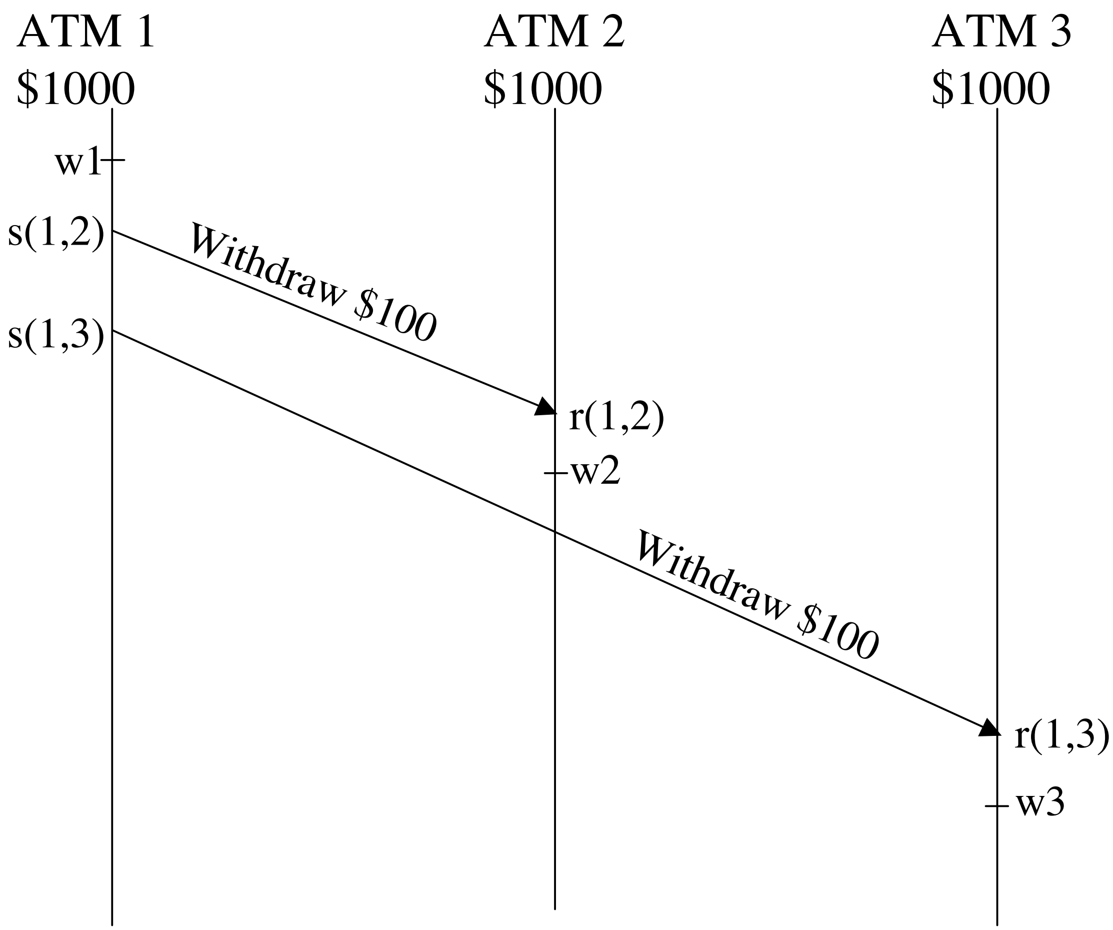
