文章编号:1671-4598(2018)07-0271-05 DOI:10.16526/j.cnki.11—4762/tp.2018.07.060 中图分类号:TP3 文献标识码:A

基于空间应用的 TTE 时钟同步算法研究

徐乾舜，华更新
（北京控制工程研究所，北京 100190）

摘要：时间触发以太网络（time—triggered ethernet，TTE）作为新型的分布式交换网络，具有强实时、高带宽、高安全性的特点，而 TTE 时钟同步是 TTE 网络实现的前提条件，通过分析 TTE 时间同步过程以及其过程中存在的噪声，总结了影响时间同步精度的因素，建立了时钟噪声模型，并且采用 kalman 滤波等滤波算法得到了良好效果，最后在滤除噪声影响之后使用两种渐进时钟补偿算法进行时钟补偿，并通过仿真验证了算法的可行性。

关键词：时间触发以太网；时间同步；kalman 滤波；渐进补偿

Research on TTE Clock Synchronization Algorithm Based on Space Application

Xu Qianshun，Hua Gengxin
（Beijing Institute of Control Engineering，Beijing 100190，China）

Abstract：Time triggered Ethernet as a new distributed switching network has the characteristics of strong real—time，high bandwidth and high security，and TTE clock synchronization is the prerequisite for constructing the TTE network. Analyzing the TTE time synchronization process and the existing noise，summarizes the factors that affect the time synchronization precision，builds a clock noise model，and get a good result using the Kalman filter filtering algorithm. Finally this paper use two progressive compensation algorithm to compensate the clock offset after removing the effect of noise，and the feasibility of the algorithm is verified by simulation.

Keywords：time—triggered ethernet (TTE)；clock synchronization；kalman filtering；progressive compensation

0 引言

近些年来，在许多关键领域比如航天航空、工业控制等对实时性提出了更高的要求。广泛应用于航空领域的航空电子全双工交换式以太网（avionics full duplex switched ethernet，AFDX）[1]采取了一些措施提高了在一些关键领域的时间确定性[2]，但由于其还是事件触发的机制，并不能保证完全的时间确定性。时间触发以太网（time—triggered ethernet，TTE）作为新型的分布式交换网络，在普通以太网的基础上增加时间触发机制，解决了普通以太网中时间不确定性问题，具有强实时、高带宽、高安全性的特点，是下一代航天航空数据网络的重要发展方向。

TTE 网络要想实现时间触发以及实时调度的功能，关键在于建立一个全局统一同步的时钟。只有在保证一定精度范围内的时钟同步的前提下，才能使各个节点在规定的时刻正确收发数据消息。因此研究 TTE 时钟同步算法成为实现 TTE 网络的关键之一，然而，目前国内外对于 TTE 时间补偿算法研究较少，AS6802 协议[4]虽然对 TTE 的时钟同步模式进行了描述，并对协议中关键的流程比如时间同步过程、固化以及压缩算法等进行了详细介绍，但却没有为时钟同步提供具体的补偿方法。

本文的工作在于首先分析了 TTE 时间同步过程，以及时间同步过程中存在的不确定性，这些不确定性会直接影响时间同步精度，并建立了时钟模型以及不确定性带来的抖动噪声数学模型，根据噪声的统计特性采用了 kalman 滤波等三种滤波算法，得到了较好的效果。最后在滤除噪声的影响之后，采用了两种渐进时钟补偿方法对时钟进行了补偿，并通过仿真分析验证了算法的可行性。

1 TTE 时钟同步分析

1.1 时钟同步过程

TTE 网络时钟同步节点主要分为三种：CM 节点、SM 节点以及 SC 节点，时钟同步过程如图 1 所示。

1）SM 在 sm _ dispatch _ pit 时刻提交 PCF 帧，并在 sm _ send _ pit 时刻发送 PCF 帧。

2）CM 在 cm _ receive _ pit 时刻接收到来自 SM 的 PCF 帧，并且使用时钟固化算法，使得 PCF 的接收顺序和其发送顺序一致，得到时钟固化点 cm _ permanence _ pit。

3）CM 使用时间压缩算法计算时间偏差的均衡值 cm _ cr。

4）CM 根据 PCF 的预计接收时间 cm _ scheduled _ pit 以及均衡值 cm _ cr 计算得到时钟偏差并进行修正，然后在 cm _ dispatch _ pit 时刻提交 PCF 帧，在 cm _ send _ pit 发

