Time in Wireless Embedded Systems

Thomas Schmid

thomas.schmid@ucla.edu

Dissertation Chair: Mani B. Srivastava
Importance of time in Wireless Embedded Systems

• Passage of time
  ‣ Polled MAC Protocols
  ‣ Ranging

• Synchronization of time
  ‣ post-facto vs. pro active
  ‣ local vs. global
  ‣ Scheduled MAC Protocols
  ‣ Localization
  ‣ Coordinated action
Ingredients of Time

- Clock source
  - Measures passage of time

- Synchronization strategy
  - Measure clock offset
    - Clock granularity
    - Network / System characteristics
  - Resynchronization rate
    - Stability of clock
    - Ambient characteristics
### Prior Work
Prior Work
Prior Work

• Improvements in time sources
  ‣ New quartz cuts, TCXO, MCXO, chip-scale atomic

• Synchronization Strategies
  ‣ 2 message synchronization exchange
  ‣ RX-RX for wireless
  ‣ Post-facto synchronization
  ‣ Explicit ambient sources
  ‣ Rate Adaptive Time Synchronization
State of the Art

- **Time Sources**
  - Lots of power and $$
    - Chip-scale atomic clocks / rubidium clocks
    - <10 ppb clock stability, hundreds of mW to several Watts
  - Lots of power, $$, and outdoors
    - GPS
      - 10 ns synchronization accuracy, hundreds of mW

- **Synchronization Strategies**
  - Wired (Ethernet)
    - Precision Time Protocol, IEEE 1588
    - < 100 ns synchronization accuracy
Low-Power Wireless Systems

- In 2004, the Flooding Time Synchronization Protocol (FTSP) achieved on average 1.5 μSecond single-hop synchronization accuracy on the Mica2 platform.
- It was not low-power since the 7.37 MHz crystal had to be on all the time, no low-power listening possible.
- Assumed temperature to be constant between resynchronization attempts.
- Used elaborate timestamping and regression techniques to cope with communication jitters.
... and How Hardware Solved it

SFD Measurements from two Tmote Skys

Time [μs]

Normalized Count
... and How Hardware Solved it

SFD Measurements from two TMote Skys

Normalized Count

0.005

0.01

0.015

0.02

0.025

Time [μs]

3.06 3.08 3.1 3.12 3.14 3.16 3.18 3.2 3.22 3.24 3.26

3.14μs
... and How Hardware Solved it

SFD Measurements from two TMote Skys

Normalized Count

0.005
0.01
0.015
0.02
0.025

Time [μs]

3.06 3.08 3.1 3.12 3.14 3.16 3.18 3.2 3.22 3.24 3.26

3.14μs

0.191μs
The TI CC2420 brings offset measurement to a new accuracy level

Clock source errors become more significant!
Goal of this Dissertation
Goal of this Dissertation

High Accuracy  sub-μSecond
Goal of this Dissertation

High Accuracy  

sub-\(\mu\)Second

Robust to Disconnects
Goal of this Dissertation

High Accuracy  sub-\mu\text{Second}

Robust to Disconnects

Resource Constraint  \mu\text{W}
Dissertation Overview

1. Dual Clock Systems for Local Temperature Compensation
   ‣ Crystal Compensated Crystal Based Timer (XCXT)

2. Influence of Temperature on Time Synchronization Accuracy
   ‣ New models of how temperature influences the accuracy
   ‣ Interaction of time resolution, temperature change, estimation accuracies

3. Autonomous Temperature Calibration
   ‣ Temperature Compensated Time Synchronization (TCTS)

4. Power and Accuracy Aware Timer Unit
   ‣ Back to the dual-clock design
   ‣ High-Low Timer
Dissertation Overview

1. Dual Clock Systems for Local Temperature Compensation
   - Crystal Compensated Crystal Based Timer (XCXT)

2. Influence of Temperature on Time Synchronization Accuracy
   - New models of how temperature influences the accuracy
   - Interaction of time resolution, temperature change, estimation accuracies

3. Autonomous Temperature Calibration
   - Temperature Compensated Time Synchronization (TCTS)

4. Power and Accuracy Aware Timer Unit
   - Back to the dual-clock design
   - High-Low Timer
### Dissertation Overview

1. Dual Clock Systems for Local Temperature Compensation
   - Crystal Compensated Crystal Based Timer (XCXT)

2. Influence of Temperature on Time Synchronization Accuracy
   - New models of how temperature influences the accuracy
   - Interaction of time resolution, temperature change, estimation accuracies

