文章编号:1671-4598(2022)03-0229-05 DOI:10.16526/j.cnki.11—4762/tp.2022.03.038 中图分类号:TP393 文献标识码:A

基于 RTEthernet 改进的时钟同步算法

陈佳佳，张凤登，张宇辉
（上海理工大学 光电信息与计算机工程学院，上海 200093）

摘要：以太网其庞大的网络系统在复杂的环境中存在网络链路延迟，节点时钟的漂移，同步能力差等问题；通过研究 RTEthernet 协议的起源和工作原理，考虑到影响实时以太网时间同步精密度的时钟拜占庭故障、网络传输延迟和漂移率等 3 个因素，建立了符合 RTEthernet 协议的通信模型；对 FTA 时钟同步算法在故障下时钟同步精密度损失率提升较少的问题进行了研究，引入了滑动窗口技术，提出了容错滑动窗口（FTSW，fault tolerant sliding window）算法；容错滑动窗口算法能进一步提高分布式系统在进行时钟同步是对故障节点的容错能力；最后，使用 CANoe 仿真工具对 FTSW 算法进行仿真验证，FTSW 算法的容错性优于 FTA 时钟同步算法算法，且在系统（7 个节点）中存在两个拜占庭故障的情况下，同步后的精密度损失率降低了 7.1%。

关键词：RTEthernet；FTSW；以太网；拜占庭故障；时钟同步

Improved Clock Synchronization Algorithm Based on RTEthernet

CHEN Jiajia，ZHANG Fengdeng，ZHANG Yuhui
（School of Optical Information and Computer Engineering，Shanghai University of Technology，Shanghai 200093，China）

Abstract：Ethernet with its huge network system has problems of network link delay，drift of node clocks and poor synchronization capability in a complex environment. By studying the origin and working principle of RTEthernet protocol，a communication model conforming to RTEthernet protocol is established considering three factors such as clock Byzantine faults，network transmission delays and drift rates，that affect the precision of real-time Ethernet time synchronization. The problem that the FTA clock synchronization algorithm has less improvement in clock synchronization precision loss rate under fault is studied，the sliding window technology is introduced，and the Fault-Tolerant Sliding Window（FTSW）algorithm is proposed. The Fault-Tolerant Sliding Window algorithm can further improve the fault tolerance of the distributed system to the faulty nodes when clock synchronization is performed. Finally，the FTSW algorithm is simulated using the CANoe simulation tool to verify that the FTSW algorithm is more fault-tolerant than the FTA clock synchronization algorithm，and in the case of two Byzantine faults in the system（seven nodes），the precision loss rate after the synchronization decreases by 7.1%.

Keywords：RTEthernet；FTSW；ethernet；byzantine failure；clock synchronization

0 引言

RTEthernet（Real-Time Ethernet）是一种可以兼容传统以太网协议的实时以太网。RTEthernet 在以往的物理层和数据链路层之上增添了会话层，并且使用了“通信循环”的概念来分时传输实时性消息和非实时性消息。

国外在时钟同步理论研究的起步是比较早的。1978 年，Leslie Lamport 在文献［1］中提出了逻辑时钟的概念，逻辑时钟这一概念的提出给内部时钟同步技术的发展提供了前提条件，通过在分布式实时系统内部建立一个虚拟的全局时间来实现各个节点之间的时间同步。

2004 年，Kopetz 等人在其著作［2］中提出了将状态同步与速率同步相结合的方法，并对主从式，非主从式同步进行了实验。实验表明，将状态同步与速率同步相结合能有效提高同步的精密度。对于不同的系统模型、同步精密度要求和容错能力等方面，衍生出了各种各样的时钟同步算法。综合远程时钟读取技术、算法容忍故障类型、系统通信模型等方面，可将算法大致分成了三类[3-4]：确定性、概率型和统计型。

通过对 RTEthernet 协议通信原理及其基础时钟同步算法进行了研究，并且考虑拜占庭故障对系统时钟精密度的影响，引入了滑动窗口技术来提升时钟同步算法容错性，提出了具有更高容错性的时钟同步算法——滑动窗口时钟同步算法。