收稿日期：2018-03-26； **修回日期**：2018-05-15。

作者简介：徐乾舜(1993-)，男，硕士研究生，主要从事时间触发以太网络方向的研究。

送新的整合的 PCF 帧。

5）SM/SC 在 smc _ receive _ pit 接收到新的 CM 发送来 PCF 帧之后，采用时间固化算法保证 PCF 的接收顺序和其发送顺序一致，得到固化时间 smc _ permanence _ pit。

6）SMC 根据预计接收时间 smc _ scheduled _ pit 以及实际固化时间 smc _ permanence _ pit 得到时钟偏差，进行时间补偿。

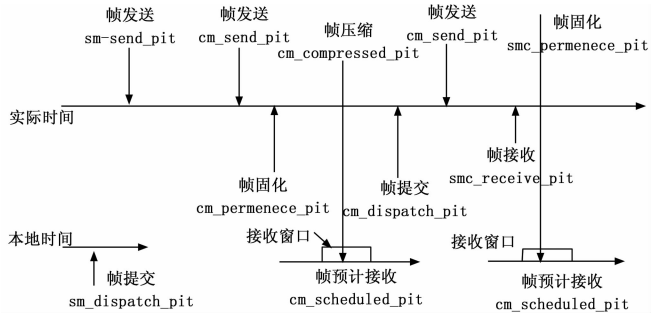

图 1 TTE 时钟同步过程

1.2 时钟同步精度影响因素

在深入分析 TTE 时钟同步的过程后，本文研究总结出影响 TTE 时间同步精度的 3 种主要因素，如图 2 所示。

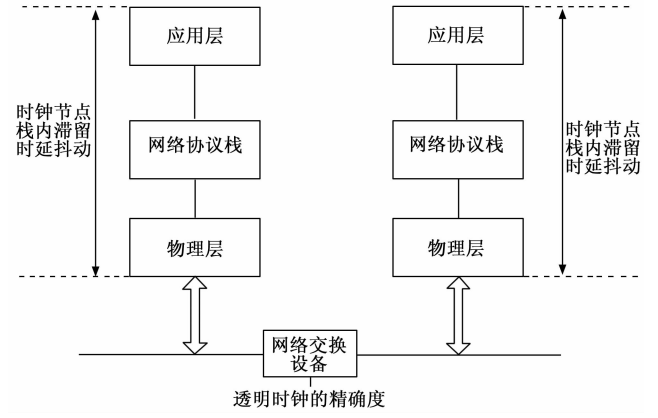

图 2 同步精度影响示意图

主要包括以下方面：

1）TTE 时钟节点的栈内滞留时延抖动。操作系统及协议栈的滞留时延抖动影响 TTE 报文的收发时间戳标记的精确性，从而直接降低时钟同步精度。

2）透明时钟精确度。TTE 时钟同步算法是基于透明时钟进行固化计算的，透明时钟的精确度是影响 TTE 时钟同步精度的关键之一，如果透明时钟记录的网络延迟与实际网络延迟越精确，那么透明时钟就越精确，同步算法计算出的时钟偏差就越准确，必然就会得到更高的时钟同步精度。

3）晶振稳定性。TTE 时钟节点晶振的不稳定性因素也会降低同步精度。

1.2.1 TTE 协议栈的滞留时延抖动

操作系统以及协议栈会带来时延抖动，TTE 报文在经过协议栈时会进行封装和拆封报文的工作，这些过程带来的时延抖动也是不确定的，这样就会造成 TTE 报文时间戳标记不准确，进而降低时间同步精度。

通过分析，TTE 时间戳标记的位置一般有以下几种：

1）应用层：采用在应用层 socket 接口处打时间戳的方式。传统的时间同步协议如 NTP 协议就采用软件方式在应用层加盖时间戳，受协议栈内滞留时延对时间戳的精确性影响较大，时间同步精度只能达到 ms 量级。

2）网卡驱动层：采用在网卡驱动层打时间戳的方式是软件方式获得时间戳较好的位置。利用网络接口中断服务程序的入口处进行时间戳标记，操作系统的计时特性和负载决定了该方法的时间戳准确性。