3. Autonomous Temperature Calibration
   - Temperature Compensated Time Synchronization (TCTS)

4. Power and Accuracy Aware Timer Unit
   - Back to the dual-clock design
   - High-Low Timer
Local Clock Sources
Relative Frequency Drift of Clock Sources vs. Cost

- Quartz Osc (≤1 mW)
- TCXO (≤10 mW)
- OCXO (~1 W)
- Atomic Clock (>1 W)
Local Clock Sources

Relative Frequency Drift of Clock Sources vs. Cost

- Quartz Osc (≤1 mW)
- TCXO (≤10 mW)
- XCXT
- OCXO (~1 W)
- Atomic Clock (>1 W)

Frequency Drift [ppm] vs. Price [$]
Temperature Dependence of Crystals

- AT-Cut quartz crystal in most digital devices today
- Characteristic cubic temperature curve with slope dependent on slight variations in cut angle
Differential Drift (I)

AT-cut Temperature vs. Frequency Drift

- Temperature [°C]
- Frequency Drift [ppm]

Graph showing the relationship between temperature and frequency drift for AT-cut crystals.
Differential Drift (1)

AT-cut Temperature vs. Frequency Drift

Frequency Drift [ppm]

Temperature [°C]

-40 -20 0 20 40 60 80
-30 -25 -20 -15 -10 -5 0 5 10 15 20

Wednesday, September 16, 2009
Differential Drift (I)

AT-cut Temperature vs. Frequency Drift

Temperature [°C]
-40 -20 0 20 40 60 80
Frequency Drift [ppm]
-30 -25 -20 -15 -10 -5 0 5 10 15 20

δC

δC₁₀
Differential Drift (1)

AT-cut Temperature vs. Frequency Drift

\[ \delta C_{12} \]

\[ \delta C_{10} \]

Temperature [C]

Frequency Drift [ppm]
Differential Drift (2)

Frequency Drift vs. Difference of Frequency Drift

\[ \delta C_{10} \text{ [ppm]} \]

\[ \delta C_{12} \text{ [ppm]} \]

\( \theta 8' \text{ vs } 1' \)  \( \theta 8' \text{ vs } 4' \)  \( \theta 8' \text{ vs } 6' \)
Differential Drift Algorithm

- Differential Drift algorithm has two parts
  1. One time Calibration
  2. Runtime compensation
- Calibration is similar to TCXO calibration at manufacturing time
- Runtime compensation is similar to TCXO’s runtime compensation

**Important difference:**
Differential Drift *represents* the quartz temperature, while a TCXO has to measure this *indirectly* through a temperature sensor.
Compared to Simulation?

- **Simulation:**
  - mean: -0.00109ppm
  - stddev: 0.14ppm

- **XCXT Implementation:**
  - mean: 0.47ppm
  - stddev: 0.31ppm
XCXT Power Consumption

- Average Current: 474 μA at 3.001V
- Average Power: 1.4 mW compared to 6 mW of a TCXO
What can be improved

- Replaces temperature sensor of TCXO
- This is all local and ignores the communication capabilities of wireless embedded systems
- Improvements:
  - Use communication to calibrate your clocks
  - Use a low and high frequency crystal for power efficiency
Effects of Temperature on Time Synchronization
Clock error

\[ c(t) \]
counter in an embedded system

\[ \lfloor f_0 \cdot t \rfloor \]
floor operator due to digital nature of system

\[ g(\kappa(\tau)) \]
value of counter after \( t \) seconds

\( \kappa(\tau) \) is the temperature at time \( \tau \), and \( g(\cdot) \) is the frequency error for a particular temperature
Effect of Temperature on Time Synchronization

- Temperature changes the local clock frequency

\[ c(t) = \left[ f_0 \cdot t + f_0 \cdot \int_0^t g(\kappa(\tau)) \, d\tau \right] \]

- Synchronization error consists of two parts
  1. Error due to quantization
  2. Error induced by change in temperature

- The longer the resynchronization interval, the more significant the second error becomes
Theoretic Upper Bound