1 系统结构及原理

1.1 RTEthernet 体系结构

RTEthernet 是保留了传统 Ethernet 的物理层和数据链

收稿日期:2021-09-17； **修回日期**:2021-10-20。
基金项目:国家自然科学基金(71840003);上海市自然科学基金项目(15ZR1429300)。
作者简介:陈佳佳(1997-),男,江苏南通人,硕士,主要从事分布式实时系统、实时以太网、深度学习方向的研究。
张凤登(1963-),男,上海人,博士生导师,教授,主要从事现场总线技术、汽车电子学等方向的研究。
引用格式:陈佳佳,张凤登,张宇辉.基于 RTEthernet 改进的时钟同步算法[J].计算机测量与控制,2022,30(3):229-233,271.

路层的实时以太网。在这个基础上引进了“全局时间”的概念，同时还用了 ISO/OSI 参考模型的会话层进行报文的再封装。RTEthernet 不仅可以兼容传统以太网，而且可以通过用时分多路访问（TDMA，time division multiple access）的通信方式进行基础报文的传送，提高了传统以太网的实时性和安全性。ISO/OSI、Ethernet 和 RTEthernet 的体系结构如图 1 所示。

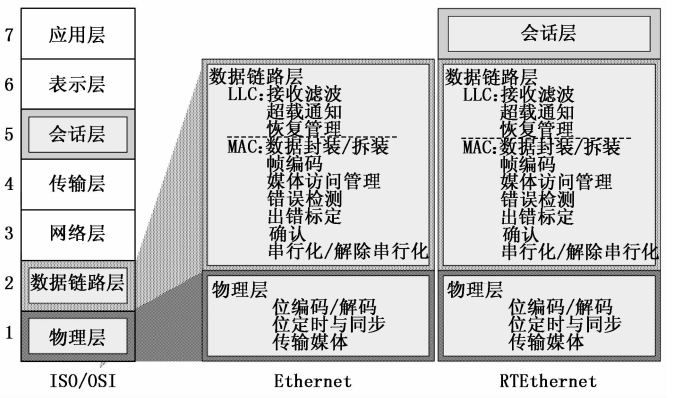

图 1 ISO/OSI、Ethernet 和 RTEthernet 的体系结构

1.2 RTEthernet 通信结构

RTEthernet 协议以确定性定时通信为基础，其通信方法是基于周期性重复的循环通信。因此，需要提前建立循环通信和时间窗口的大小，RTEthernet 的整体情况表示如图 2 所示。图中的矩阵从宏观上展现了 RTEthernet 的工作原理。

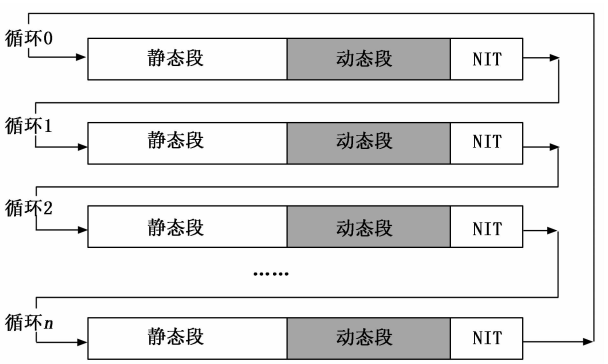

图 2 RTEthernet 通信原理

2 同步时钟的概念及 FTA 算法

2.1 时钟同步

时钟同步算法的主要目标是要保证处理器的时钟差异不超过某个定值。在保证一定时钟同步精度的前提下，降低节点在时钟同步过程中的能量消耗，延长节点的生命周期是设计时钟同步算法的首要考虑。保证分布式系统中各个节点的时钟彼此同步成为保证分布式系统实时性的基础。

2.2 误差来源

分布式实时系统中的节点需要交换各自的时钟信息来实现建立全局时间。它们通过通信网络连接，在发送和接受各自本地时钟时都需要一个真实存在的处理时间。因此在时钟同步中需要考虑通信延迟的存在，不然将会直接影响到最终算法的性能。而且节点中的处理器、时钟、通信的链路等组件都有可能发生故障。故障主要可以分成七类，其中最重要的是时钟拜占庭故障。分布式系统中的任意节点都有可能遭遇某一种故障或几种故障混合，这时我们称该节点为故障节点，否则称之为无故障节点。文中考虑的是一般性故障和最恶劣情况（拜占庭故障，如图 3 所示）下的情况。若分布式系统可以通过某时钟同步算法容忍最恶劣情况下的故障，则说明这个算法是有效的。时钟同步算法的容错性主要体现在系统对拜占庭故障节点的容忍性这方面。

