(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 104639273 A
(43)申请公布日 2015.05.20

(21)申请号 201310558216.5

(22)申请日 2013.11.08

(71)申请人 沈阳高精数控技术有限公司
地址 110168 辽宁省沈阳市东陵区南屏东路16号

(72)发明人 林浒 杨磊 郑飚默 韩旭 韩文业

(74)专利代理机构 沈阳科苑专利商标代理有限公司 21002
代理人 许宗富

(51)Int.Cl.
H04J 3/06(2006.01)

权利要求书2页 说明书5页 附图2页

(54)发明名称

一种适用于LAN网络下通信设备的时间同步方法

(57)摘要

本方法提出了一种适用于LAN网络下通信设备的时间同步方法，该方法以优化设计的中间层同步模型为基础，包含不同阶段的优化方法，对所在环境下的节点时钟进行同步。该中间层模型是同步方法的关键要素，它对系统内线程进行不同的功能划分并通过合理的线程规划，使初次同步与周期性的再同步过程能够顺畅地完成，并将同步延迟限制在合理的范围内。

CN 104639273 A

1. 一种适用于 LAN 网络下通信设备的时间同步方法，其特征在于，在通信网络和各节点之间构建一个中间层，用于同步相关线程的管理，包括以下步骤：

在初次同步之前，指定一个节点为主节点，其余节点参照该节点的时钟来进行同步；

每一个从节点在启动初次同步之前在主节点中注册；

在所有节点注册完毕后，主节点启动初次时间同步；

经过了一轮双向通信后，初次时间同步的结束时间会被记录在主节点的配置文件中，开始对与后续再同步之间的间隔时段计时；

当从节点启动中间层模型后，发送注册消息到主节点，主节点在配置文件中记录下各节点的注册信息，并选定一个从节点进行一轮双向通信以测量当前环境下的平均延迟 T_{delay}；

主节点向注册节点发送反馈消息，该消息包含了单向延迟 $T_{delay}/2$，同时主节点挂起 $T_{delay}/2$ 的时间；

从节点收到反馈消息后，启动主系统线程开始同步。

2. 按权利要求 1 所述的适用于 LAN 网络下通信设备的时间同步方法，其特征在于：所述从节点的时钟状态信息在同步过程中被记录下来，保存在主节点的配置文件中。

3. 按权利要求 1 所述的适用于 LAN 网络下通信设备的时间同步方法，其特征在于：所述时间同步过程，滞后节点在重新启动后向主节点发送注册消息；主节点在注册阶段以外的时间接收到某从节点的注册消息时，会向该节点反馈一个包含主节点时钟状态信息及单向通信延迟 $T_{delay}/2$ 的消息；等待中的滞后节点收到该消息后，根据主节点时钟与延迟修正自己的时钟并按再同步周期启动计时线程等待至下一次再同步。

4. 按权利要求 1 所述的适用于 LAN 网络下通信设备的时间同步方法，其特征在于：所述中间层中的每一个系统线程启动后都按次序分配一个时间节片，通过调整所述时间节片的顺序，保证在任意时刻只会有一个节点中的通信线程处于运行状态。

5. 按权利要求 1 所述的适用于 LAN 网络下通信设备的时间同步方法，其特征在于：还包括对节点内部时钟累积的相对偏移量进行修正，包括以下步骤：

假设当前再同步将于时刻 $T_{i,k}$ 开始，表示节点 k 中的第 i 次再同步，所有节点，包括主节点，都将于时刻 $T_{i,k}-t_{Dev}-t_{resp}$ 进入再同步状态，其中 t_{Dev} 表示主节点中所记录的与从节点时钟的最大偏移量，t_{resp} 而则是从节点中系统进入同步状态的响应时间，则在时刻 $T_{i,k}$ 到来之前，所有节点都已准备就绪；

启动再同步任务，主节点广播包含当前时钟状态信息的再同步消息，持续了 $T_{delay}/2$ 后进入监听状态，等待从节点的反馈消息；同时，从节点开始对监听状态计时，如果经过 $2t_{Dev}+t_{resp}+T_{delay}/2$ 时间之后还没有收到再同步消息，则进行超时处理；如果接收到再同步消息，则从节点读取其中的主节点时钟状态信息并对本地时钟做偏移量的修正。

