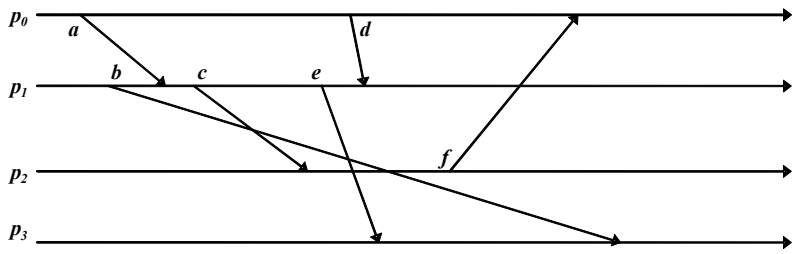
CONSISTENT CUTS

39

39

Temporal distortions

- What does “now” mean?

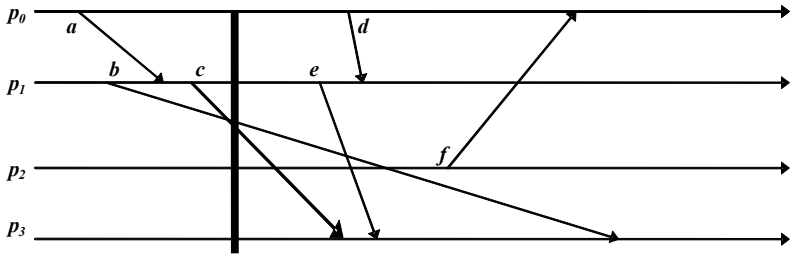


40

40

Temporal distortions

- What does “now” mean?

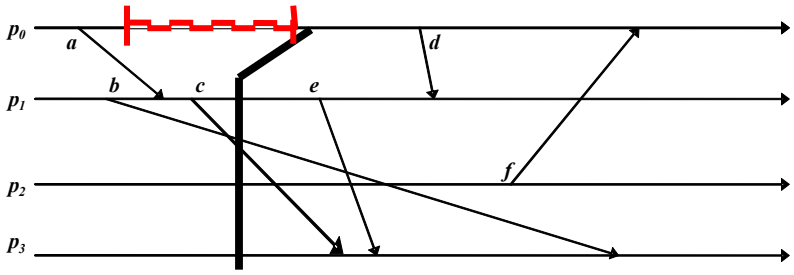


41

41

Temporal distortions

- Timelines can “stretch”…



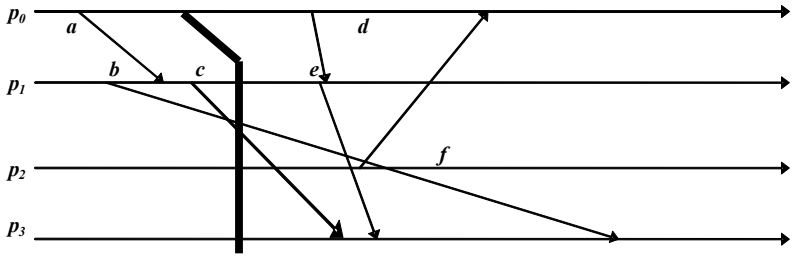
- … caused by scheduling effects, message delays, message loss…

42

42

Temporal distortions

- Timelines can “shrink”



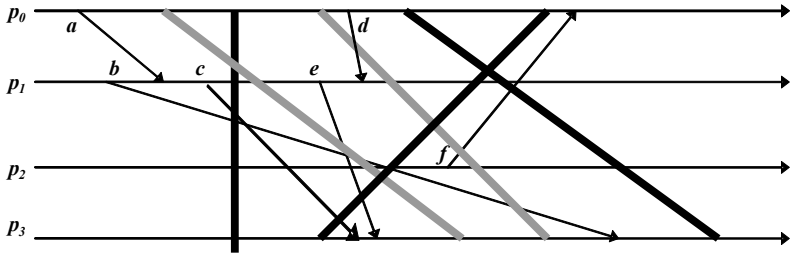
- E.g. something lets a machine speed up

43

43

Temporal distortions

- Cuts represent instants of time.



- But not every “cut” makes sense

44

44

Consistent cuts and snapshots

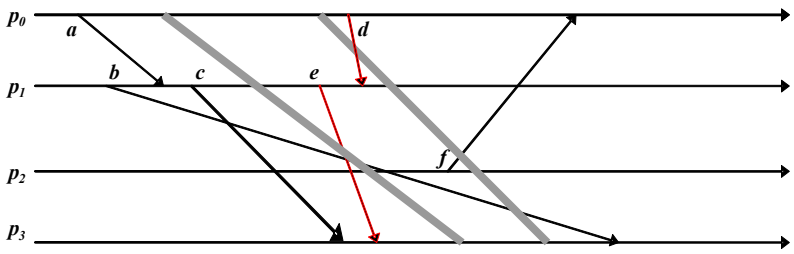
- Identify system states that "might" have occurred in real-life
  - Avoid capturing "inconsistent" states
    - Receive without a send
  - This is the problem with the gray cuts

45

45

Temporal distortions

- Red messages cross gray cuts "backwards"

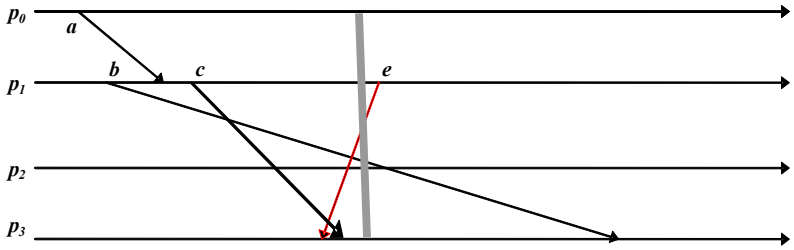


46

46

Temporal distortions

- Red messages cross gray cuts “backwards”