3）MII 接口处：MAC 层和物理层之间的 MII 接口也称为媒体独立接口，在 MII 接口处侦听捕获时间戳可获得较好的时间戳准确性。

4）物理层内，将硬件时间戳功能集成到 PHY 芯片中。

其中前两种为纯软件时间戳方式，成本较低，但精度不高。采用硬件方式打时间戳精度较高，能有效提高 TTE 时钟同步精度。

1.2.2 透明时钟精确度

透明时钟主要通过测量数据帧在设备以及链路上的驻留时间，并将其累加到对应的消息本身或跟随消息的修正域内。在时间触发以太网中，各节点将透明时钟的值累加、存储在 PCF 的透明时钟域中，用来存储 PCF 从源节点到目的节点经过的总时延。

透明时钟精确度主要包括三类延迟：节点发送延迟、中继延迟以及接收延迟。

其中三类延迟都与 MAC 层和 PHY 层对 PCF 帧的封装解析有关，具有较高的不确定性，此外还与本地时钟晶振稳定性、MAC 和 PHY 芯片的性能和工作模式有关。而网络链路延迟是由物理介质属性决定的，延迟和不确定性都很小。

因此采用精确计算透明时钟的方法以及选用性能好且稳定的 MAC 和 PHY 芯片都能提高透明时钟的精确度。

1.2.3 晶振的不稳定性

晶振由于制造工艺的不同，本身就存在一定的频率偏差，加上环境因素的影响，频率偏差进一步加大。如果 TTE 时间节点选用这种频率偏差较大晶振那么在一段时间后相位偏差会变得很大，可能会影响网络正常工作，也是 TTE 不能容忍的。因此晶振的不稳定性也是影响时间同步精度的一个重要因素。

1.3 关键函数

1.3.1 固化函数

时间触发以太网通过时间固化实现了“延迟固化”功能。该算法的作用就是通过 PCF 帧的到达时间确定该帧的发送时间。由于各个节点的时钟抖动以及网络参数等原因，使得 PCF 帧之间的实际延迟存在不确定偏差，故采用时间固化函数来消除不确定偏差。

固化函数作用于 CM、SM、SC 所有节点，具体流程

如下：

1）接收节点（CM/SM/SC）接收到 PCF，读取本地时钟值记为 1）接收节点（CM/SC/SM）接收到 PCF，读取本地时钟值记为 receive _ pit，并启动时间固化算法；

2）读取 PCF 的透明时钟域得到透明时钟值 pcf _ transparent _ clock，即为 PCF 在网络中传输的实际总延迟。

3）计算固化补偿延迟 permanence _ delay：

$$\text{permanence\_delay} = \text{max\_transmission\_delay} - \text{pcf\_transparent\_clock} \quad (1)$$

4）计算固化时间点 permanence _ pit：

$$\text{permanence\_pit} = \text{receive\_pit} + \text{permanence\_delay} \quad (2)$$

如上面算法，在接收端时间固化算法：

1）考虑了所有 PCF 的时延得到整个通信网络的最大网络时延 max _ transmission _ delay；

2）计算了时序保持时延 permanence _ delay。可将网络抖动造成的时延偏差转换到网络时延中。

在时序保持功能下：透明时钟机制可以提取网络抖动产生的时延；因为所有的 PCF 帧到达目的节点后都是在 permanence _ pit 时间点节点才认为该 PCF 是有效可用的帧，故可以将所有 PCF 的网络时延看作恒值 max _ transmission _ delay。

1.3.2　压缩算法

TTE 采用压缩算法实现网络节点间时间偏差均衡值的计算，用以时间同步。该算法通过 CM 整合同一个周期内 SMs 发来的 IN 帧固化时间点（permanence _ pit），通过调用压缩算法得到这些 IN 帧的均衡值（compression _ pit），并在等待一段时间后形成一个新的 PCF 帧广播到网络中其他 SM 和 SC 节点进行时间同步。

压缩函数由 3 个阶段组成：收集阶段、计算阶段和延迟阶段。

1）收集阶段。

当一个 PCF 帧固化之后并且当前没有相应集成周期处于激活状态的压缩函数正在收集 PCF 帧时，则从该时刻起开始压缩函数。压缩函数开始之后，开启第一个观察窗口（observation _ window，OW）开始接受 PCF 帧。收集阶段遵循以下规则。（n 代表当前 OW，$OW_{\max}$ 代表最大观察窗口）