6. 按权利要求 5 所述的适用于 LAN 网络下通信设备的时间同步方法，其特征在于：对本地时钟做偏移量的修正，具体为：

如果本地时钟落后于主节点时钟，则进行时钟优化修正；

如果本地时钟不落后于主节点时钟，则将所有正在运行的进程挂起一段时间，这段时间等于计算得到的时钟偏移量，然后再将任务恢复。

7. 按权利要求 6 所述的适用于 LAN 网络下通信设备的时间同步方法，其特征在于：所

述时钟优化修正，具体为：

将时钟回拨到 $-t_{Dev}/2$，也就是比标准时钟慢 $t_{Dev}/2$ 的时间，则时钟被回拨了 $3/2t_{Dev}$。

一种适用于 LAN 网络下通信设备的时间同步方法

技术领域

[0001] 本发明涉及通信技术领域，具体的说是一种适用于 LAN 网络下通信设备的时间同步方法

背景技术

[0002] 在实时网络环境下，通常对同步方法的精度要求非常高。通常情况下，同步过程都是通过网络节点内部的本地时钟与专门的时钟服务器，如 UTC（Universal Time Coordinate）等时钟源，之间的通信实现。为了达到更高的精度，如 GPS 接收器等辅助硬件会被安装在节点上，从而实现时间信息的及时发送与接收，同步完成后的时钟偏差能够被控制在毫秒级。但在很多情况下，因为昂贵的费用，硬件设备很难被应用在网络中的各个节点上。各节点可以通过与被指定的相对标准时钟交换时间信息，这种同步方式被称之为内部同步。

[0003] 同时各个节点内的时钟频率也不可避免地有差异存在，即便上轮同步已将节点间的时钟偏差修正，在接下来的一段时间内，误差还会不断地累积，由此可见，周期性的再同步是必不可少的，在不借助硬件的基础上，将网络间的时钟误差控制在合理的范围内。

[0004] 针对不同网络环境下的时间同步，已经有了广泛的研究与应用，包括在小规模的工业现场，以及大型的本地互联网。但随着网络环境的日趋复杂，一些功能上的不足也逐渐暴露出来。比如应用最为广泛的同步协议 NTP(Network Time Protocol)，虽然能够满足互联网间多方的同步需求，但在实时性较高的分布式环境下，同步效果不能令人满意。NTP 比较适用于精度要求不高的软实时网络，对节点间时钟误差的控制并不严格，不能完全适用于 LAN 环境下的设备间时间同步要求。

发明内容

[0005] 针对现有技术中存在的上述问题，本发明提供一种适用于 LAN 网络下通信设备的时间同步方法。

[0006] 本发明为实现上述目的所采用的技术方案是：一种适用于 LAN 网络下通信设备的时间同步方法，在通信网络和各节点之间构建一个中间层，用于同步相关线程的管理，包括以下步骤：

[0007] 在初次同步之前，指定一个节点为主节点，其余节点参照该节点的时钟来进行同步；

[0008] 每一个从节点在启动初次同步之前在主节点中注册；

[0009] 在所有节点注册完毕后，主节点启动初次时间同步；

[0010] 经过了一轮双向通信后，初次时间同步的结束时间会被记录在主节点的配置文件中，开始对与后续再同步之间的间隔时段计时；

[0011] 当从节点启动中间层模型后，发送注册消息到主节点，主节点在配置文件中记录下各节点的注册信息，并选定一个从节点进行一轮双向通信以测量当前环境下的平均延迟

T_{delay}；

[0012] 主节点向注册节点发送反馈消息，该消息包含了单向延迟 $T_{delay}/2$，同时主节点挂起 $T_{delay}/2$ 的时间；

[0013] 从节点收到反馈消息后，启动主系统线程开始同步。

[0014] 所述从节点的时钟状态信息在同步过程中被记录下来，保存在主节点的配置文件中。

[0015] 所述时间同步过程，滞后节点在重新启动后向主节点发送注册消息；主节点在注册阶段以外的时间接收到某从节点的注册消息时，会向该节点反馈一个包含主节点时钟状态信息及单向通信延迟 $T_{delay}/2$ 的消息；等待中的滞后节点收到该消息后，根据主节点时钟与延迟修正自己的时钟并按再同步周期启动计时线程等待至下一次再同步。