- In a nutshell: the cut includes a message that “was never sent”

47

47

DISTRIBUTED LOGGING

48

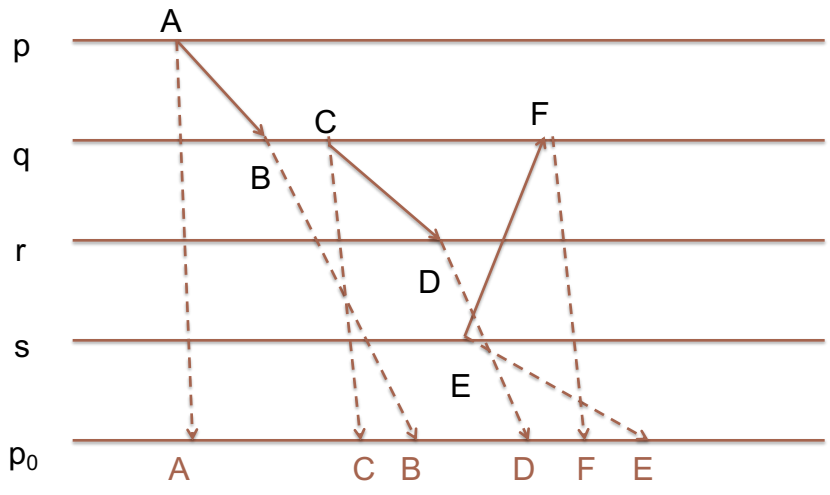
48

Distributed Logging

- We have n processes $p_1,\ldots,p_n$
- We want to use a monitor process p_0 to build a trace of the system for debugging purposes
- Protocol: every time an event e happens at a process p_i, it sends a notification of that event to p_0