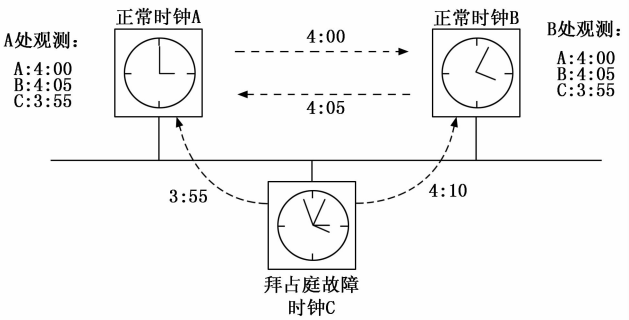

图 3 时钟拜占庭故障

2.3 时钟同步的衡量指标

为了衡量系统中各个时钟的行为是否达到同步，引入了准确度 Ψ、时钟偏差 Δ 和精密度 ϕ 这几个指标来衡量。偏差 Δ：具有相同分辨率的两个时钟在同一单位微节拍上的偏移，即两个时钟速率的相对差，表示为：

$$\Delta_{ab}^{i} = | r(MT_a^i) - r(MT_b^i) | \tag{1}$$

在上式中，i 表示微节拍；时钟 a，b 在第 i 个微节拍上的偏差用 $\Delta_a^i b$ 来表示；$r(MT_a^i)$ 和 $r(MT_b^i)$ 分别表示时钟 a，b 在第 i 个微节拍上对应的时钟微节拍数。精密度 ϕ：在给定的真实时间间隔 $[t_1, t_2]$ 内，任意两个时钟间的最大偏差，表示为：

$$\phi_i = \max\{ \Delta_a^i b \} \tag{2}$$

把有限时间间隔上的最大 ϕ_i 称为时钟集合的精密度。如果精密度 ϕ_i 小于某个期望值，就表示系统的内部各个节点之间相互同步，反之，就没有达到同步。

准确度 Ψ：在某段真实时间段内时钟 a 相对于参考时钟 r 的偏差，用 Ψ_i^a 表示。

$$\Psi_i^a = \max | C_a(\mu T_a^i) - C_r(\mu T_a^i) | \tag{3}$$

其中：$C_a(\mu T_a^i)$ 表示时钟 a 在第 i 个微节拍的长度，$C_r(\mu T_a^i)$ 表示时钟 a 在第 i 个微节拍上对应参考时钟的微节拍数。某时间段内外部参考与给定时钟时钟的最大偏差常用时钟的准确度来衡量。

2.4 FTA 算法

传统的容错算法 FTA 容错均值算法是通过对修正项进

行筛除提高修正项的准确性。FTA 算法是一种单轮算法，能够对不一致的信息进行处理，避免因为信息不一致引入的错误。该算法是从传统的“平均技术”发展而来的，并且在其中融入了去除极值的思想，于是形成了基础的时钟同步算法，FTA 算法在一定基础上提高了分布式系统时钟同步的容错性，算法过程如图 4 所示。

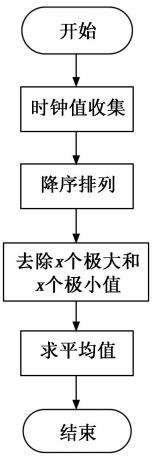

图 4　FTA 算法同步过程

3　滑动窗口时钟同步算法

3.1　滑动窗口技术

滑动窗口是一种流量控制技术。在早期的网络通信中，通信双方没有考虑网络的拥堵情况而直接发送数据导致了中间结点阻塞丢包，双方都不能发送数据，所以就有了滑动窗口机制来解决此问题[5]。后来，由这一技术衍化而来的滑动窗口算法不仅大量运用到了通信领域，在其他各个领域之中也得到了广泛地应用，例如图像处理，轨迹预测及其他进行数据预测、优化的领域。滑动窗口算法和滑动窗口技术是一样的，只是应用的场景不一样，可以根据需要调整窗口的大小，也可以固定窗口的大小[6]。

