Synchronous Algorithms II

Transient Messages and Distance Between LPs

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

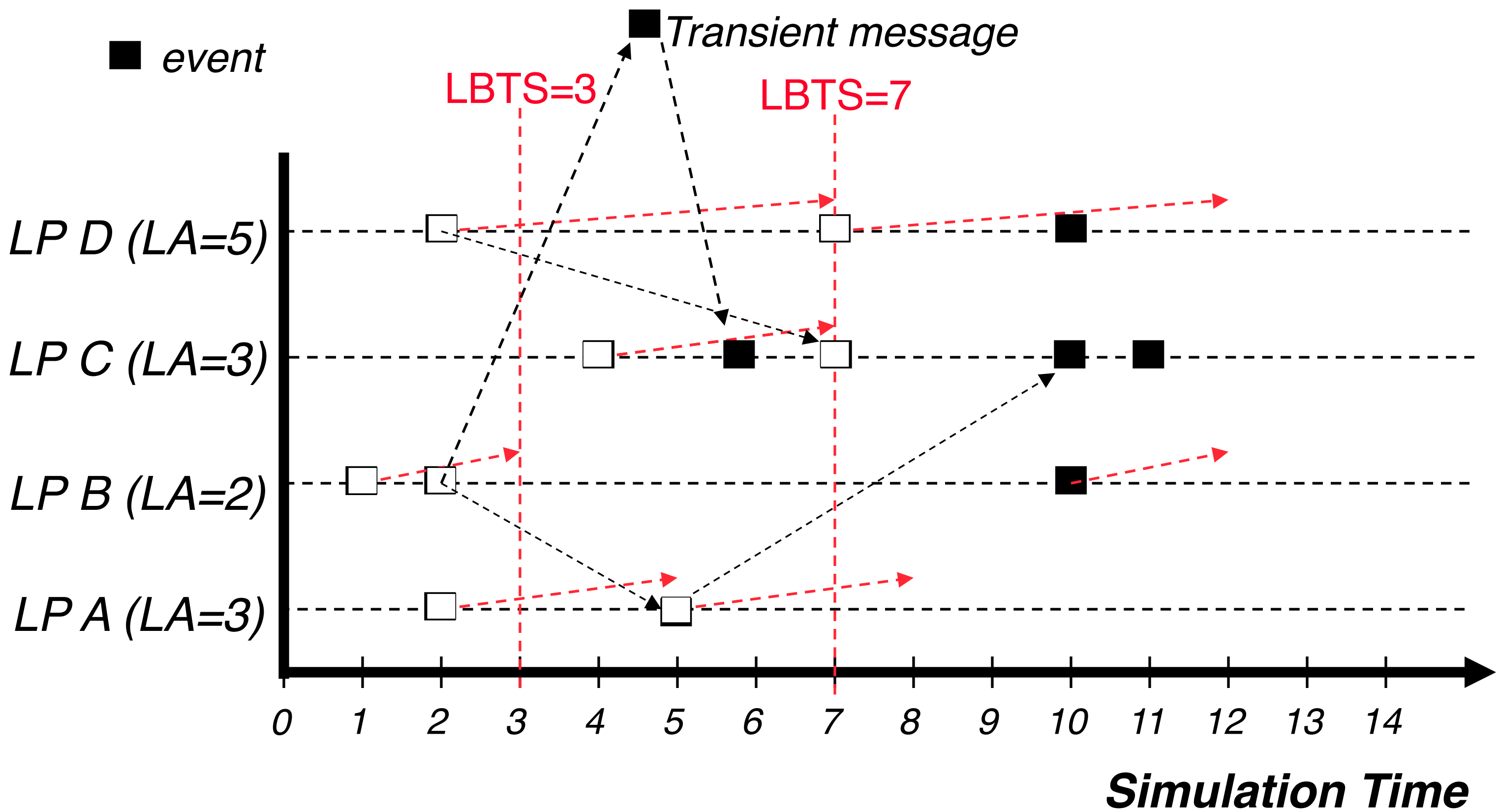
- Transient Messages
  - Transient Message Problem
  - Flush Barrier
  - Tree Implementation
  - Butterfly Implementation
- Distance Between Processes
  - Potential Performance Improvement
  - Distance Matrix