[0016] 所述中间层中的每一个系统线程启动后都按次序分配一个时间节片，通过调整所述时间节片的顺序，保证在任意时刻只会有一个节点中的通信线程处于运行状态。

[0017] 还包括对节点内部时钟累积的相对偏移量进行修正，包括以下步骤：

[0018] 假设当前再同步将于时刻 $T_{i,k}$ 开始，表示节点 k 中的第 i 次再同步，所有节点，包括主节点，都将于时刻 $T_{i,k}-t_{Dev}-t_{resp}$ 进入再同步状态，其中 t_{Dev} 表示主节点中所记录的与从节点时钟的最大偏移量，t_{resp} 而则是从节点中系统进入同步状态的响应时间，则在时刻 $T_{i,k}$ 到来之前，所有节点都已准备就绪；

[0019] 启动再同步任务，主节点广播包含当前时钟状态信息的再同步消息，持续了 $T_{delay}/2$ 后进入监听状态，等待从节点的反馈消息；同时，从节点开始对监听状态计时，如果经过 $2t_{Dev}+t_{resp}+T_{delay}/2$ 时间之后还没有收到再同步消息，则进行超时处理；如果接收到再同步消息，则从节点读取其中的主节点时钟状态信息并对本地时钟做偏移量的修正。

[0020] 所述对本地时钟做偏移量的修正，具体为：

[0021] 如果本地时钟落后于主节点时钟，则进行时钟优化修正；

[0022] 如果本地时钟不落后于主节点时钟，则将所有正在运行的进程挂起一段时间，这段时间等于计算得到的时钟偏移量，然后再将任务恢复。

[0023] 所述时钟优化修正，具体为：

[0024] 将时钟回拨到 $-t_{Dev}/2$，也就是比标准时钟慢 $t_{Dev}/2$ 的时间，则时钟被回拨了 $3/2t_{Dev}$。

[0025] 本发明具有以下有益效果及优点：

[0026] 1. 精确度高。在本发明中，LAN 环境中的设备经过初次同步，再同步及后续的偏移量修正，使得网络环境中的所有设备保持精确的时钟一致。

[0027] 2. 实时性强。本发明中以优化设计的中间层同步模型为基础，通过重新设计线程结构，对线程进行不同的功能划分以及合理的时序规划，在系统底层完成了设备同步过程，并将同步延迟限制在合理的范围内，使得该网络环境的实时性得到了保证。

附图说明

[0028] 图 1 为本发明方法应用的中间层模型的基本结构图；

[0029] 图 2 为本发明方法应用的内部线程结构图；

[0030] 图 3 为本发明方法中节点线程访问序列示意图；

[0031] 图 4 为本发明方法中节点间再同步流程示意图；

[0032] 图 5 为本发明方法中再同步中的时钟偏移量修正示意图。

具体实施方式

[0033] 本发明构建了一个实时中间层，用于管理同步相关线程(用于启动所对应的同步功能模块)，并对调用各个方法的线程做合适的规划以维护系统的一致性。与此同时，中间层模型还负责处理节点时钟所提供的详细的时间信号，用于各个实时任务的启动与终止，为当前 LAN 环境中的设备提供精确的时钟同步功能。

[0034] 该中间层维护了一种线程结构，该结构保持了进程间的独立性以及不同功能的模块化设计，在方法执行时便于进行相关的可视化分析。该结构主要包含两种类型的线程，应用线程和系统线程。应用线程执行应用调用的各个方法(包括事件驱动和时间驱动)，系统线程则是 RT 内核通过实时中断周期性的启动，另有一部分系统线程是随模型一块启动。

[0035] 系统线程包括(1)计时线程：周期性线程，负责对该层次中的其它线程按计时顺序进行调度并查验是否存在超出运行时限的线程存在，当线程切换时便会调用计时线程并赋予相应的时间片，节点的系统在启动时便会生成计时线程，这个时间甚至早于中间层模型的启动；(2)通信进程：管理来自通信网络中的消息收发，并调用其它相关线程；(3)本地 I/O 线程：负责本地读写操作，如缓存区读写和磁盘读写。在该线程的帮助下，能够很好地对 I/O 操作以外的程序运行进行准确快速的开销分析，并且便于 I/O 操作的管理；(4)主系统线程：该线程主要用于管理时间分片，包括对占用时间分片的实时任务的调度规划以及空闲时间分片的有序回收。在计时线程启动并开始任务管理之后，主系统线程会为其分配空闲的时间分片，对于其他提出请求的实时线程也同样如此。因此从概念上来说，主任务线程是具有代表性的管理其他线程的主线程。

