Lecture 17

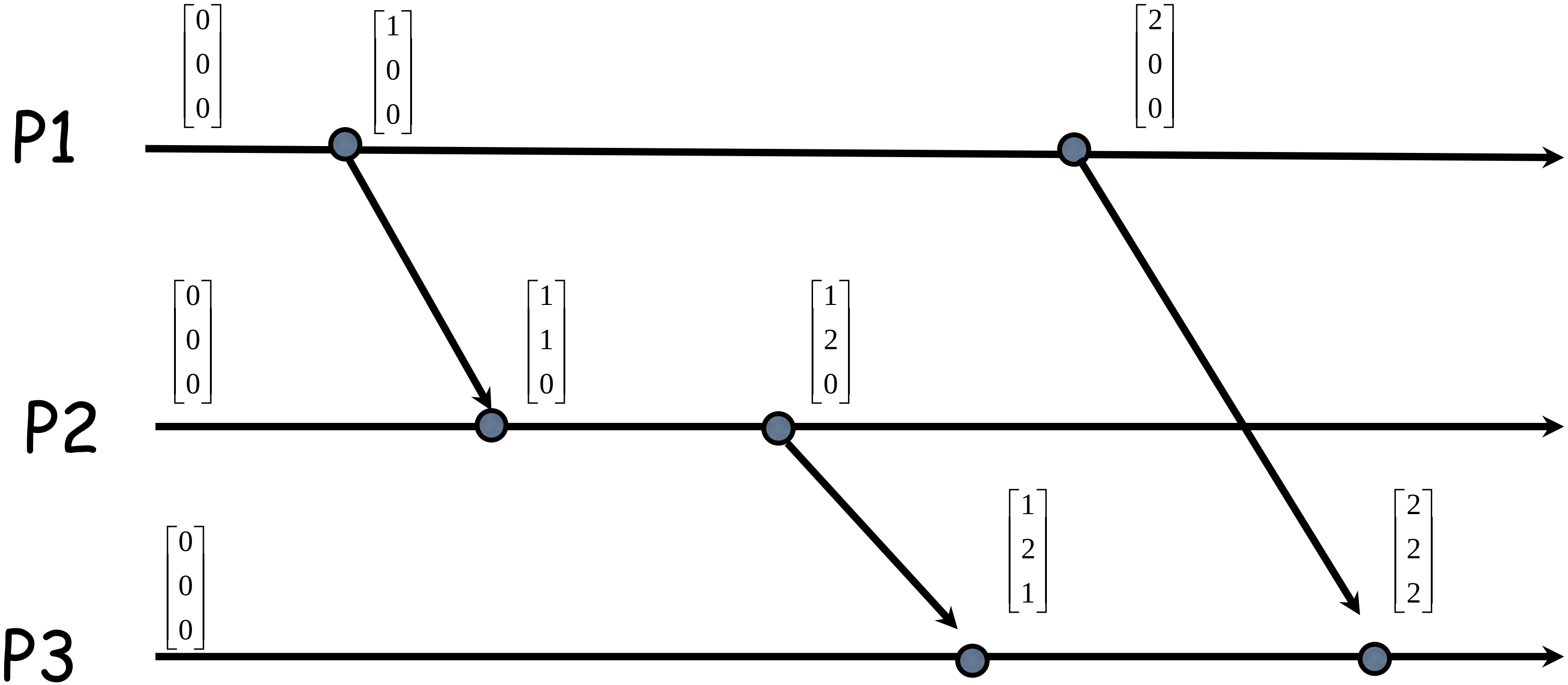
- Administration

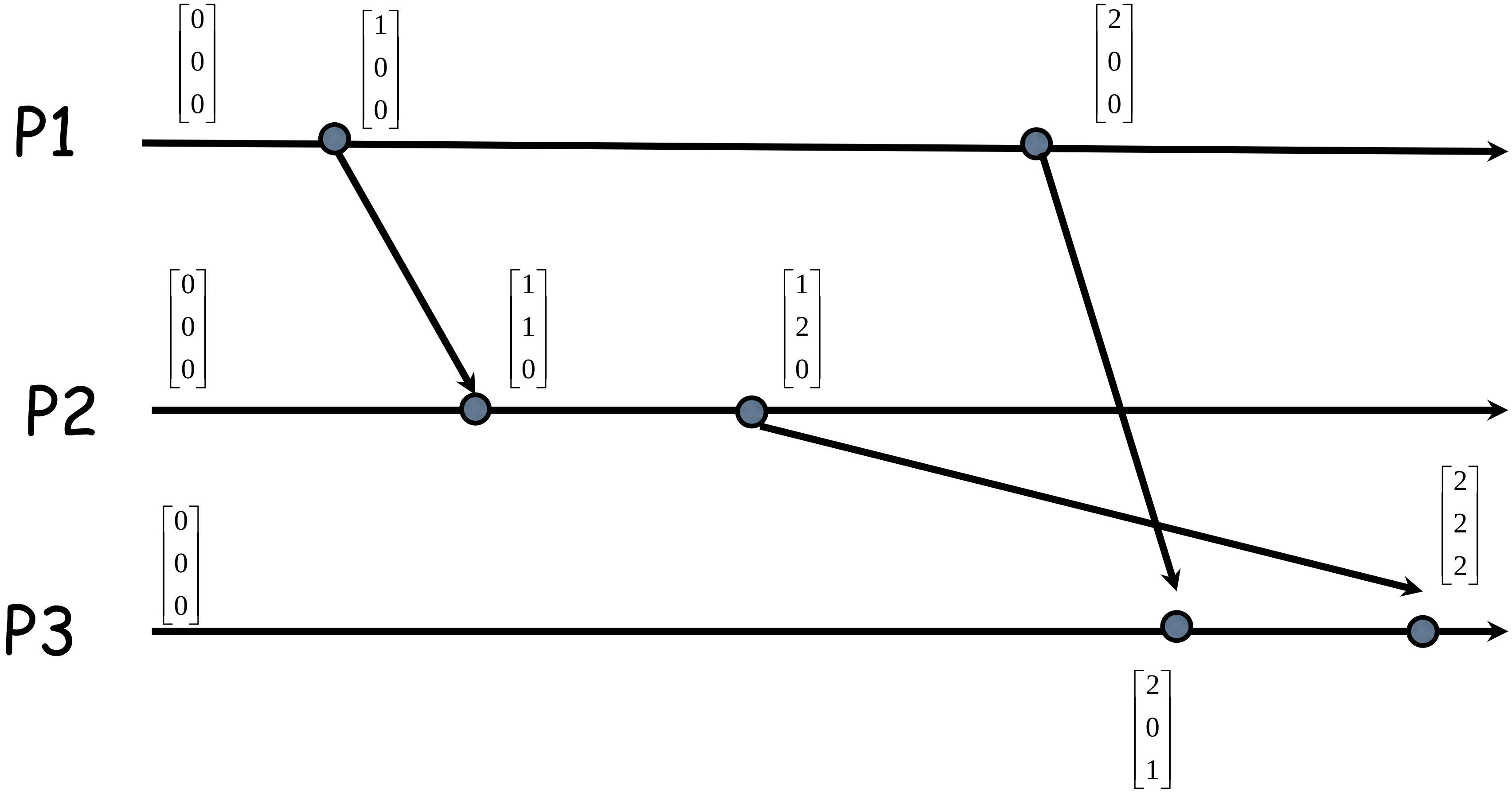
Vector Clock

- R1: Internal events Ci = Ci + d (d>0, typically d=1)

- R2: Piggyback local logical clock on every message, when received do the following:

    - Ci = max(Ci, C value in message)
    - Execute R1 (updates time)
    - Deliver the message


P1
P2
P3
[0 0 0]
[1 0 0]
[2 0 0]
[0 0 0]
[1 1 0]
[1 2 0]
[0 0 0]
[1 2 1]
[2 2 2]



P1
$\begin{bmatrix}0\\0\\0\end{bmatrix}$
$\begin{bmatrix}1\\0\\0\end{bmatrix}$
$\begin{bmatrix}2\\0\\0\end{bmatrix}$
P2
$\begin{bmatrix}0\\0\\0\end{bmatrix}$
$\begin{bmatrix}1\\1\\0\end{bmatrix}$
$\begin{bmatrix}1\\2\\0\end{bmatrix}$
$\begin{bmatrix}2\\2\\2\end{bmatrix}$
P3
$\begin{bmatrix}0\\0\\0\end{bmatrix}$
$\begin{bmatrix}2\\0\\1\end{bmatrix}$