（1）当 $n=1$ 时，如果当前 OW 内收集到固化 PCF 帧小于 2，那么停止收集，否则开启下一个 OW。

（2）当 $n \geqslant 2, n \leqslant OW_{\max}$ 时，如果当前 OW 没有收集到固化的 PCF 帧，那么停止收集，否则开启下一个 OW。

（3）当 $n = OW_{\max}$，则该 OW 结束之后停止收集。

2）计算阶段。

该阶段根据收集阶段收集到的 PCF 固化时间点计算出压缩校验值（compression _ correcion）input 表示在收集阶段的第 i 个 PCF 时间固化点和第一个 PCF 时间固化点的差值。

compression _ correcion 计算方法如下：（k 表示所收集的 PCF 固化点数目，$th+$ 是顺次第 f 个，$th-$ 是倒次第 f 个）：

$$\begin{aligned}
&k=1：\text{compression_correcion} = \text{input}_1；\\
&k=2：\text{compression_correcion} = (\text{input}_1 + \text{input}_2)/2；\\
&k=3：\text{compression_correcion} = \text{input}_2；\\
&k=4：\text{compression_correcion} = (\text{input}_2 + \text{input}_3)/2；\\
&k=5：\text{compression_correcion} = (\text{input}_2 + \text{input}_4)/2；\\
&k>5：\text{compression_correcion} = (\text{input}_{th+} + \text{input}_{th-})/2；
\end{aligned} \quad (3)$$

3）延迟阶段。

该阶段在计算阶段之后延迟 compression _ correcion 的时间得到压缩时间点（cm _ compressed _ pit）。在压缩时间点到来的时刻，新的 PCF 帧需准备好。

2　时钟噪声模型

TTE 时钟同步过程由于采用了 PCF 帧交换的方式，这样将无法避免的带来时延抖动，其中主要存在协议栈时延抖动、物理层、硬 MAC 层等时延抖动。采用在链路层打时间戳的方式可以减少协议栈带的时延抖动，但是仍存在驱动中断、物理层以及硬 MAC 层等时延抖动[10]，并且各种延迟带来的抖动均为随机不确定的。本文中假设这些延迟抖动满足高斯正态分布，主要原因有：

1）文献［3］在 Mica2Dot 平台上，采用中断的方式在发送与接收首帧数据时记录时间戳。通过实验证明了时钟的频率偏移和时钟同步过程中的噪声均满足高斯分布。

2）存在于硬件电路中的随机噪声主要为热噪声，满足高斯分布。

3）相互独立的随机抖动根据中心极限定律叠加之后趋近于高斯分布。

时钟的状态行为通常可以用两种模型来表述，一种是确定性模型，另一种是随机模型。其中，确定性时间模型一般利用实际观测到的时间数据采用多项式拟合得到[8]。而随机模型则通常采用随机差分方程表示，其主要受频率偏移、过程噪声等的影响。

本文考虑常见的只存在调频白噪声（调相随机游走噪声）和调频随机游走噪声两种噪声的时钟模型。噪声方差分别为 σ_α^2 和 σ_β^2，时间尺度为 τ 的艾伦方差及其关系如式（4）所示[5]：

$$\sigma_y^2(\tau) = \frac{\sigma_\alpha^2}{\tau} + \frac{\sigma_\beta^2 \tau}{3} \quad (4)$$

相应的两状态时钟模型用下面两个差分等式表示：

$$\alpha(t+\tau) = \alpha(t) + \beta(t) \cdot \tau + \omega_\alpha(t) \quad (5)$$

$$\beta(t+\tau) = \beta(t) + \omega_\beta(t) \quad (6)$$

其中：$\alpha(t)$ 是相位偏差，$\beta(t)$ 是频率偏差，$\omega_\alpha(t)$ 是调频白噪声，$\omega_\beta(t)$ 是调频随机游走噪声。

