Lookahead

Richard M. Fujimoto
Professor

Computational Science and Engineering Division
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0765, USA

http://www.cc.gatech.edu/~fujimoto/

Outline

- Null message algorithm: The Time Creep Problem
- Lookahead
  - What is it and why is it important?
  - Writing simulations to maximize lookahead
- Changing lookahead
- Avoiding Time Creep
- Repeatability and Simultaneous Events

Deadlock Avoidance Using Null Messages

Null Message Algorithm (executed by each LP):

Goal: Ensure events are processed in time stamp order and avoid deadlock

WHILE (simulation is not over)

wait until each FIFO contains at least one message

remove smallest time stamped event from its FIFO

process that event

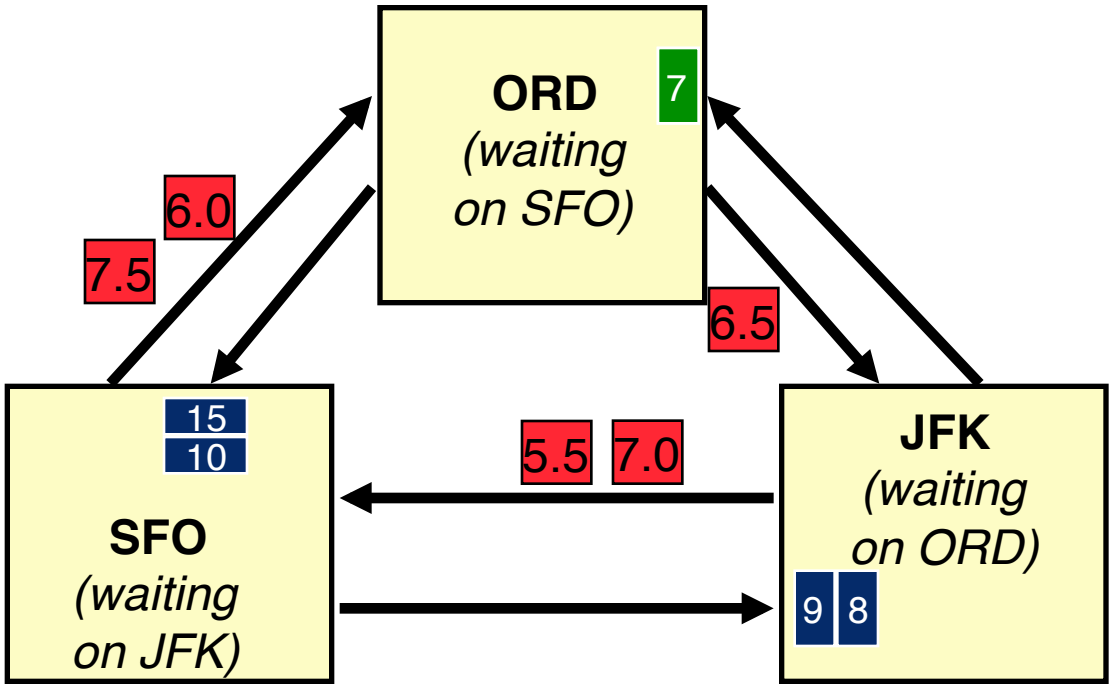
send null messages to neighboring LPs with time stamp indicating a lower bound on future messages sent to that LP (current time plus minimum transit time between airports)

END-LOOP

Variation: LP requests null message when FIFO becomes empty

- Fewer null messages
- Delay to get time stamp information

The Time Creep Problem



Null messages:

JFK: timestamp = 5.5

SFO: timestamp = 6.0

ORD: timestamp = 6.5

JFK: timestamp = 7.0

SFO: timestamp = 7.5

ORD: process time stamp 7 message

Five null messages to process a single event!

