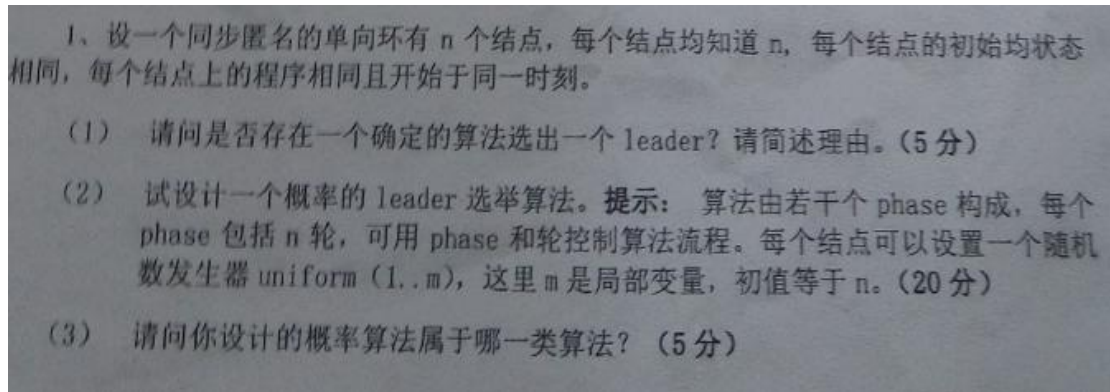


1.



(1)由 Lemma3.1 可得。(同步匿名非均匀)

假设 R 是大小为 $n>1$ 的环 (非均匀)， A 是其上的一个匿名算法，它选中某处理器为 leader。因为环是同步的且只有一种初始配置，故在 R 上 A 只有唯一的合法执行。

Lemma3.1: 在环 R 上算法 A 的容许执行里，对于每轮 k ，所有处理器的状态在第 k 轮结束时是相同的。Note: 每个处理器同时宣布自己是 Leader。

(2)

n 个节点每个节点随机产生一个 $(1-n)$ 的随机数作为自己的 id，然后将它发送，若绕场一周后回到了该节点，该节点就用那个 id;

否则重新产生随机数，直到绕场一周回到该点;

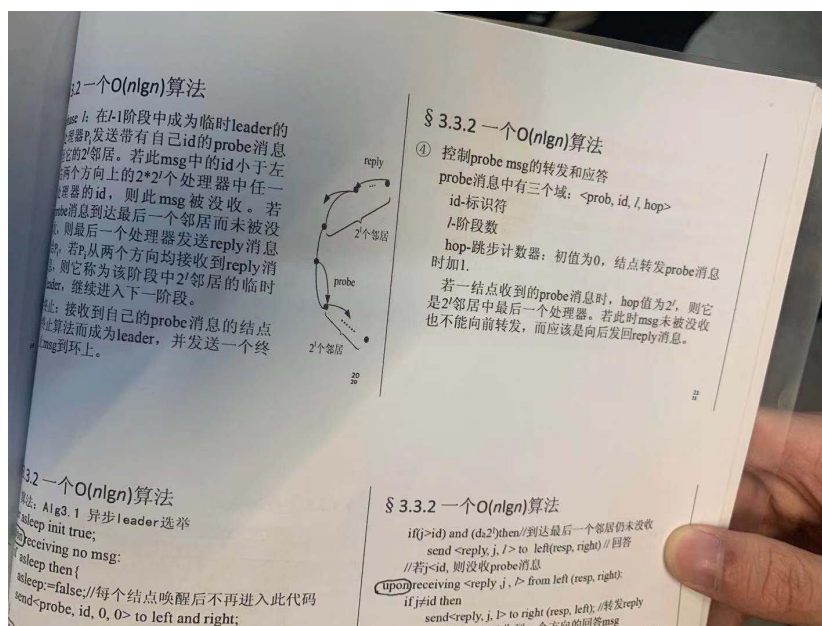
对于每个节点若收到的 msg 中的 id 和自己产生的 id 一样就没收它，反之转发该 msg;