[0036] 在由计时线程调用的方法中，设定了一种适用于 LAN 环境的访问序列来调整节点通信时的线程调用次序，提升线程的工作效率。在该序列中，各节点在启动了中间层模型后，才会由通信线程向网络发送消息。中间层中的每一个系统线程启动后都按次序分配一个时间节片，通过合理地调整分配给通信线程的时间节片的顺序，能够保证在任意时刻，只会有一个节点中的通信线程处于运行状态。

[0037] 在该方法设定了基于中间层模型的初次同步机制，在初次同步之前，会有一个节点被指定为主节点，其余节点会参照该节点的时钟来进行同步。这些从节点的时钟状态信息会在同步过程中会被记录下来，保存在主节点的配置文件中。在指定了主节点后，每一个从节点都需要在启动初次同步之前在主节点中注册本机。在所有节点注册完毕后，主节点则会启动初次时间同步。经过了一轮双向通信后，初次同步的结束时间会被记录在主节点的配置文件中，开始对与后续再同步之间的间隔时段计时。当从节点启动中间层模型后，会发送注册消息到主节点，主节点在配置文件中记录下各节点的注册信息，并选定一个从节点进行一轮双向通信以测量当前环境下的平均延迟 T_{delay}。之后主节点会向注册节点发送反馈消息，该消息包含了单向延迟 $T_{delay}/2$，同时主节点会挂起 $T_{delay}/2$ 的时间以保证各从节点尽可能地收到反馈消息。从节点收到反馈消息后，启动主系统线程开始同步。为了能够重新加入到同步过程中来，滞后节点在重新启动后需要向主节点发送注册消息。主节点在注册阶段以外的时间接收到某从节点的注册消息时，会向该节点反馈一个包含主节点时钟状

态信息及单向通信延迟 $T_{delay}/2$ 的消息。等待中的滞后节点收到该消息后，根据主节点时钟与延迟修正自己的时钟并按再同步周期启动计时线程等待至下一次再同步，至此，滞后节点便能够重新加入到同步过程中来。

[0038] 本发明设计了基于中间层模型的时钟再同步方法，对节点内部时钟累积的相对偏移量进行修正。假设当前再同步将于时刻 $T_{i,k}$ 开始，表示节点 k 中的第 i 次再同步，所有节点，包括主节点，都将于时刻 $T_{i,k}-t_{Dev}-t_{resp}$ 进入再同步状态，其中 t_{Dev} 表示主节点中所记录的与从节点时钟的最大偏移量，t_{resp} 而则是从节点中系统进入同步状态的响应时间，默认为一个很小的常量，则在时刻 $T_{i,k}$ 到来之前，所有节点都已准备就绪。之后，由计时线程调用相关线程启动再同步任务，主节点广播包含当前时钟状态信息的再同步消息，持续了 $T_{delay}/2$(单向通信延迟)后进入监听状态，等待从节点的反馈消息。同时，从节点开始对监听状态计时，如果经过 $2t_{Dev}+t_{resp}+T_{delay}/2$ 时间之后还没有收到再同步消息，则进行超时处理。

[0039] 接收到再同步消息之后，从节点读取其中的主节点时钟状态信息并对本地时钟做偏移量的修正。修正方式根据时钟偏移的不同分为两种情况 ：如果本地时钟落后于主节点时钟，只需要把时钟往前拨即可 ；如果比主节点时钟要快，为了防止计时任务的重复运行，不能简单地把时钟回拨，需要将所有正在运行的进程挂起一段时间，这段时间等同于计算得到的时钟偏移量，然后再将任务恢复。