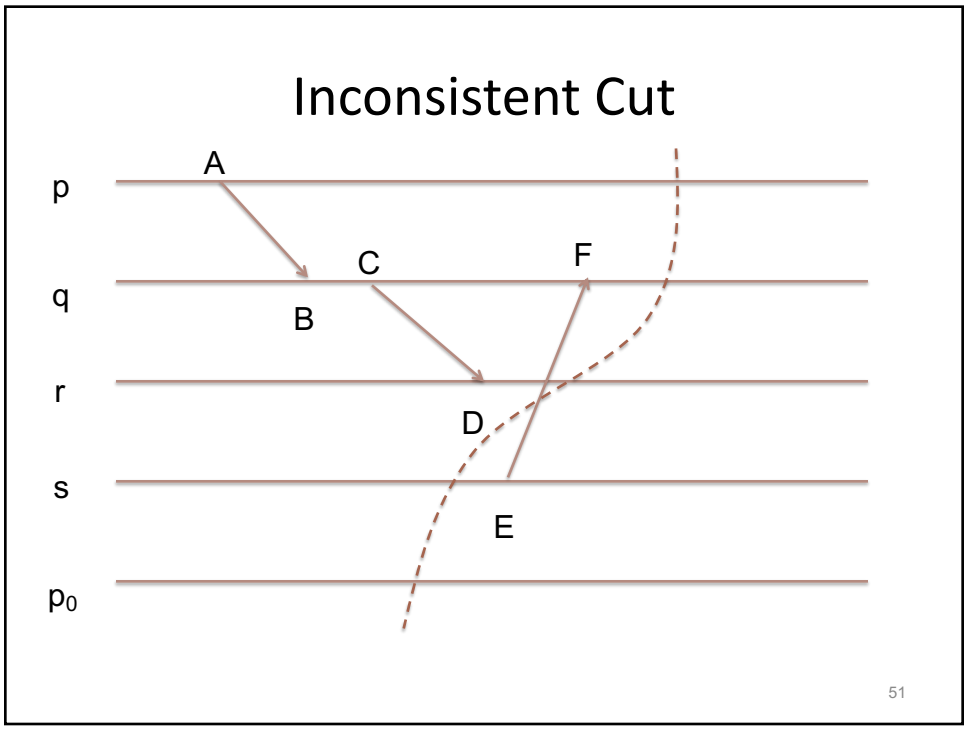
49

49

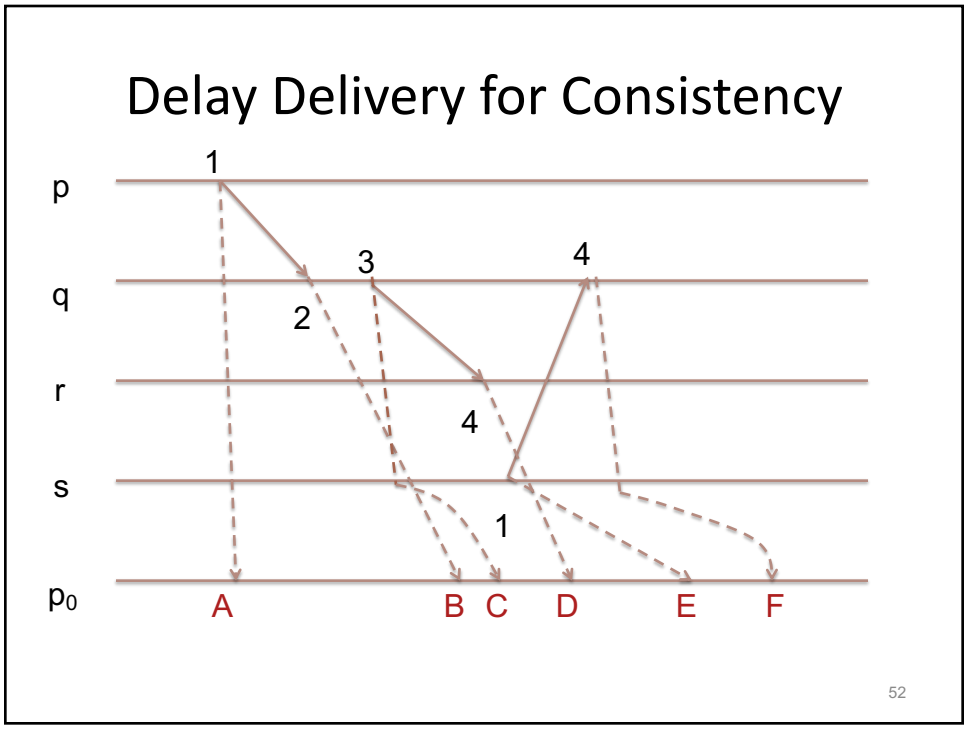


50

50


Inconsistent Cut
p
q
r
s
p_0
A
B
C
D
E
F
51

51


Delay Delivery for Consistency
p
q
r
s
p_0
1
2
3
4
4
1
A
B
C
D
E
F
52

52

Clock Condition

- **Clock Condition:**
  - $e \rightarrow e'$ implies $LT(e) < LT(e')$
- **Delivery Rule 1 (DR1):**

  At time t, deliver all received messages with timestamps up to t, in increasing timestamp order
- Clock condition ensures consistent observations

53

53

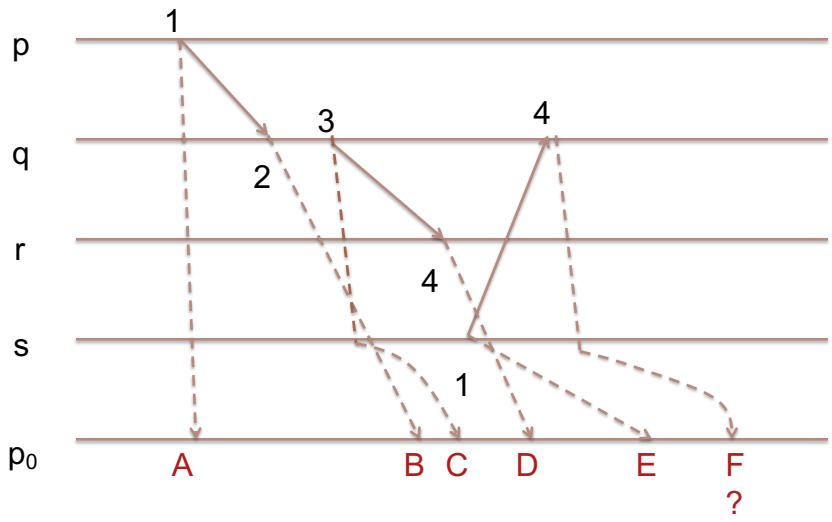
Gap Detection

- We cannot deliver a message m with TS(m) = t unless we are certain that no message m' with TS(m') < t can be received
- **Gap Detection:**
  - Given two events e and e'
  - Given LT(e) < LT(e')
  - Determine whether an event e'' exists such that LT(e) < LT(e'') < LT(e')

54

54

Safe to Deliver?



55

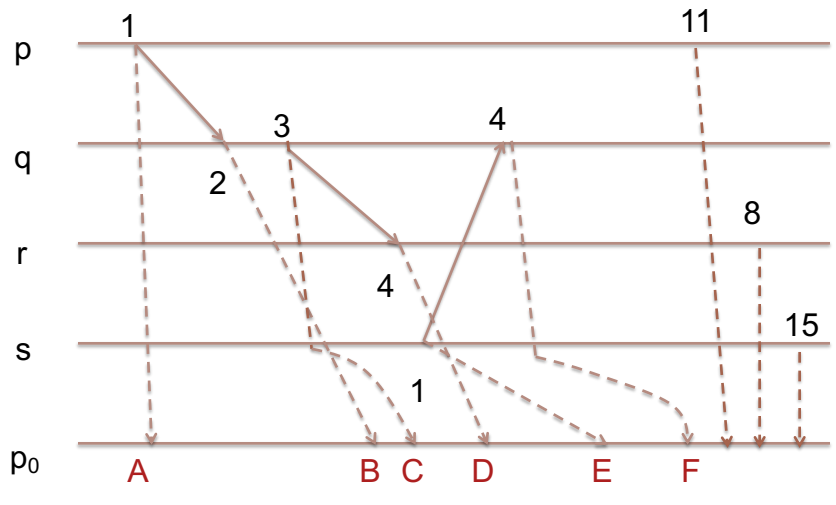
55

Stable Messages

- Message m received at p is **stable** if no future messages with smaller timestamps will be received at p
- **Delivery Rule 2 (DR2):** Deliver all stable messages at p0 in increasing time-stamp order
- With *FIFO* channels, stability is assured once messages with greater timestamps are received from every other process

56

56



57

57

Problem

- Delivery Rule 2 is too conservative
  - We have to see later messages from every other process before delivering a message from p