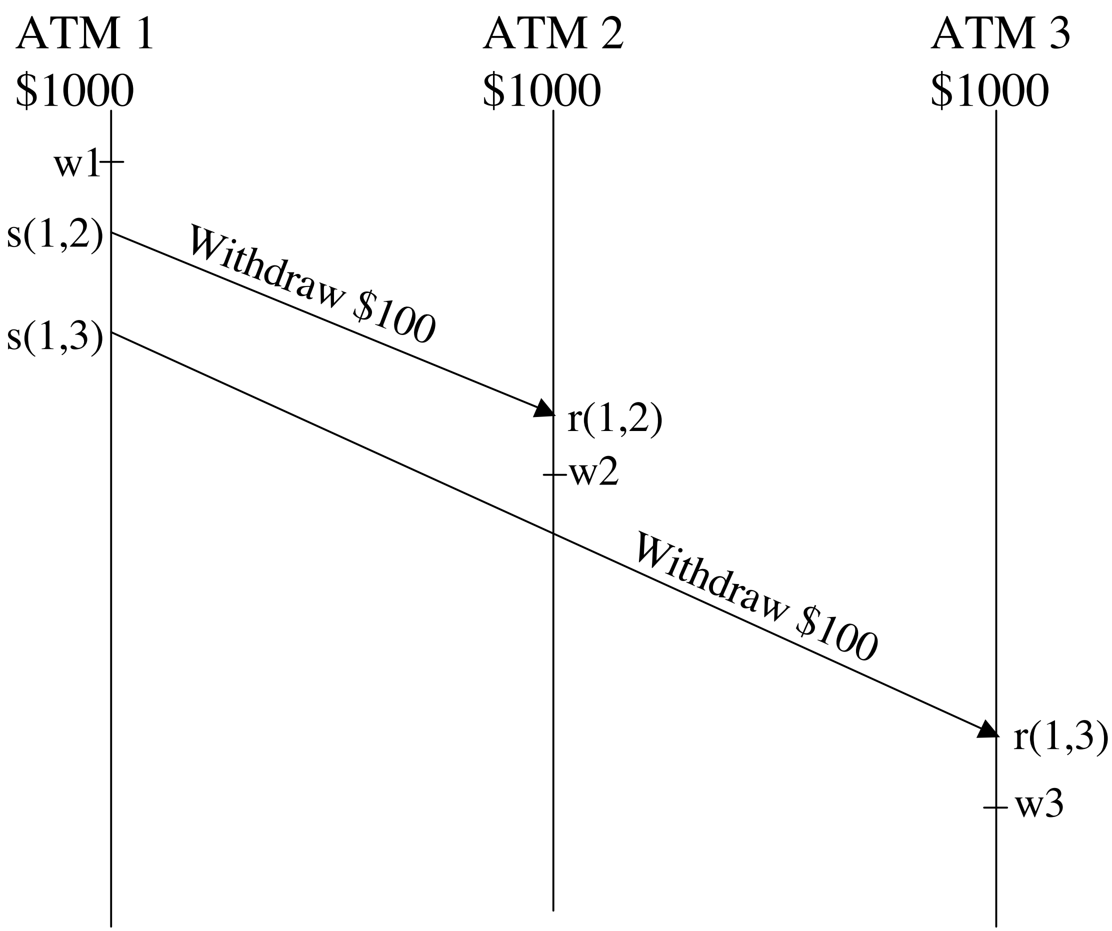
Logical Time