0.5

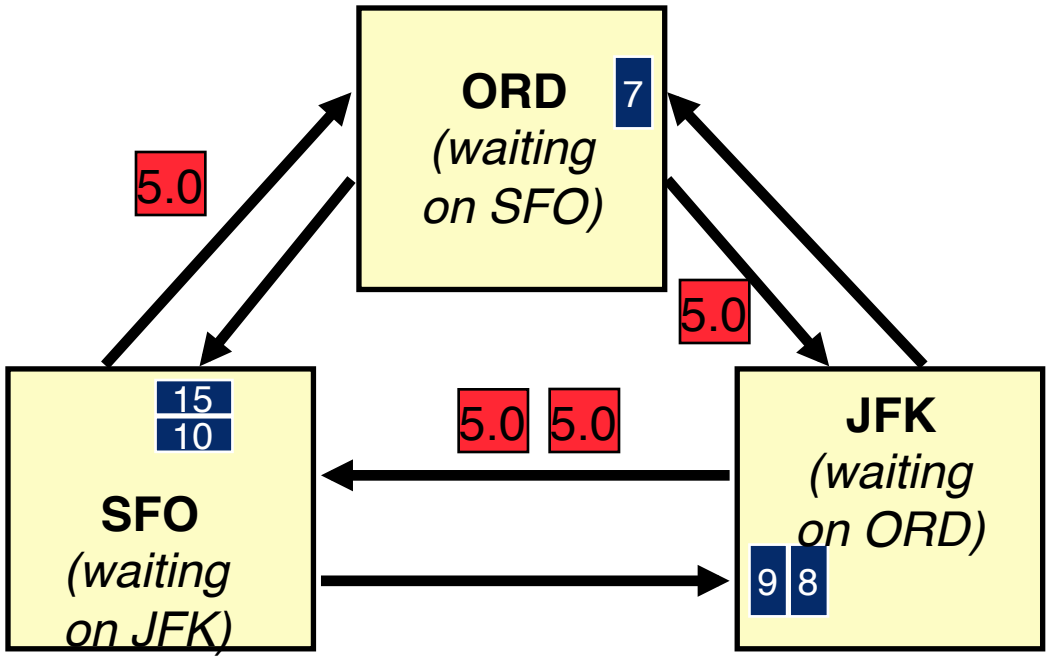
Assume minimum delay between airports is ~~3~~ units of time

JFK initially at time 5

Many null messages if minimum flight time is small!

Livelock Can Occur!

Suppose the minimum delay between airports is zero!



Livelock: un-ending cycle of null messages where no LP can advance its simulation time

There cannot be a cycle where for each LP in the cycle, an incoming message with time stamp T results in a new message sent to the next LP in the cycle with time stamp T (zero lookahead cycle)

Outline

- Null message algorithm: The Time Creep Problem
- Lookahead
  - What is it and why is it important?
  - Writing simulations to maximize lookahead
- Changing lookahead
- Avoiding Time Creep
- Repeatability and Simultaneous Events

Lookahead

The null message algorithm relies on a "prediction" ability referred to as *lookahead*

- "ORD at simulation time 5, minimum transit time between airports is 3, so the next message sent by ORD must have a time stamp of *at least* 8"

