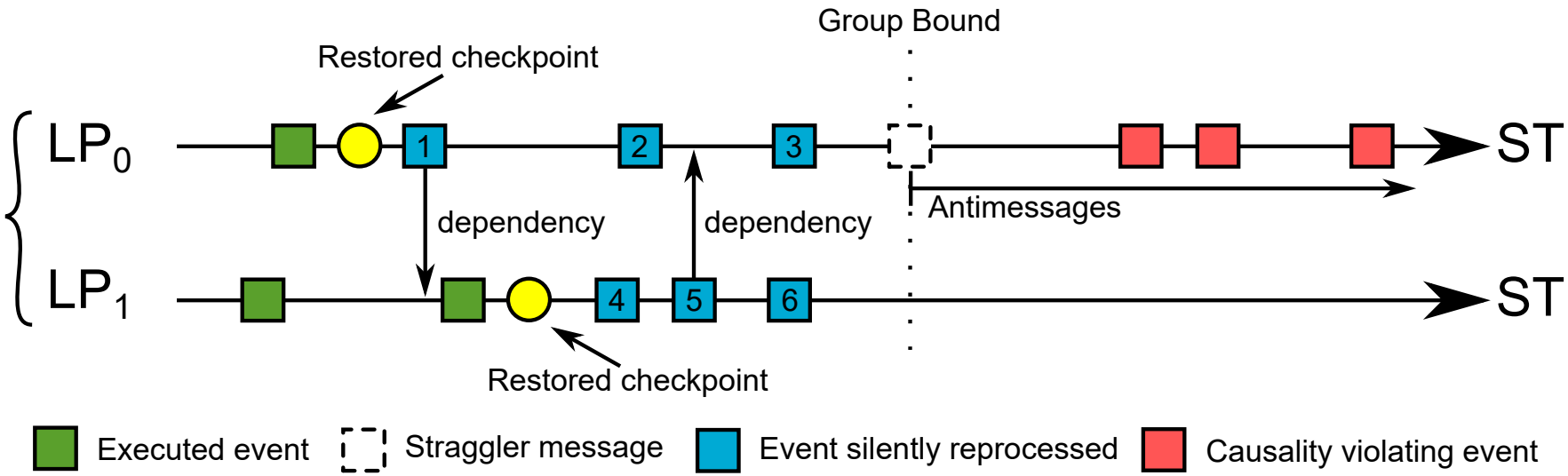
Rolling back a GLP: Straggler Messages

- In case a straggler hits a LP in a GLP, the GLP must be taken into account
- Rolling back other LPs is not an option!
- A GLP is a *unique logical object*



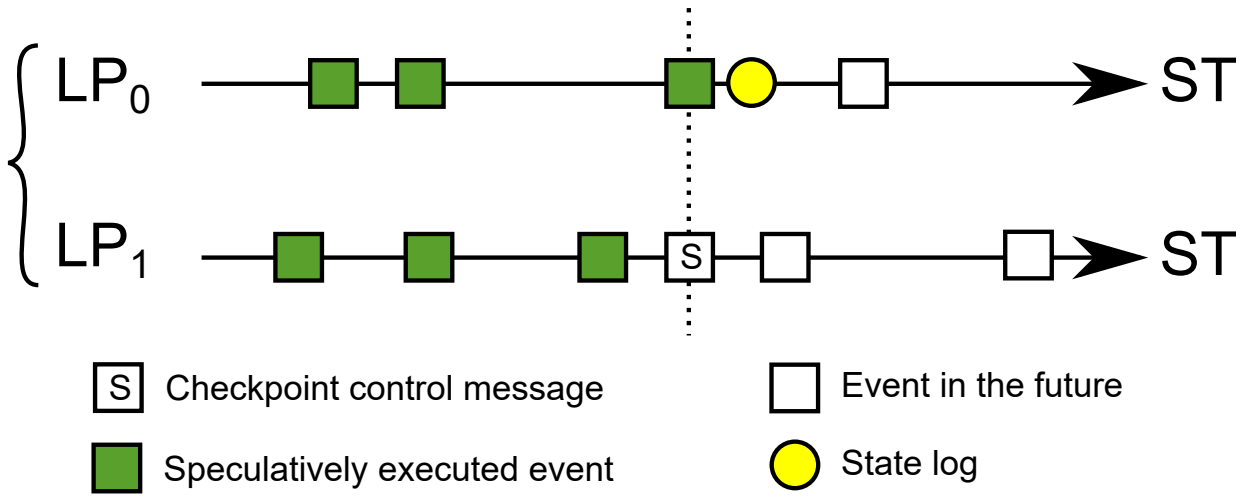
Cannot execute a rollback traditionally

- State saving/restore must be handled differently
- Two different anomalies might arise
- They are generated by inter-LP ECS dependencies