滑动窗口技术是在给定的特定窗口大小的数组或字符串上执行要求的操作。该技术可以将一部分问题中的嵌套循环转变为一个单一循环，因此它可以减少时间复杂度。下面举例简单说明滑动窗口技术的原理。如图 5 所示，我们假定有一组数：－2，3，1，5，4，0，7，1，－1，求出波动最小相邻的 4 个元素。因此，我们将窗口大小设定为 4，当滑动窗口每次划过数组时，计算当前滑动窗口中所有元素的方差。图 5 可以方便看出滑动窗口技术可以用来解决一些满足一定条件的连续区间的问题。由于区间的连续性，当区间发生变化时，可以通过既有的计算结果对搜索空间进行裁剪，从而减少了重复计算，节省了时间。

3.2　容错滑动窗口算法的描述

根据文中对容错能力的改进，下面以为伪代码的形式描述第 i 轮同步中节点 k 使用容错滑动窗口算法进行时钟同步的过程：

1）函数及符号的定义，见表 1：

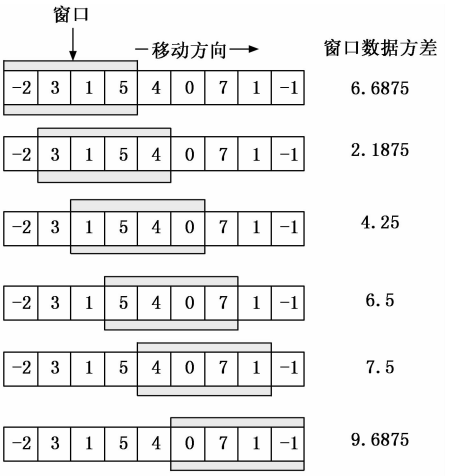

图 5　滑动窗口示例

表 1　伪代码参数

符号	说明
f	容忍的故障节点数
$P_k^i[k]$	节点 k 在第 i 轮时钟同步时读取的所有时钟差值数列
T_k^{NOW}	表示节点 k 当前的逻辑时钟值
$ARR_k^i(q)$	表示 k 节点收到节点 q 的时钟值(用时钟 k 度量)
ADJ_k^i	节点 k 在第 i 轮时钟同步的校正项
T_k^i	第 i 轮中 k 节点应该发送同步消息的逻辑时刻
T^i	表示进入第 i 轮的逻辑时钟值
$descend(P_k^i)$	将数列 P_k^i 降序排列的函数
$discard(P_k^i)$	表示去掉 P_k^i 中 $f/2$ 个最大值与最小值的函数
$median(P_k^i)$	求数列 P_k^i 中值的函数
$variance(P_k^i)$	求数列 P_k^i 方差的函数
$W_j[k]$	表示第 j 个含有 f 个元素的数列
$Q_k^i[k]$	$discard(descend(P_k^i[k]))$
$S[j]$	存放滑动窗口计算出的方差数列
S_max	表示方差最大的窗口元素组成的数列
$MED(P_k^i)$	表示求数列 P_k^i 中值的函数
$rest(P, Q)$	获取集合 P 中 Q 集合的补集
$send()$	发送时钟值函数
$SWA_j()$	确定滑动窗口中元素的函数

2）算法描述

算法：容错滑动窗口算法

输入：重同步时间

输出：本地节点即将应用的校正项

1. WhileT_k^NOW＝T^{i}do/＊当任一节点当前的逻辑时钟进入第 i 轮的循环时开始准备时钟同步＊/

2. $ARR_k^i(k)=T_k^N$OW //每个节点记录当前的时钟值

3. ifT_k^NOW＝T_k^ithen /＊如果任一节点的时钟值等于重同步时间，则将自己的时钟值发送给其他所有节点＊/

4. send(message)；

5. $P_k^i[k]=ARR_k^i(q)$

6. end if

7. ifT_k^NOW＝T_NIT^{i}then

8. Q_k^i[k]=discard(descend(P_k^i[k]))//将获取到的时钟值进行降序排列,然后去掉 f/2 个最大和最小的值

9. whilej≤n−2f+1then//获取所有窗口内的方差

10. W_j[k]= SWA_j(Q_k^i[k])

11. S[j]=variance(W_j[k])

12. end

13. MED=median(rest(Q_k^i[k],S_max))//获取方差最大集合的补集,然后求得该补集的中值

14. $ADJ_k^i=T_k^i-MED+\delta$//计算校正项

15. return(ADJ_k^i)