- Assume synchronization exchange similar to TPSN or IEEE 1588
- One can show that the synchronization error is bounded by
  \[ \varepsilon_Q(t = T) \leq 2f_0T \cdot (1 + \delta_{\text{max}}) - \frac{(f_0T(1 + \delta_{\text{min}}) - 1)^2}{f_0T + 1} - (f_0T - 1) \]
- \( T \): time between resynchronization attempts
- \( f_0 \): nominal clock frequency
- \( \delta_{\text{min}}/\delta_{\text{max}} \): minimum and maximum frequency error
We can now bound the time synchronization error interval between synchronization messages. We can see that there are two different error sources of the employed clock circuit. Putting it all together we find that the maximum bound the integral over the temperature dependent frequency error by

$$\epsilon_Q(t = T) \leq 2f_0T \cdot (1 + \delta_{max}) - \frac{(f_0T(1 + \delta_{min}) - 1)^2}{f_0T + 1} - (f_0T - 1)$$

**Figure 3.2** illustrates the worst case synchronization accuracy for a given time limit behavior.
We can now bound the time synchronization error:

\[ \varepsilon_Q(t = T) \leq 2f_0T \cdot (1 + \delta_{\text{max}}) - \frac{(f_0T(1 + \delta_{\text{min}}) - 1)^2}{f_0T + 1} - (f_0T - 1) \]

Worst Case Time Synchronization Error

The graph illustrates the worst case synchronization accuracy for a given time interval. From the definition, we know that maximum synchronization error is bounded by

- Maximum Synchronization Error \[\text{tics}\]
- Resynchronization Interval \[T \text{ [s]}\]

Different cases are shown for different frequencies:
- \(f_0=32\text{kHz}\)
- \(f_0=1\text{MHz}\)
- \(f_0=8\text{MHz}\)
We can now bound the time synchronization error

\[ \epsilon_Q(t = T) \leq 2f_0T \cdot (1 + \delta_{\text{max}}) - \frac{(f_0T(1 + \delta_{\text{min}}) - 1)^2}{f_0T + 1} - (f_0T - 1) \]

**Worst Case Time Synchronization Error**

\[ f_0T \cdot (2\delta_{\text{max}} - 2\delta_{\text{min}} - \delta^2_{\text{min}}) \]
We can now bound the time synchronization error regions:

\[ \varepsilon_Q(t = T) \leq 2 f_0 T \cdot (1 + \delta_{\text{max}}) - \frac{(f_0 T (1 + \delta_{\text{min}}) - 1)^2}{f_0 T + 1} - (f_0 T - 1) \]

Worst Case Time Synchronization Error

\[ T^* = \frac{4}{f_0 \cdot (2 \delta_{\text{max}} - 2 \delta_{\text{min}} - \delta^2_{\text{min}})} \]

4 Tics
A Better Bound

• It is unrealistic that the frequency error changes from $\delta_{\text{min}}$ to $\delta_{\text{max}}$ within the resynchronization period.

• Two approaches
  1. Scanning temperature traces (Analytic)
  2. Simulation
     - Enhanced version of Castalia, to simulate and find the maximum change in frequency error for different synchronization periods.
     - Modifications available as Open Castalia at http://projects.nesl.ucla.edu/~thomas/opencastalia.html
Analytic vs. Simulation

Maximum Synchronization Error [s]

Resynchronization Interval T [s]

1kHz Analytic + 32kHz Analytic *
1kHz Simulation 32kHz Simulation
Conclusion

- There is an optimal resynchronization rate $T^*$
  - Below $T^*$ no gain is made due to quantization
  - Above $T^*$ you loose accuracy due to change in temperature
- Dual between time synchronization accuracy and frequency error estimation
  \[
  \delta_Q = \frac{(t_{A2} - t_{B2}) - (t_{A1} - t_{B1})}{t_{A2} - t_{A1}} = \frac{\int_{t_{A1}}^{t_{A2}} g(\kappa(\tau))d\tau}{T}
  \]
- This suggests that there is also an optimal point to find current frequency error
  - Below $T^*$ quantization degrades estimation accuracy
  - Above $T^*$ you loose accuracy due to change in temperature
Frequency Error Estimation Accuracy

![Graph showing Frequency Error Estimation Accuracy vs Synchronization Interval.