The Transient Message Problem

```
/* synchronous algorithm */
N_i = time of next event in LP_i
LA_i = lookahead of LP_i
WHILE (unprocessed events remain)
    receive messages generated in previous iteration
    LBTS = min (N_i + LA_i)
    process events in with time stamp ≤ LBTS
    barrier synchronization
    endDO
```

- A transient message is a message that has been sent, but has not yet been received at its destination
- The message could be "in the network" or stored in an operating system buffer (waiting to be sent or delivered)
- The synchronous algorithm fails if transient message(s) remain after the processes are released from the barrier

Transient Message Example



Message arrives in LP C's past!

Flush Barrier

No process will be released from the barrier until

- All processes have reached the barrier
- Any message sent by a process before reaching the barrier has arrived at its destination

Revised algorithm:

WHILE (unprocessed events remain)

receive messages generated in previous iteration

LBTS = min (N_i + LA_i)

process events in with time stamp ≤ LBTS

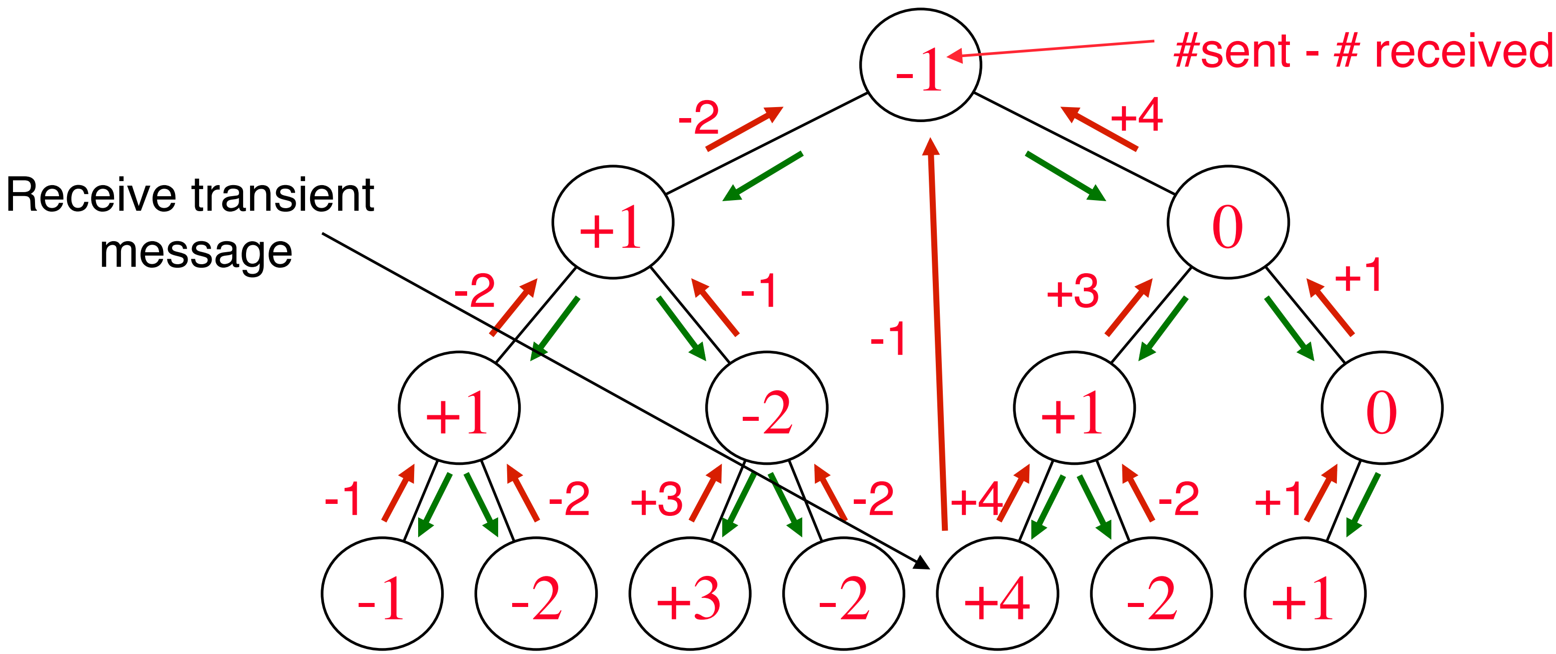
flush barrier

endDO

Implementation

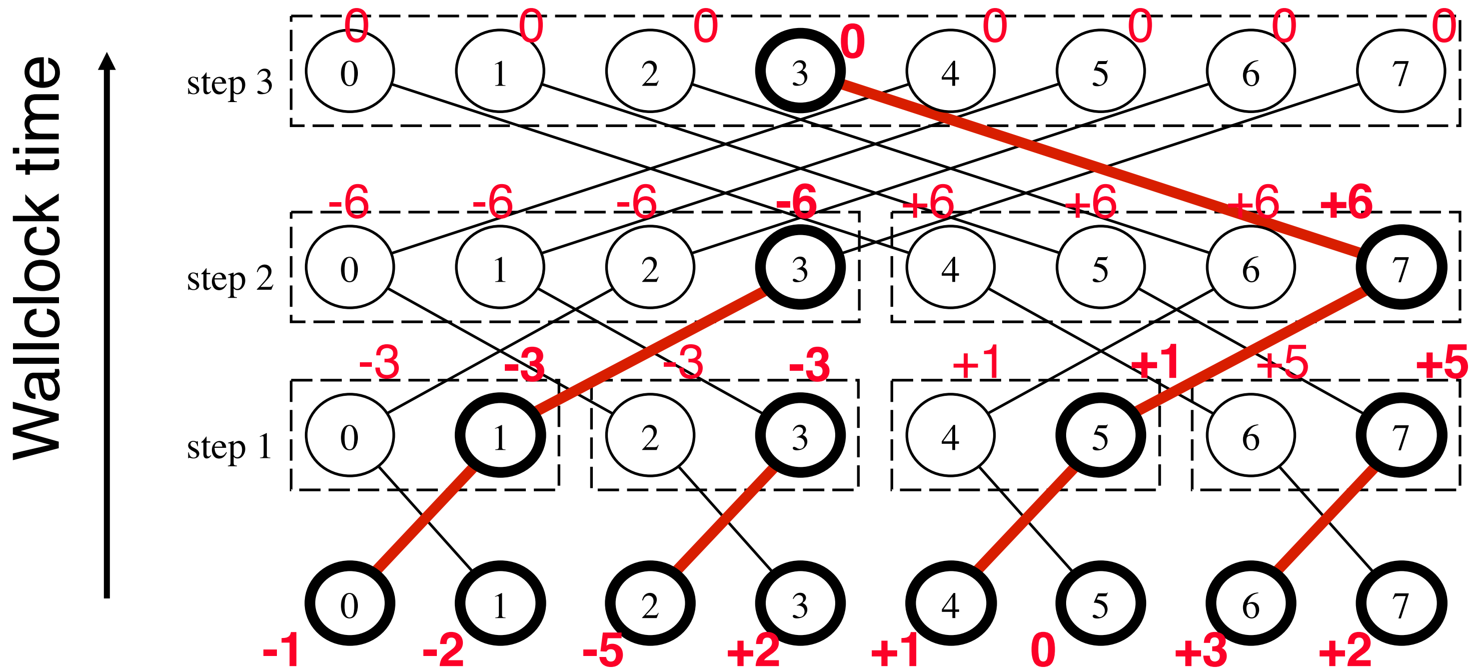
- Use FIFO communication channels
- Send a “dummy message” on each channel; wait until such a message is received on each incoming channel to guarantee transient messages have been received
  - May require a large number of messages
- Another approach: message counters
  - $Send_i$ = number of messages sent by LP_i (this iteration)
  - Rec_i = number of messages received by LP_i (this iteration)
  - There are no transient messages when
    - All processes are blocked (i.e., at the barrier), and
    - $\sum Send_i = \sum Rec_i$

Tree: Flush Barrier



- When a leaf process reaches flush barrier, include counter (#sent - #received) in messages sent to parent
- Parent adds counters in incoming messages with its own counter, sends sum in message sent to its parent
- If sum at root is zero, broadcast "go" message, else wait until sum is equal to zero
- Receive message after reporting sum: send update message to root