16. end if

17. end while

上述过程概括的来说，可将算法分为 4 个阶段：时钟值收集阶段、去极值阶段、滑动窗口阶段和求中值阶段。分布式实时以太网将通信的循环划分为了静态段、动态段和 NIT 段。每个节点在通信循环中 NIT 的开始时刻执行 FTSW 算法，如图 6 所示。

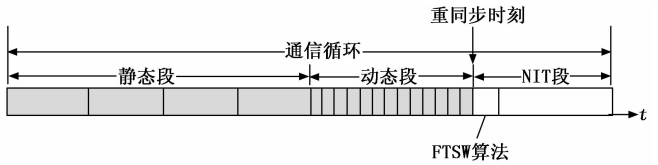

图 6 NIT 段的 FTSW 算法

下面将分别介绍 FTSW 算法的各个阶段。

1）时钟值收集阶段：所有要参与时钟同步的节点在通信循环的静态段时隙的开始处，通过通信帧帧头的第 4 位，向时钟集合中的所有其他参与者表明它将发送是同步帧，该同步帧的作用是参与网络的同步[7]。然后利用静态段发送的同步帧测量时间偏差来得到每个节点的时钟估计值，并存储在一个数列中，然后将数列中的时钟值按照降序排列，如图 7 所示。

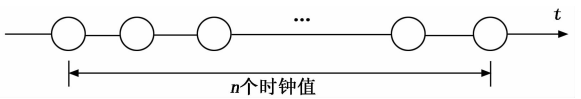

图 7 时钟值收集阶段

每个节点时钟在第 r 轮的重同步，可将得到的时钟值构造成一个时钟值矩阵，我们以节点 k 为例，节点 k 保存该矩阵的列和对角线上的元素。

$$\boldsymbol{X}_k^r=\begin{bmatrix} x_{1,1}^r & & x_{1,k}^r & & \\ & x_{2,2}^r & x_{2,k}^r & & \\ & & \cdot & \cdot & \\ & & x_{k,k}^r & & \\ & & \cdot & \cdot & \\ & & x_{n,k}^r & & x_{n,n}^r \end{bmatrix} \tag{4}$$

对于无故障时钟有：

$$x_{i,k}^r=x_{i,i}^r+(\Delta_{i,k}^r+d^e)(1-\delta_{i,k}) \tag{5}$$

其中：$\Delta_{i,k}^r$表示时钟 i 和时钟 k 之间的差异；d^e 表示估计的消息延迟；

$$\delta_(i,k)=\begin{cases}1 & \text{如果} \quad i=k\\ 0 & \text{如果} \quad i\neq k\end{cases}$$

2）去极值阶段：将 X_k^r 中的第 k 列时钟值按照降序排列后去除 $f/2$ 个最大和 $f/2$ 个最小的时钟值，其中 f 表示系统能够容忍的最大故障时钟个数。

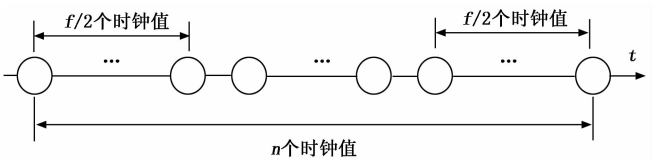

图 8 去极值阶段

3）滑动窗口阶段：滑动窗口的大小为 f，并且 f 为该系统能够容忍的故障节点数。该窗口从去极值后的时钟值序列的最左端开始，也就是第一个窗口是由第 $f/2+1$ 个元素至第 $3f/2$ 个元素组成的。然后，将窗口逐元素的向右滑动，直至窗口的最右边元素为第 $n-f/2$ 个元素，即去极值后序列中的最右边元素。在这个过程中，一共得到了 $n-2f+1$ 个窗口的数据集。最后，从这 $n-2f+1$ 个数据集中找到方差最大的窗口，如图 9 所示。