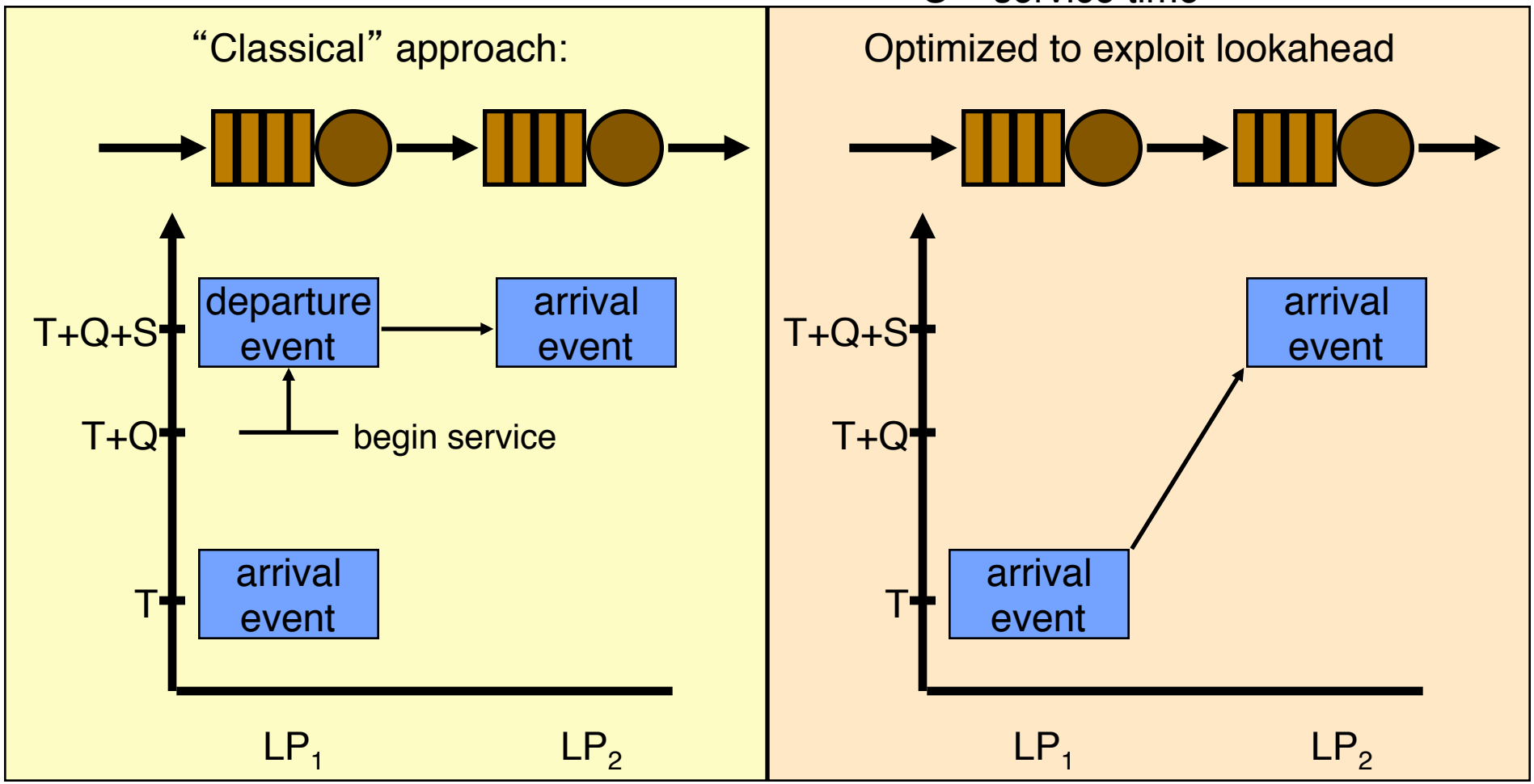
Lookahead is a constraint on LP's behavior

- **Link lookahead**: If an LP is at simulation time T, and an outgoing link has lookahead L_i, then any message sent on that link must have a time stamp of at least $T+L_i$
- **LP Lookahead**: If an LP is at simulation time T, and has a lookahead of L, then any message sent by that LP must have a time stamp of at least T+L
  - Equivalent to link lookahead where the lookahead on each outgoing link is the same