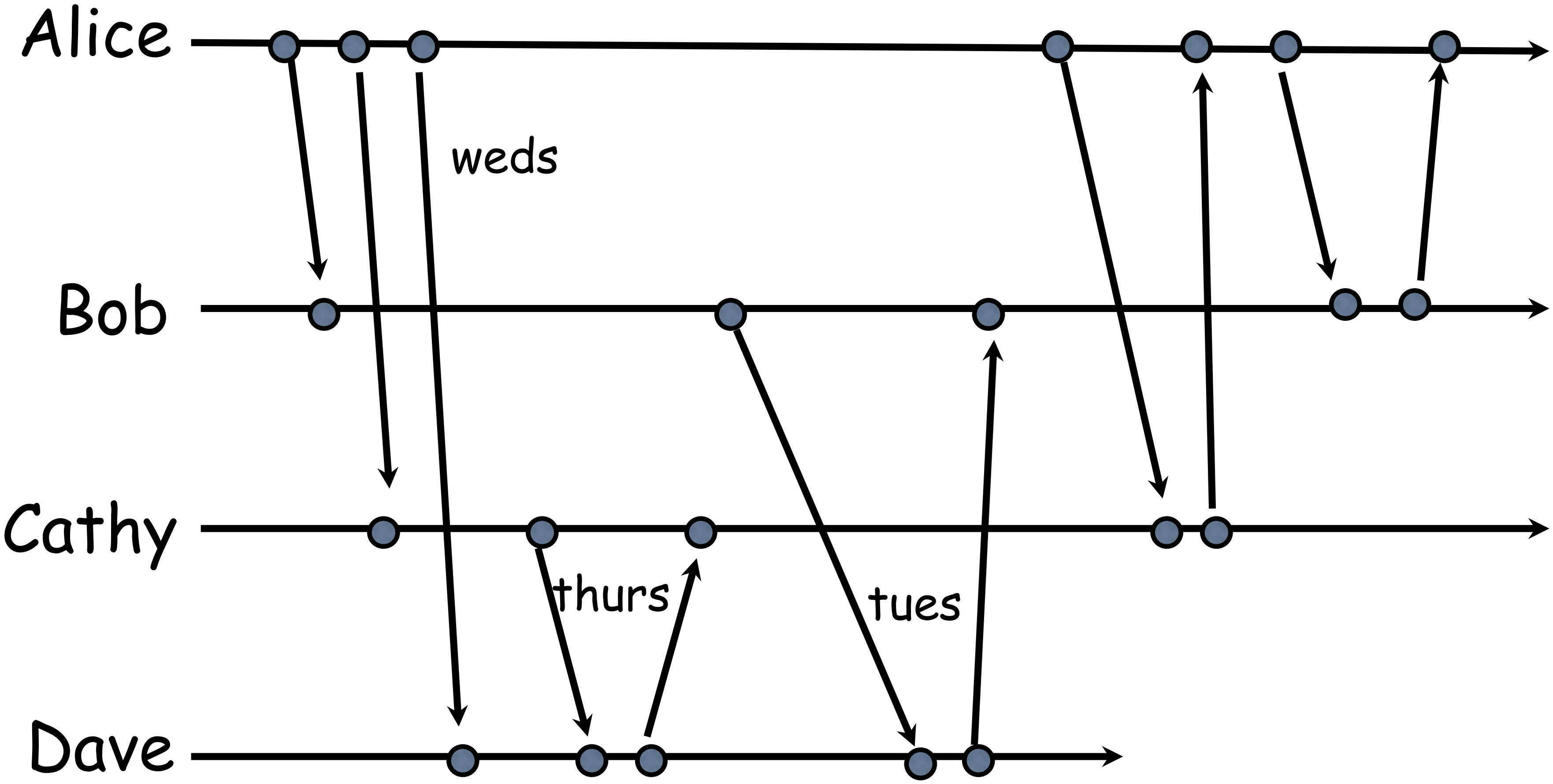
Problem

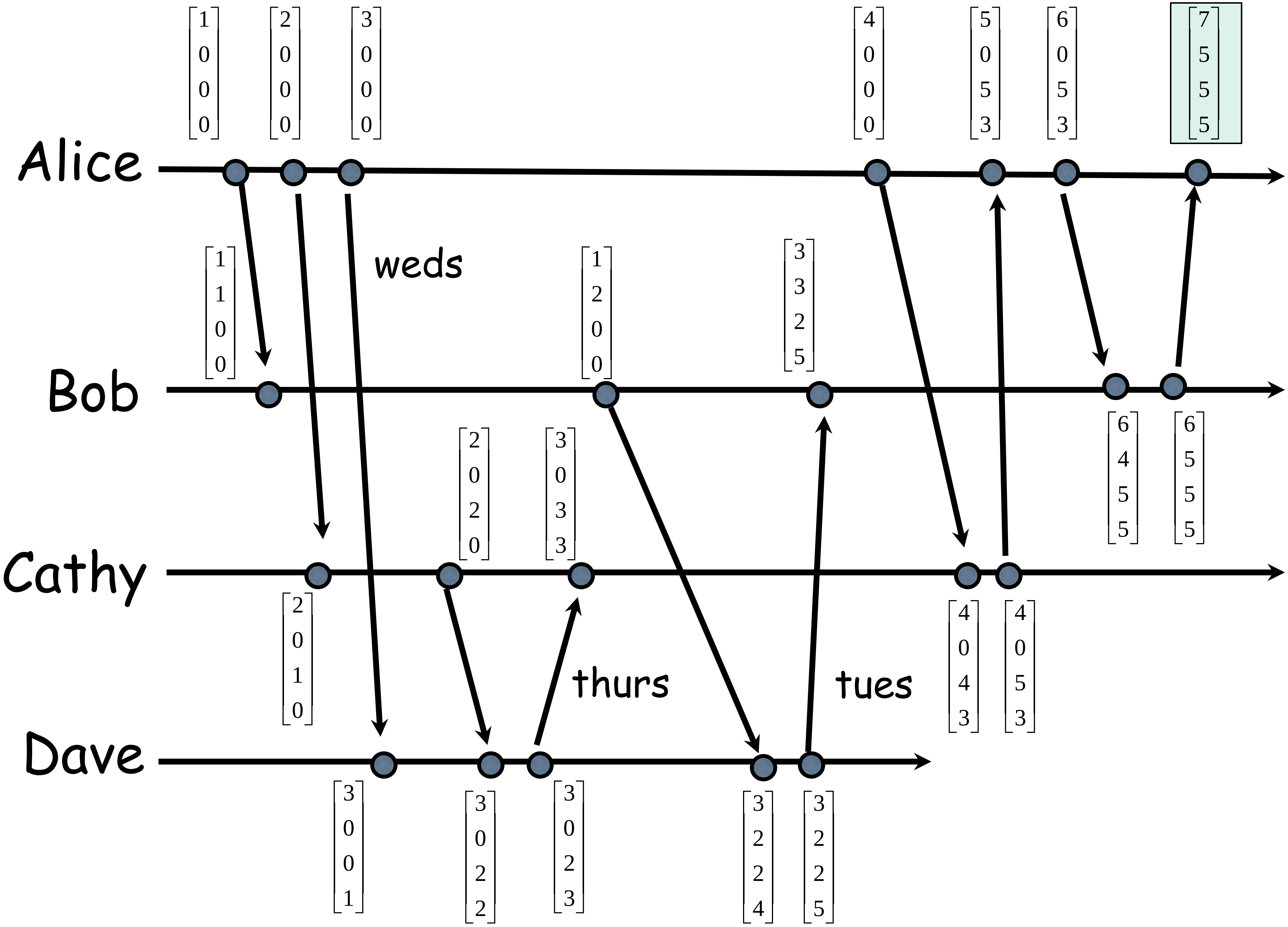
Alice, Ben, Cathy, and Dave are planning to meet next week for dinner. The planning starts with Alice suggesting they meet on Wednesday. Later, Dave discuss alternatives with Cathy, and they decide on Thursday instead. Afterwards, Dave exchanges email with Ben, and Dave they end up deciding on Tuesday. Alice pings everyone again to find out whether they still agree with her Wednesday suggestion. She gets back mixed message from Cathy and Bob and Dave can't be reached.

