CS649
Sensor Networks
IP Lecture 9: Synchronization

I-Jeng Wang
http://hinrg.cs.jhu.edu/wnsn06/
Outline

- Description of the problem: axes, shortcomings
- Reference-Broadcast Synchronization
  - Reference-Broadcast Synchronization
  - Federation of broadcast domains
  - Post-facto synchronization
- Alternatives
  - TPSN
  - FTSP
Does timesync matter?

- Internet Time Synchronization
  - Critical in some contexts (e.g. crypto, distributed packet traces)
  - A convenience in many other contexts
- Sensor Network Synchronization
  - Fundamental to its purpose: data fusion
  - *Physical time* needed to relate events in the *physical world*
Heterogeneity

• Time sync is critical at many layers
  • Beam-forming, localization, distributed DSP
  • Data aggregation & caching
  • TDMA guard bands
  • “Traditional” uses (debugging, user interaction…)
• But time sync needs are non-uniform
  • Maximum Error
  • Lifetime
  • Scope & Availability
  • Efficiency (use of power and time)
  • Cost and form factor
Beam-forming, localization, distributed DSP: small scope, short lifetime, high precision
Target tracking: larger scope, longer lifetime, but lower required precision
Isn’t this solved?

- NTP (Network Time Protocol)
  - Ubiquitous in the Internet
  - Variants appearing in sensor networks
- 802.11 synchronization
  - Precise clock agreement within a cluster
- GPS, WWVB, other radio time services
- High-stability oscillators (Rubidium, Cesium...
NTP

• The “gold standard” -- used by millions
• The basic idea: measure round-trip-time

![Diagram showing round-trip time between Host A and Host B with timestamps t0, t1, t2, and t3.]

• One-way delay: \( \Delta = \frac{(t_3 - t_0) - (t_2 - t_1)}{2} \)

• Offset: \( t_1 - (t_0 + \Delta) = \frac{t_1 - t_0 + t_2 - t_3}{2} \)
Basic Problems

- With NTP-like schemes
  - Assumes the same forward and reverse path
  - Assumes delay is deterministic
- With all network-based schemes
  - Hard to measure the exact time of events
- With all forms of time synchronization
  - Clocks run at different rates
  - Clocks change rates over time (drift, skew)
  - Basic tradeoff: longer experiments mean more data collected, but older data is less useful
So what’s wrong?

- Existing work is a **critical** building block

**BUT...**

- This isn’t the Internet
  - Important assumptions no longer hold
    - (fewer resources available for synchronization...)
  - Sensor apps have stronger requirements
    - (...but we have to do better than the Internet anyway)

- **Energy, energy energy:**
  - Listening to the network is no longer free; even occasional CPU use can have a major impact
Infrastructure vs. Ad-Hoc

• “NTP provides UTC to the entire Internet”
• Infrastructure isn’t ubiquitous in sensor nets
  • GPS doesn’t work indoors, in the forest, underwater, on Mars...
• What happens without infrastructure?
“Mundane” Reasons

- Cost
  - We can’t put a $500 Rubidium oscillator or a $50 GPS receiver on a $5 sensor node
- Form factor
  - Nodes are small, extra components are large
- Not actually a mundane limitation if it changes the economics of the sensor net
Leveraging the Medium

- Strict layering and levels of abstraction prevent us from exploiting domain knowledge
- Wireless networks often use network interfaces with physical-layer broadcasts
- Reference Broadcast Synchronization takes advantage of this to remove most of the non-determinism from the critical path
Problem: Many sources of unknown, nondeterministic latency between timestamp and its reception

At the tone: $t=1$

NIC

Send time

Access Time

Propagation Time

Physical Media

Receiver

Receive Time

Sender
Reference Broadcasts*

Sync 2 receivers *with each other*, NOT sender with receiver

RBS reduces error by removing much of it from the critical path.

Traditional critical path: From the time the sender reads its clock, to when the receiver reads its clock.