58

58

VECTOR TIME

59

59

Ordering Relations

- Ordered set (A, $\leq$)
- **Total order:** order is
  - *Total*: for all x,y, either $x \leq y$ or $y \leq x$
  - *Symmetric*: $x \leq y$ and $y \leq x$ implies $x = y$
  - *Transitive*: $x \leq y$ and $y \leq z$ implies $x \leq z$
- **Partial order:** weaken totality to *reflexivity*: $x \leq x$ for all x
- **Preorder:** ordering relation is transitive, not refl
  - $x < y$ and $y < z$ implies $x < z$

60

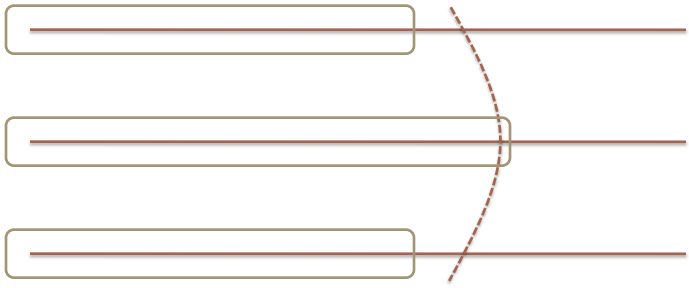
60

Potential Causality

- If A happens before B, A→B, then LT(A)<LT(B)
- But converse might not be true:
  - If LT(A)<LT(B) can't be sure that A→B
  - Total order placed on what is a partial order

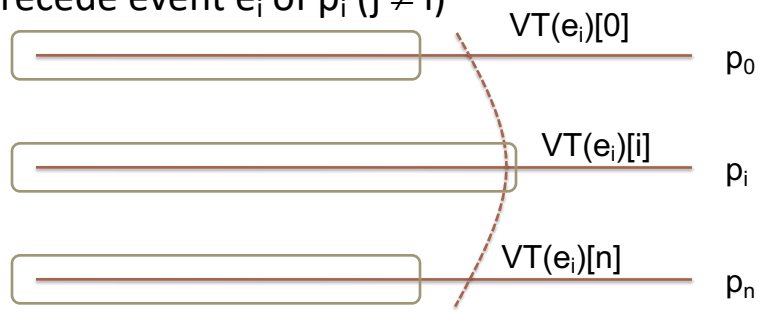
Vector Clocks

- Here we treat timestamps as a list
  - One counter for each process
  - Vector of n counters represents a "cut" of the executions of n processes



Operational Interpretation

- $VT(e_i)[i]$ = number of events p_i has executed up to and including e_i
- $VT(e_i)[j]$ = number of events of p_j that causally precede event e_i of p_i ($j \neq i$)



