# Time Warp Operating System

Parth Nain & Jay Lin

# Table of Contents


1. Virtual Time
    a. Implementation
    b. Time Scale Comparison, Leslie Lamport
2. Time Warp
    a. Global Control Mechanism
    b. Rollback definition
3. Time Warp Operating System (TWOS)
    a. Discrete Event Simulations
    b. Architecture
    c. Comparisons with Typical O.S.
    d. Rollback in TWOS
4. Summary


Papers discussed:

- "Distributed Simulation and the Time Warp Operating System" (David R. Jefferson et al)
- "Virtual Time" (David R. Jefferson)

# Virtual Time

- Virtual time: Proposed as a new paradigm for distributed computation, 1985
- 1D coordinate system that extends globally (across all processors)
- Values:

  $-\infty$, the virtual time value in every system before computation begins

  $\infty$, all operations have been completed, and termination is ready

  $(-\infty,\infty)$ = valid timestamps for events & messages

- Real time comes with various side effects for parallelized or distributed systems
- Advantage of virtual time is total ordering and handling of simultaneous events

# Virtual Time Implementation

- Numerous local virtual clocks, each hold own set of coordinates in the global system
- All primitive actions have both a coordinate in the time system, and a coordinate for which processor performed the action
  - Changing variables
  - Sending messages
- Sender stamps messages are stamped with:
  - Sender name
  - Virtual send time
  - Receiver name
  - Virtual receive time

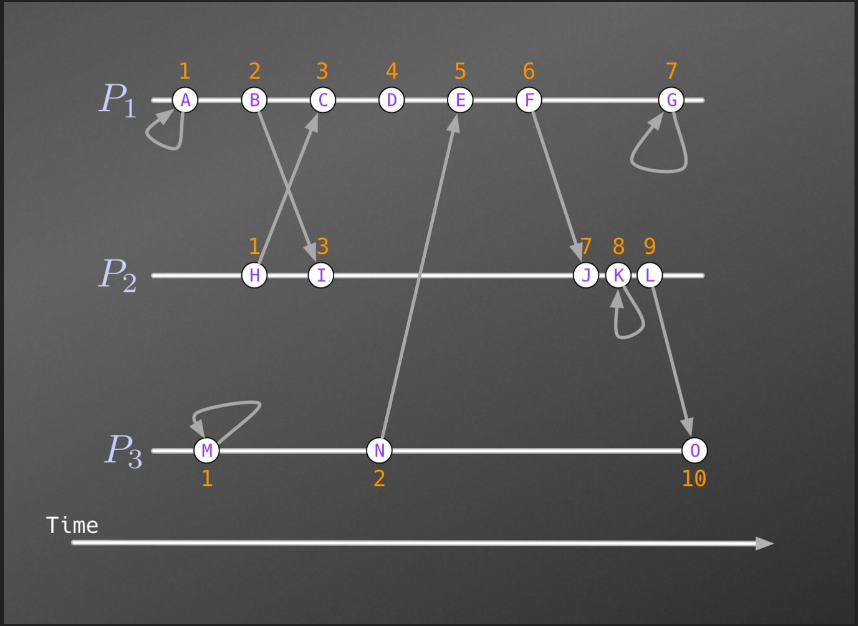
# Virtual Time Implementation cont.

- Lamport’s Clock Conditions
  - Virtual send time of each message < its virtual receive time
  - Virtual time of each event in a process < virtual time of the next event in the process
    - These conditions must be satisfied by all Virtual Time systems
- Expected Event Operations
  - Process may…
    - receive any # of messages stamped with their own location and timestamp, and read contents
    - read the virtual clock of the event
    - update its state variables
    - send any # of messages to other processes

# Time Scale Comparison

Leslie Lamport

- Showed that simple relationships (before/after) only form partial order, and distinct events can become incomparable if concurrent in declaration
- Algorithm involves labeling each event from a totally ordered set
- Virtual time reverses this algorithm: assumes that all events will be labeled, and then leaves room to resolve problems by rolling back later



# Time Warp Mechanisms

## Local Control Mechanism

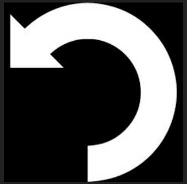
- Abstractly, global time is expected to slowly progress forward as a simulation completes (a singular concept)
- In implementation, numerous local clocks with differing values are interacting, still expected to move forward together
- Each process can receive messages and order them by receive time, and cycle through receiving-executing-sending. This will proceed unless a misordered message appears.