Exploiting Lookahead in Applications

Example: Tandem first-come-first-serve queues

T = arrival time of job
Q = waiting time in queue
S = service time

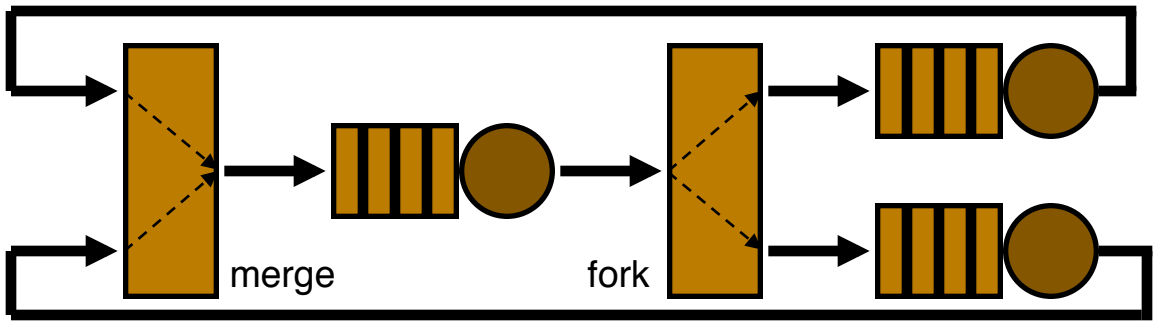


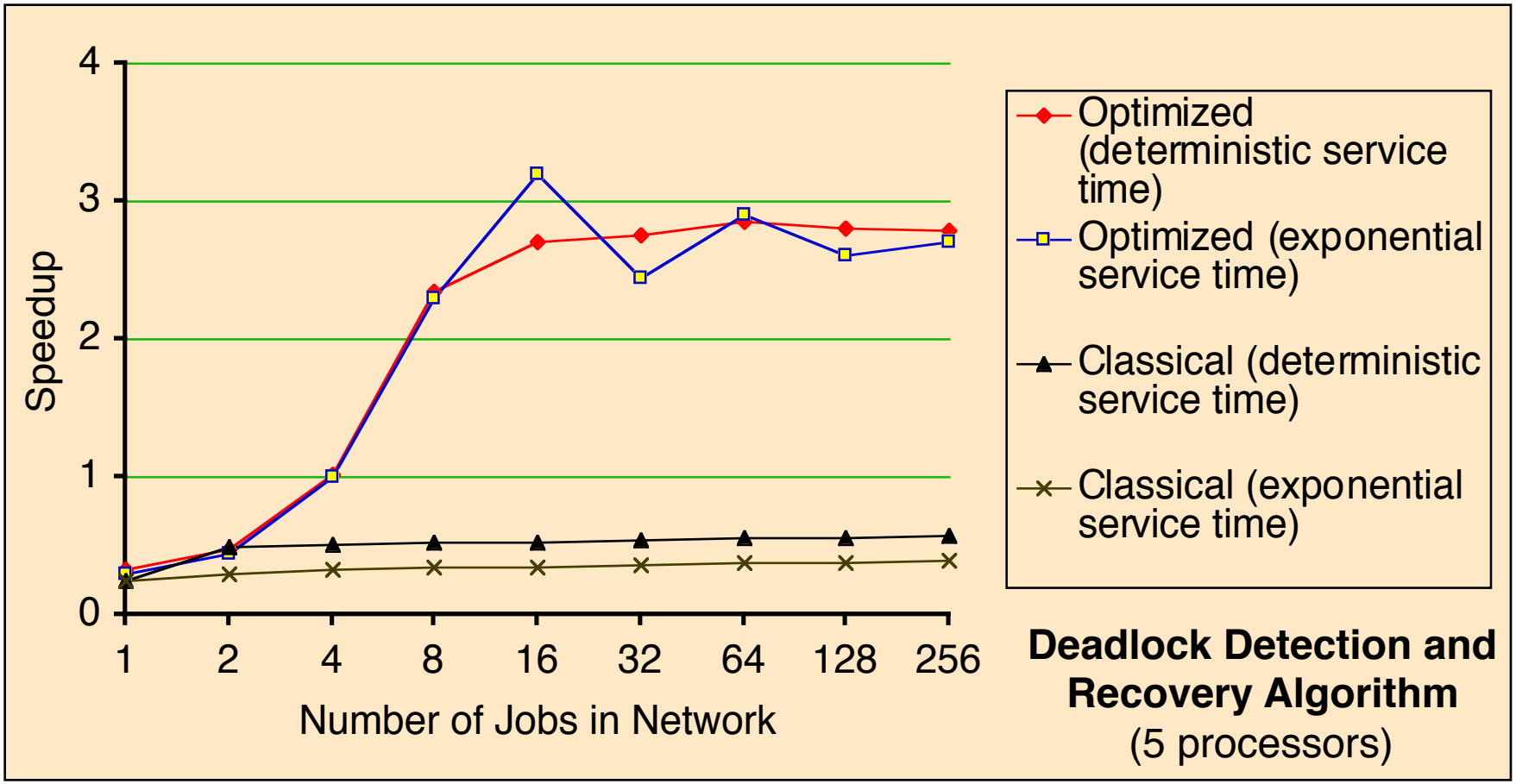
The degree to which the program can exploit lookahead is critical for good performance

