因果有序的多播（Causally Ordered Multicasting）

- **与全序多播的关系**

  因果有序的多播比之前提到的全序多播更弱。尤其是如果两个消息互相没有任何关系，并不关心以那种顺序发送给应用程序。

- **观察**

  使用向量时钟，可以确保所有因果先于某个消息的所有消息接收后才传送这个消息。

  Deliver，将消息从自己的消息队列中取出，传送给上层的应用程序，应用程序会进一步处理这条消息例如进行计算或者显示等，并不是指的通过网络发送消息，参见参考书《分布式系统原理与泛型》第六章，181页，第三段

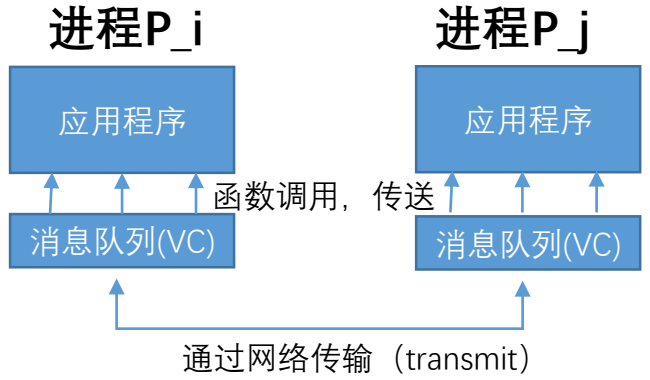
- **调整**

  仅当P_i发送消息时才增加VC[i]，当P_j接收到消息后才调整VC_j；

- **P_j 推迟发送消息 m直到：**

  1. ts(m)[i] = VC_j[i] + 1
  2. ts(m)[k] ≤ VC_j[k]，k ≠ i

问题： *“P_j推迟发送消息直到...”*不准确，P_j并没有足够的信息决定它是否推迟发送消息， 描述算法的时候，应当引入两个进程进行叙述，单单说"P_j推迟发送消息直到"让人困惑.



Adjustment

P_i increments $VC_i[i]$ only when sending a message, and P_j "adjusts" VC_j when receiving a message (i.e., effectively does not change $VC_j[j]$).

P_j postpones delivery of m until:

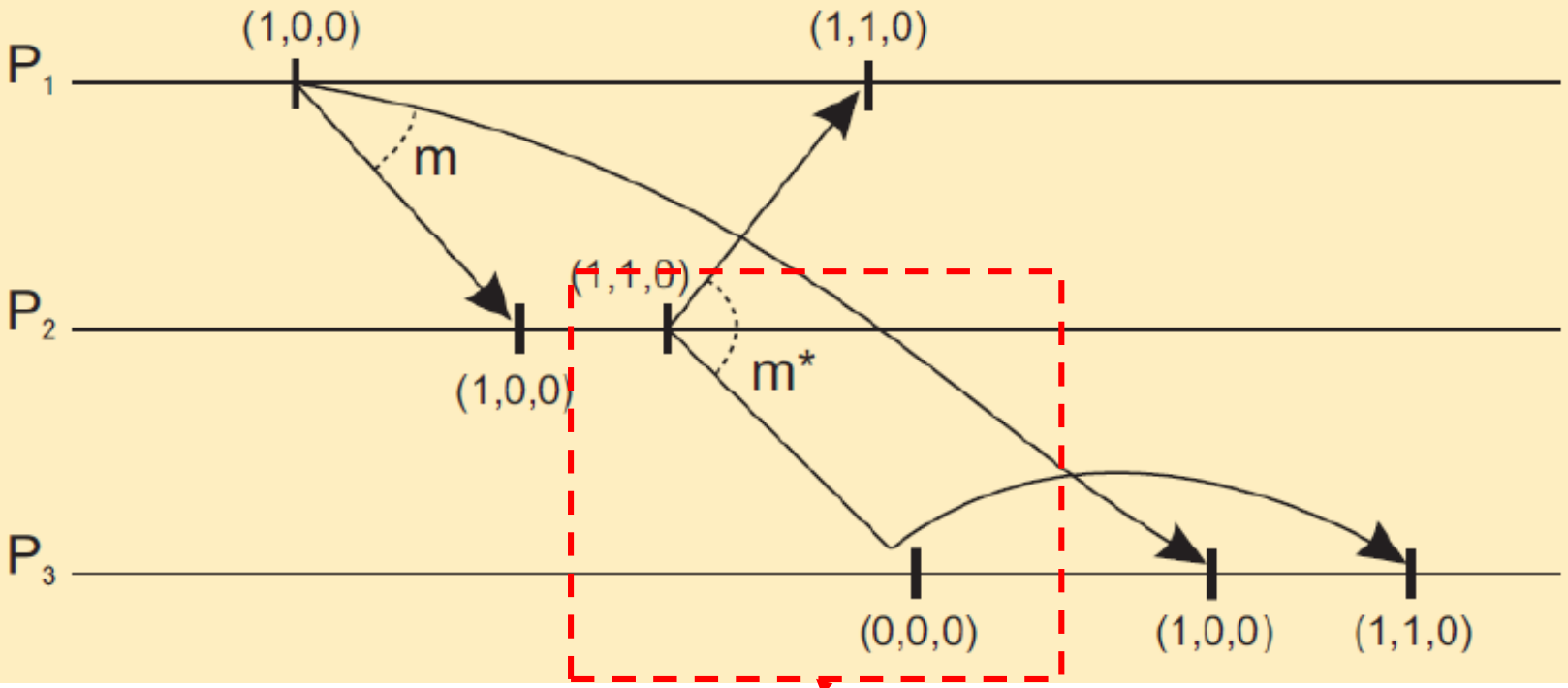
1. $ts(m)[i] = VC_j[i] + 1$
2. $ts(m)[k] \leq VC_j[k]$ for all $k \neq i$

上一页PPT参考了原书作者的PPT，但是翻译上有一些差别，但是这里的delivery是指的向上层应用传递。这个条件约束是发生在两个进程之间的即进程 i和进程 j，而且是进程 i向进程 j发送消息。

也可以理解成Pj接收并传递该消息

因果有序的多播（Causally Ordered Multicasting）

➢ 强制因果有序的通信



在P2和P3交互过程中，根据前一页PPT中的约束条件，p2相当于是Pi，P3相当于是Pj， Pj 是接收消息的进程，当消息(1,1,0)传到的时候，不满足条件2，因此P3进程不能将消息发送给应用程序，先缓存到消息队列中或者拒收等待P2重传，P3进程不会广播。等条件满足的时候P3从消息队列中取出消息传递给应用程序。