63

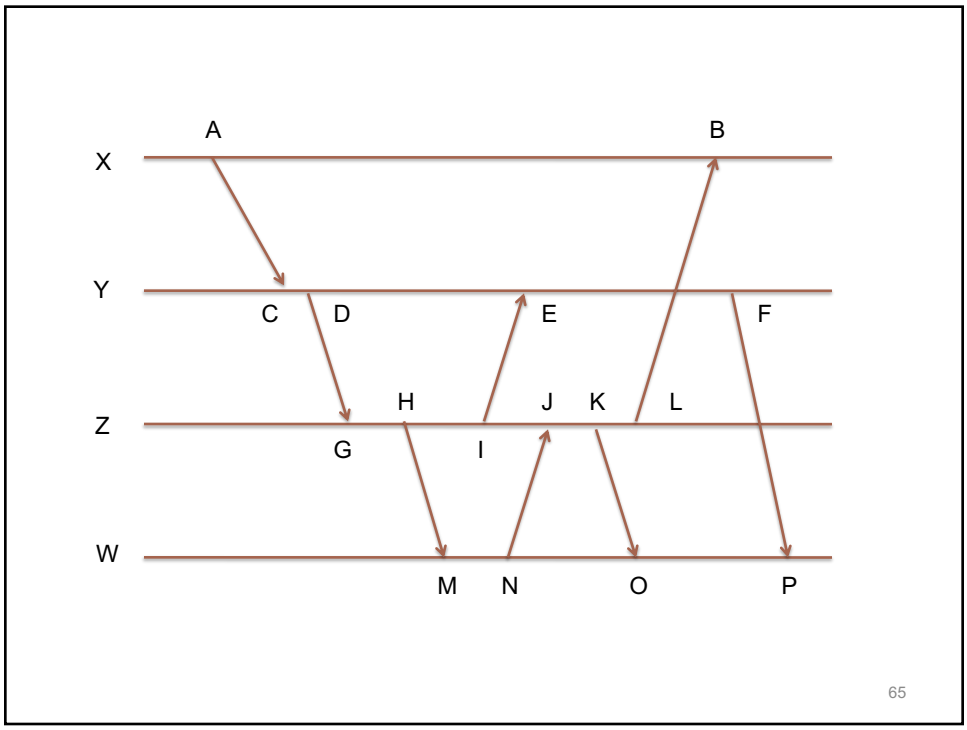
63

Vector Clocks

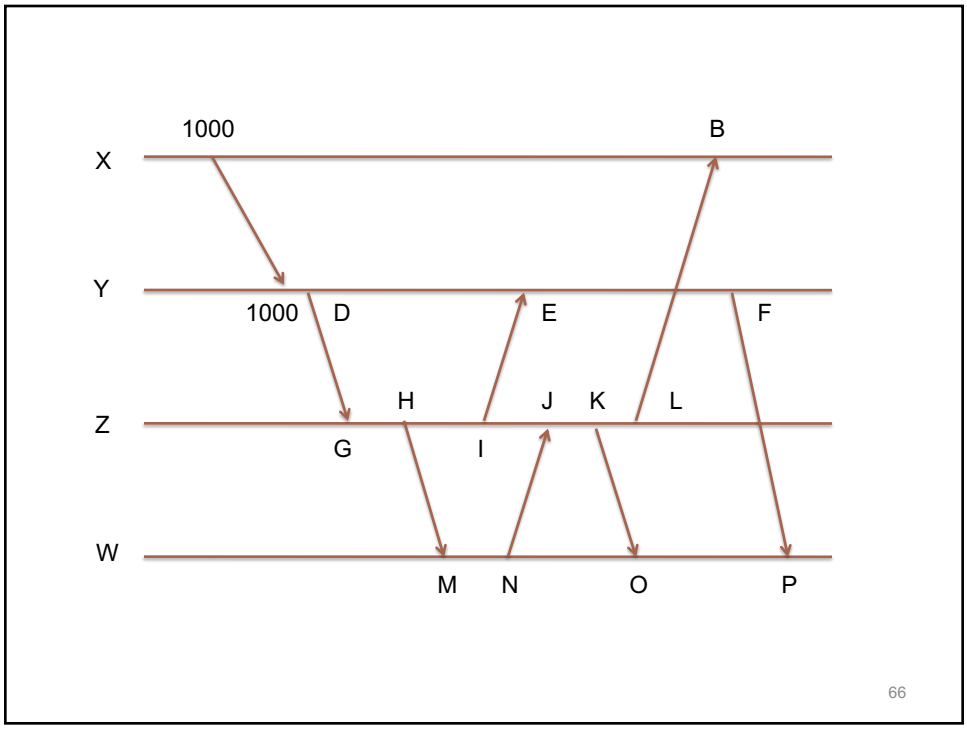
- Rules for managing vector clock
  - When event happens at p, increment $VT_p[index_p]$
    - Normally, also increment for snd and rcv events
  - When sending a message, set $TS(m)=VT_p$
  - When receiving, set $VT_q=max(VT_q, TS(m))$