3　时钟补偿算法

时间同步节点的本地时钟依靠其内部晶振实现，节点内部的计数器通过捕获晶振脉冲进行累加，完成对时间的计数。由于晶振本身的频率误差，启动时间不可能完全同

步以及节点工作环境的影响，随着时间的流逝，不同节点间会出现一定的时间偏差。

节点时钟模型[11]是对时间同步协议性能进行有效分析的数学理论基础。由于节点晶振频率的不稳定性，通常节点 i 在某个 t 时刻的本地时间可以表示为公式（7）[12]：

$$C_i(t) = \frac{1}{f_0}\int_{t_0}^{t} f_i(t)\mathrm{d}t + C_i(t_0) \tag{7}$$

其中：f_0 为节点晶振的标准频率，它是表征晶体物理特性的常量，$f_i(t)$ 表示节点在时刻 t 的频率。通常由于制造工艺不同，$f_i(t)$ 和 f_0 存在微小差别。t_0 表示节点的初始时间，$C_i(t_0)$ 表示节点在 t_0 时刻的本地时间。

如果不考虑节点晶振频率变化以及外界环境影响，即认为在一定时间内节点晶振频率是保持不变的，那么节点的时钟可以表示为公式（8）：

$$C_i(t) = \frac{f_i}{f_0}(t - t_0) + C_i(t_0) \tag{8}$$

其中：$k = f_i/f_0$ 为相对频率，$C_i(t_0)$ 为节点在 t_0 时刻的本地时间，在理想情况下，时间变化速率 $r(t) = \mathrm{d}c(t)/\mathrm{d}t = 1$，然而，在实际应用中，外界环境影响和晶振本身的频率变化导致公式（8）不能满足实际应用环境。但是晶振频率波动幅度也是有限度的，公式（9）可以描述这个变化幅度：

$$1 - \sigma \leqslant k \leqslant 1 + \sigma \tag{9}$$

其中：σ 为绝对频差上界，由制造厂商提供，取值范围通常在 1～100 ppm，即节点在 1 s 内会有 1～100 μs 的时间偏移[9]。

3.1 线性渐进补偿算法

根据文献［6］，同步主节点 SM_k 的时钟模型为：

$$C(t)\_SM_k = \varphi_k + \alpha_k(t) \times t + n_k(t) \tag{10}$$

其中：φ_k 为相位偏差，$\alpha_k(t)$ 为时钟频率，$n_k(t)$ 为随机噪声。

图 3 给出了参考时钟与本地时钟的关系，其中 $c_1(t)$ 表示本地时钟，它与参考时钟存在一个初始的相位偏差 $\alpha(0)$ 以及频率偏移 β。$c_2(t)$ 是只进行周期性相位偏移补偿后的同步时钟，由于不进行频率补偿，$c_2(t)$ 与参考时钟一直存在 β 的频率偏移，在进行相位偏移补偿后，同步误差变小，但随着时间增大，同步误差会逐渐变大，最后呈现出锯齿状的特性。而 $c_3(t)$ 是只进行频率偏移补偿，它与参考时钟的频率接近，从 $c_3(t)$ 我们可以知道，如果本地时钟与参考时钟初始存在较小的相位偏移，并且保证本地时钟与参考时钟频率相近，那么就会得到较高的同步精度。

线性渐进补偿算法如图 4 所示：图中 $c_1(t)$ 与参考时钟的斜率偏移为 β，斜率为 $1+\beta$。假设 K 时刻 $c_1(t)$ 与参考时钟的相位偏移为 $\alpha(k)$，如果采用直接补偿算法，那么补偿后的结果为 $c_2(t)$，该时钟在 K 时刻出现了 $\alpha(k)$ 的跳变，此后仍保持原有的斜率。而 $c_3(t)$ 是采用线性渐进补偿算法之后的补偿结果，在调整期间，$c_3(t)$ 将按指定频率进行调节，在剩下的同步间隔里，仍按原有的 $1+\beta$ 斜率继续运行。