# Global Control Mechanism


Local clocks alone leave many problems unresolved...

- Is progress guaranteed with rollbacks happening?
  - **Yes**. Each local process regularly estimates GVT (Global Virtual Time), the lowest bound for all virtual times present in any sent message.
  - This is a commitment horizon that cannot be rolled back.
- How can all processes detect termination?
  - Termination is $+\infty$, when all timestamps reach $+\infty$, GVT will hit that value as well and termination will be signalled to all processes.
- Handling errors and output
  - Run-time errors mark a given state with an error flag, attempt to roll-back. Any errors timestamped before GVT will be presented after the simulation.
  - Output (ex: printing to a screen) is only performed when GVT passes the receive time of a message; cannot be rolled back.

# Rollback



Undo any consequence of an incorrect action, such as infinite loops and errors.
Rollback of a message = “un-sending”.

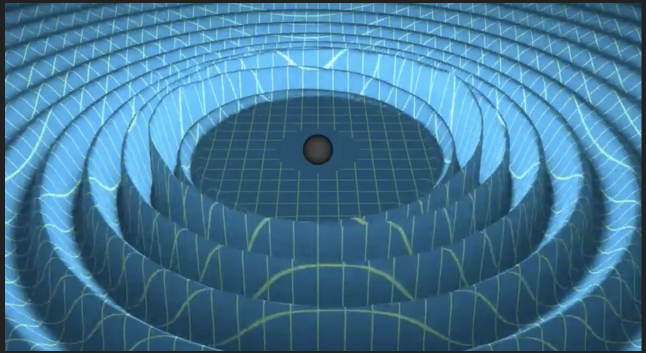
- Anti-message: message w/ negative (-) sign attached to it.
- Annihilation: anti-message is sent, and cancels out with the positive pair during a sender rollback.

(cancels out, msg doesn’t exist anymore)

# Rollback cont.

Message-antimessage pair per message process P → Q.

- (+) goes to receiving process Q's input queue
- (-) goes to sending process P's output queue
  - Eventually garbage-collected if P doesn't rollback and timestamp < GVT (commitment time)
- If P rollback,
  - P's Local Virtual Time will point towards an earlier section of the input/output queues.
  - Any formerly emanated messages cancel/annihilate with the anti-message. May cause a chain of rollbacks



"Ripple effect" possibly caused by rollbacks

# Time Warp OS

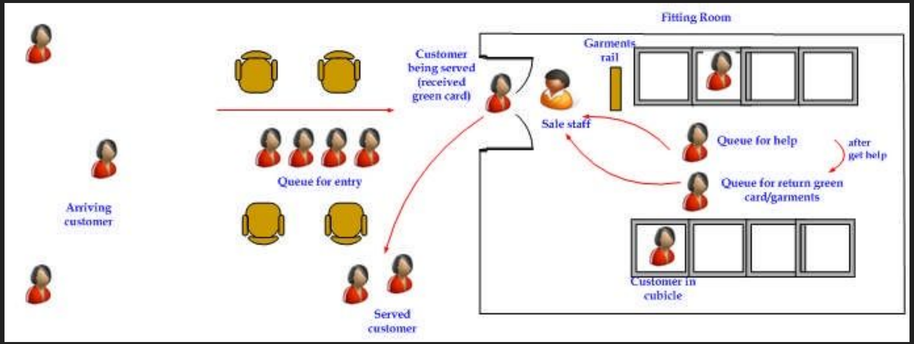
Developed in the 1980s in hopes of being able to concurrently run discrete-event simulations at an optimal speed.

- Different from typical OS; performs process synchronization using a general distributed process rollback mechanism.
- Written with C and assembly.

# Discrete Event Simulations (DES)

Simulates the behavior of a real-life system by modeling the system as a sequence of events, observing its state changes in time.

- Example real-life system: healthcare services, retail, military projects…
- Computationally $$$, can take days to complete w/out parallelism.