Butterfly: Flush Barrier



For (i = 1 to log N)

send local counter to partner at step i

wait for message from partner at step i

local counter = local counter + counter in message

End-for

If local counter not zero after last step:

- Send update messages up butterfly
- Alternatively, abort and retry

Outline

- Transient Messages
  - Transient Message Problem
  - Flush Barrier
  - Tree Implementation
  - Butterfly Implementation
- Distance Between Processes
  - Potential Performance Improvement
  - Distance Matrix

Identifying Safe Events

WHILE (unprocessed events remain)

receive messages generated in previous iteration

LBTS = min (N_i + LA_i) /* time of next event + lookahead */

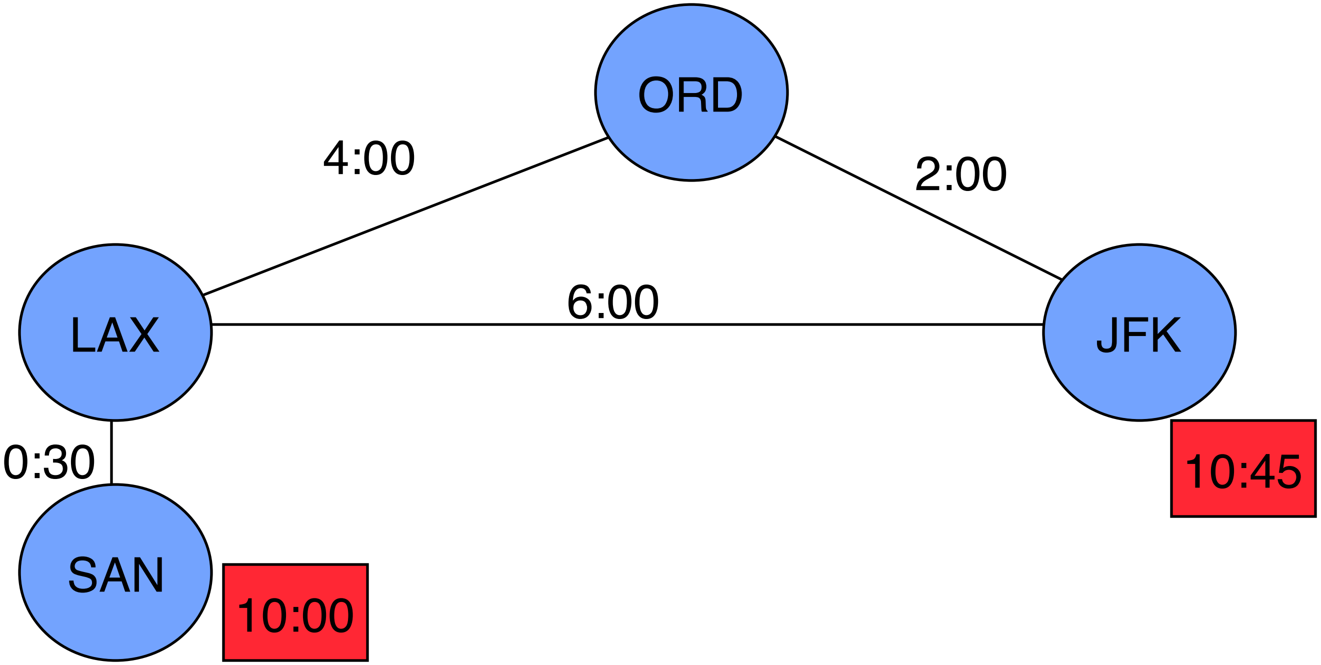
process events in with time stamp ≤ LBTS*

flush barrier /* barrier + eliminate all transient messages */

If all processes are blocked and there are no transient messages in the system, LBTS = min (N_i + LA_i) for each process where N_i and LA_i are the time of the next unprocessed event and lookahead, respectively, for LP_i

- Overly conservative estimate for LBTS
- Does not exploit "locality" in physical systems (things far away can't affect you for some time into the future)

