4210 Simulation & Modeling

PDES Introduction

The Time Warp Mechanism

Maria Hybinette, UGA

- Optimistic Synchronization
- Time Warp
  - » Local Control Mechanism
    - – Rollback
    - – Event cancellation
  - » Global Control Mechanism
    - – Global Virtual Time
    - – Fossil Collection

Golden rule

The Synchronization Problem

Local causality constraint: Events within each logical process must be processed in time stamp order

Observation: Adherence to the local causality constraint is sufficient to ensure that the parallel simulation will produce exactly the same results as the corresponding sequential simulation*

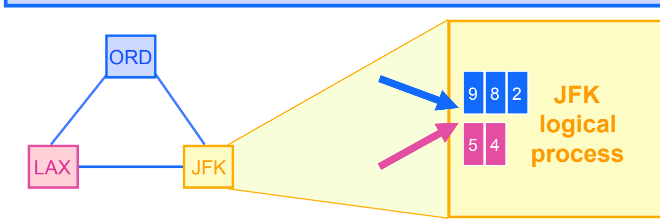
Synchronization Algorithms

- Conservative synchronization: avoid violating the local causality constraint (wait until it's safe)
  - » 1st generation: null messages (Chandy/Misra/Bryant)
  - » 2nd generation (later): time stamp of next event
- Optimistic synchronization: allow violations of local causality to occur, but detect them at runtime and recover using a rollback mechanism
  - » Time Warp (Jefferson & Sowizral)

** provided events with the same time stamp are processed in the same order as in the sequential execution*

Time Warp Algorithm

- **Assumptions:**
  - » logical processes (LPs) exchanging time stamped events (messages)
  - » dynamic network topology, dynamic creation of LPs
  - » messages sent on each link need not be sent in time stamp order
  - » network provides reliable delivery, but need not preserve order when received
- **Basic idea:**
  - » process events w/o worrying about messages that will arrive later
  - » detect out of order execution, recover using rollback



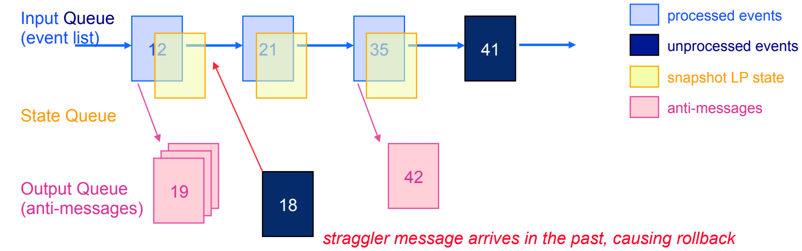
process all available events (2, 4, 5, 8, 9) in time stamp order

Time Warp Algorithms

- **Many have been proposed, will cover fundamental concepts:**
  - » **rollback, anti-messages, Global Virtual Time (GVT).**
  - » **Initially assume 'non-zero' look-ahead**
- **Time Warp Structure:**
  - » **local control mechanism: implemented within each processor, mostly independent of other processors**
  - » **global control mechanism: used to reclaim memory and used to commit operations such as I/O that cannot be rolled back: requires a distributed computation involving all processors in the system.**

Time Warp: Local Control Mechanism

Each LP: process events in time stamp order, like a sequential simulator, except:
(1) do NOT discard processed events (backs up a history) and
(2) add a rollback mechanism

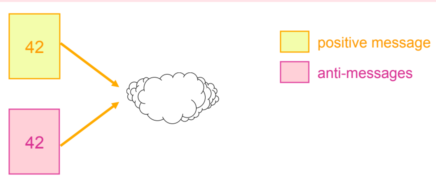


Problem: Need to account for messages received in the LP's past.
Approach: Rollback and then re-compute
Sub Problem : Rollback changes to state variables performed by events
Solution: checkpoint state or use incremental state saving (state queue)

Sub Problem: Rollback previously sent messages
Solution: Anti-messages and message annihilations (output queue)

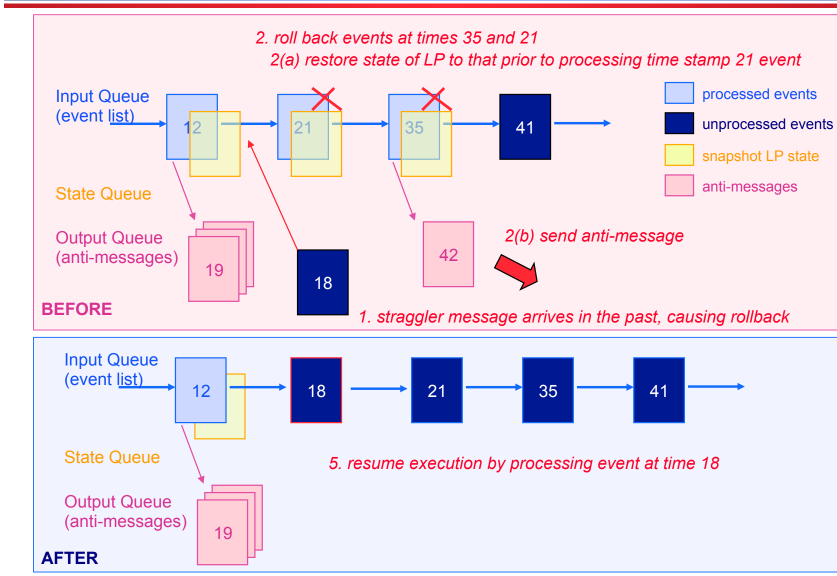
Anti-Messages

Undo message sends by 'unsending' a previously sent message



- Each **positive (regular) message** sent by an LP has a corresponding **anti-message**
  - » An **anti-message** is an identical (copy) to its positive message, except for a sign bit.
- ***Rule of cancellation***: When an anti-message and its matching positive message meet in the same queue, the two annihilate each other (analogous to matter and anti-matter).
- **Mechanism:**
  - » To undo the effects of a previously sent (positive) message, the LP need only send the corresponding anti-message
  - » Message send: in addition to sending the message, leave a copy of the corresponding anti-message in a data structure in the sending LP called the output queue.