# DES: How to speed them up?

1. Run different parts of the simulation in parallel, using a time-stepped approach.
    a. Easy for regular systems that contain objects that change state at regular time intervals, like logical circuits with a clock.
    b. Very difficult to emulate systems that are not driven by regular time intervals, or irregular.
2. Event-triggered approach
    a. Should be used over the time-stepped method in the case of irregular systems as mentioned above, because time intervals are irregular.
    b. TWOS comes into play as a solution

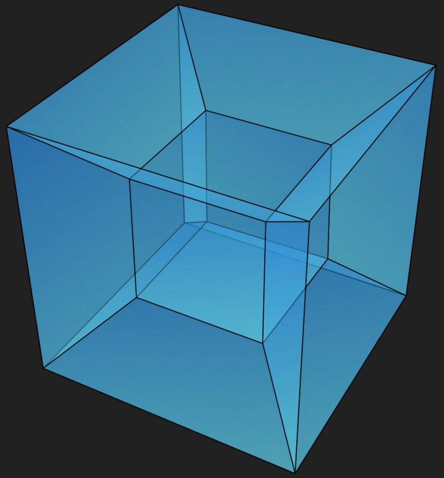
# Time Warp OS: background

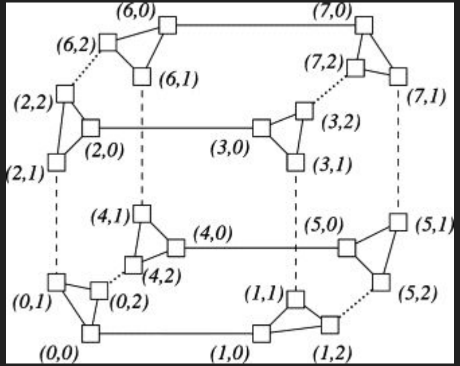
Initially designed to be run on a hypercube, an arrangement of a network of processors running in parallel.

- TWOS can utilize any # of processors.

Why a standalone OS and not just on top of another OS?

- Rollback requires a whole re-designing of operating system scheduling, synchronization, flow control, memory management… etc.






Example diagram of a hypercube topology, each square is a node, or processor connected to a fixed # of neighbours

# Time Warp OS: Architecture

Top Layer = "executive layer"

- C code implementing Time Warp
- Portable (code not dependent on and can work on different platforms)

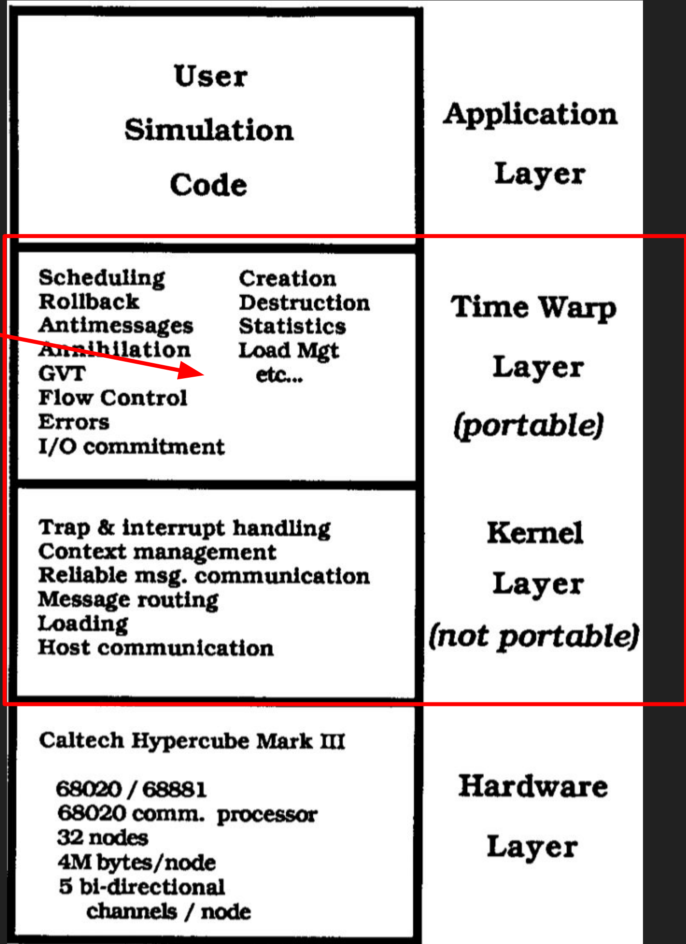



Figure 5: Structure of TWOS

# Differences b/t Time Warp and typical OS

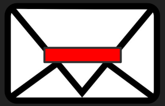
| Typical OS | TWOS |
| --- | --- |
| 1. FIFO message queueing, preserves message sending order<br>2. Blocks sender in the case of preventing overflow of receiver's memory; easier storage management<br>3. Channels for processes to communicate<br>4. Deletes message as soon as receiver reads it to save memory | 1. Virtual timestamp order message queuing<br>2. Difficult storage management; usually runs with memory almost full<br>3. No channels; any process send to any process<br>4. Can't delete message before commitment time (GVT); rollback might require it to be read again |

# Processes in TWOS

Each process has an input queue, output queue, and event history (snapshot queue).