Lookahead Affects Performance

Parallel Simulation of a Central Server Queueing Network

Speedup:
Sequential execution time / parallel execution time



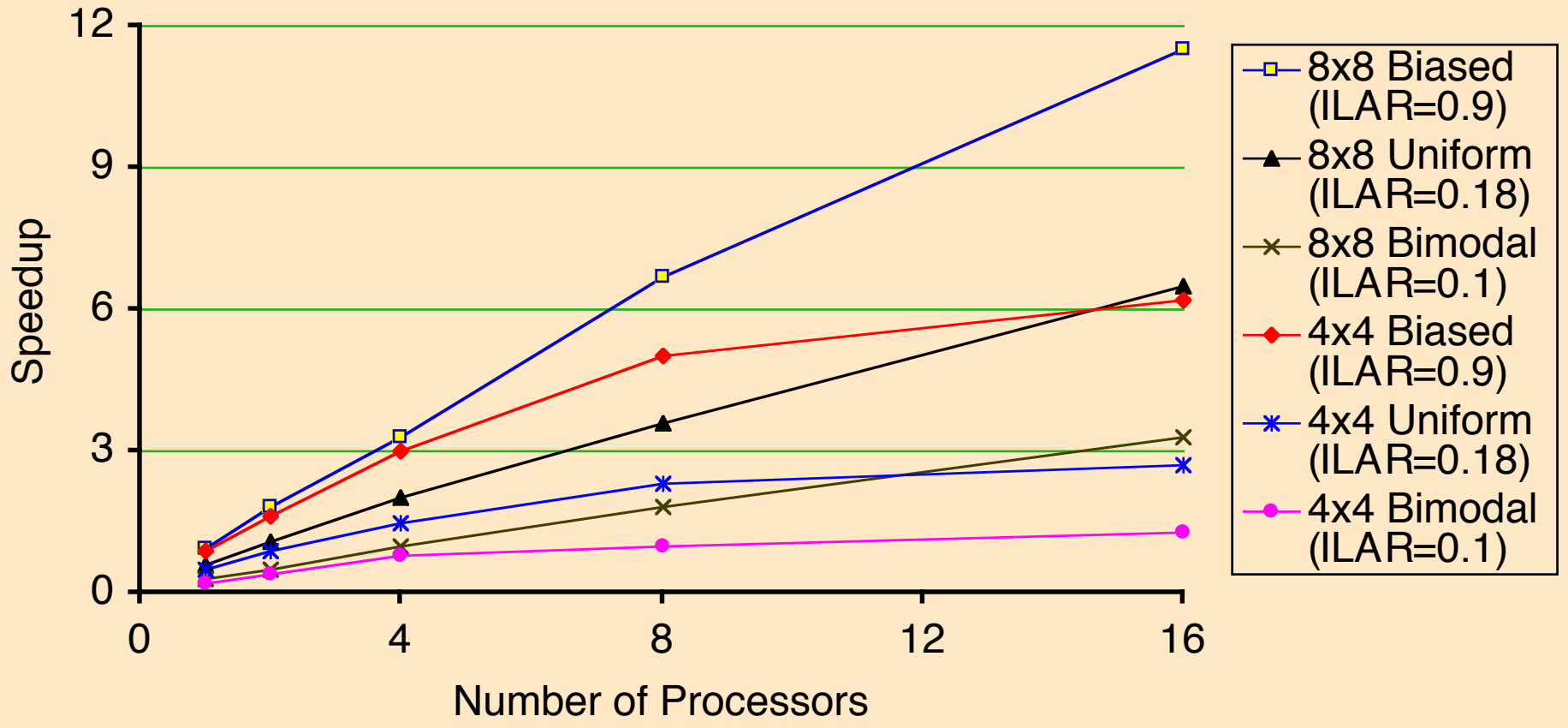


Deadlock Detection and Recovery Algorithm
(5 processors)

Null Message Algorithm: Speed Up

- toroid topology
- message density: 4 per LP
- 1 millisecond computation per event

- vary time stamp increment distribution
- ILAR=lookahead / average time stamp increment



Conservative algorithms live or die by their lookahead!