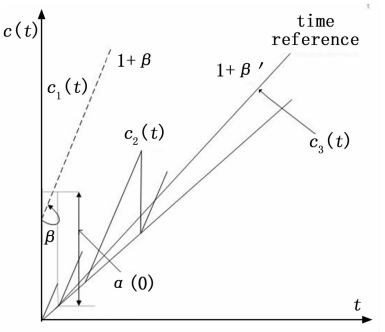

图 3 时钟同步示意图

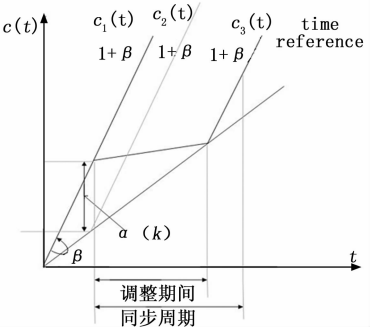

图 4 线性渐进补偿算法

3.2 PI 控制补偿算法

采用上文描述的线性渐进补偿算法调节相位偏差，虽然算法比较简单，但是实现精度不是很高，接下来介绍一种 PI 控制补偿算法分别对相位偏移以及频率偏移进行时钟补偿。

PI 调节器是一种线性控制器，它根据给定值与实际输出值构成控制偏差，将偏差的比例和积分通过线性组合构成控制量，对被控对象进行控制。其控制方程为式（12）：

$$u(t) = K_p\left[e(t) + \frac{1}{T_I}\int_0^t e(t)\mathrm{d}t\right] \tag{11}$$

其中：$u(t)$ 为 PI 控制器的输出，。$e(t)$ 为调节器的输入，K_p 为比例系数，T_I 为积分时间常数。通常采用的 PI 控制器的离散形式如下：

$$u(k) = K_p\left[e(k) + \frac{T_s}{T_I}\sum_{j=0}^{k} e(j)\right] = K_p e(k) + K_i\sum_{j=0}^{k} e(j) \tag{12}$$

其中：$k = 0,1,2\cdots\cdots$ 表示采样序列，T_s 表示采样周期，K_i 为积分系数。

控制器由比例和积分两个校正环节组成，各环节的作用如下：

比例调节作用：按比例反应系统的偏差，系统一旦出现了偏差，比例调节立即产生调节作用用以减少偏差。比例作用大，可以加快调节，减少误差，但是过大的比例，使系统的稳定性下降，甚至造成系统的不稳定。

积分调节作用：使系统消除稳态误差，提高无误差度。因为有误差，积分调节就进行直至无差，积分调节停止，积分调节输出一常值。积分作用的强弱取决于积分时间常

数 T_I，T_I 越小，积分作用就越强。反之 T_I 大则积分作用弱，加入积分调节可使系统稳定性下降，动态响应变慢。

PI 数字调节器通常可以分为位置式控制器和增量式控制器。式（12）所示的就是位置式控制算法，控制器的输出直接调节被控对象。这种算法的优点是计算精度比较高，缺点是每次都要对 $e(k)$ 进行累加，很容易出现积分饱和的情况，造成控制对象的不稳定，因此通常采用改进的抑制积分饱和的位置式控制算法，改进算法如下：

$$\begin{aligned} U(n) &= K_p \cdot e(n) + I_n(n-1) \\ I_n(n) &= I_n(n-1) + K_i \cdot e(n) + K_{sat} \cdot e_{pi} \\ e_{pi} &= U_s - U(n) \end{aligned} \tag{13}$$

其中：

$$\begin{aligned} &\text{当 } U(n) \geqslant U_{\max} \text{ 时 } U_s = U_{\max}; \\ &\text{当 } U(n) \leqslant U_{\max} \text{ 时 } U_s = U_{\min} \end{aligned} \tag{14}$$

式中，U_s 表示抑制积分饱和算法的输出，$U(n)$ 表示本次 PI 调节器的计算结果，K_i 表示积分系数，K_p 表示积分系数，K_{sat} 表示抗饱和系数，$I_n(n)$ 为本次积分累加和，$U_{\max}$ 和 $U_{\min}$ 分别表示调节器输出的最大值和最小值，一般根据实际应用场景确定，本文中 $U_{\max}$ 和 $U_{\min}$ 可分别时间偏差 offset 的最大估计值和最小估计值。使用这种算法，可以将调节器的输出限定在需要的范围内，保证控制量不出现异常值。图 5 为抑制饱和位置式算法的流程图。