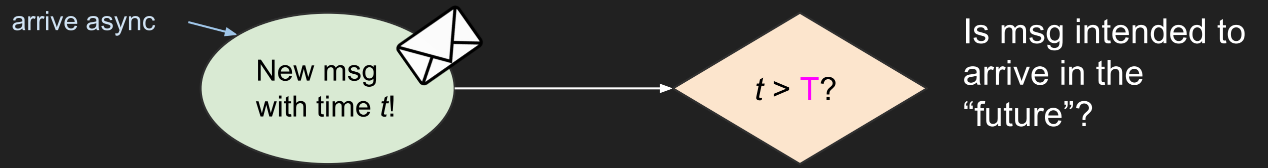
If a process Q receives a message from P whose timestamp *t* < its LVT (occurs in its past):

1. Q rollback its own event execution to that timestamp (look at snapshot queue)
2. Q also rollback its own input queue to that point
3. Send anti-messages to cancel (annihilate) all messages whose timestamp > *t*



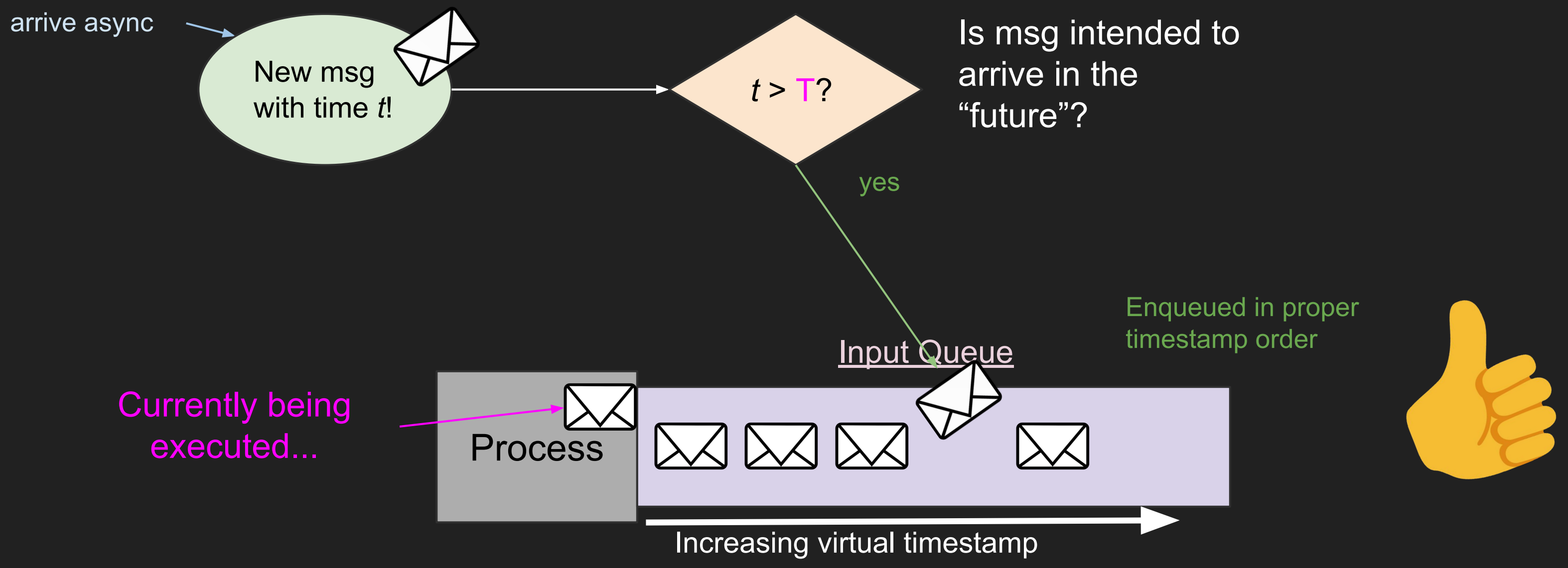
# Rollback in TWOS

Optimistic synchronization: allows for temporary causal violations, assume that the messages in the input queue are the always true "next" messages to execute.