每个节点重复上述过程直至 n 节点的 id 均不相同;

于是得到一个同步非匿名环，可选举一个 id 最大的 leader。

(3)Las Vegas 算法

2.为啥 leader 选举算法选取两边的 2^l 个处理机而不是其他底数



答: 如果用其他数作为底数，如 3，复杂的仍为 $n \log n$ ，不过 \log 的底数是 3。另一方面，如果底数选得过大，有可能造成算法时间接近 n^2 。例如，如果有环有十个处理机，底数也选为 10，那么，prob 消息将有 200 个。

3. 下图的数字为何不是 210 而是 110

除 self 和 k 之外，循环其他节点
是否 $\text{earliest}[k][i] < \text{earliest}[i][i]$ 安全
若 Yes，则此时 block[k] 中的消息（队头）可以发出去了

§ 4.2.3 因果通信

■ 分析
什么样的 msg 是非阻塞的？
当阻塞队列 k（指 blocked[k]）非空时，检测其能否解除阻塞：
设队头 msg 是 m，在 self 上若其他节点 i（除 self 和 k 之外）无更早时戳 $\text{earliest}[i]$ （即比 $m.\text{timestamp} = \text{earliest}[k]$ 更小）的 msg 可能被传递，则消息 m 是非阻塞的。此时，m 可安全传递，m 出队（行 9）。

4) 问题
上述算法可能会发生死锁：若一节点长时间不发送你要的 msg，会发生死锁。
因此，上述因果通信算法通常被用于组播的一部分。

§ 4.2.3 因果通信

■ 算法执行例子（Causal Multicast）
因为协议只对发送事件的因果序感兴趣，故一节点只有发送 msg 时对向量时戳做增量操作。

① 处理接收消息：当 p3 从 p1 接收 m2 时，修改 $\text{earliest}[1]$ 为 (1,1,0)，m2 入阻塞队列 blocked[1]；//line 6, 7

② 处理阻塞：blocked[1] $\neq \emptyset$ ，故 while 循环中 k=1，self=3，i=2， $\text{not_earliest}(\text{earliest}[2], \text{earliest}[1], 2)$ ，前者早于后者，表示 p2 上有一个更早的 msg 还没有达到 p3，返回假。m2 被阻塞。

在 p3 上
 $\text{earliest}[1] \quad \text{earliest}[2] \quad \text{earliest}[3]$
(1,0,0) (0,1,0) (0,0,1)

接收 m2 //k=1, i=2, 阻塞 m2
 $(1,0,0) \geq (0,0,0) \quad (0,0,1)$

清空 delivery-list
block[p] 是否空
Yes
 $\text{earliest}[p] = m.\text{timestamp}$
No
block[p] 是否空
Yes
block[k] 的队头为 m'
No
delivery-list

答：

因为协议只对发送事件的因果序感兴趣，故一节点只有发送 msg 时对向量时戳做增量操作。

因此在发送 m2 之前 p1 时间戳为 010，发送时变为 110。

4. 同步环问题。算法 3.2. 下图

为何非唤醒 msg 要延迟 $2^i - 1$ 轮？

如何修改算法 3.2 来改善时间复杂性？

答：

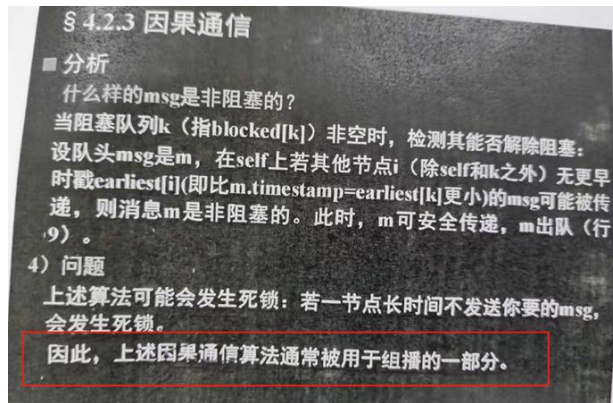
(1) 降低消息复杂度 (id 最小的节点被选举为 Leader，Leader 节点消息的转发速度最快，则可以使得其他非 leader 的转发 msg 被淹没)。

