

实验八：时间同步

一 实验目的

- 1、掌握 TPSN 时间同步协议的基本原理。
- 2、在节点上实现 TPSN 时间同步协议。

二 实验原理

1、TPSN 协议基本介绍

TPSN (Timing-sync Protocol for Sensor Networks)，即传感网络的时间同步协议，该协议通过一对包含了节点时间信息的请求和应答报文对网络中的两个节点进行同步。TPSN 协议的实现有两个条件：

- 1、每个传感器节点都有一个唯一的 ID；
- 2、节点之间的无线通信链路是双向的，通过双向消息交换实现节点间的时间同步。

其优点在于在 MAC 层消息开始发送到无线信道时才给消息添加时标，消除了访问时间带来的时间同步误差，另外考虑了传播时间和接收时间，利用双向消息交换计算消息的平均延迟，提高了时间同步的精度。

TPSN 的双向通信过程如图 1-1 所示。

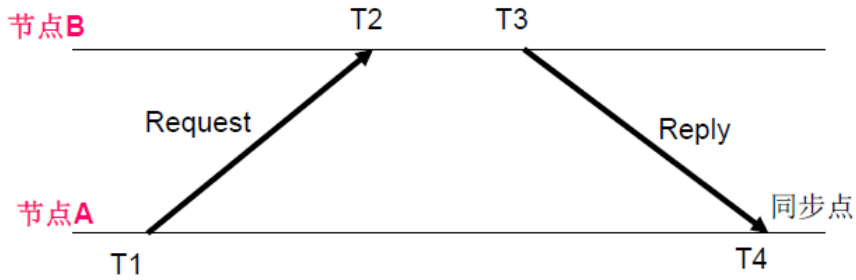


图 1-1 TPSN 的双向通信过程

由图分析可知， T_1 、 T_4 用节点 A 的本地时间记录， T_2 、 T_3 用节点 B 的本地时间记录。节点 A 记录下当前时刻 T_1 ，然后向节点 B 发送一个同步请求报文，节点 B 在接收到该报文后，记录下接收到时刻 T_2 ，并立即向节点 A 返回一个同步应答报文，把 T_2 和该报文的发送时刻 T_3 嵌入在报文中，当节点 A 接收到该报文时，记录下接收到时刻 T_4 。令 Δ 为当节点 A 的本地时刻为 T_1 时，节点 A 和 B 之间的时偏。由于 $T_1 \sim T_4$ 时间比较短，可认为当节点 A 的本地时刻为 T_4 时，其与节点 B 之间的时偏没有变化。假设报文的传输延迟相同，均为 d ，则可以得到，

$$\begin{cases} T_2 = T_1 + \Delta + d \\ T_4 = T_3 - \Delta + d \end{cases} \quad (1-1)$$

由式 (1-1) 可以得到，

$$\begin{cases} \Delta = \frac{(T_2 - T_1) - (T_4 - T_3)}{2} \\ d = \frac{(T_2 - T_1) + (T_4 - T_3)}{2} \end{cases} \quad (1-2)$$

根据式 (1-2)，我们只需在节点 A 和节点 B 双向通信的过程中依次获取 T_1 、 T_2 、 T_3 、 T_4 四个系统时间，即可计算出 Δ 和 d 。我们以节点 B 的系统时间为基准，校正节点 A 的系统时间。在 T_3 时刻打印节点 B 的时间，在 T_4 时刻计算并打印节点 A 的校正时间，即可完成两个节点的时间同步。

在程序设计思路，考虑到任务要求是通过两个节点通过通信以实现 TPSN 协议，因此可以在 TinyOS 自带的示例程序——BlinkToRadio 的基础上进行修改。首先需要将发送的包分为两类：请求包和回复包，然后按照 TPSN 协议的思路分别定义两种包的具体内容。其次在主程序中利用 Timer 组件提供的 `getNow()` 接口，在合适的时候获取节点各自不同的本地时间： T_1 、 T_2 、 T_3 、 T_4 。 T_2 、 T_3 存入回复包，发出请求的节点则保存 T_1 、 T_4 ，并在获取到回复包中的 T_2 、 T_3 之后计算出两个节点间的时间差 Δ 以及传播时延 d ，从而修改本地时间，达到时间同步的目的。