- Each event is assigned a logical time from a totally ordered set T
- The logical times for the events must respect any possible dependencies between events
  - If event A happens before event B at some process or in some channel, then the logical time of A must precede the logical time of B

Logical Time



Logical Time



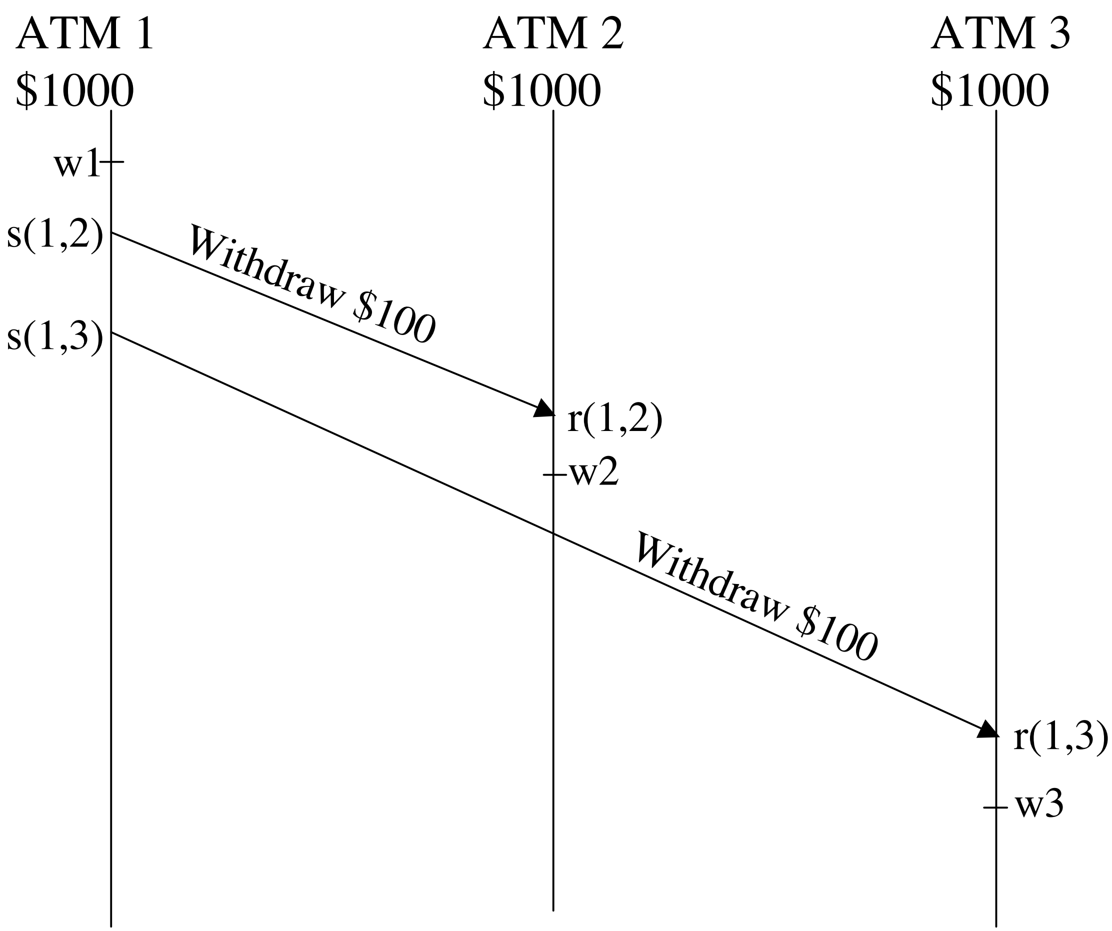
Evt	Time		
w1	1	1	1
s(1,2)	2	2	2
s(1,3)	3	5	3
r(1,2)	4	3	6
w2	5	4	7
r(1,3)	6	6	4
w3	7	7	5

w1 < s(1,2) < s(1,3)
s(1,2) < r(1,2)
r(1,2) < w2
s(1,3) < r(1,3)
r(1,3) < w3

Convenient Fact (Theorem 18.1)

- Take any allowable assignment of logical times to an execution $s_0 a_1 s_1 a_2 s_2 \ldots$
  - That is, ltimes are in order for a_i's at the same process and for sends and receives
- If the execution is reordered by logical times, it looks exactly the same to each process