(2) 方案 1：添加 Relay 变量，保证消息在转发节点不延迟，时间复杂度由 $O(n \cdot 2^i)$ 降为 $O(N \cdot 2^i + n \cdot N)$ ，N 为自发唤醒的节点数。

方案 2：原算法延迟函数为 $f(id) = 2^{id}$ ，时间复杂度为 $O(n \cdot 2^i)$ 。通过重新定义延迟函数来降低时间复杂度，如 $f(id) = c \cdot id$ 等。

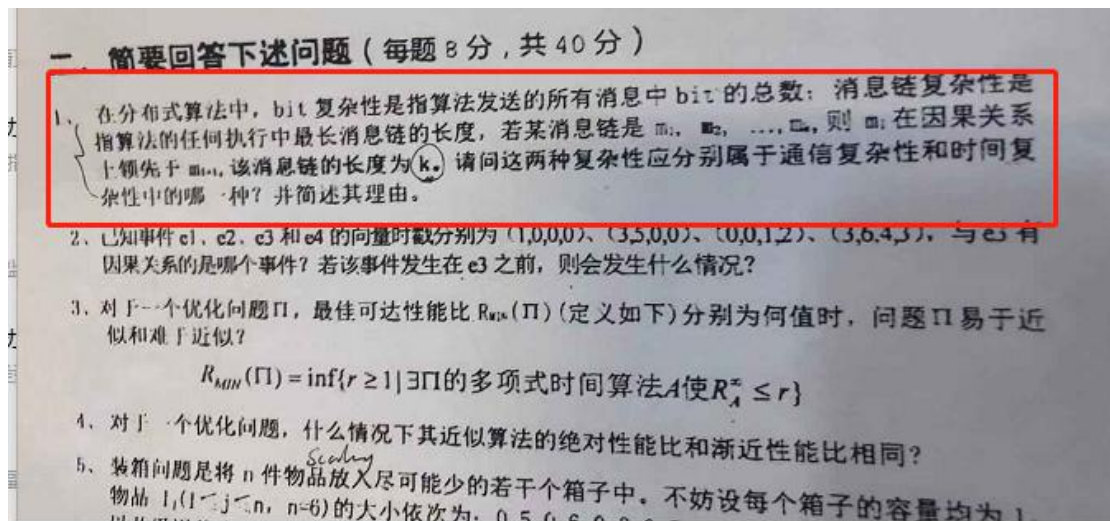
Note：思考方案 2 中消息复杂度和时间复杂度的关系！

5. 为啥被用于组播?



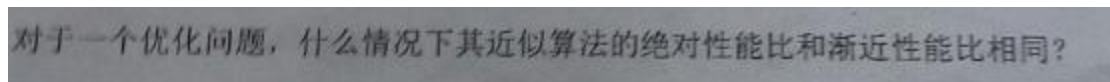
答: 因为本地节点的初始化在本地节点完成, 有处理阻塞的机制。

6.



bit 复杂性属于通信复杂性, 消息链复杂性属于时间复杂性; 若在一个分布式算法中每个 msg 信息的 bit 数目相同, 则 msg 的个数就等于 bit 的总数除以一个 msg 的 bit 数目, 则 bit 复杂性可以等价为 msg 复杂性; 消息链复杂性是最长消息链的长度, 在同步系统中它就是最大轮数, 异步系统中假定任何执行的 msg 延迟至多是一个单位时间, 它就是计算直到终止时间的最大运行时间, 在同, 异步系统中皆为时间复杂性。

7



答:

具有 Scaling 性质的问题, 近似算法的绝对性能比和渐近性能比是相同的。