Rollback: Receiving a Straggler Message

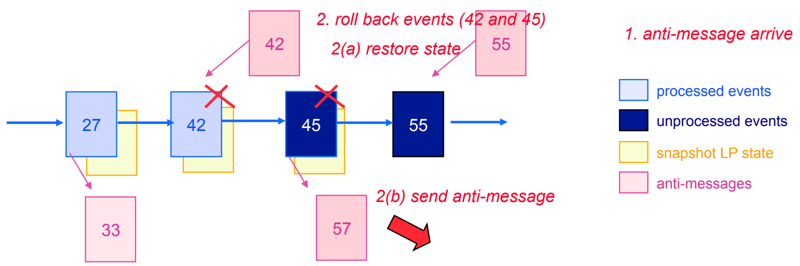


Processing Incoming Anti-Messages

Case I: Corresponding message has not yet been processed
- » annihilate message/anti-message pair

Case II: Corresponding message has already been processed
- » rollback to time prior to processing message (secondary rollback)
- » annihilate message/anti-message pair

Case III: Corresponding message has not yet been received
- » queue anti-message
- » annihilate message/anti-message pair when message is received

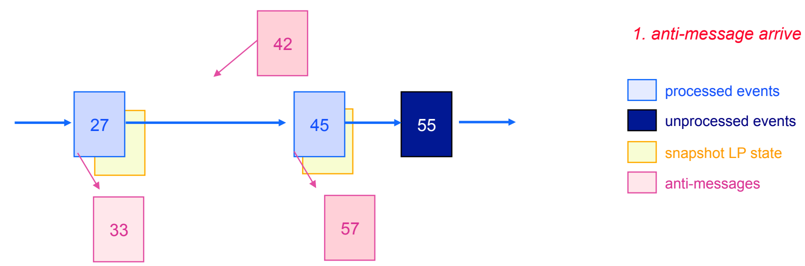


Processing Incoming Anti-Messages

Case I: Corresponding message has not yet been processed
- » annihilate message/anti-message pair

Case II: Corresponding message has already been processed
- » rollback to time prior to processing message (secondary rollback)
- » annihilate message/anti-message pair

Case III: Corresponding message has not yet been received
- » queue anti-message
- » annihilate message/anti-message pair when message is received



LP Simulation Example

`Now:` current simulation time

`InTheAir:` number of aircraft landing or waiting to land

`OnTheGround:` number of landed aircraft

`RunwayFree`: Boolean, true if runway available

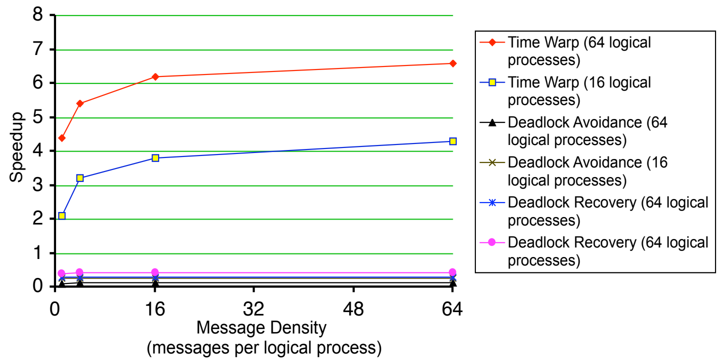
```
Arrival Event:
    InTheAir := InTheAir+1;
    if( RunwayFree )
      RunwayFree:=FALSE;
      Schedule Landed event(local) @ Now + R;

Landed Event:
    InTheAir := InTheAir-1;
    OnTheGround := OnTheGround + 1;
    Schedule Departure event(local) @ Now + G;
    if( InTheAir > 0 ) Schedule Landed event(local) @ Now + R;
    else RunwayFree := True;
Departure Event: (D = Delay to reach another airport)
    OnTheGround := OnTheGround - 1;
    Schedule Arrival Event (remote) @ (Now+D) @ another airport
```


Global Virtual Time and Fossil Collection

- A mechanism is needed to:
  - » reclaim memory resources (e.g., old state and events)
  - » perform irrevocable operations (e.g., I/O)
- Observation: A lower bound on the time stamp of any rollback that can occur in the future is needed.

- Global Virtual Time (GVT) is defined as the minimum time stamp of any unprocessed (or partially processed) message or anti-message in the system. GVT provides a lower bound on the time stamp of any future rollback.
  - » storage for events and state vectors older than GVT (except one state vector) can be reclaimed
  - » I/O operations with time stamp at GVT can be performed.
- Observation: The computation corresponding to GVT will not be rolled back, guaranteeing forward progress.

Time Warp and Chandy/Misra Performance



- eight processors
- closed queuing network, hypercube topology
- high priority jobs preempt service from low priority jobs (1% high priority)
- exponential service time (poor lookahead)

Summary

- **Optimistic synchronization: detect and recover from synchronization errors rather than prevent them**
- **Time Warp**
  - » **Local control mechanism**
  - » **Rollback**
  - » **State saving**
  - » **Anti-messages**
  - » **Cascaded rollbacks**
- **Global control mechanism**
  - » **Global Virtual Time (GVT)**
  - » **Fossil collection to reclaim memory**
  - » **Commit irrevocable operations (e.g., I/O)**