64

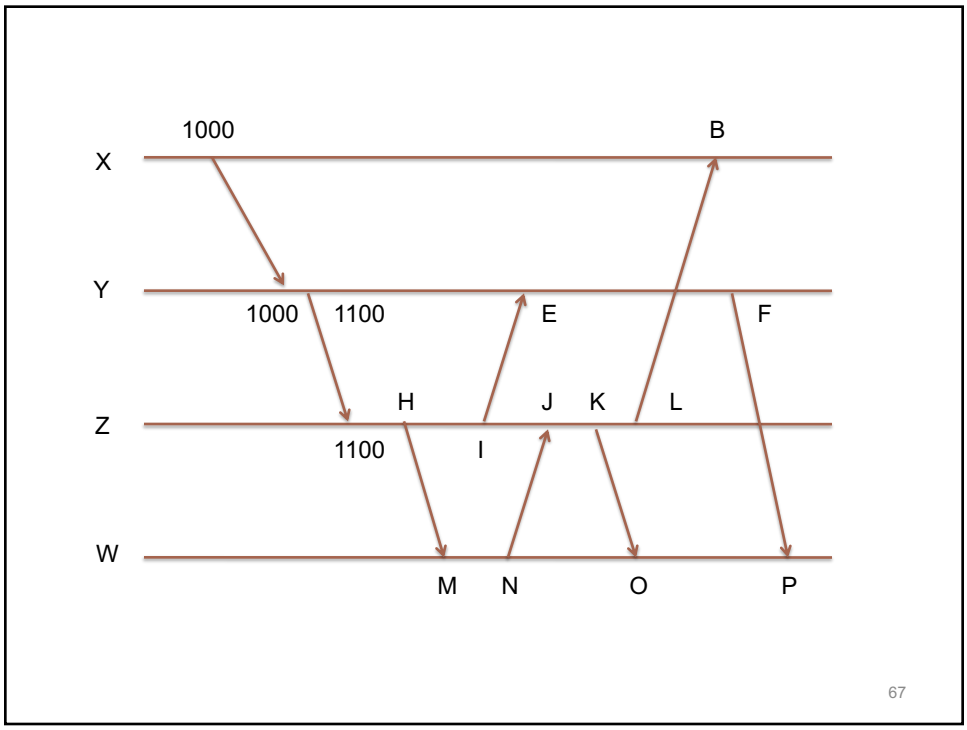
64


A
B
X
Y
C
D
E
F
H
J
K
L
Z
G
I
W
M
N
O
P
65

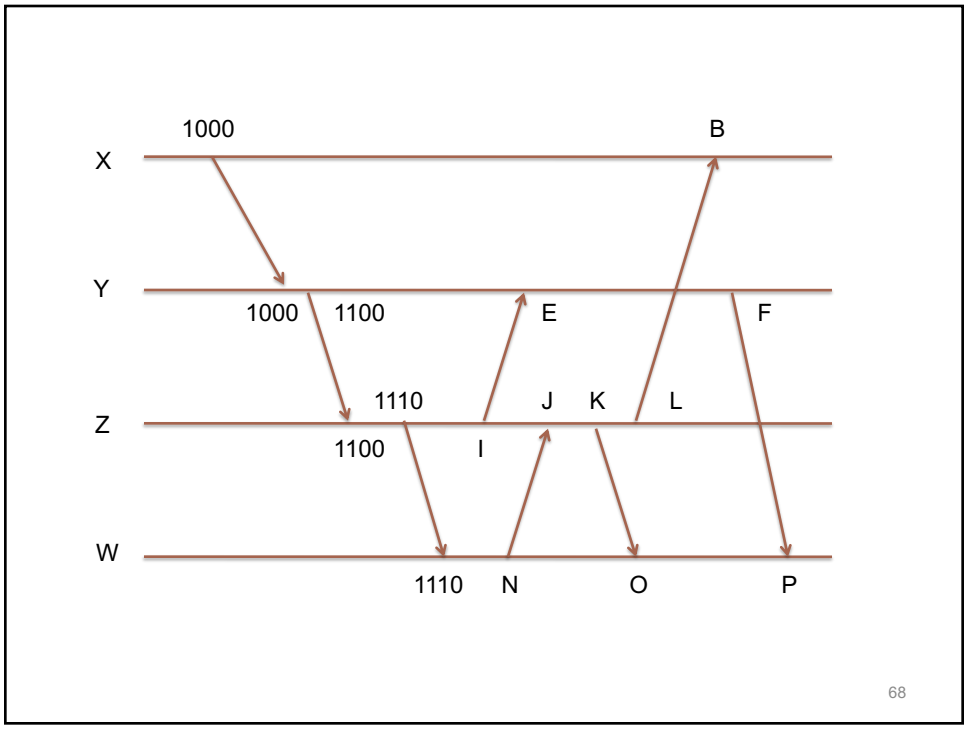
65


1000
B
X
Y
1000
D
E
F
H
J
K
L
Z
G
I
W
M
N
O
P
66

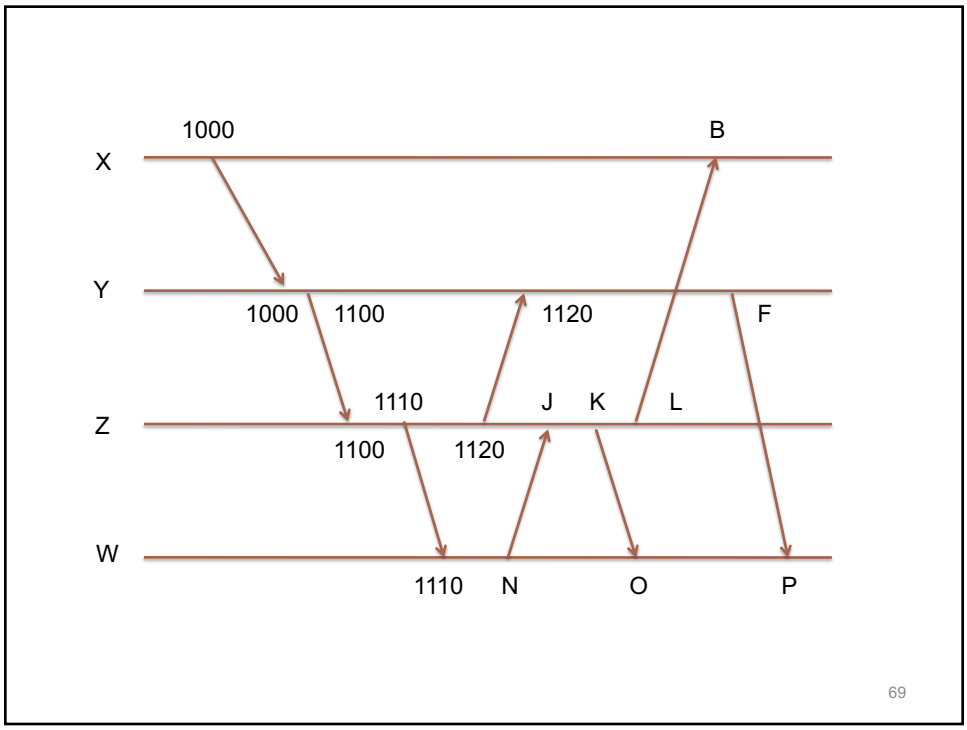
66


1000
B
X
Y
1000
1100
E
F
H
J
K
L
Z
1100
I
W
M
N
O
P
67

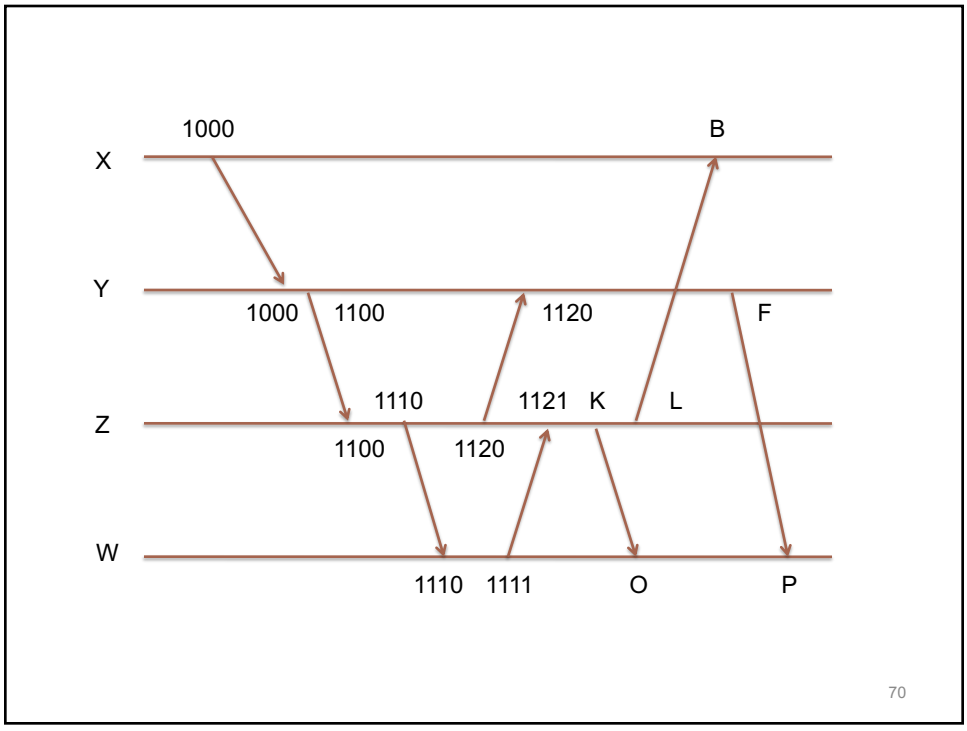
67


1000
B
X
Y
1000
1100
E
F
1110
J
K
L
Z
1100
I
W
1110
N
O
P
68

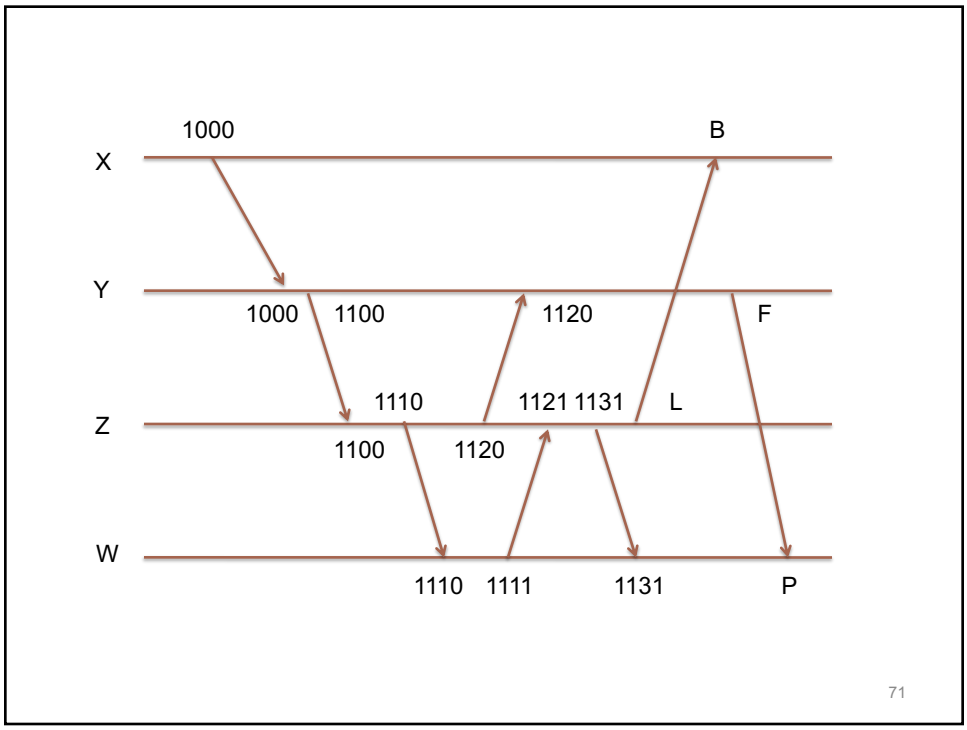
68


1000
B
X
Y
1000
1100
1120
F
1110
J
K
L
Z
1100
1120
W
1110
N
O
P
69

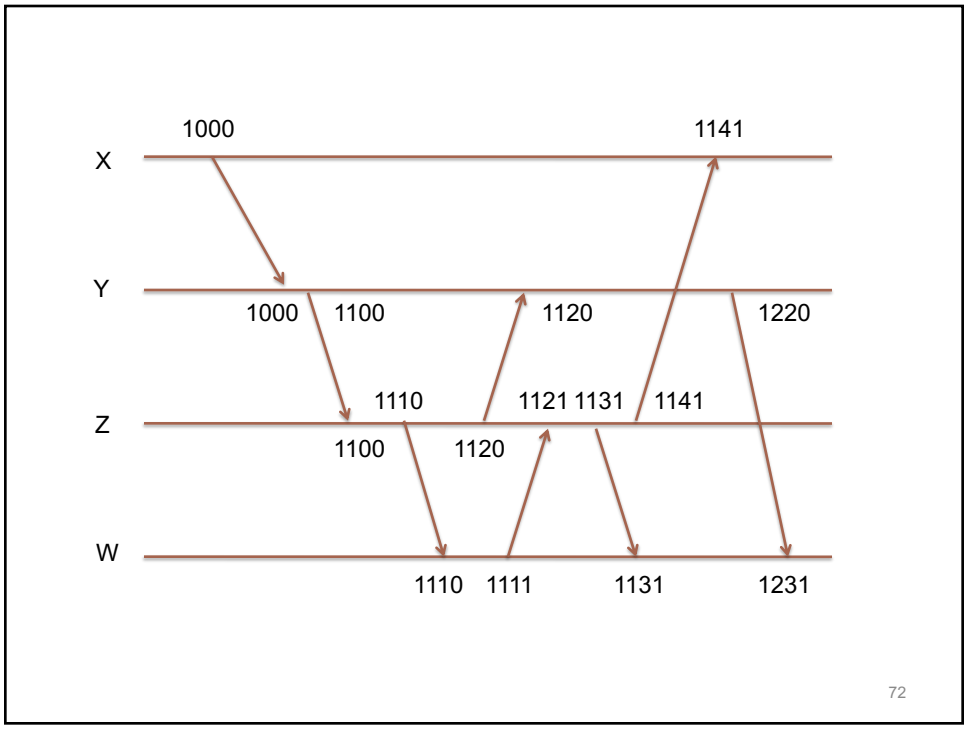
69


1000
B
X
Y
1000
1100
1120
F
1110
1121
K
L
Z
1100
1120
W
1110
1111
O
P
70

70


1000
B
X
Y
1000
1100
1120
F
1110
1121 1131
L
Z
1100
1120
W
1110
1111
1131
P
71

71


1000
1141
X
Y
1000
1100
1120
1220
1110
1121 1131
1141
Z
1100
1120
W
1110
1111
1131
1231
72

72

Rules for comparison of VTs