RBS: Only sensitive to the differences in receive time and propagation delay.
Receiver Determinism
Basic Mechanism Description

- Some node sends $m$ broadcast “reference” messages
- Each of $n$ receivers records the time the reference message was received
- Receivers exchange their observations
- Receiver $i$ computes offset to receiver $j$ as the average of phase offsets

\[
Offset[i, j] = \frac{1}{m} \sum_{k=1}^{m} \left( T_{j,k} - T_{i,k} \right)
\]
Clock Skew

• Clocks are implemented using digital oscillators
  • Accuracy: difference between expected and actual frequency (parts per million)
  • Stability: variations in frequency over short and long timescales
• Result: Phase difference changes over time due to frequency differences
• Solution: Instead of averaging phase offsets, use least-squares linear regression
  • Assumption: Frequency difference is constant
Regression experiment
Comparison to NTP

- Second implementation:
  - Compaq IPAQs (small Linux machines)
  - 11mbit 802.11 PCMCIA cards
- Ran NTP, RBS-Userspace, RBS-Kernel
  - NTP synced to GPS clock every 16 secs
  - NTP with phase correction, too; it did worse (!)
- In each case, asked 2 IPAQs to raise a GPIO line high at the same time; differences measured with logic analyzer
NTP Comparison: Low Network Load

Cumulative Error Probability

Synchronization Error with Light Network Load

- RBS-Kernel
- RBS-Userspace
- NTP-Kernel
- NTP-Userspace

Error (usec)
NTP Comparison: High Network Load

RBS degraded slightly (6µs to 8us); NTP degraded severely (51µs to 1542µs)
Multi-Hop RBS

- Some nodes broadcast RF synchronization pulses
- Receivers in a neighborhood are synced by using the pulse as a time reference. (The pulse senders are not synced.)
- Nodes that hear both can relate the time bases to each other
Time Routing

The physical topology can be easily converted to a logical topology; links represent possible clock conversions.

Use shortest path search to find a “time route”; Edges can be weighted by error estimates.
Multi-Hop RBS

Error (and std dev) over multiple hops, in usec

- 1 Hop: 1.85 +/- 1.28
- 2 Hop: 2.73 +/- 1.91
- 3 Hop: 2.73 +/- 2.42
- 4 Hop: 3.68 +/- 2.57
“Post-Facto” Sync

- Most protocols stay synced all the time
- Post-facto sync:
  - Clocks start out unsynchronized
  - A set of receivers waits for an interesting event
  - Locally timestamp an event when it happens
  - After the fact, reconcile clocks

- Avoids wasting energy on unneeded sync; it’s easier to predict the past than future
Post-Facto Sync

7µsec error after 60 seconds of silence

- Similar to NTP in many ways -- uses round-trip time measurement with 2 packets
- Achieves a network-wide synchronization by constructing a tree and synchronizing each node with its parent
- Depends on being able to modify the MAC, to do time-stamping very close to transmission
- Demonstrates 2x better performance than RBS based on analysis and experimentation
RBS vs TPSN on Accuracy

- Uncertainties in Radio Message Delivery
  - Send Time
  - Access Time
  - Transmission Time
  - Propagation Time
  - Reception Time
  - Receive Time
- RBS (receiver-receiver synchronization)
  - Eliminate impacts of the send and access time
  - Can remove the receive time with minimal OS modification
  - Source of errors: propagation and reception time
  - Does not require access to the low-levels of the OS
- TPSN (sender-receiver synchronization)
  - Remove the send, access, and receive time by MAC-layer time-stamping
  - Eliminate the propagation time via two-way handshakes
  - Require construction of a tree (level discovery phase)
The Flooding Time Synchronization Protocol

M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi

SenSys 2004
Summary

- Achieve a network-wide synchronization through one-way radio broadcast
  - Does not compensate for propagation errors as in TPSN
- MAC Layer Time-stamping
- Clock drift management
- Multi-hop time synchronization
Analysis of Delay in Transmission and Reception

