Time Synchronization in Wireless Sensor Networks

Koninis Christos

Research Academic Computer Technology Institute
University of Patras, Greece

October 14th, 2009

Outline

Time Synchronization

Time synchronization algorithms provide a mechanism to synchronize the local clocks of the nodes in the network, to a global clock or relative to each other.

- The requirements for synchronization algorithms in WSN are more strict due to the constrains
- Small number of messages
- High accuracy/precision
- Small impact when we have interference (dropped packets)

Time Synchronization Applications

- Time synchronization is need in many WSN applications
  1. Cryptography: Authentication schemes often depend on synchronized time to ensure freshness, preventing replay attacks and other forms of circumvention.
  2. Data Fusion: Synchronized clocks are needed to resolve to otherwise ambiguous cases.
  3. Radio scheduling: From a low-power TDMA radio schedule.
  4. Target tracking: Tracking in the physical world requires a common reference frame.
  5. Coordinated actuation: Systems that incorporate coordinated actuation require synchronized time.
  6. Logging and Debugging: During design and debugging, it is often necessary correlate logs of many different nodes activities to understand the global system's behavior.

Traditional Time Synchronization Methods

The traditional synchronization methods are based in the NTP and GPS, but are impractical in the WSN environment.

- NTP: NTP is an ubiquitously adopted protocol for Internet time synchronization, but it it can take a long time before it achieves its optimal accuracy, when the network connectivity is poor. Also it listens to the network all the time and it assumes a continuous ability to transmit packets during normal operation.
- GPS: A GPS receiver provide an external, persistent, global clock. But there are energy and cost considerations.

Time Synchronization Methods

Many network synchronization algorithms have been proposed over the years, but most share the same basic design.

- A server sends a message containing its current value to a client.
- The client sends a request followed by a server response.
- The client measures the total round-trip time and estimates the one-way latency.
- The client may use a clock drift compensation algorithm to adjust the clock beetween.

Time Synchronization Methods

Many network synchronization algorithms have been proposed over the years, but most share the same basic design.

- A server sends a message containing its current value to a client.
- The client sends a request followed by a server response.
- The client measures the total round-trip time and estimates the one-way latency.
- The client may use a clock drift compensation algorithm to adjust the clock beetween.

Sources of time Synchronization Error

The enemy of precise network time synchronization is non-determinism. Latency estimates are confounded by random events that lead to asymmetric round-trip message delivery delays.

- Send Time - The time spent at the sender to construct the message. This includes kernel protocol processing and variable delays introduced by the operating system.
- Access Time - Delay incurred waiting for access to the transmit channel. This is specific to the MAC protocol in use.
- Propagation Time - The time needed for the message to transit from sender to receivers once it has left the sender. This time is very small as it is simply the physical propagation time of the message through the media.
- Receive Time - Processing required for the receiver's interface to receive the message from the channel and notify the host of its arrival.

Reference Broadcast Synchronization introduction

RBS is an time synchronization protocol which instead of estimating the error, exploits the broadcast channel available to remove as much as possible from the critical path.

- RBS is used to synchronize a set of receivers with one another.
- Nodes periodically send messages to their neighbors using physical-layer broadcast.
- The message has no timestamp, nor it is important exactly when it is sent.
- The recipients use the message arrival time as a point of reference for comparing their clocks.
- This method removes the sender's nondeterminism from the critical path and improves accuracy.

RBS Protocol - phase offset estimation

The RBS protocol used to compute the phase offset of n receivers is:

- A transmiter broadcasts m reference packets.
- Each of the n receivers records the time that the reference was observed, according to its local clock.
- The receivers exchange their observations.
- Each receiver i can compute its phase offset to any other receiver j as the average of the phase offsets implied by each pulse received by both nodes i and j.

RBS Protocol - clock skew estimation

- All clocks don't run at the same rate due to frequency differences in the oscillators.
- The phase difference between two clocks will change over time.
- We need a clock skew estimation algorithm.
- Instead of averaging the phase offsets from multiple observations we perform a least-squares linear regression.
- This offers a fast way to find the best fit line through the phase error observations over time.

Outline

The Wiselib Time Synchronization Concept

The Time Synchronization Concept

```
concept TimeSynchronization
{
   typedef ... OsModel;
   typedef ... Radio;
   typedef ... Os;

   void set_os( Os* os )
   void enable( void );
   void disable( void );
};
```

The Time Synchronization Concept - Possible Future Extensions

How the application will know when/if the timer is synchronized?

- Add an function for polling the protocol.
- Add a callback, we can register functions that the protocol will call when it is synchronized.

The Time Synchronization Concept - Possible Future Extensions

The Time Synchronization Concept

```
concept TimeSynchronization
{
   typedef ... OsModel;
   typedef ... Radio;
   typedef ... Os;

   void set_os( Os* os )
   void enable( void );
   void disable( void );

   boolean isSynchronized();

   template<class T, void (T::*TMethod)( void )>
   static inline int reg_recv_callback( Os *os, T *obj_pnt );

   static inline void unreg_recv_callback( Os *os, int idx );

};
```

Wiselib + Time Synchronization Challenges

When developing Time Synchronization with Wiselib we have to deal with some basic challenges.

- Extra latency when using high level OS calls.
- Timestamping cannot be done in low layers (e.g. MAC).

Future solution to this problem will include.

- Concept and implementations for accessing lower layers of the radio stack.

Time Synchronization Example

Time Synchronization Example

```
typedef wiselib::iSenseOsModel Os;
typedef wiselib::rbs_synchronization<Os,Os::Radio,Os::Debug,OS::Clock> rbs_synchron
typedef wiselib::tpsn_synchronization<Os,Os::Radio,Os::Debug,OS::Clock> tpsn_synchr

class iSenseDemoApplication
{
...
private:
rbs_synchronization_t rbs;
tpsn_synchronization_t tpsn;
...
void iSenseDemoApplication::boot(void)
{
rbs.enable();
rbs.disable();
tpsn.enable();
tpsn.disable();
}
...
}
```