Logical Time

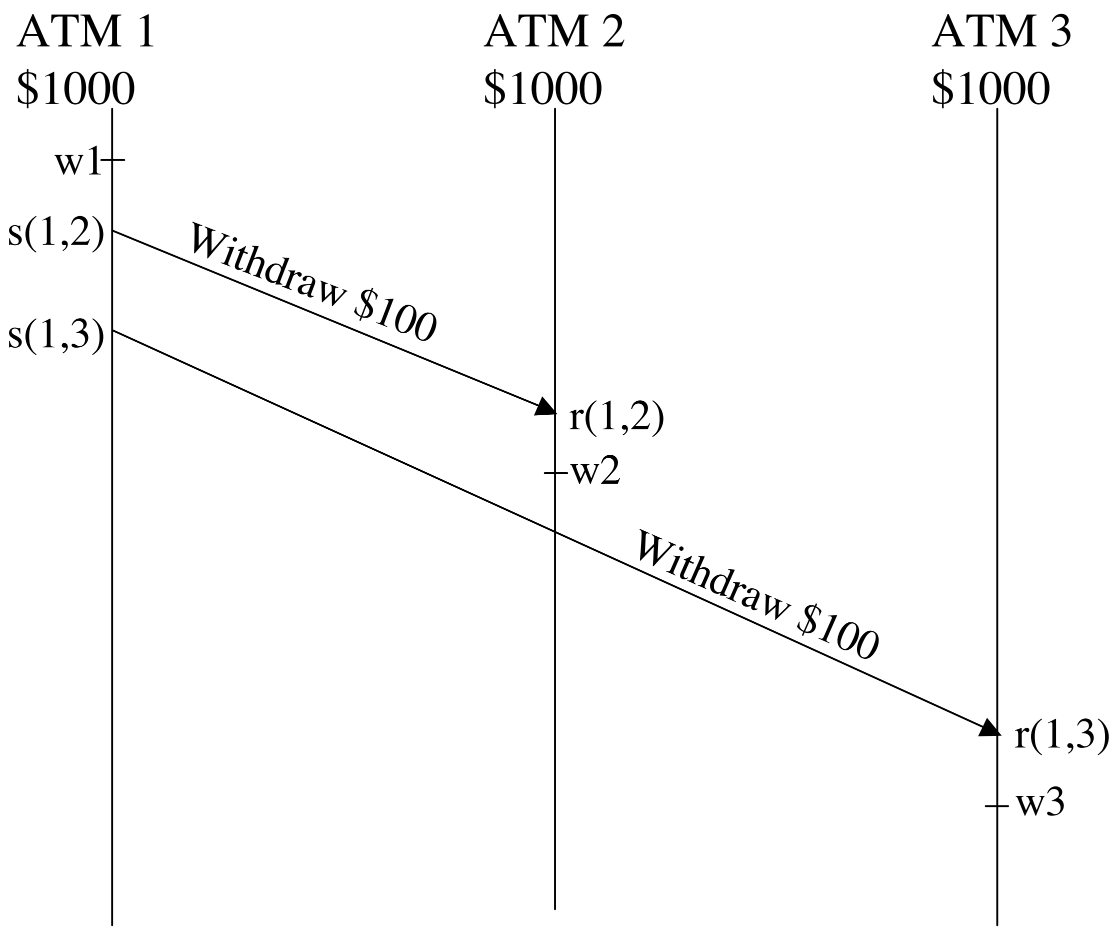


Evt	Time		
w1	1	1	1
s(1,2)	2	2	2
s(1,3)	3	5	3
r(1,2)	4	3	6
w2	5	4	7
r(1,3)	6	6	4
w3	7	7	5