Example



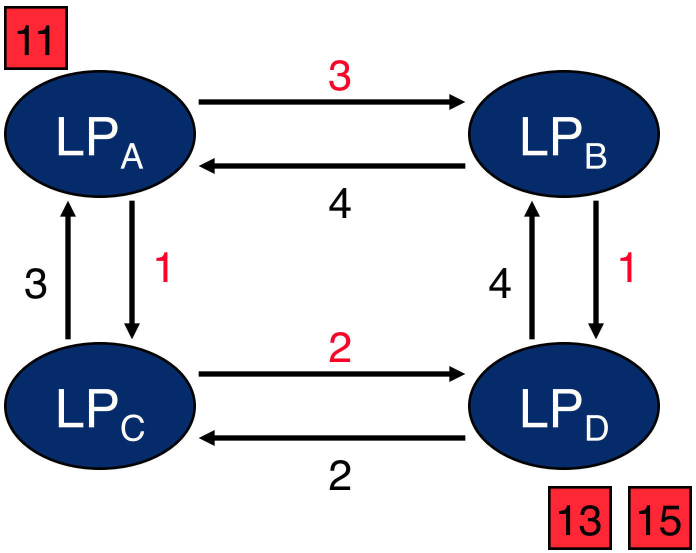
- Lookahead = minimum flight time to another airport
- Can the two events be processed concurrently?
  - Yes because the event @ 10:00 cannot affect the event @ 10:45
- Simple synchronous algorithm:
  - LBTS = 10:30 (10:00 + 0:30)
  - Cannot process event @ 10:45 this iteration
- Algorithm does not consider LP topology

Distance Between LPs

- Associate a lookahead with each link: L_{AB} is the lookahead on the link from LP_A to LP_B
  - Any message sent on the link from LP_A to LP_B must have a time stamp of $T_A + L_{AB}$ where T_A is the current simulation time of LP_A
- A path from LP_A to LP_Z is defined as a sequence of LPs: LP_A, LP_B, …, LP_Y, LP_Z
- The lookahead of a path is the sum of the lookaheads of the links along the path
- D_{AB}, the minimum distance from LP_A to LP_B is the minimum lookahead over all paths from LP_A to LP_B
- The distance from LP_A to LP_B is the minimum amount of simulated time that must elapse for an event in LP_A to affect LP_B

Distance Between Processes

The distance from LP_A to LP_B is the minimum amount of simulated time that must elapse for an event in LP_A to affect LP_B



Distance Matrix:

D [i,j] = minimum distance from LP_i to LP_j

	LP_A	LP_B	LP_C	LP_D
LP_A	4	3	1	3
LP_B	4	5	3	1
LP_C	3	6	4	2
LP_D	5	4	2	4

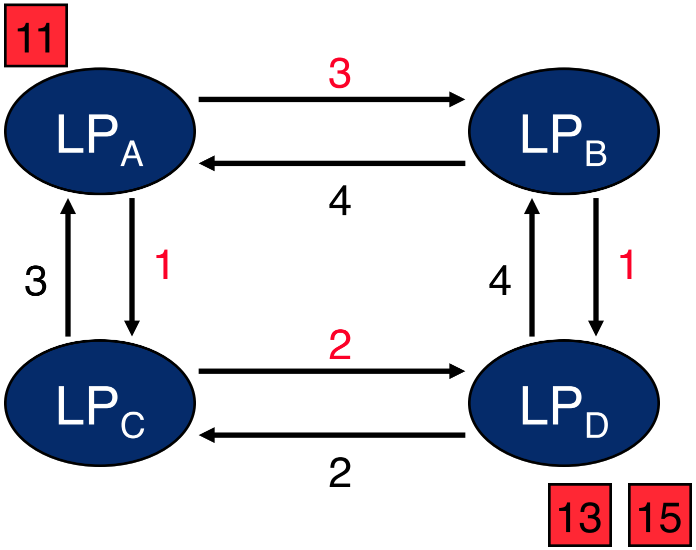
min (1+2, 3+1)

- An event in LP_Y with time stamp T_Y depends on an event in LP_X with time stamp T_X if $T_X + D[X,Y] < T_Y$
- Above, the time stamp 15 event depends on the time stamp 11 event, the time stamp 13 event does not.