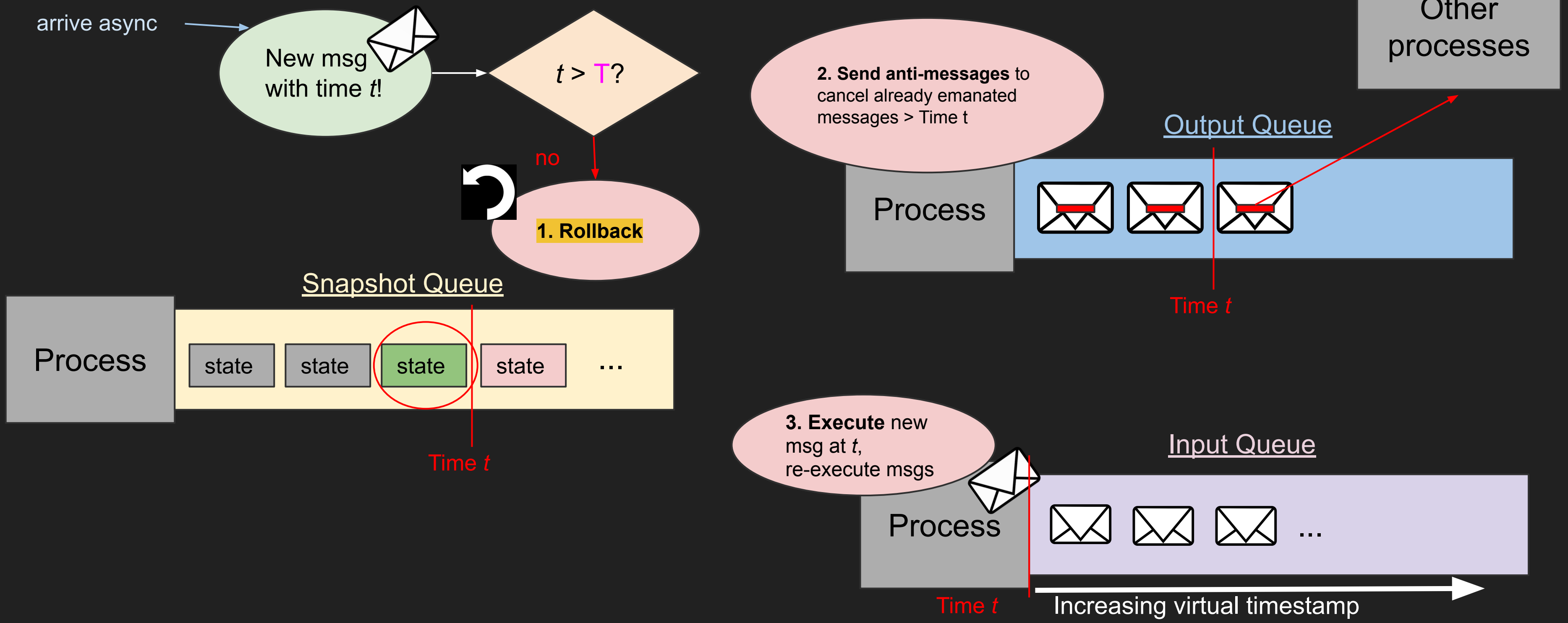



T = highest timestamp message processed so far

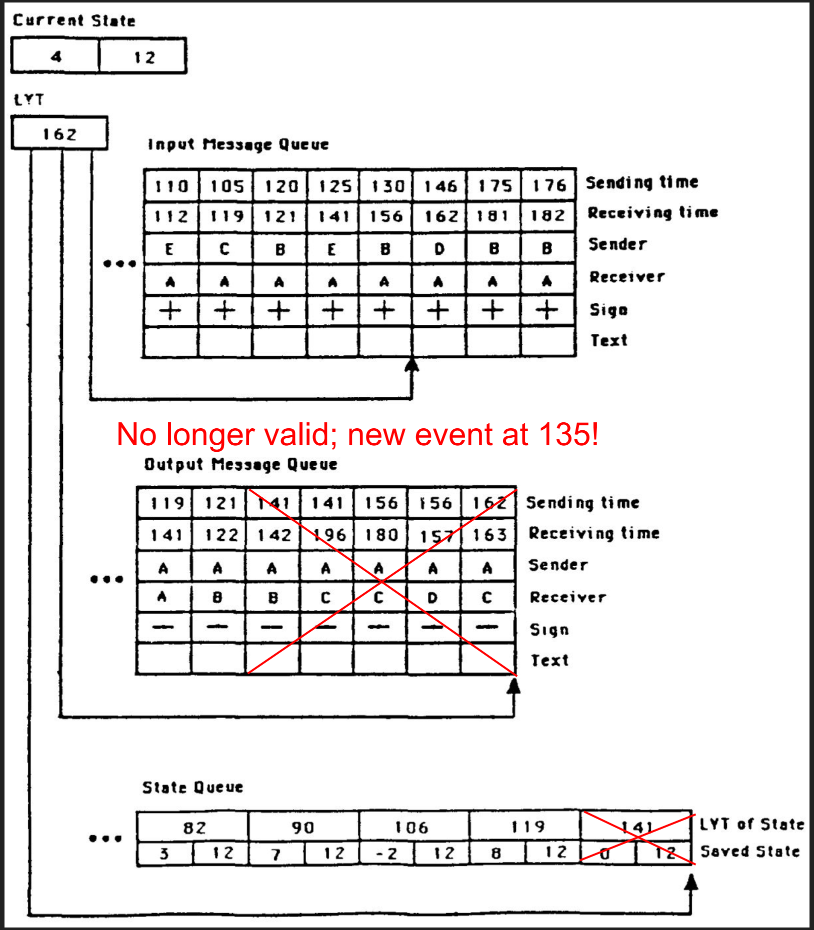
# Rollback in TWOS

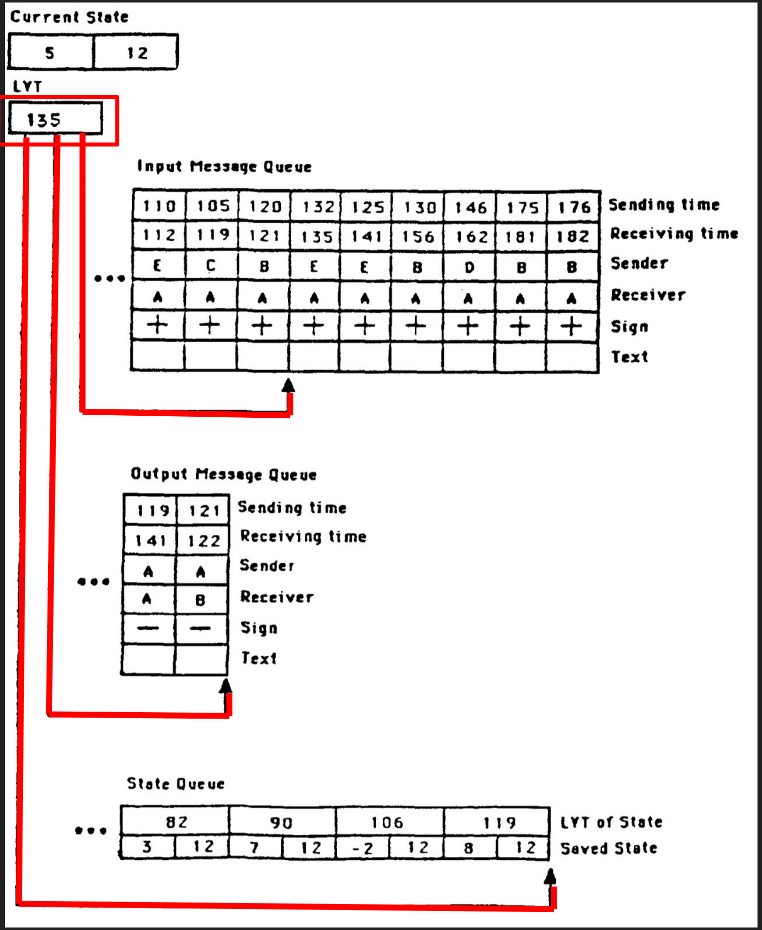


# Rollback in TWOS cont.

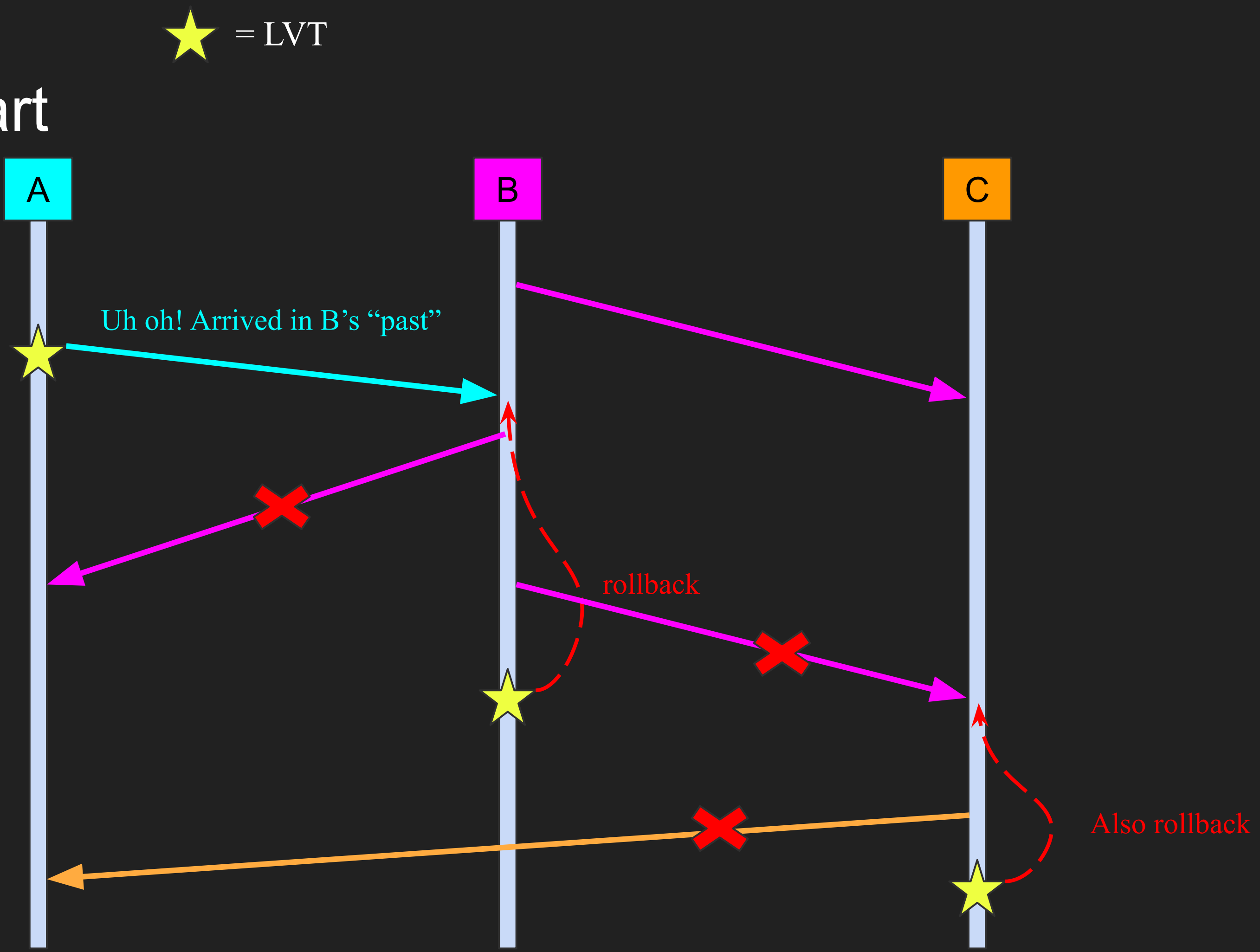


# More detailed look at process queues and rollback

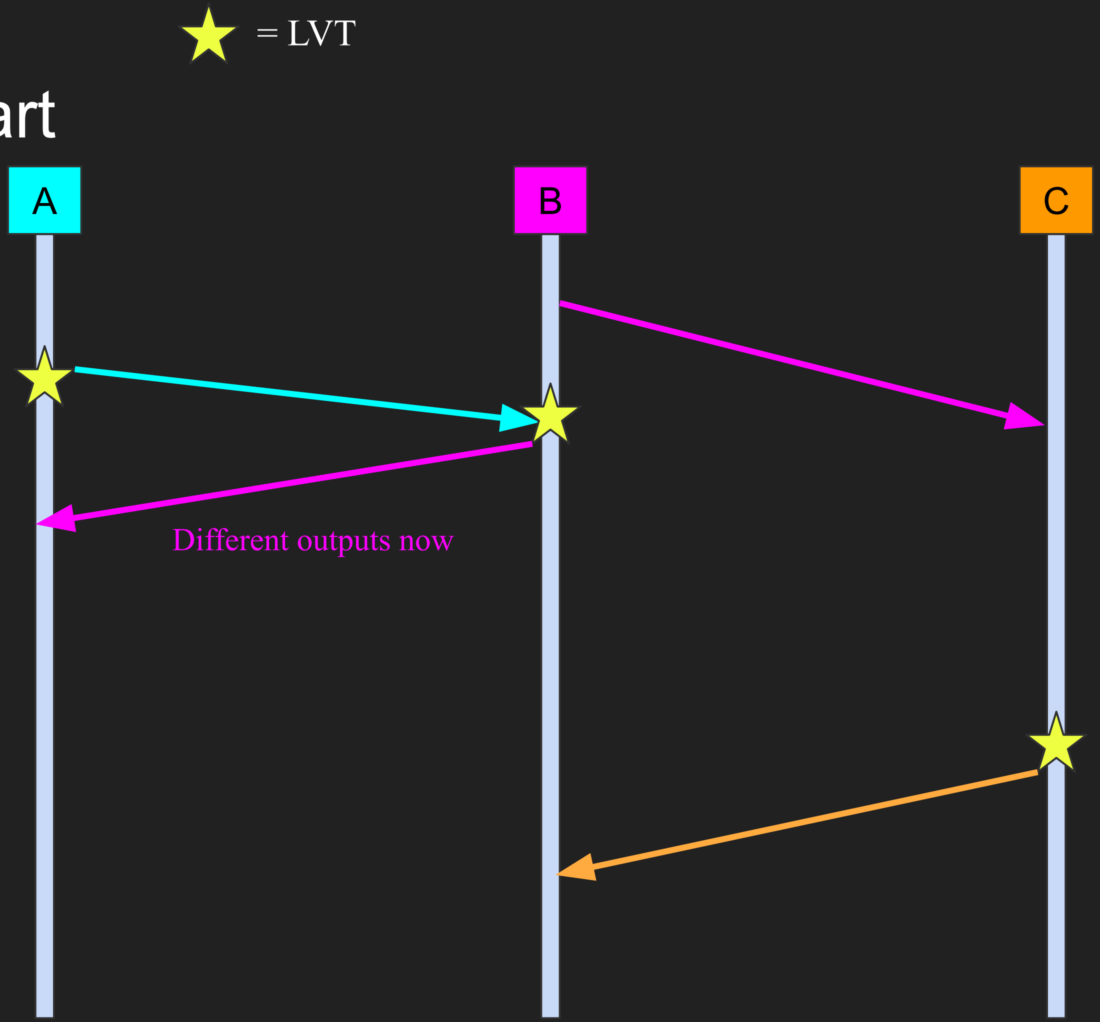


# Timing Chart



# Timing Chart



# TWOS impact and summary

Significant contribution to the speedup of running simulations in distributed systems.