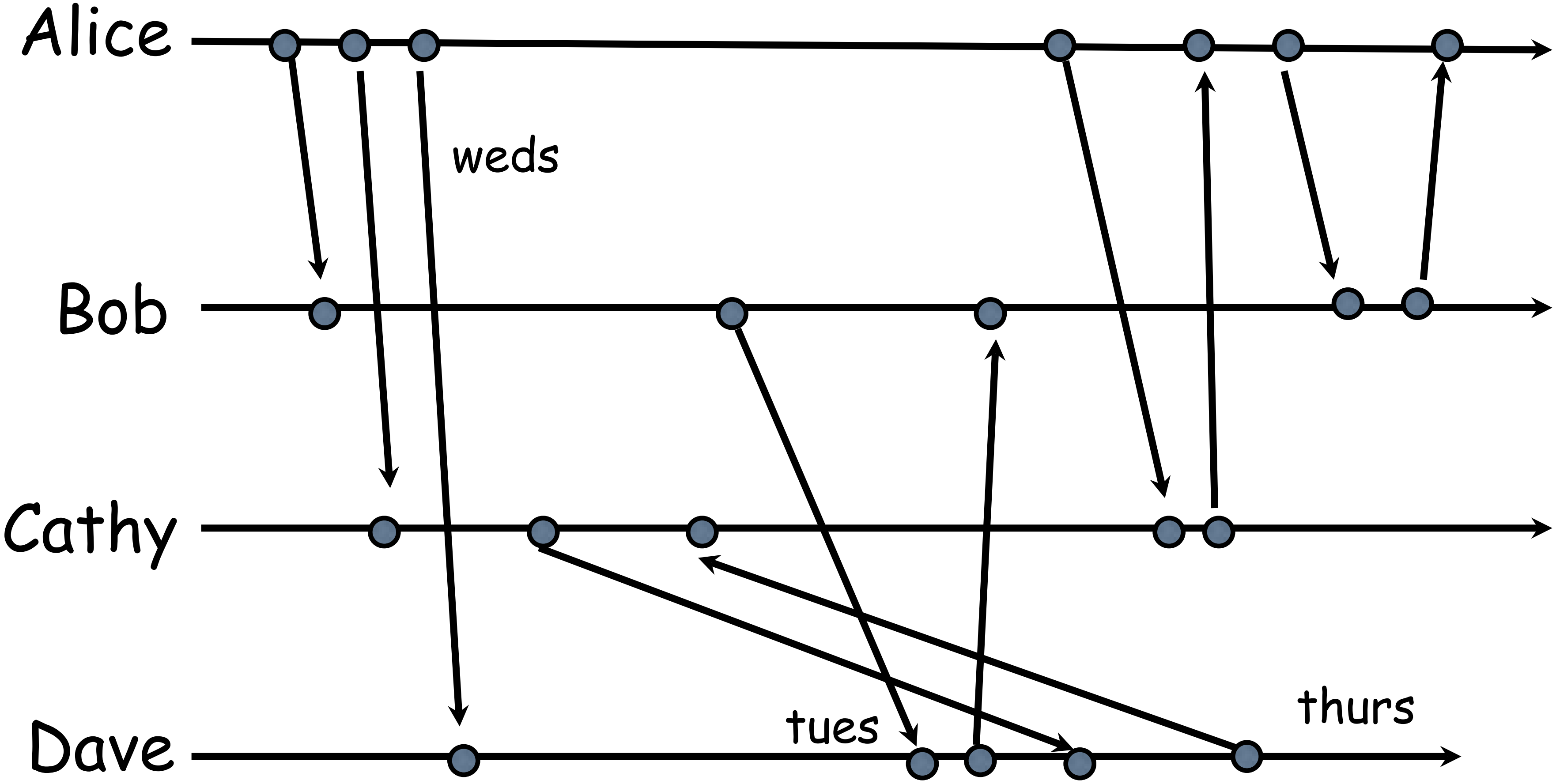
Luckily, Alice, Ben, Cathy and Dave are all taking 416, so they knew to attach their vector clock to every message they sent.