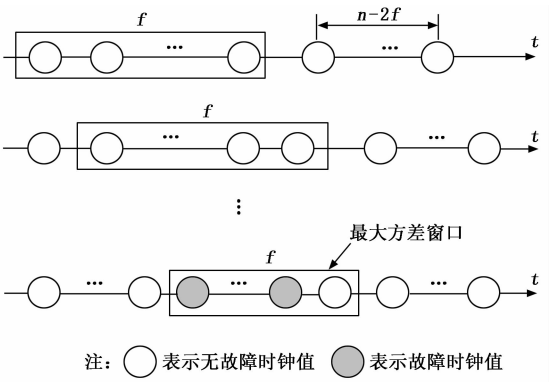

图 9 滑动窗口阶段

4）求中值阶段：利用滑动窗口阶段得出了方差最大的时钟值集，然后将这些时钟值剔除，最后对剩余的时钟值进行求中值，从而减轻时钟拜占庭故障对最终校正项的影响。

上述算法概括的来说，先将远程读取到的时钟值进行降序排列，然后去除 f 个极端值（$f/2$ 个最大值和最小值），当 f 是奇数时将去除的极大值个数向上取整，去除的极小值个数向下取整。然后，对剩余的时钟值利用滑动窗口的思想求出方差最大的窗口，然后得到其补集。最后求取该补集的中值，若此时补集为偶数时求取中间两个值的均值，然后对该中值向下取整。最终，利用简单的代数运算就可以求得本地时钟的校正项。

从算法的实现上来讲，容错滑动窗口算法在 FTA 算法的基础上采用了容错中值的思想并且增加了求最大方差窗口的过程，这一过程有效地提高了系统的容错能力，这是

该算法的显著特点。

4　算法仿真验证

4.1　系统模型的构成

实验用 7 个节点组成一个总线型的拓扑结构，该网络中的各个节点模拟 RTEthernet 协议进行通信。将节点 3 和 6 设置成拜占庭故障节点，把 1，2，4，5，7 节点设置成无故障节点，如图 10 所示为仿真系统拓扑结构。

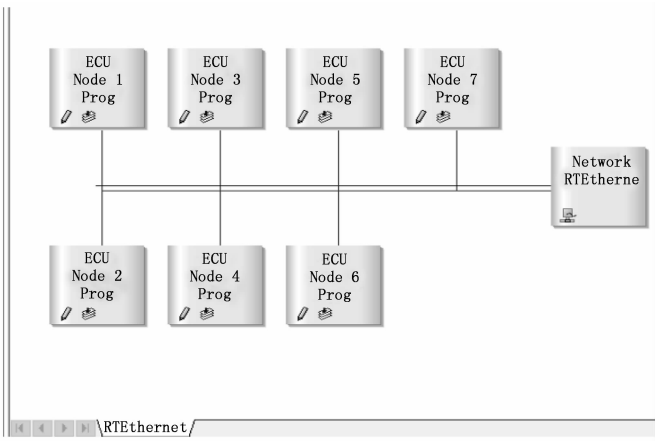

图 10　仿真系统拓扑结构

4.2　系统参数设置

根据文献［8－10］研究成果和 RTEthernet 通信模型对局部参数进行设置：系统中各节点初始同步的精密度设置为 $\phi_{mt}=20\times10^{-6}$ s，各个节点的最大时钟漂移率为 $\rho=10^{-4}$ s，系统中传输延迟最小的为 5×10^{-6} s，延迟最多的为 10×10^{-6} s。即设置分布式系统内的消息传输延迟范围为［5，10］μs，因此该系统传输延迟的不确定度为 2.5×10^{-6} s，消息延迟范围的中间值为 $\delta=7.5\times10^{-6}$ s。然后，考虑每个循环中包含了 ST 段和 DYN 段以及 NIT 段，将重同步周期长度设置为 5 ms。节点参数设置如表 2 所示。