2. 实验中用到的接口和命令：

(1) 打印库

TinyOS 打印库 (`printf library`)：把调试信息从运行在节点上的 TinyOS 应用程序发送到 PC 显示屏上。

TinyOS 应用程序的调试工作非常不便。TOSSIM 仿真只能在 PC 上确认程序的逻辑功能。一旦运行在真实硬件上，不可预知的问题非常之多且难以避免。仅仅依靠节点上的三位 LED 灯来跟踪程序的运行，显然是不够的。最佳的解决方法就是在屏幕终端打印调试信息。

`printf` 库提供了一种终端打印功能。把节点连接到 PC 机的串口，调用打印命令（其语法风格同 C 语言），就可以在屏幕终端打印出调试信息。开发者只需：（1）包含 `PrintfC` 组件到顶层配件；（2）在任何调用 `printf()` 命令的组件里包含 “`printf.h`” 头文件。

整个打印库由 4 个位于 `tos/lib/printf` 中的文件组成：一个模块、一个配件、一个接口以及一个头文件。

`MainC.nc`：系统组件 `MainC` 的副本，可以自动地链接打印系统；

`PrintfC.nc`：为 TinyOS 程序提供打印功能的配件；

`PrintfP.nc`：实现打印功能的模块；

`printf.h`：头文件，指明打印消息的格式和刷新缓冲的大小。

请注意：在任何 TinyOS 应用程序里，只需简单地包含 `printf.h` 头文件，就可以调用 `printf` 命令和 `printfflush` 命令实现打印功能。

开发者以分布的方式在 TinyOS 应用程序的组件里提供字符串到 `printf()` 命令。这些字符串缓冲在 `PrintfP` 组件的一个中心位置，然后通过 `printfflush()` 命令以 TinyOS 串口消息的形式刷新到 PC 上。`printf()` 和 `printfflush()` 命令可以用 C 的形式调用，只需简单地包含 “`printf.h`” 头文件。默认的缓冲大小是 250 个字节（可通过 `makefile` 修改）。当已缓冲达一半以上就会自动地刷新，也可以直接通过 `printfflush()` 命令实现刷新。自动刷新可以满足大多数应用程序的要求，但当那些程序的代码有时间敏感时，还是建议直接刷新。

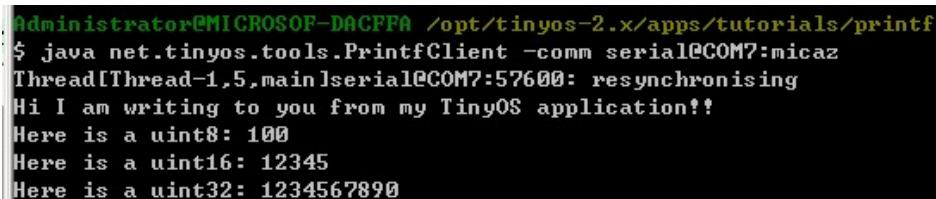
注意：TinyOS 2.1 和 TinyOS 2.0 的 `printf` 库略有不同，TinyOS 2.1 的 `Printf` 库不必在顶层配件里明确地绑定打印库组件 `PrintfC`。如下所示：

```
configuration TestPrintfAppC {
}
implementation {
    components MainC, TestPrintfC;
    TestPrintfC.Boot -> MainC;
}
```

只要组件包含 `printf.h` 头文件，就可以在该组件里调用 `printf` 和 `printfflush` 命令。

```
java net.tinyos.tools.PrintfClient -comm serial@COMN:telosb
```

```
event void Boot.booted() {  
    printf("Hi I am writing to you from my TinyOS application!!\n");  
    printf("Here is a uint8: %u\n", dummyVar1);  
    printf("Here is a uint16: %u\n", dummyVar2);  
    printf("Here is a uint32: %ld\n", dummyVar3);  
    printfflush();  
}
```



```
Administrator@MICROSOFT-DACFFA /opt/tinyos-2.x/apps/tutorials/printf  
$ java net.tinyos.tools.PrintfClient -comm serial@COM7:mica2  
Thread[Thread-1,5,main]serial@COM7:57600: resynchronising  
Hi I am writing to you from my TinyOS application!!  
Here is a uint8: 100  
Here is a uint16: 12345  
Here is a uint32: 1234567890
```