- We'll say that $VT_A \leq VT_B$ if
  - $VT_A[i] \leq VT_B[i]$ for all i
- And we'll say that $VT_A < VT_B$ if
  - $VT_A \leq VT_B$ but $VT_A \neq VT_B$
  - That is, for some i, $VT_A[i] < VT_B[i]$
- Examples?
  - [2,4] ≤ [2,4]
  - [1,3] < [7,3]
  - [1,3] is "incomparable" to [3,1]

73

73

Vector time and happens before

- If $A \rightarrow B$, then $VT_A < VT_B$
  - Write a chain of events from A to B
  - Step by step the vector clocks get larger
- If $VT_A < VT_B$ then $A \rightarrow B$
  - Two cases: if A and B both happen at same process p, trivial
  - If A happens at p and B at q, can trace the path back by which q "learned" $VT_A[p]$
- Otherwise A and B happened concurrently

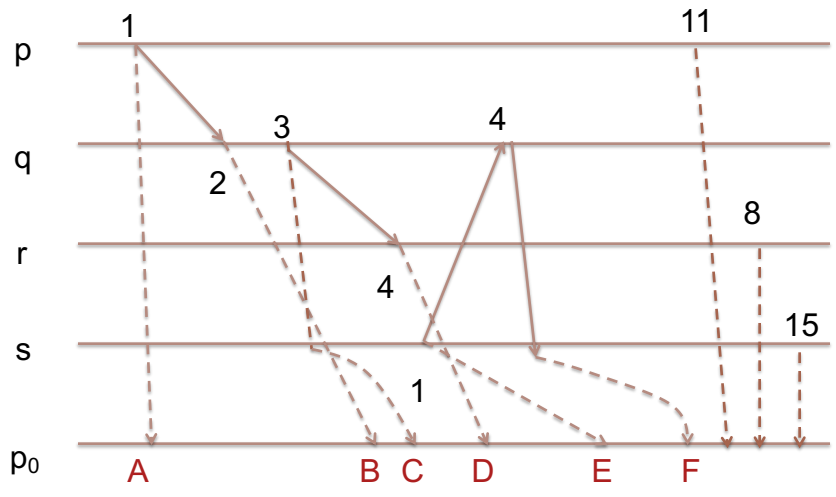
74

74

DISTRIBUTED LOGGING AND VECTOR TIME

75

75



76

76

Clock Condition

- Delivery Rule 2 too conservative