Computing LBTS

$LBTS_i = \min(N_j + D_{ji})$ (all j) where N_i = time of next event in LP_i

(assuming all LPs blocked, no transient messages)



Distance Matrix:

D [i,j] = minimum distance from LP_i to LP_j

	LP_A	LP_B	LP_C	LP_D
LP_A	4	3	1	3
LP_B	4	5	3	1
LP_C	3	6	4	2
LP_D	5	4	2	4

min (1+2, 3+1)

$LBTS_A = 15$ [min (11+4, 13+5)]

$LBTS_B = 14$ [min (11+3, 13+4)]

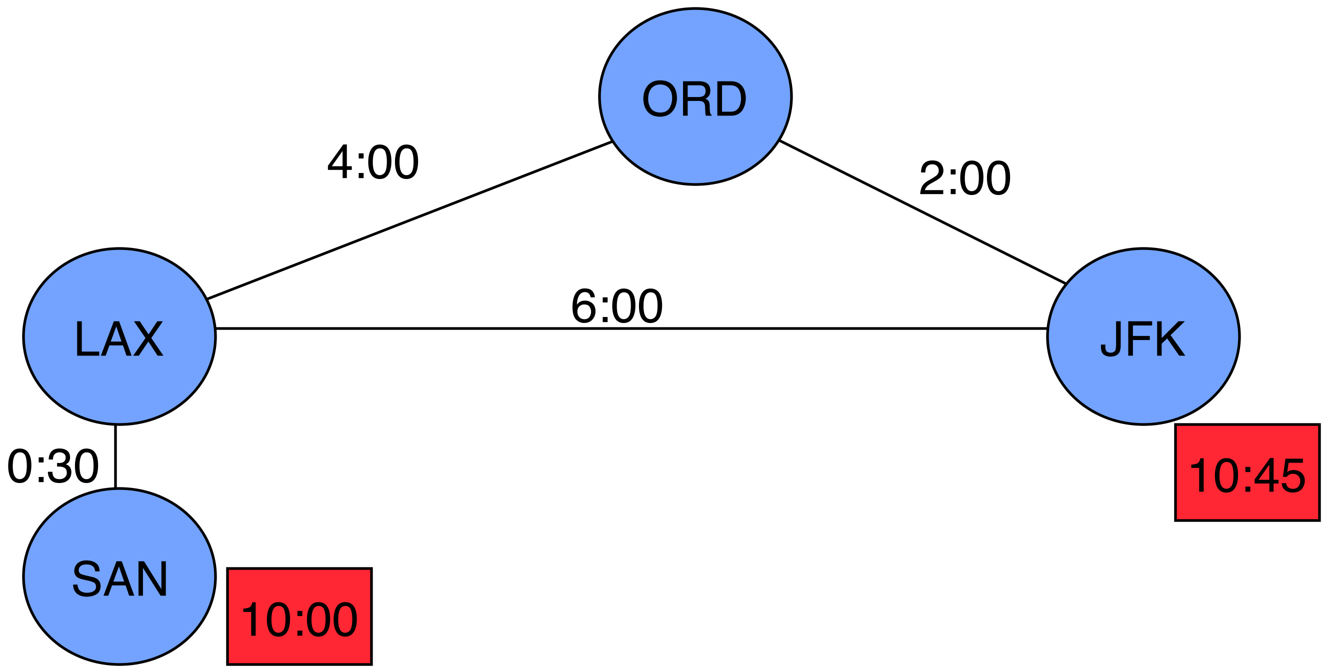
$LBTS_C = 12$ [min (11+1, 13+2)]

$LBTS_D = 14$ [min (11+3, 13+4)]

Need to know time of next event of every other LP

Distance matrix must be recomputed if lookahead changes

Example



- Using distance information:
  - $D_{SAN,JFK}$ = 6:30
  - $LBTS_{JFK}$ = 14:30 (10:00 + 6:30, 10:45 +4:00)
  - Event @ 10:45 can be processed this iteration
  - Concurrent processing of events at times 10:00 and 10:45

Summary

- Transient messages must be accounted for by the synchronization algorithm
  - Flush barrier
  - Send and receive counters
- Distance between LPs
  - Exploit locality in physical systems to improve concurrency in the simulation execution
  - Increased complexity, overhead
  - Lookahead and topology changes introduce additional complexities