分布式系统 HW4

1．当某个节点要使其时钟与另一个节点的时钟同步时，通常，一个较好的想法是还要把以前的度量（偏差）考虑进去。为什么？请给出这样的一个示例。

因为时钟同步不仅需要考虑当前的时钟差异，还需要考虑过去的时钟偏差。如果只考虑当前的时钟差异，可能无法准确地同步两个节点的时钟，因为它们的时钟可能在过去的某个时刻已经存在了较大的偏差。通过考虑以前的度量，可以更准确地估计和调整时钟的偏差，从而实现更精确的时钟同步。

假设有两个节点A和B，它们的时钟在过去的某个时刻发生了较大的偏差。如果只考虑当前的时钟差异，节点A可能会认为它的时钟比节点B慢了1秒钟，而节点B可能会认为它的时钟比节点A快了1秒钟。然而，如果考虑以前的度量，可能会发现节点A在过去的某个时刻比节点B慢了2秒钟，而节点B在过去的某个时刻比节点A快了2秒钟。通过考虑这个过去的度量，可以更准确地调整两个节点的时钟，以实现更精确的时钟同步。

2．分布式系统可能有多个互相独立的资源。假设进程0要访问资源A而进程1要访问资源B。Ricart和Agrawala的算法会导致死锁吗？请解释原因。

Ricart和Agrawala的算法不会导致死锁。该算法是一种分布式互斥算法，用于实现进程对共享资源的互斥访问。在该算法中，每个进程在访问资源之前都会发送请求消息给其他进程，并等待其他进程的回复。只有当进程收到其他进程的回复后，才能访问资源。这种协议确保了互斥性，因为只有一个进程能够同时访问资源。

死锁是指多个进程互相等待对方释放资源而无法继续执行的情况。在Ricart和Agrawala的算法中，进程发送请求消息后等待回复，但不会互相等待对方释放资源。因此，该算法不会导致死锁。

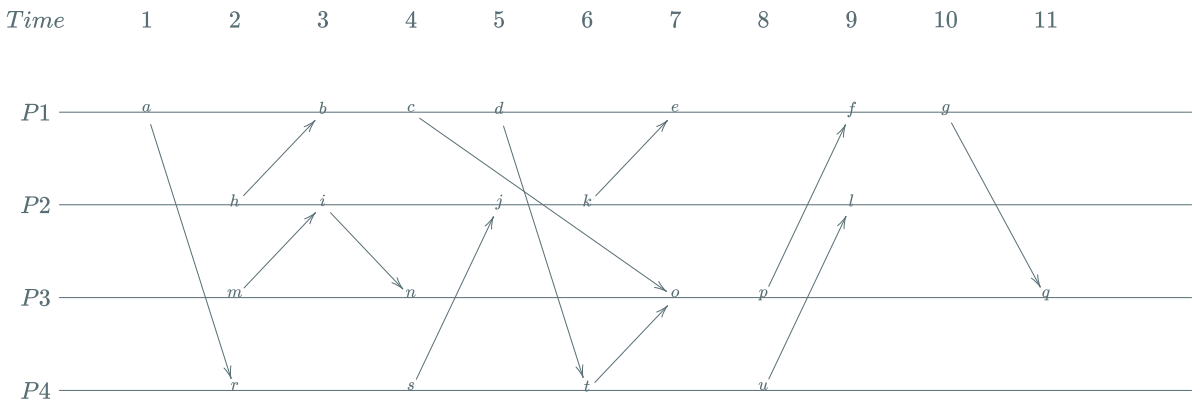
3．假设两个进程同时检测到协作者崩溃，并且它们都使用Bully算法主持一个选举。这时将发生什么？

如果两个进程同时检测到协作者崩溃，并且它们都使用Bully算法来主持选举，那么它们会进行选举来决定新的主节点。

在Bully算法中，每个进程都有一个唯一的标识符，并且较大标识符的进程具有更高的优先级。当一个进程检测到协作者崩溃时，它会发送选举消息给其他进程，并等待回复。如果没有回复，该进程会宣布自己成为新的主节点。如果其他进程回复并宣布自己是更高优先级的进程，那么该进程会放弃选举并承认更高优先级的进程作为新的主节点。

因此，如果两个进程同时检测到协作者崩溃并开始选举，它们会相互发送选举消息，并等待对方的回复。由于其中一个进程具有更高的优先级，它将在选举过程中获胜，并成为新的主节点。另一个进程将放弃选举，并承认更高优先级的进程作为新的主节点。

4．请标出下图中各个事件的逻辑时钟和向量时钟；



Name	Logical Clock
a	1
b	2
c	3
d	4
e	6
f	8
g	9
h	1
i	2
j	4
k	5
l	7
m	1
n	3
o	6
p	7
q	10
r	2
s	3
t	5
u	6

Name	V1	V2	V3	V4
a	1	0	0	0
b	2	1	0	0
c	3	1	0	0
d	4	1	0	0
e	5	4	1	2
f	6	4	4	3
g	7	4	4	3
h	0	1	0	0
i	0	2	1	0
j	1	3	1	2
k	1	4	1	2
l	4	5	1	4
m	0	0	1	0
n	0	2	2	0
o	4	1	3	3
p	4	1	4	3
q	7	6	5	3
r	1	0	0	1
s	1	0	0	2
t	4	1	0	3
u	4	1	0	4

5．互斥的解决方案包括集中式算法、非集中式算法、分布式算法以及令牌算法，请给出不同算法每次进/出需要的消息数，并解释原因。

不同的互斥解决方案包括集中式算法、非集中式算法、分布式算法和令牌算法。每种算法在进入和离开临界区时所需的消息数会有所不同。以下是每种算法的消息数和解释：

1. 集中式算法：
    - 进入：进程发送请求消息给中央服务器，服务器检查是否可以进入临界区，然后发送回复消息给进程。
    - 离开：进程发送释放消息给中央服务器。
      集中式算法每次进入和离开临界区只需要2条消息，一条请求消息和一条回复消息。
2. 非集中式算法（例如Lamport算法）：

- 进入：进程向其他进程发送请求消息，并等待其他进程的回复。
- 离开：进程发送释放消息给其他进程。
  非集中式算法每次进入和离开临界区需要n条消息，其中n是进程的数量。因为进程需要与其他所有进程进行通信，以确保进入和离开的正确顺序。

3. 分布式算法（例如Ricart和Agrawala算法）：
   - 进入：进程向其他进程发送请求消息，并等待其他进程的回复。
   - 离开：进程发送释放消息给其他进程。
     分布式算法每次进入和离开临界区需要n-1条消息，其中n是进程的数量。因为进程只需要与其他n-1个进程进行通信，以确保进入和离开的正确顺序。
4. 令牌算法：
   - 进入：进程等待接收令牌消息，一旦接收到令牌，就可以进入临界区。
   - 离开：进程将令牌消息发送给下一个进程。
     令牌算法每次进入和离开临界区只需要1条消息，即接收或发送令牌消息。令牌算法通过传递一个特殊的令牌来确保进程按顺序进入和离开临界区。

消息数的差异是由于不同算法的设计和通信方式不同。集中式算法和令牌算法只需要少量的消息，因为它们利用中央服务器或令牌来控制进入和离开的顺序。而非集中式算法和分布式算法需要与其他进程进行通信，以协调顺序，因此需要更多的消息。