Lookahead and the Simulation Model

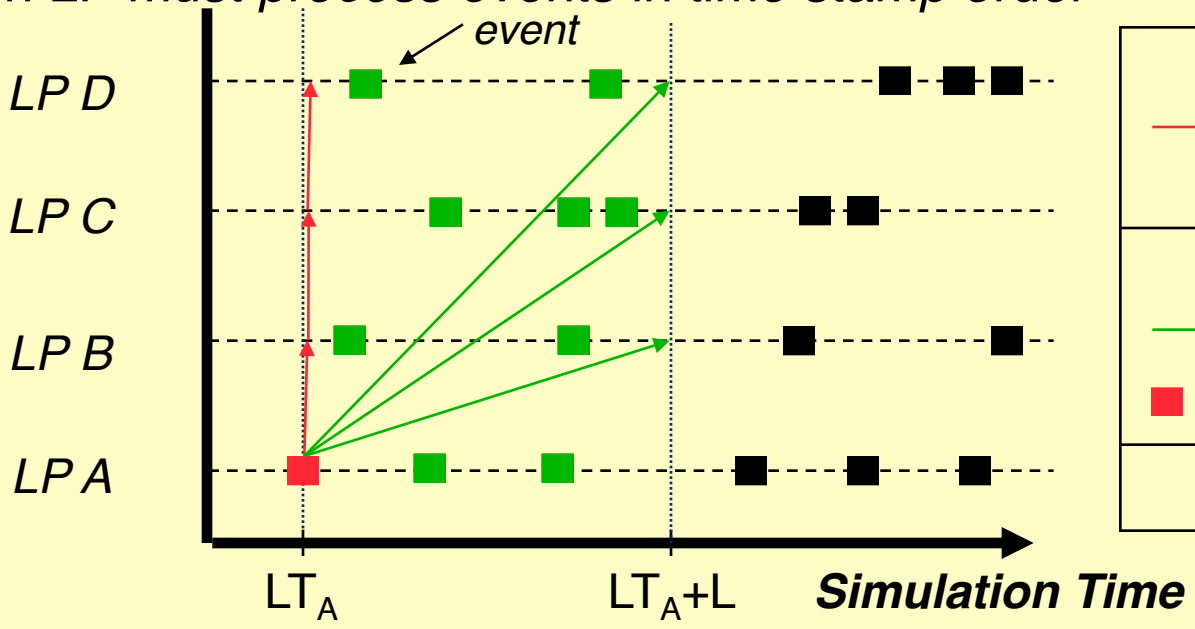
Lookahead is clearly dependent on the simulation model

- could be derived from physical constraints in the system being modeled, such as minimum simulation time for one entity to affect another (e.g., a weapon fired from a tank requires L units of time to reach another tank, or maximum speed of the tank places lower bound on how soon it can affect another entity)
- could be derived from characteristics of the simulation entities, such as *non-preemptable* behavior (e.g., a tank is traveling north at 30 mph, and nothing in the federation model can cause its behavior to change over the next 10 minutes, so all output from the tank simulator can be generated immediately up to time "local clock + 10 minutes")
- could be derived from tolerance to temporal inaccuracies (e.g., users cannot perceive temporal difference of 100 milliseconds, so messages may be timestamped 100 milliseconds into the future).
- simulations may be able to *precompute* when its next interaction with another simulation will be (e.g., if time until next interaction is stochastic, pre-sample random number generator to determine time of next interaction).

Observation: time-stepped simulations implicitly use lookahead; events in current time step are considered independent (and can be processed concurrently), new events are generated for the next time step, or later.

Why Lookahead is Important

problem: limited concurrency
each LP must process events in time stamp order



without lookahead
possible message
OK to process

with lookahead
possible message
OK to process

not OK to process yet

Each LP A using logical time declares a lookahead value L; the time stamp of any event generated by the LP must be ≥ LT$_A$+ L

- Lookahead is used in virtually all conservative synchronization protocols
- Essential to allow concurrent processing of events

Lookahead is necessary to allow concurrent processing of events with different time stamps (unless optimistic event processing is used)