- Interrupt Handling Time
- Encoding Time
- Decoding Time
- Byte Alignment Time

<table>
<thead>
<tr>
<th>Time</th>
<th>Magnitude</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send and Receive</td>
<td>0 – 100 ms</td>
<td>nondeterministic, depends on the processor load</td>
</tr>
<tr>
<td>Access</td>
<td>10 – 500 ms</td>
<td>nondeterministic, depends on the channel contention</td>
</tr>
<tr>
<td>Transmission / Reception</td>
<td>10 – 20 ms</td>
<td>deterministic, depends on message length</td>
</tr>
<tr>
<td>Propagation</td>
<td>&lt; 1μs for distances up to 300 meters</td>
<td>deterministic, depends on the distance between sender and receiver</td>
</tr>
<tr>
<td>Interrupt Handling</td>
<td>&lt; 5μs in most cases, but can be as high as 30μs</td>
<td>nondeterministic, depends on interrupts being disabled</td>
</tr>
<tr>
<td>Encoding plus Decoding</td>
<td>100 – 200μs, &lt; 2μs variance</td>
<td>deterministic, depends on radio chipset and settings</td>
</tr>
<tr>
<td>Byte Alignment</td>
<td>0 – 400μs</td>
<td>deterministic, can be calculated</td>
</tr>
</tbody>
</table>
Time Stamping

- Using periodic radio broadcast to synchronize receivers to the sender
- Time stamp of the sender is embedded in the transmission message
- Each broadcast provide a reference point (a global-local time pair) to each receiver for estimating the clock offset between the sender and the receiver
- The proposed time stamping mechanism reduces the jitter of interrupt handling and encoding/decoding times
  - Achieved 1.4\(\mu\)s average error and 4.2\(\mu\)s maximum error in experiments
Clock Drift Management

• The offset between two local clocks can change in a linear fashion due to clock drifts.
• Linear regression can be used to estimate the skew from multiple reference points as done in RBS.
Multi-hop Time Synchronization

• Basic scheme:
  • A single root is required for global synchronization
  • Each node synchronized itself based on multiple received reference points
  • Once a node is synchronized, it broadcasts synchronization messages to its neighbors

• Synchronization Message Format
  • timeStamp
  • rootID
  • seqNum: set and increment by the root; each node inserts the most recent received seqNum to its broadcast messages

• Managing Redundant Information: message filtering
• The root election problem: through broadcast messages without additional handshakes
Experiment Results

- $ID_1$ off
- Reset random nodes
- Turn off odd-ID nodes
- Turn on odd-ID nodes
# Accuracy Comparisons (Single Hop)

<table>
<thead>
<tr>
<th></th>
<th>RBS</th>
<th>TPSN</th>
<th>FTSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send and Receive</td>
<td>✓/✓✓</td>
<td>✓/✓✓</td>
<td>✓/✓✓</td>
</tr>
<tr>
<td>Access</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Transmission / Reception</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Propagation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Interrupt Handling</td>
<td>✓✓✓✓</td>
<td></td>
<td>✓✓✓✓</td>
</tr>
<tr>
<td>Encoding plus Decoding</td>
<td>✓✓✓✓</td>
<td>✓✓✓✓</td>
<td>✓✓✓✓</td>
</tr>
<tr>
<td>Byte Alignment</td>
<td>10-30µs</td>
<td>10-30µs</td>
<td>1-2µs</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>Magnitude</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send and Receive</td>
<td>0 – 100 ms</td>
<td>nondeterministic, depends on the processor load</td>
</tr>
<tr>
<td>Access</td>
<td>10 – 500 ms</td>
<td>nondeterministic, depends on the channel contention</td>
</tr>
<tr>
<td>Transmission / Reception</td>
<td>10 – 20 ms</td>
<td>deterministic, depends on message length</td>
</tr>
<tr>
<td>Propagation</td>
<td>&lt; 1 µs for distances up to 300 meters</td>
<td>deterministic, depends on the distance between sender and receiver</td>
</tr>
<tr>
<td>Interrupt Handling</td>
<td>&lt; 5 µs in most cases, but can be as high as 30 µs</td>
<td>nondeterministic, depends on interrupts being disabled</td>
</tr>
<tr>
<td>Encoding plus Decoding</td>
<td>100 – 200 µs, &lt; 2 µs variance</td>
<td>deterministic, depends on radio chipset and settings</td>
</tr>
<tr>
<td>Byte Alignment</td>
<td>0 – 400 µs</td>
<td>deterministic, can be calculated</td>
</tr>
</tbody>
</table>
Comparisons– Multiple Hops

- RBS and FTSP are more robust to topology changes than TPSN since no network structure needs to be maintained for multi-hop synchronization
- Communication overhead for each synchronization per node
  - RBS: 1.5 messages (0.5 for reference broadcast, 1 for time stamp exchange)
  - TPSN: 2 messages (1 to parent and one response)
  - FTSP: 1 messages
- Convergence?