为了使用打印库，`tos/lib/printf` 目录必须在包含路径内。最简单的方法是在顶层应用程序的 `makefile` 里增加下面这行代码：

```
CFLAGS += -I$(TOSDIR)/lib/printf
```

```
CFLAGS += -DPRINTF_BUFFER_SIZE=XXX
```

必须在每个调用 `printf()` 命令的组件里，包含 “`printf.h`” 头文件。

(2) 在 `BlinkToRadio` 程序基础上进行修改，但是 `BlinkToRadio` 节点间的通信方式为广播，而在一个实验室里有很多组同学同时实验，大家都采用广播的话，节点间的干扰是必然的。所以请将广播改成了单播，并且把 `AMsend` 的组号按照学号进行修改。另外，将一些获取时间和计算的单步操作设置为 `atomic` 原子操作避免被抢占，保证准确获得实验结果。

三 实验报告要求

- ①实验名称
- ②实验内容说明
- ③程序源代码，包含必要的注释
- ④实验步骤，实验中出现的問題，观察到的结果
- ⑤实验总结

实验九：远程环境监控

一 实验目的

1. 掌握传感器数据读取方法
2. 掌握静态路由的配置方法
3. 掌握数据融合方法
3. 掌握上位机程序设计方法

二 实验要求

两个传感器节点采集温度、湿度或光照强度信息（如果节点上传感器缺失，则采集电压数据），通过静态路由传到中继节点，中继节点将收到的传感数据相同属性取最大值后转发给基站，基站节点通过串口与 PC 连接，将采集到的温湿度和光照强度和节点 ID、采样时间一起存入文本或数据库中，同时通过图形化界面显示出来。

三 实验提示

这个实验我们至少需要 4 个节点，两个作为传感器节点，采集光照强度或者电压的数据，在 Oscilloscope 程序基础上修改；一个作为中继节点，在 BlinkToRadio 程序上修改，另一个作为基站节点，通过串口与 PC 连接，使用 BaseStation 程序。

Oscilloscope 程序将传感器采集到的数据显示在电脑上。安装此程序的节点周期性的采样传感器（通过 DemoSensorC 读的电压值或者 HamamatsuS1087ParC 读的光照强度值），当采集到 10 个数据时，把它作为一个数据包广播出去。安装有 Oscilloscope 程序的节点每发送一个数据包，就切换第二盏灯；若是从其他节点那接收到数据包（用于告之自己的序列号或是更改了自己的采样频率），则切换第三盏灯；为了防止无线收发器之间的竞争，会切换第一盏灯。

基站程序（BaseStation）是 TinyOS 中一个最基本也是最有用的程序，它在串口和无线收发器之间扮演着桥梁的作用。当节点从串口收到数据包时，程序马上将该数据包发送给收发器，同时切换（原本亮的，变灭；反之亦然）LED 0；相反，当它从无线收发器接收到数据包时，程序会将此数据包发送给串口，同时切换 LED 1；若出现丢包（当一个节点发送速率大于另一个节点的接受速率），则切换 LED 2。

传感节点采用 TinyOS 默认的防碰撞算法，传感节点和中继节点都必须设计静态路由协议。

为了将节点 ID、采样时间、光照强度等数据存入文本，需要对 Oscilloscope.java 进行修改，在图形化显示之前将数据包中的数据和获取的系统时间通过文件流写入文本。如果要存入数据库，则要连接 MySQL 数据库。在 Oscilloscope.java 中，数据包 omsg 有各种 get 方法，可以获得版本、周期、节点 ID、数据包数量、采集数据等信息。只需取得其中的节点 ID 和采集数据，并利用 PrintWriter 写入文本即可，此处应注意捕获异常。

四 实验报告要求

- ①实验名称
- ②实验内容说明
- ③程序源代码，包含必要的注释
- ④实验步骤，实验中出现的問題，观察到的结果
- ⑤实验总结