- Speedup of simulations actually required more rollbacks contrary to initial beliefs.

Drawbacks of TWOS: high memory usage, huge overhead for rollbacks.

- Studies only done w/ limited # of processors (what happens on a larger scale?).
- Performance analysis experiments were only done on the Mark III computer (hypercube) in 1987; doesn't show us Time Warp's performance in other environments.

Personal remarks:

- TWOS should be tested on environments where memory is scarce/ on larger models (w/ more processors).
- Overall, the *Virtual Time* paper was much easier to digest because of its use of diagrams.

# References


1. David Jefferson, Brian Beckman, Fred Wieland, Leo Blume, Mike DiLoret, Phil Hontalas, Pierre Laroche, Kathy Sturdevant, Jack Tupman, Van Warren, John Wedel, Herb Younger, and Steve Bellonot, "Distributed Simulation and the Time Warp Operating System" Technical Report, UCLA, Agust, 1987.
2. Jefferson, D.R., "Virtual Time", ACM Transactions on Programming Languages and Systems, Vol.7 No.3, 1985, pages 404-425.
3. Stewart Robinson. 2004. *Simulation: The Practice of Model Development and Use.* John Wiley & Sons, Inc., Hoboken, NJ, USA.
4. https://www.britannica.com/technology/hypercube
5. https://www.youtube.com/watch?v=QpHNPk6oSDs&ab_channel=DhananjaiRao

# Q&A