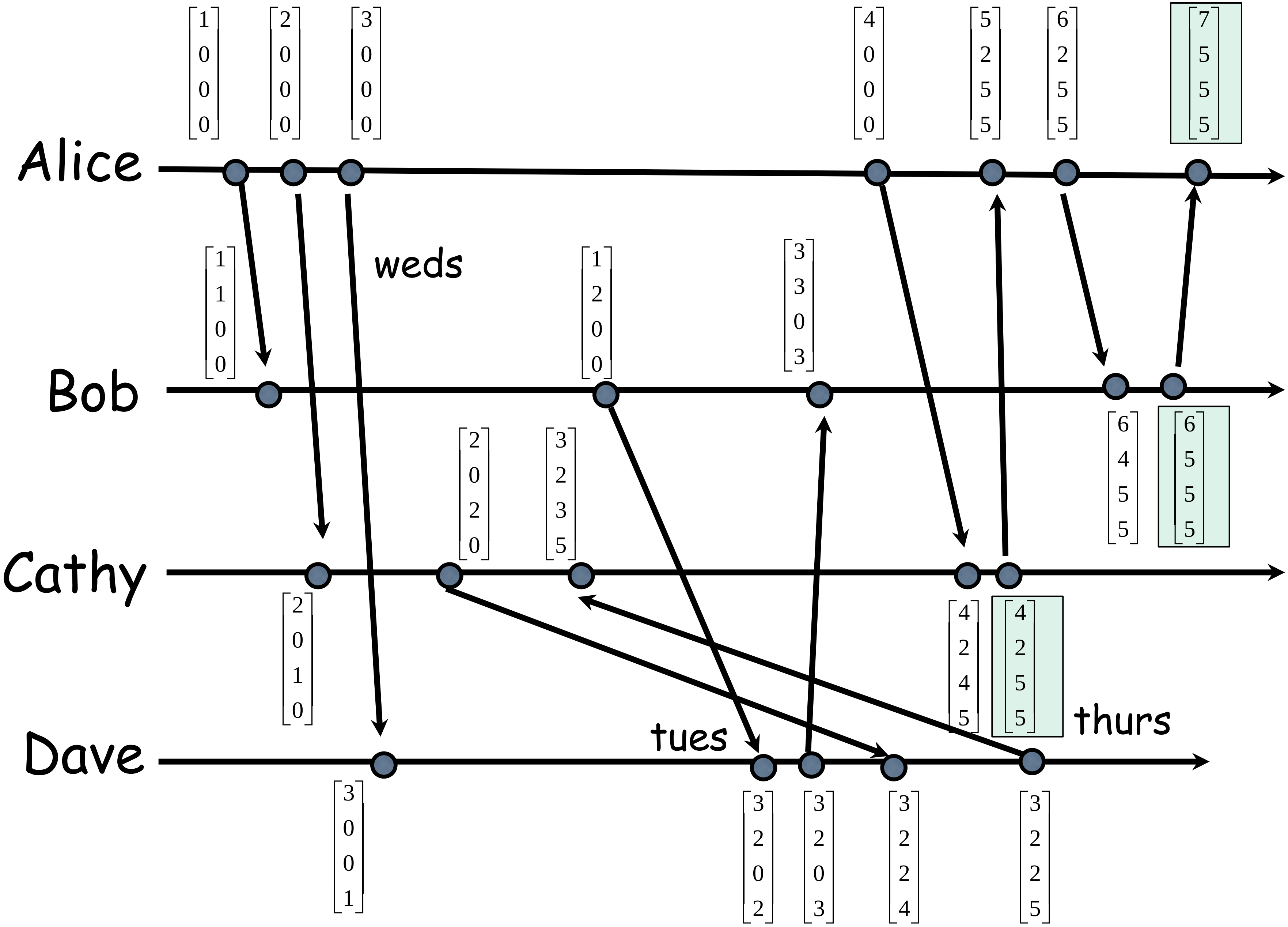
Answer the following questions:

1. What day do they meet?
2. Does Alice need to send another message? Is so, to whom?


Alice
Bob
Cathy
Dave
weds
thurs
tues


Alice
$\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$
$\begin{bmatrix}2\\0\\0\\0\end{bmatrix}$
$\begin{bmatrix}3\\0\\0\\0\end{bmatrix}$
$\begin{bmatrix}4\\0\\0\\0\end{bmatrix}$
$\begin{bmatrix}5\\0\\5\\3\end{bmatrix}$
$\begin{bmatrix}6\\0\\5\\3\end{bmatrix}$
$\boxed{\begin{bmatrix}7\\5\\5\\5\end{bmatrix}}$
weds
Bob
$\begin{bmatrix}1\\1\\0\\0\end{bmatrix}$
$\begin{bmatrix}1\\2\\0\\0\end{bmatrix}$
$\begin{bmatrix}3\\3\\2\\5\end{bmatrix}$
$\begin{bmatrix}6\\4\\5\\5\end{bmatrix}$
$\begin{bmatrix}6\\5\\5\\5\end{bmatrix}$
Cathy
$\begin{bmatrix}2\\0\\2\\0\end{bmatrix}$
$\begin{bmatrix}3\\0\\3\\3\end{bmatrix}$
$\begin{bmatrix}2\\0\\1\\0\end{bmatrix}$
$\begin{bmatrix}4\\0\\4\\3\end{bmatrix}$
$\begin{bmatrix}4\\0\\5\\3\end{bmatrix}$
thurs
tues
Dave
$\begin{bmatrix}3\\0\\0\\1\end{bmatrix}$
$\begin{bmatrix}3\\0\\2\\2\end{bmatrix}$
$\begin{bmatrix}3\\0\\2\\3\end{bmatrix}$
$\begin{bmatrix}3\\2\\2\\4\end{bmatrix}$
$\begin{bmatrix}3\\2\\2\\5\end{bmatrix}$