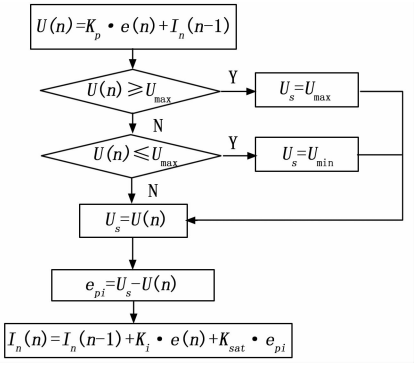

图 5　抑制积分饱和的 PI 算法

图 6 为 PI 控制器相位偏移调整原理框图，经过滤波算法的 TTE 时钟节点的相位偏移 offset 作为控制器的输入，利用 PI 控制器对本地时钟进行补偿。

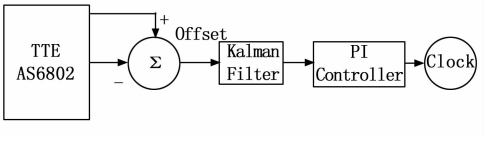

图 6　PI 控制器的相位偏移调整原理图

由前面分析可知，如果只进行相位偏移补偿，那么时钟同步精度将与同步周期密切相关，如果能同时对相位偏移以及频率偏移进行补偿则能达到更好的效果。图 7 为时钟频率偏移补偿调整原理图，与相位偏移调整系统相比，增加了频率偏移估计器来计算频偏估计值。

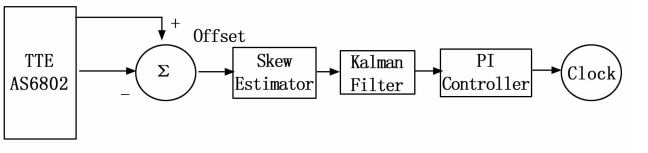

图 7　PI 控制器的频率偏移调整原理图

4　仿真分析

根据前面分析建立的时钟模型以及噪声模型，设初始相位偏差为 100 ns，频率偏差为 50 ppm，即节点 1 s 会出现 50 μs 的时间偏移[9]。由物理层抖动、链路抖动等带来的时间戳不确定性高斯噪声设为 10^{-7}。采样时间间隔为 5 m，采样 2 000 个点即 10 s，同步周期设为 10 ms，时钟固有频率均为 100 MHz，利用 Matlab 进行仿真。

如图 8 所示，相位偏移和频率偏移均叠加了高斯噪声，此外，相位偏移还受到频漂的影响。

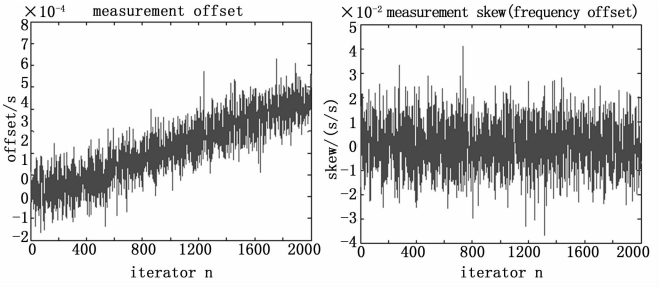

图 8　具有高斯噪声的相位偏移和频率偏移

采用 kalman 滤波、自适应 kalman 滤波[7]以及均值滤波三种常见的滤波方法进行滤波得到的结果如图 9 所示：

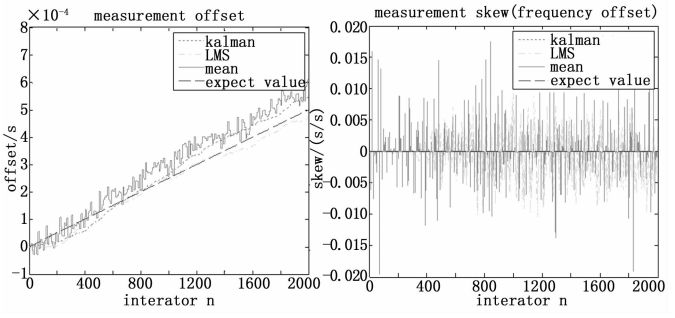

图 9　滤波后的相位偏移和频率偏移

均值滤波方法算法简单容易实现，但相较于卡尔曼滤波以及 LMS 滤波方法效果更差，从图 9 可以看出 kalman 滤波方法具备更好的追踪性能。

线性渐进补偿算法的补偿结果如图 10 所示。

利用线性渐进补偿算法进行线性补偿后，相位偏移能达到亚微秒量级，但由于没进行频率偏移补偿，频率偏移仍然较大。

PI 控制的补偿算法补偿结果如图 11 所示。

使用 PI 控制的补偿算法补偿之后，相位偏移达到了亚微妙量级，频率偏移也达到了良好的效果。

（下转第 281 页）