- Clock Condition:
  - $e \rightarrow e'$ implies LT(e) < LT(e')
  - Possible: LT(e) < LT(e'), but not ($e \rightarrow e'$)
- Logical clocks give potential causality
  - Hence the need to wait for stability

77

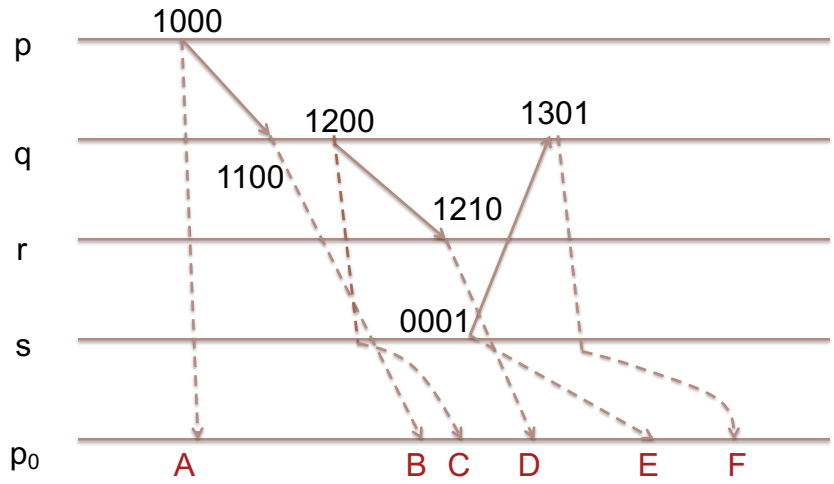
77

Strong Clock Condition

- **Delivery Rule 3 (DR3):**
  - Deliver messages all of whose causal predecessors have been delivered
- Relies on: **Strong Clock Condition**
  - $e \rightarrow e'$ if and only if VT(e) < VT(e')

78

78



F can be delivered once we have received
1 message from p, 2 from q, one from s

79

79

Causal Delivery

- We have used causal delivery at monitor process to construct consistent observations
- Causal delivery may also be used in general message delivery
- Deadlock problems with point-to-point (requires matrix clock)
- Vector clock used with causal broadcast

80

80