Alice
Bob
Cathy
Dave
weds
tues
thurs

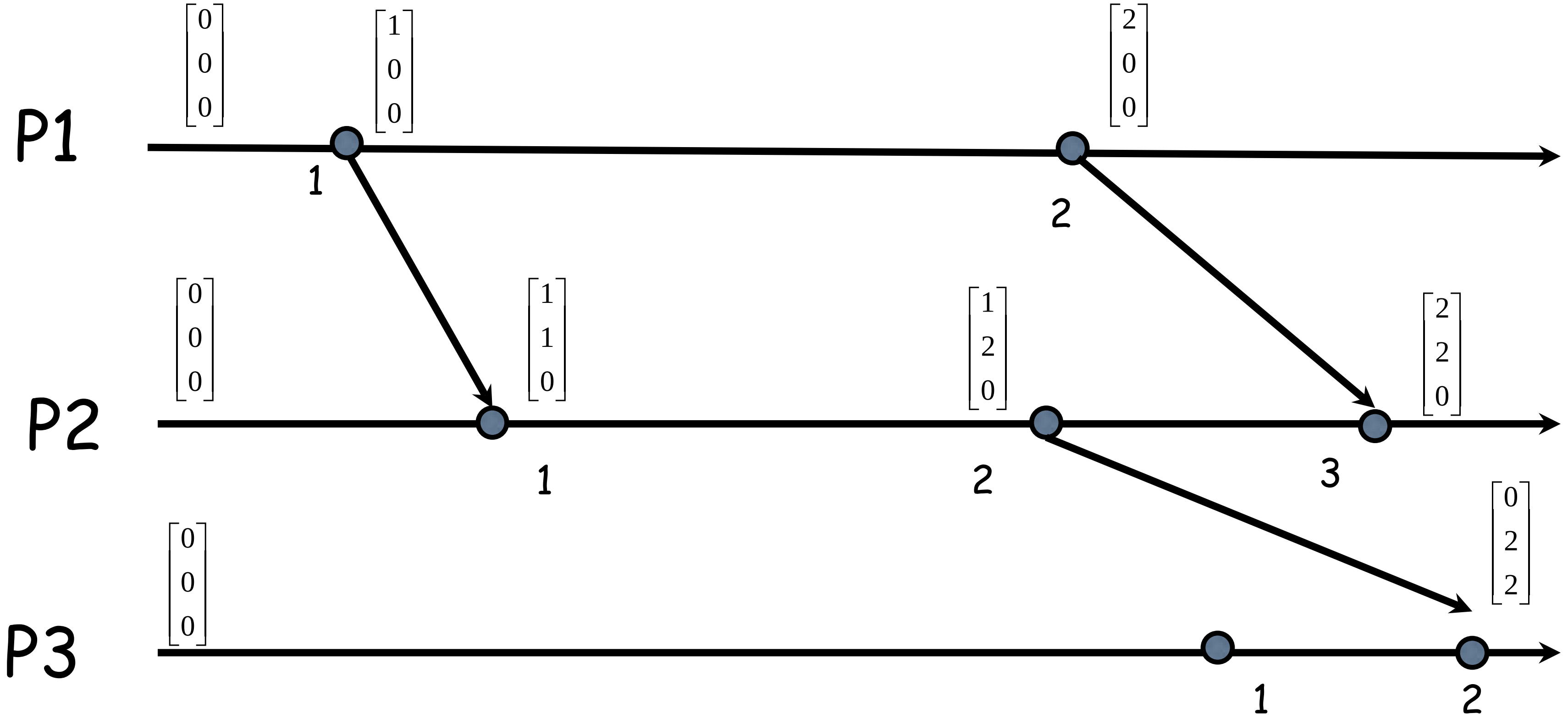

Alice
1 0 0 0
2 0 0 0
3 0 0 0
4 0 0 0
5 2 5 5
6 2 5 5
7 5 5 5
weds
1 1 0 0
1 2 0 0
3 3 0 3
Bob
6 4 5 5
6 5 5 5
2 0 2 0
3 2 3 5
Cathy
2 0 1 0
4 2 4 5
4 2 5 5
thurs
tues
Dave
3 0 0 1
3 2 0 2
3 2 0 3
3 2 2 4
3 2 2 5

Efficient Implementations

- Sending differences
  - m – need to keep previous values
  - m + good for limited communication

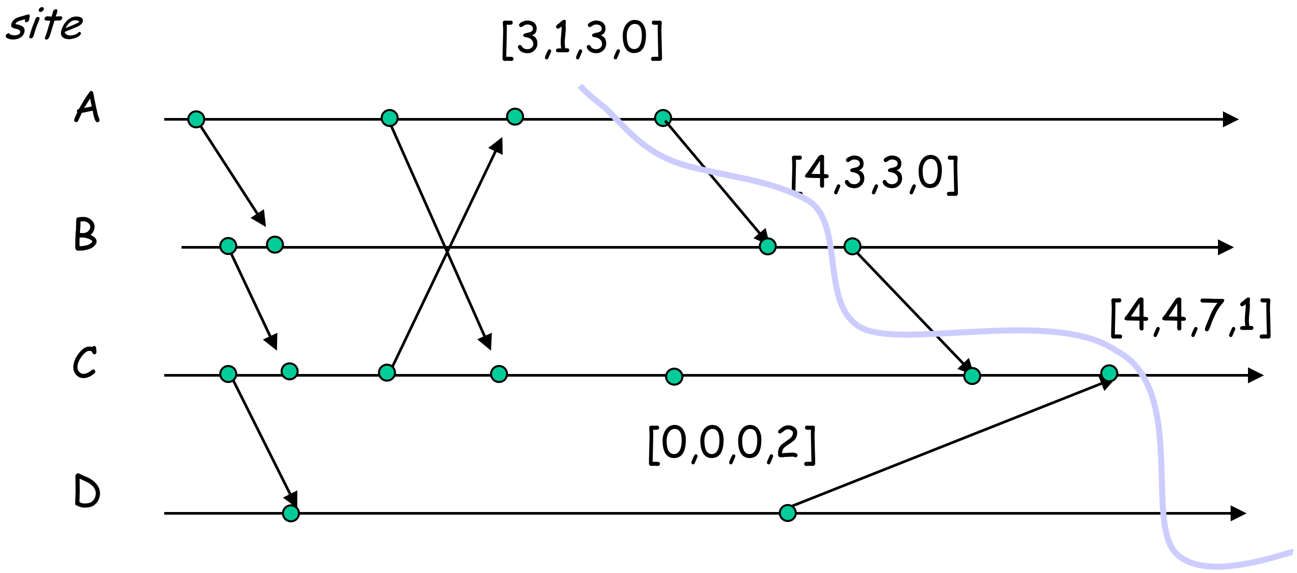
- Remember previous Lamport clock values
  - m + very little information sent
  - m - still need to store O(n)
  - m - no transitivity information

Sending only local clock



Matrix Condition

- Build matrix of each node's vector time
- Matrix captures *a* view of system state



$$\begin{matrix} [3 & 1 & 3 & 0] \\ [4 & 3 & 3 & 0] \\ [4 & 4 & 7 & 1] \\ [0 & 0 & 0 & 2 \end{matrix}$$

Matrix Clock

i. Rule 1

$$mt_i[i,i] = mt_i[i,i] + d \qquad (d > 0)$$

ii. Rule 2

1. Update its global logical time:

2. Update Pi's row with Pj's values

$$1 \le k \le n : mt_i[i,k] = \max(mt_i[i,k], mt[j,k])$$

3. Update the other rows; Deliver the message

$$1 \le k,l \le n : mt_i[k,l] = \max(mt_i[k,l], mt\ [k,l])$$

Matrix Clocks

Process A finding out from B what it knows about C

Note the MAX operation "forgets" information along one of the causal paths