- **Estimation Error [ppm]**
- **Synchronization Interval [s]**
- **Sim f₀ = 1 kHz**
- **Sim f₀ = 32 kHz**
- **δₑ f₀ = 1 kHz**
- **δₑ f₀ = 32 kHz**
- **Temperature Limit**

Wednesday, September 16, 2009
Frequency Error Estimation Accuracy

![Graph showing frequency error estimation accuracy over synchronization interval.](image)

- **Quantization**
  - Temperature
  - Sim $f_0 = 1\text{kHz}$
  - Sim $f_0 = 32\text{kHz}$
  - $\delta_E f_0 = 1\text{kHz}$
  - $\delta_E f_0 = 32\text{kHz}$

**Synchronization Interval [s]**

- Time in Wireless Embedded Systems

- Temperature Limit

**Estimation Error [ppm]**

- Frequency Error Estimation Accuracy - 95% Confidence Intervals

- $\text{Sim } f_0 = 1\text{kHz}$
- $\text{Sim } f_0 = 32\text{kHz}$
- $\delta_E f_0 = 1\text{kHz}$
- $\delta_E f_0 = 32\text{kHz}$

- Temperature Limit

**Notes:**

- $f = 32\text{kHz}$
- $\text{Sim } f_0 = 1\text{kHz}$
- $\text{Sim } f_0 = 32\text{kHz}$
- $\delta_E f_0 = 1\text{kHz}$
- $\delta_E f_0 = 32\text{kHz}$
Frequency Error Estimation Accuracy

![Graph showing synchronization interval vs. estimation error and quantization error.](image)

- **Quantization**: 
  - Sim $f_0 = 1\text{kHz}$
  - Sim $f_0 = 32\text{kHz}$

- **Estimation Error**: 
  - $\delta_E f_0 = 32\text{kHz}$
  - $\delta_E f_0 = 1\text{kHz}$

- **Temperature Limit**

---

Wednesday, September 16, 2009
Temperature Compensated Time Synchronization (TCTS)
Several Forces at Play

- Measuring offsets: solved through new radio hardware
- Cannot increase resynchronization interval above $T^*$
  - Higher error in synchronization accuracy
  - Higher error in frequency estimation
- We have to temperature compensate the local clock source
  - Use TCXO for local clock source
  - TCXO’s are expensive in $$ and power!
Several Forces at Play

• Measuring offsets: solved through new radio hardware

• Cannot increase resynchronization interval above $T^*$
  ‣ Higher
  ‣ Higher

• We have to temperature compensate the local clock source
  ‣ Use TCXO for local clock source
  ‣ TCXO’s are expensive in $$ and power!
Temperature Compensated Time Synchronization

- Use communication to learn frequency vs. temperature curve
- Use local temperature knowledge to compensate for temperature induced frequency error
- Measuring local temperature is cheaper than communication

- Assumptions
  - Root or master node has accurate time (GPS, TCXO, etc)
  - No assumption on synchronization mechanism!
Power to Measure Temperature

- Sensirion SHT11
  - 220ms, 287μJ
- MSP430 On-Chip
  - 35ms, 66.5μJ

- Or XCXT for direct crystal temperature measurement
Power to Measure Temperature

- Sensirion SHT11
- 220ms, 287μJ

- MSP430 On-Chip
- 35ms, 66.5μJ

Send 1 radio message ~600μJ

- Or XCXT for direct crystal temperature measurement
Calibration

- During calibration, use time synchronization with a resynchronization rate of $T^*$ for maximal precision
- Store current frequency error and temperature in a table
- Go to compensation
Compensation

- Measure temperature
  - *If* temperature is not in the calibration table, go to calibration
  - *Else*, use frequency error estimate from table for compensation
- After $T > T^*$ resynchronize to correct for accumulated time errors
State Diagram

- **Init**
- **Compensate**
- **Calibrate**

**Known Drift**

**Unknown Drift**

**Measure Temperature**
Evaluating TCTS

- Resynchronization interval is not constant
- When TCTS starts, it often needs to calibrate
  \[ \rightarrow \text{synchronization interval near } T^* \]
- Over time, TCTS is more often in compensation state where
  synchronization interval is additively increased
  \[ \rightarrow \text{longer synchronization intervals} \]
- If at resynchronization the error between estimated global
time and global time is bigger than \( \varepsilon \), multiplicative decrease
Average Hourly Beacon Interval of TCTS

![Graph showing the average hourly beacon interval over simulation time with temperature on the side. The graph illustrates the relationship between beacon interval and temperature over different periods of simulation time.](image-url)
Robustness