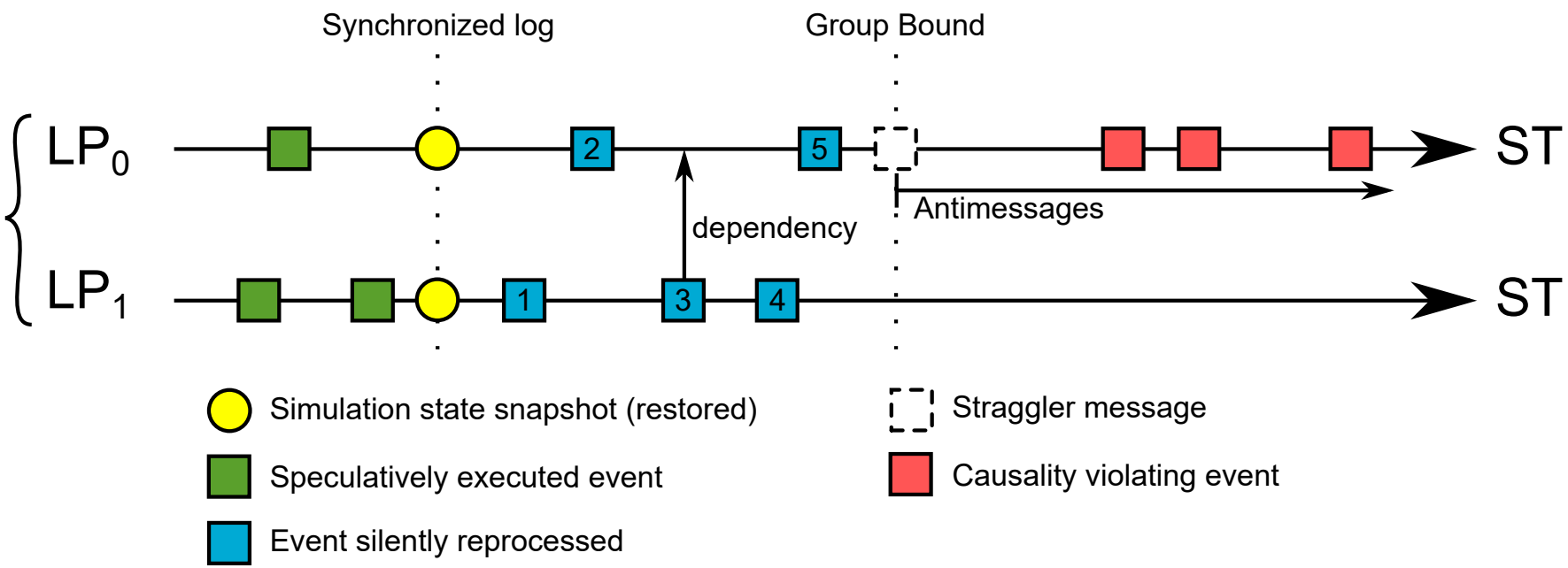


Group checkpoint

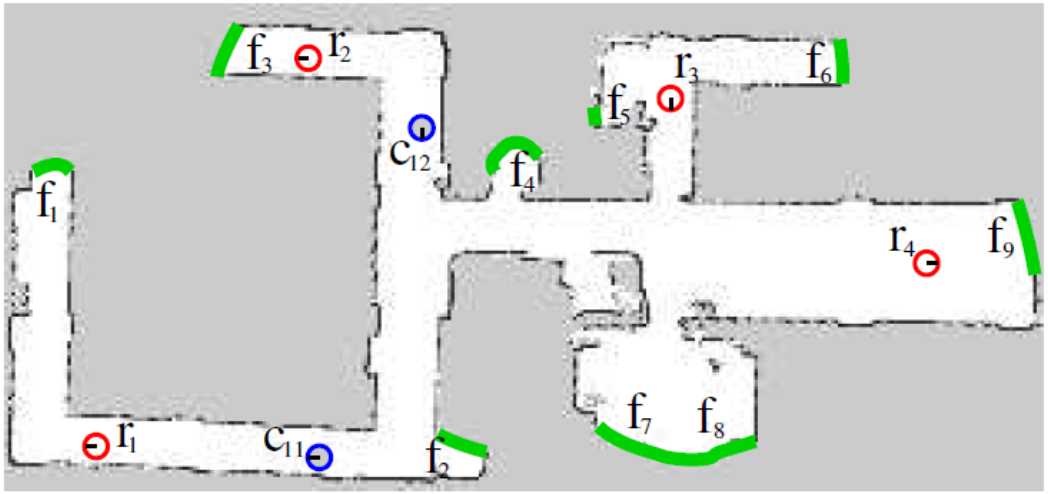
- A GLP is a unique object (again!)
- We use control messages to synchronize logging