[0040] 针对普通修正部分，偏移量 t_{Dev} 在当前的再同步过程中得到修正，但由时钟频率所决定的，在下一次再同步之前，偏移量还是会逐渐恢复到 t_{Dev} 附近，这就意味着时差始终保持在 t_{Dev} 左右。更理想的优化修正部分，是将时钟回拨到 $-t_{Dev}/2$，也就是比标准时钟慢 $t_{Dev}/2$ 的时间。在这种情况下，时钟被回拨了 $3/2t_{Dev}$ 的时间而不是 t_{Dev}。这对后续的再同步来说是一项不错的改进，之后从节点时钟偏移量会在 $-t_{Dev}/2$ 到 $t_{Dev}/2$ 之间浮动，主从节点间的时差将会维持在 $t_{Dev}/2$ 的范围内。

[0041] 如图 1 所示，为本发明方法所应用的中间层模型的基本结构。该结构基于 RT Linux 构建而成并带有相应的扩展接口，负责功能模块的调用以及上下层的数据交互。

[0042] 如图 2 所示，本发明方法中计时线程负责对该层次中的其它线程按计时顺序进行调度并查验是否存在超出运行时限的线程存在，当线程切换时便会调用计时线程并赋予相应的时间片 ；通信进程管理来自通信网络中的消息收发，并调用其它相关线程 ；本地 I/O 线程负责本地读写操作，如缓存区读写和磁盘读写，在该线程的帮助下，能够很好地对 I/O 操作以外的程序运行进行准确快速的开销分析，并且便于 I/O 操作的管理 ；主系统线程用于管理时间分片，包括对占用时间分片的实时任务的调度规划以及空闲时间分片的有序回收。

[0043] 如图 3 所示，本发明方法中各节点在启动了中间层模型后，才会由通信线程向网络发送消息。如前文所言，中间层中的每一个系统线程启动后都按次序分配一个时间节片，通过合理地调整分配给通信线程的时间节片的顺序，能够保证在任意时刻，只会有一个节点中的通信线程处于运行状态。

[0044] 如图 4 所示，表述了再同步流程的处理细节，包括节点中的线程规划以及时间片的分配。在节点 1 中，对再同步消息的监听状态会持续 $T_{i,1}+t_{Dev}+T_{delay}/2$ 的时间，$T_{delay}/2$ 表示了网络中的单向通信延迟，$t_{Dev}+T_{delay}/2$ 则是与主节点之间最长的通信延迟，再加上 t_{Dev} 以修正两者的时钟偏移量，再将该式从 $T_{i,1}$ 扩展到节点 m 中的 $T_{i,m}$。由此可知，从时刻

$T_{i,k}-t_{Dev}-t_{resp}$ 到时刻 $T_{i,k}+t_{Dev}+T_{delay}/2$，这段时间为节点 k 中再同步开始之前的准备时间。

[0045] 如图 5 所示，在普通修正部分，偏移量 t_{Dev} 在当前的再同步过程中得到修正，但由时钟频率所决定的，在下一次再同步之前，偏移量还是会逐渐恢复到 t_{Dev} 附近，这就意味着时差始终保持在 t_{Dev} 左右。更理想的做法，如优化修正部分，是将时钟回拨到 $-t_{Dev/2}$，也就是比标准时钟慢 $t_{Dev/2}$ 的时间。在这种情况下，时钟被回拨了 $3/2t_{Dev}$ 的时间而不是 t_{Dev}。这对后续的再同步来说是一项不错的改进，之后从节点时钟偏移量会在 $-t_{Dev/2}$ 到 $t_{Dev/2}$ 之间浮动，主从节点间的时差将会维持在 $t_{Dev/2}$ 的范围内。