Outline

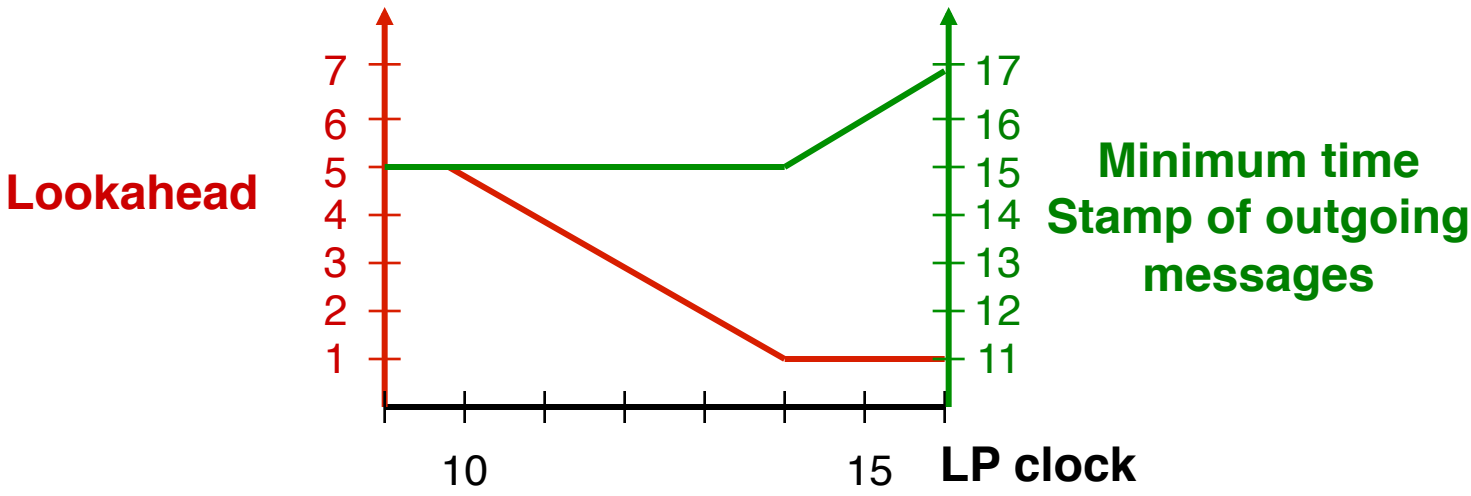
- Null message algorithm: The Time Creep Problem
- Lookahead
  - What is it and why is it important?
  - Writing simulations to maximize lookahead
- Changing lookahead
- Avoiding Time Creep
- Repeatability and Simultaneous Events

Changing Lookahead Values

- Increasing lookahead
  - No problem; lookahead can immediately be changed
- Decreasing lookahead
  - Previous time stamp guarantees must be honored
  - Lookahead cannot immediately be decreased
    - If an LP is at simulation time 10 and lookahead is 5, it has promised subsequent messages will have a time stamp of at least 15
    - If lookahead were *immediately* set to 1, it could generate a message with time stamp 11
  - Lookahead can decrease by k units of simulation time only after the LP has advanced k units of simulation time

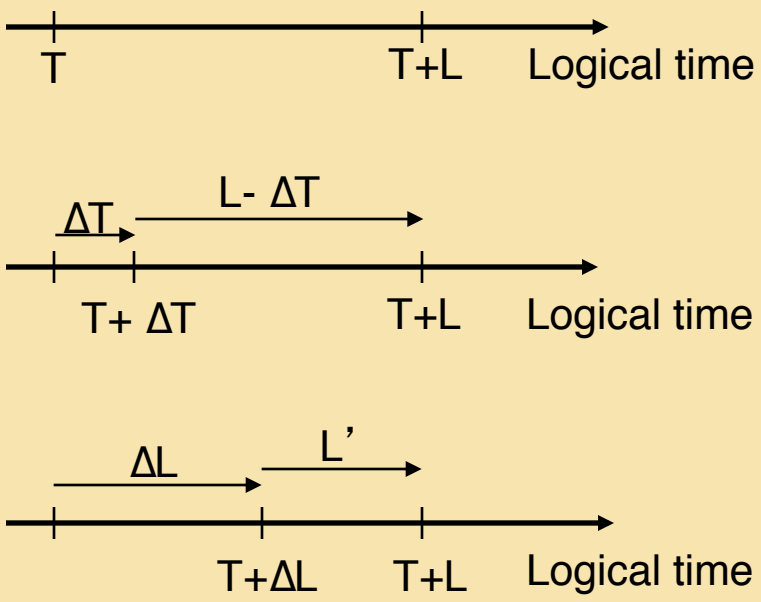
Example: Decreasing Lookahead

- SFO: simulation time = 10, lookahead = 5
- Future messages sent on link must have time stamp ≥ 15
- SFO: request its lookahead be reduced to 1



Lookahead: Example Programmer Interface

- Lookahead in the High Level Architecture standard
- Each LP (federate) must declare a non-negative lookahead value
- Any TSO sent by an LP must have time stamp at least the LP's current time plus its lookahead
- Lookahead can change during the execution (*Modify Lookahead*)
  - increases take effect immediately
  - decreased do not take effect until the federate advances its logical time