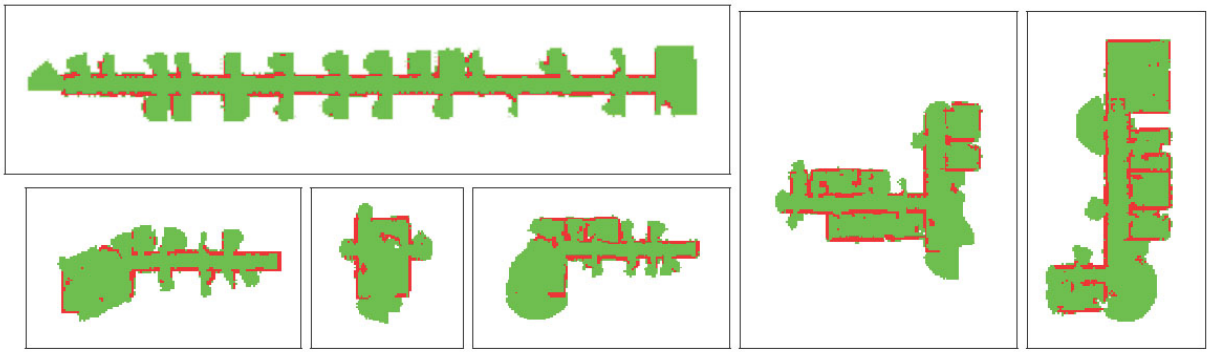
Overall Rollback Execution



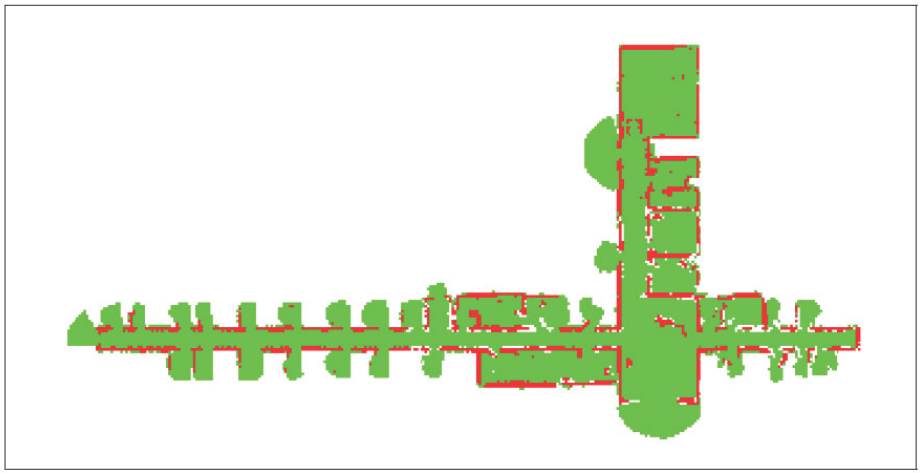
Preliminary Assessment: Distributed Multi-Robot Exploration and Mapping