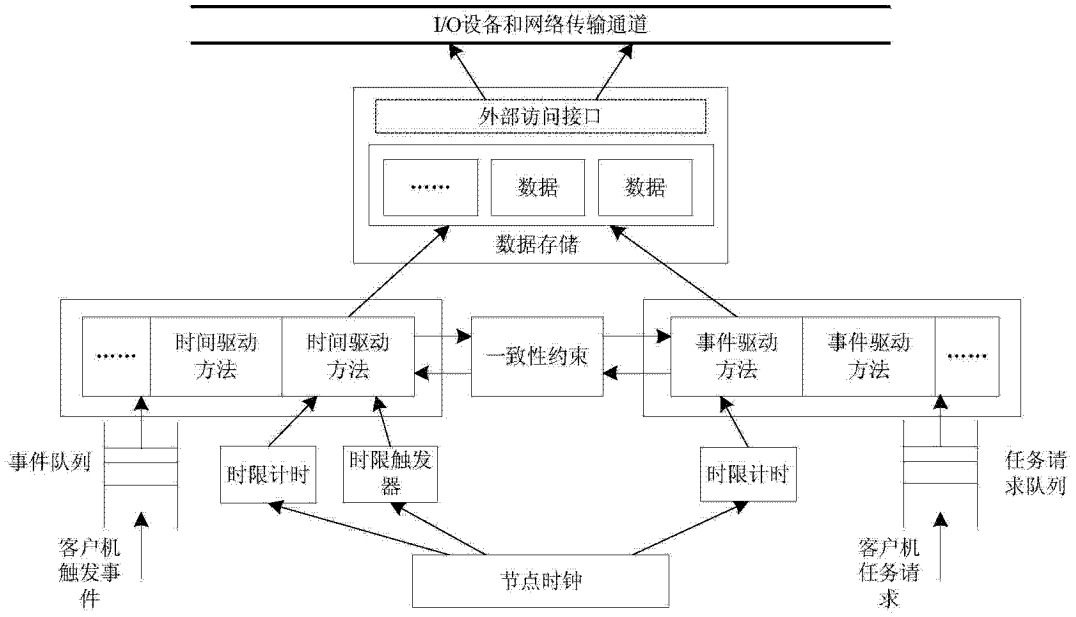
I/O设备和网络传输通道
外部访问接口
……
数据
数据
数据存储
……
时间驱动方法
时间驱动方法
一致性约束
事件驱动方法
事件驱动方法
……
事件队列
时限计时
时限触发器
时限计时
任务请求队列
客户机触发事件
节点时钟
客户机任务请求

图 1

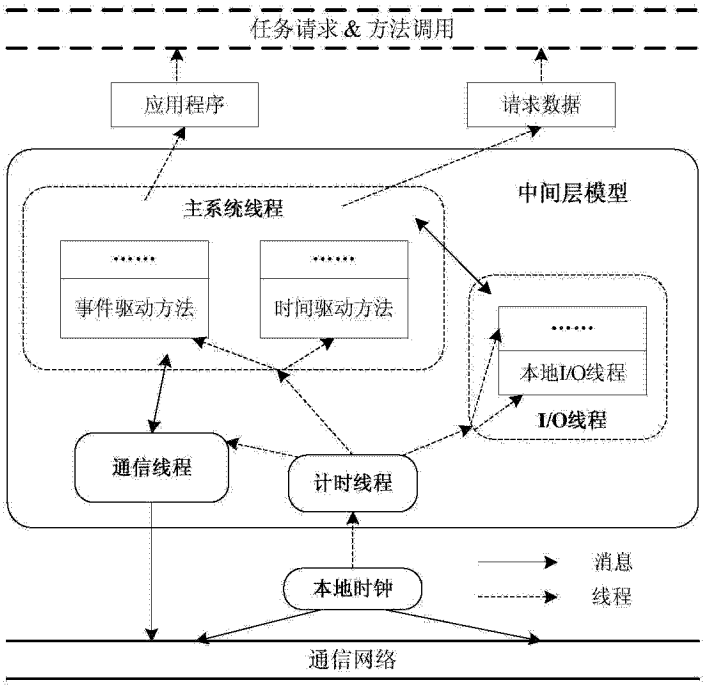
任务请求 & 方法调用
应用程序
请求数据
中间层模型
主系统线程
……
事件驱动方法
……
时间驱动方法
……
本地I/O线程
I/O线程
通信线程
计时线程
本地时钟
消息
线程
通信网络