w1 < s(1,2) < s(1,3)
s(1,2) < r(1,2)
r(1,2) < w2
s(1,3) < r(1,3)
r(1,3) < w3

Send/receive diagram

aka Call Flow or Message Sequence Chart



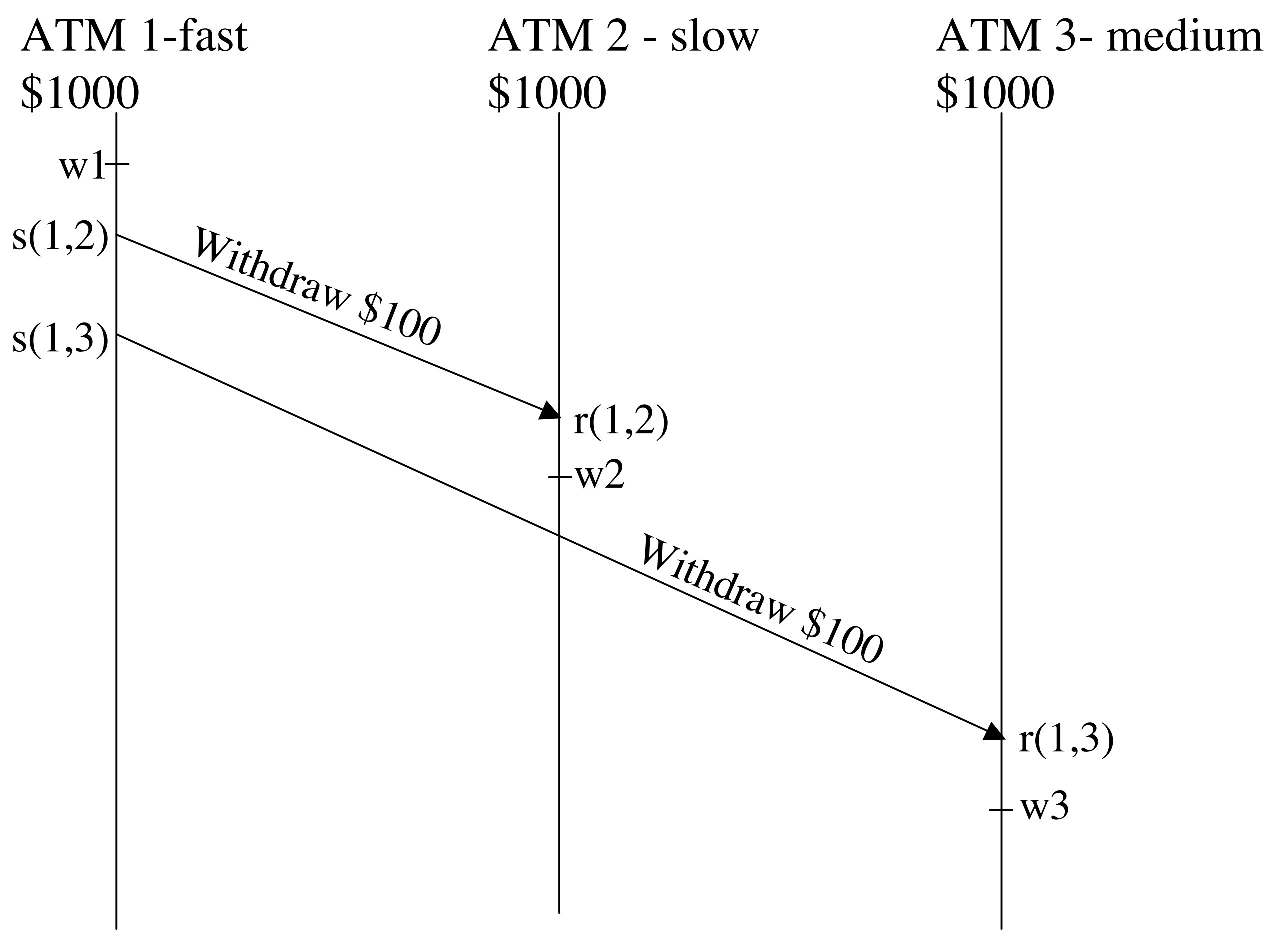
Non-blocking time-keeping

- Each host maintains a clock
- Before it sends, it timestamps the message with the next value of the clock
- When it receives it updates the clock to be strictly greater than the timestamp on the message and the local clock

Blocking time-keeping

- Each process assigns a logical time as the time of the clock + the order of events at the process
- It timestamps each message with the current time on the clock
- It holds messages in a queue until the local clock catches up