Preliminary Assessment: Distributed Multi-Robot Exploration and Mapping



(a) input maps

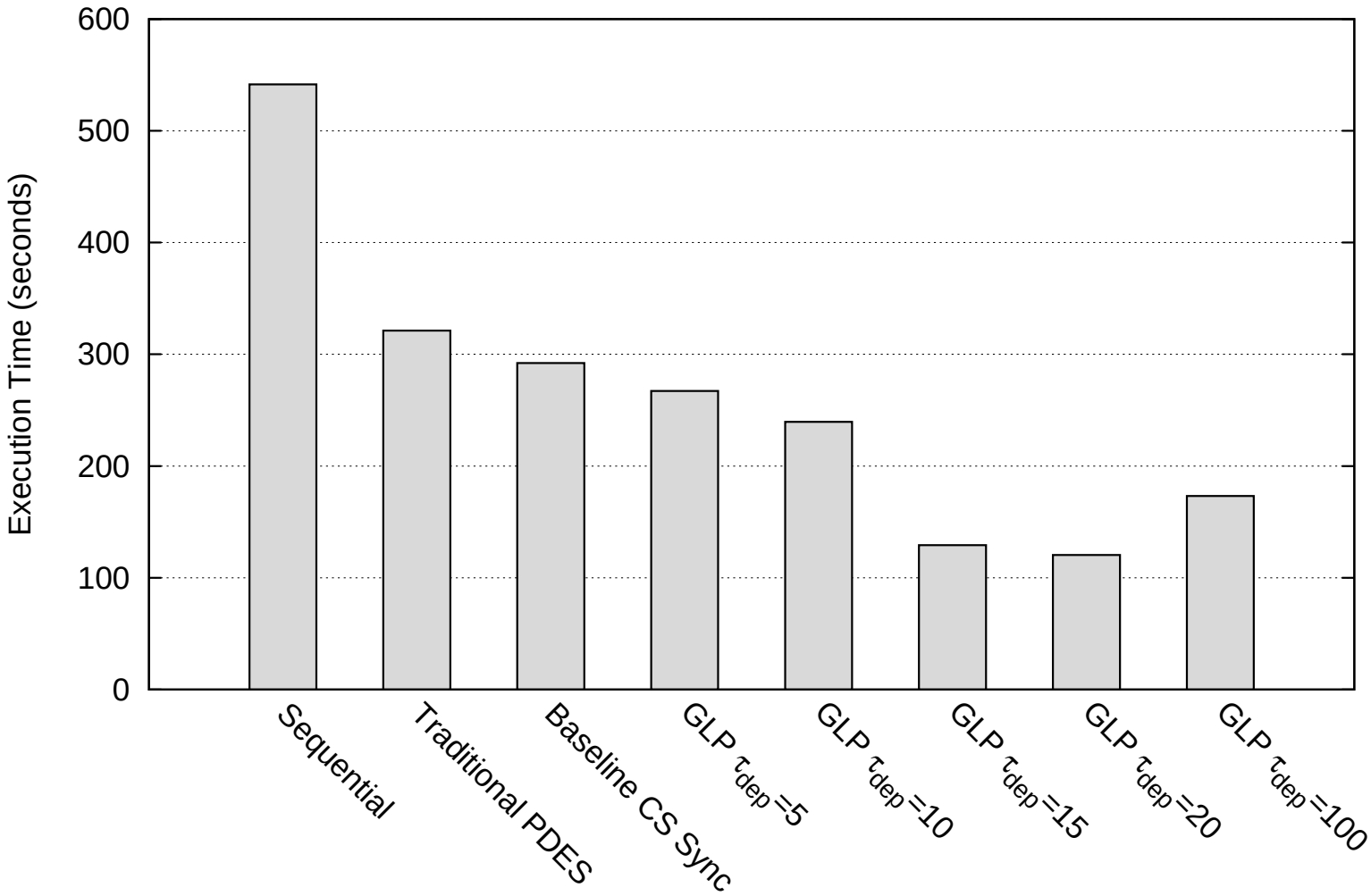


(b) merged map

Preliminary Assessment: Distributed Multi-Robot Exploration and Mapping

- The map is constructed online
- Robots explore independently, until they accidentally meet:
  1. they use their sensors to estimate their mutual physical position
  2. they create a rendez-vous point to verify the estimation's goodness
  3. if the hypothesis is verified, they exchange the so-far acquired data
  4. they form a cluster
- Clusters allow to explore collaboratively:
  - jointly define the next targets (reduce mapping time)
  - make a guess on the position of other robots (enlarge the cluster)

Results


Execution Time (seconds)
600
500
400
300
200
100
0
Sequential
Traditional PDES
Baseline CS Sync
GLP τdep =5
GLP τdep =10
GLP τdep =15
GLP τdep =20
GLP τdep =100

Thanks for your attention

Questions?

pellegrini@dis.uniroma1.it
http://www.dis.uniroma1.it/∼pellegrini
http://www.dis.uniroma1.it/∼ROOT-Sim