- Once TCTS is calibrated, it essentially provides a TCXO to the software.
- In challenged networks, where communication is not as reliable, TCTS can provide essential robustness towards communication loss.
- A legacy time synchronization protocol has to rely on the last measured frequency error.
FTSP vs. TCTS if Synchronization is Stopped

- Sync Stopped
- FTSP
- TCTS
- Temperature

![Graph showing synchronization error vs. time with temperature factor. The graph compares FTSP and TCTS under synchronization stop conditions.]
TCTS Calibration Accuracy

\[ \delta_Q \text{ [ppm]} \]

\[ \text{Temperature [C]} \]

TCTS $\delta_Q$ Estimation + Tuning Fork Model
Frequency Error Estimation Accuracy

![Graph showing frequency error estimation accuracy over synchronization interval. The graph includes lines representing different frequency sources and their synchronization intervals. The x-axis represents the synchronization interval in seconds, ranging from 10 to 10,000. The y-axis represents the estimation error in ppm, ranging from 1e-7 to 0.0001. Lines are labeled with different frequency sources: Sim $f_0=1$kHz, Sim $f_0=32$kHz, $\delta_E f_0=32$kHz, and $\delta_E f_0=1$kHz. The graph also highlights the temperature and quantization effects on the estimation error.]
Architectural Support
• $\tau = 1/(T \cdot f_0)$ Quantization limit

• Increasing $T$ is not possible because of temperature, thus we need to increase the frequency $f_0$
  
  ▶ High frequency clocks are needed for precision time synchronization
  
  ▶ High frequency clocks consume a lot of power
  
  ▶ High frequency clocks negate the benefits of ultra low-power sleep in duty cycled systems
Until now, no time synchronization could provide high precision (<30.5 µs) and low-power sleep (<200 µA) at the same time. High frequency clocks negate the benefits of ultra low-power sleep in duty cycled systems.

- \( \tau = \frac{1}{(T \cdot f_0)} \) Quantization limit

- Increasing \( T \) is not possible because of temperature, thus we...
Dual-Clock Hardware Design

- Use two crystals, one fast, one slow
- Slow crystal keeps coarse time and is always on
- Fast crystal is only turned on when high accuracy is needed (during resynchronization)
  - It interpolates between the slow crystal clock ticks
- An event is stamped with the slow crystal and fast crystal time
- Using some simple modulo operations, we can find phase offset between the slow ticks of the event, using the fast clock signal
Event time = \( c_0 \cdot \frac{f_{\text{fast}}}{f_{\text{slow}}} + \varphi \)

- After event, fast clock can be turned off
HyperFTSP

- TI MSP430 8MHz MCU (Quanto Testbed Mote)
- Slow clock: 32 kHz
- Fast clock: 8 MHz
- Dual clock enhanced FTSP with Low Power Listening
- Fast clock gets enabled on radio on/off switching
Dual-Clock Hyper FTSP Performance

- Average: 0.06μs
- StdDev: 0.9μs
The High-Low Timer

• Dual-clock system consumes all timer resources on the evaluated MCU
• Timer units of MCU are underutilized
• Implementation on low-power Actel IGLOO FPGA
• Consumes 868 of 13824 core cells of the IGLOO AGL600 FPGA (6%)
Did we Achieve the Goal?

- High Accuracy
- Sub-µSecond
- Robust to Disconnects
- Resource Constraint µW
Did we Achieve the Goal?

HyperFTSP

Robust to Disconnects

Resource Constraint $\mu$W
Did we Achieve the Goal?

HyperFTSP

XCXT & TCTS

Resource Constraint \(\mu W\)
Did we Achieve the Goal?

- HyperFTSP
- XCXT & TCTS
- TCTS & High-Low Timer
Future Work

- HLTimer Mote architecture
  - FPGA joint with a MCU, I2C for communication
  - Use of either Epic or E-Co core (E-Co: Epic Cortex)
- Large Scale TCTS Tests
  - General TCTS Extension in TinyOS that can be used with any underlying time synchronization process (FTSP, GTS, etc)
  - Multi-Hop effects in TCTS?
- Join TCTS and HLTimer
Contributions

- Dual Clock Systems for Local Temperature Compensation
  [DAC’08, ISLPED’08, SigMetrics’08, HotPower’08]

- Theory of the influence of environmental temperature on
  time synchronization
  [TOSN’09 under submission]

- Temperature Compensated Time Synchronization
  [ESL’09 accepted]