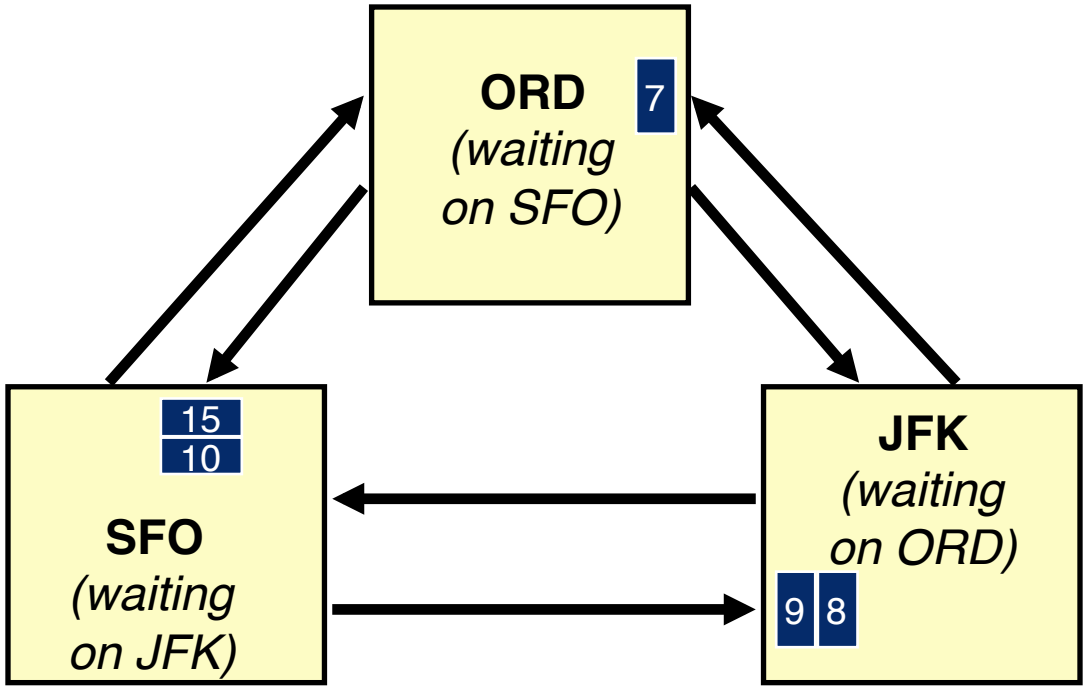


1. Current time is T, lookahead L
2. Request lookahead decrease by ΔL to L'

3. Advance ΔT, lookahead, decreases ΔT

4. After advancing ΔL, lookahead is L'

Preventing Time Creep: Next Event Time Information



Observation: smallest time stamped event is safe to process

- Lookahead creep avoided by allowing the synchronization algorithm to immediately advance to (global) time of the next event
- Synchronization algorithm must know time stamp of LP's next event
- Each LP guarantees a logical time T such that ***if no additional events are delivered to LP with TS < T, all subsequent messages that LP produces have a time stamp at least T+L (L = lookahead)***

Repeatability in Distributed Simulations

- Each logical process will process events (both local and those generated by other processors) in time stamp order
  - final result of entire distributed execution same as if all events were processed sequentially in time stamp order
- Issues
  - Repeatable event computations: must ensure the event computation is repeatable: e.g., not dependent on wallclock time
  - Simultaneous events (events with the same time stamp): must be processed in the same order with each execution
  - Interactive inputs:
    - Values must be identical, and input to the simulation at the same point in its execution on each run
    - Can assign each input a logical time value, and ensure same value is used for each execution

Simultaneous Events

- The order that simultaneous events are processed will affect the results computed by the simulation, possibly significantly
  - Simultaneous aircraft arrival events: Which airline's flight is delayed?
  - Simultaneous "detonation" events at a target: which weapon system gets credit for the kill?
  - Systematically favoring one outcome may bias results
- The modeler must be able to control the ordering of simultaneous events (not the RTI or simulation executive)!
  - Proper ordering requires domain knowledge of the simulation application

Summary

- Null message algorithm
  - Lookahead creep problem
  - No zero lookahead cycles allowed
- Lookahead
  - Constraint on time stamps of subsequent messages
  - Has large effect on performance: essential for concurrent processing of events for conservative algorithms
  - Programs must be coded to exploit lookahead
- Use time of next event to avoid lookahead creep