Logical time

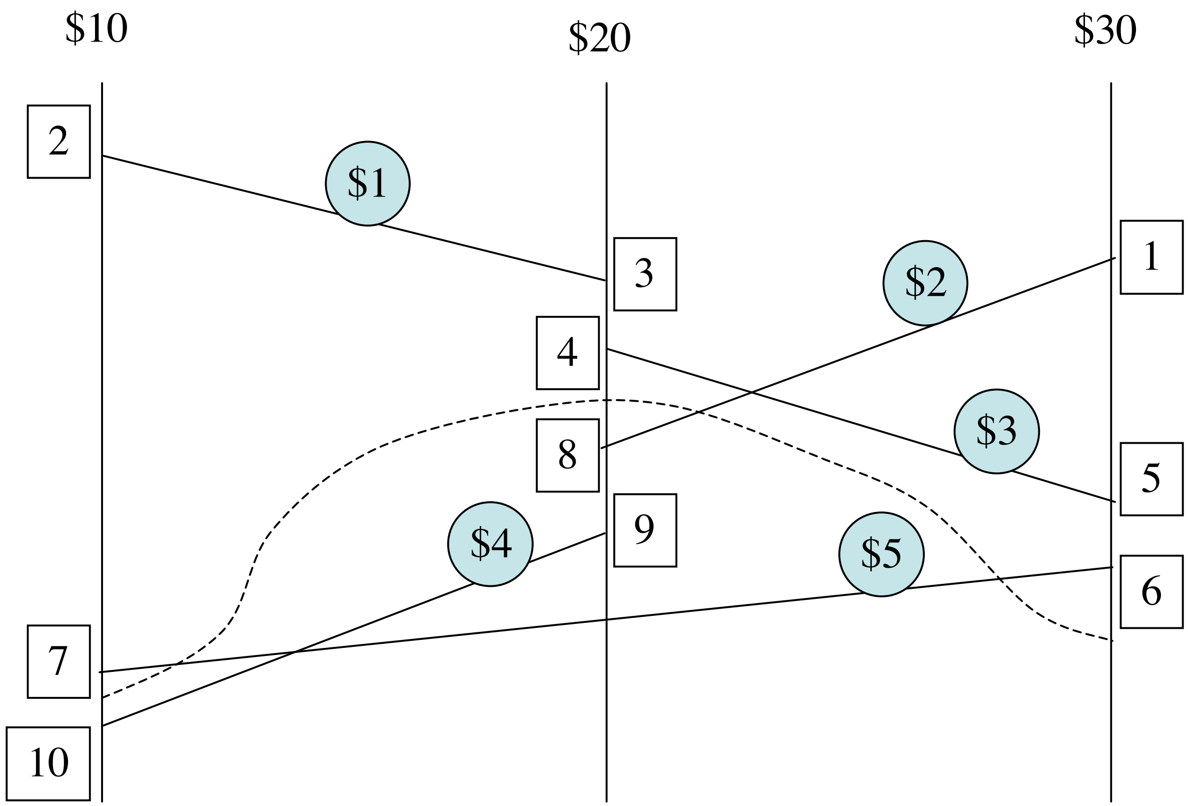


The Banking Problem

- Asynchronous send/receive network
- Banking system with a balance at each process
- Transfers between banks
- Each process has a local balance
- The total of the local balances is the correct amount in the system

The Banking Problem

Using logical times to define snapshot



The Money Counting Algorithm

- For each process of A, determine its local balance after all events with logical times before t and before any event with logical time after t
- For each channel, determine the amount of money in messages sent before t but received after t

Computing the local balance

- For process values:
  - Attach a timestamp to each send event
  - Record money value just before the first event with time $> t$
- For channel values:
  - Record incoming messages that arrive after time t until the first message sent at time $> t$
- The balance: sum of the process value and all incoming channels

Global Snapshot Problem

- A global snapshot returns a state of the system
  - States of all processes and channels
  - Looks to each process as if it was taken at the same instant everywhere
- The bank problem is a special case
- Note that the actual values computed in the bank algorithm may never have been observable