表 2　系统节点参数

节点	时钟初始值/μs	时钟偏移率/μs	宏节拍长度/μs	微节拍长度/μs	NMP/个	同步帧发送时刻/μs
1	20	35	4	1	4	40
2	5	40	4	0.5	8	80
3	0	90	4	2	2	120
4	12	30	4	0.2	20	160
5	8	25	4	0.4	10	200
6	10	70	4	4	1	240
7	16	20	4	0.8	5	280

4.3　仿真结果分析

根据上述系统拓扑结构和参数的设置，本文将实验分为两部分，第一部分是无拜占庭时钟故障情况下对 FTA 和 FTSW 算法进行收敛性验证，第二部分是存在拜占庭故障情况下对 FTA 和 FTSW 算法进行容错性验证，运行结果如下：

1）当 $f=0$ 与系统初始精密度 $\phi_{mt}=20$ μs 时，系统中的各个节点分别执行 FTA 算法和 FTSW 算法，最终将每次同步后系统的精密度绘制为如图 11 所示。

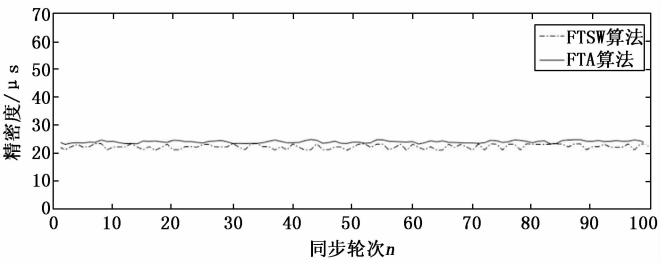

图 11　无拜占庭故障情况下两种算法的时钟同步精密度对比

从图 11 可以看出，该系统中的各个时钟通过时钟同步算法进行过周期性的时间校正后，系统的时钟精密度会趋于稳定，并且当无拜占庭故障时钟的时候 FTA 算法大于 FTSW 算法。当考虑到传输延迟的误差后，FTSW 算法的平均精密度约为 22.28 μs，FTA 算法的平均精密度约为 23.15 μs。由此可以看出，在此系统中无拜占庭故障的情况下，原始算法的精密度还是比较优异的。

2）当 $f=1$，2 和系统初始精密度 $\phi_{mt}=20$ μs 时，系统中的各个节点分别执行 FTA 算法和 FTSW 算法，对 FTA 算法和 FTSW 算法进行比较分析和容错性验证。

由图 12 可以看出，当系统中节点 3 发生拜占庭故障时，经过 FTA 算法同步后该系统精密度为 24.47 μs，当系统中的节点 3 和 6 同时发生拜占庭故障时，该系统的精密度为 26.32 μs。由此可以发现，在一个由 7 个节点组成的系统中，当系统中存在两个拜占庭故障的时候，FTA 算法的容错能力大大降低，系统的精密度下降了 13.7%。

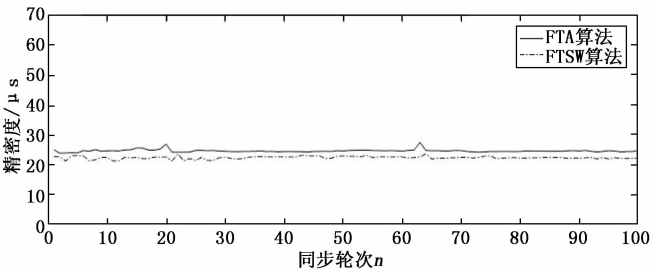

图 12　出现拜占庭故障情况下两种算法的时间同步精密度对比

此实验是在时钟同步过程中通过环境变量控制一个节点（节点 3）或者两个节点（节点 3 和 6），使其产生［0，200］之间的随机数（单位为 μs）作为该节点的本地时间信息封装到同步帧并发送到网络，试图干扰其他节点的时间同步，但该节点自身真实的本地时间还是保持原来微节拍计数器的正常计数值。从图 12 可以看出，当一个节点发生拜占庭故障时，该系统的精密度为 22.15 μs，当系统中两个节点同时发生拜占庭故障的时候，该系统的精密度为 23.75 μs。

（下转第 271 页）