图 2

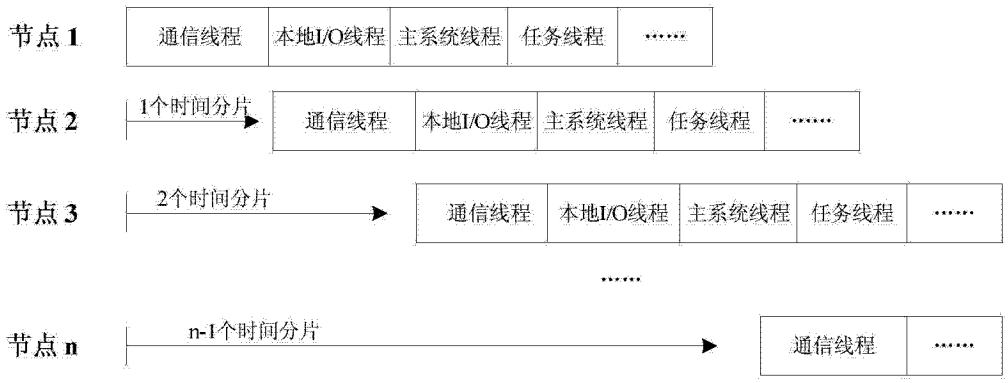
节点 1
通信线程
本地I/O线程
主系统线程
任务线程
……
节点 2
1个时间分片
通信线程
本地I/O线程
主系统线程
任务线程
……
节点 3
2个时间分片
通信线程
本地I/O线程
主系统线程
任务线程
……
……
节点 n
n-1个时间分片
通信线程
……

图 3

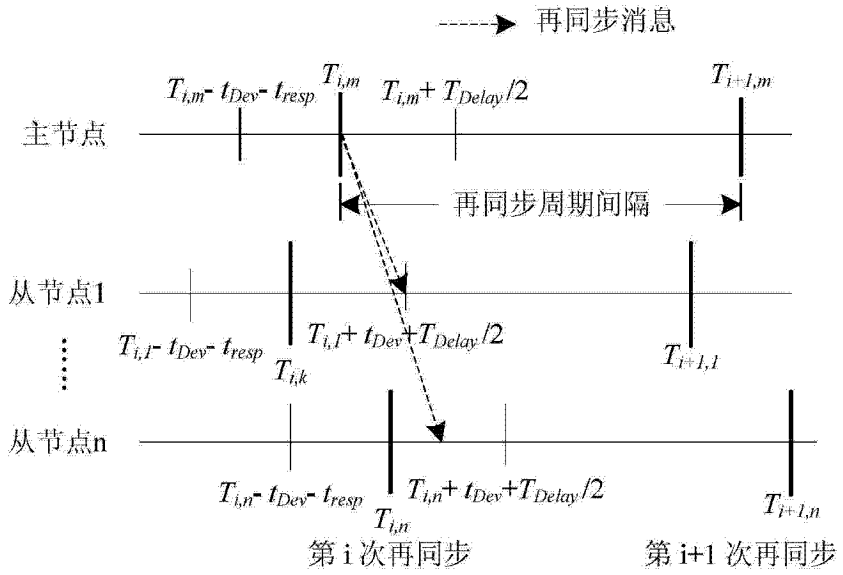
再同步消息
$T_{i,m}$- t_{Dev}- t_{resp}
$T_{i,m}$
$T_{i,m}$+ T_{Delay}/2
$T_{i+1,m}$
主节点
再同步周期间隔
从节点1
$T_{i,1}$- t_{Dev}- t_{resp}
$T_{i,1}$+ t_{Dev}+T_{Delay}/2
$T_{i,k}$
$T_{i+1,1}$
从节点n
$T_{i,n}$- t_{Dev}- t_{resp}
$T_{i,n}$+ t_{Dev}+T_{Delay}/2
$T_{i,n}$
$T_{i+1,n}$
第 i 次再同步
第 i+1 次再同步

图 4

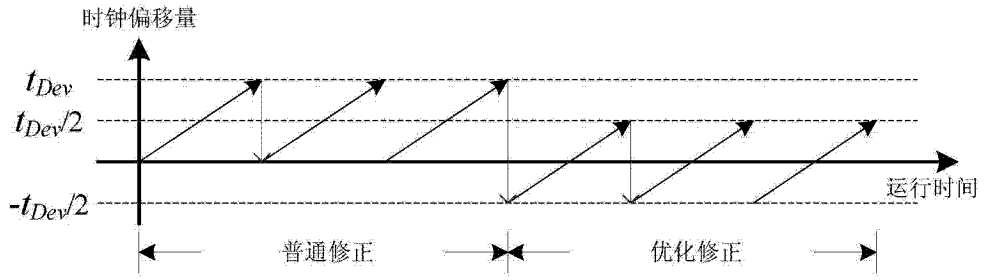
时钟偏移量
t_{Dev}
t_{Dev}/2
-t_{Dev}/2
运行时间
普通修正
优化修正

图 5