Principles of Wireless Sensor Networks

https://kth.instructure.com/courses/2912/

Lecture 11

Time Synchronization

Carlo Fischione
Associate Professor of Sensor Networks
e-mail:carlofi@kth.se
http://www.ee.kth.se/∼carlofi/

KTH Royal Institute of Technology
Stockholm, Sweden

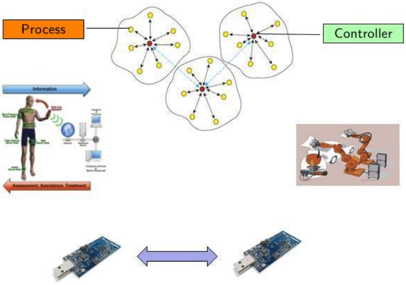
October 2, 2017

Course content

- Part 1
  - Lec 1: Introduction to WSNs
- Part 2
  - Lec 2: Wireless Channel
  - Lec 3: Physical Layer
  - Lec 4: Medium Access Control Layer
  - Lec 5: Routing
- Part 3
  - Lec 6: Distributed Detection
  - Lec 7: External seminars
  - Lec 8: Static Distributed Estimation
  - Lec 9: Dynamic Distributed Estimation
  - Lec 10: Positioning and Localization
  - Lec 11: Time Synchronization
- Part 4
  - Lec 12: Wireless Sensor Network Control Systems 1
  - Lec 13: Wireless Sensor Network Control Systems 2

Previous lecture

Application
Presentation
Session
Transport
Routing
MAC
Phy



How to estimate the position of fixed and mobile nodes?

Today's learning goals

- Which measurements are used for synchronizing the nodes?
- How to synchronize pair of nodes?
- How to synchronize a network of nodes?

Outline

- Basics of time synchronization
  - Hardware clock - Software clock
  - Message exchanges
- Synchronization protocols without drift model
- Time synchronization protocol with drift model
- Distributed clock synchronization

Basics of time synchronization

Time synchronization is defined as the procedure for at least two nodes to have a common reference clock

A typical node possesses an oscillator of a specified frequency (**hardware clock**) and a counter register, which is incremented after a certain number of oscillator pulses.

The nodes software has access only to the counter register (**software clock**).

The time distance between two increments (ticks) determines the achievable **time resolution**

Consider two nodes: node i, node j. Then,

- **Clock offset** is defined as the difference between time at node i and time at node j
- **Clock rate** is the frequency at which clock progresses
- **Clock skew** represents the difference in the frequency of two clocks

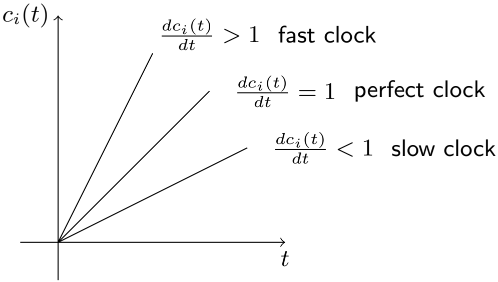
Hardware Clock and Software Clock

- Denote the nominal hardware clock of node i as $H_i(t)$.
- The software time measured at node i at time t is

$$c_i(t) = \rho_i(t)H_i(t) + \phi_i(t)$$

  where $\phi_i(t)$ is called **phase shift** and $\rho_i(t)$ is called **drift rate**.

- Ideally, $H_i(t) = t$, $\rho_i(t) = 1$, and $\phi_i(t) = 0$.



Software clock rate

Clock rate: $\dfrac{dc_i\,(t)}{dt}$

Consider a bounded drift rate at node i,

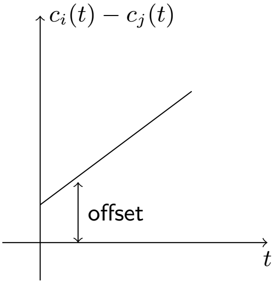
$$1 - \rho_i(t) \leq \frac{dc_i\,(t)}{dt} \leq 1 + \rho_i(t)$$

Two synchronized nodes, node i and node j, before being resynchronized can drift of at maximum $2\rho_{\max}$, that is

$$\frac{dc_i\,(t)}{dt} - \frac{dc_j\,(t)}{dt} \leq 2\rho_{\max}$$

where $2\rho_{\max} \cdot \tau_{\text{synch}} < \delta_{\max}$ and $\delta_{\max}$ a precision parameter that is equal to the maximum offset between two clocks

Modeling of the software clock differences



Offset is worsening as time goes by!

How can we model the offset between node i and node j ?

$$c_i\,(t) - c_j\,(t) = \underbrace{t_0}_{\text{constant offset}} + \underbrace{\Delta f\,(t)}_{\text{time dependent frequency offset}} \cdot t + \underbrace{\Delta\tau(t)}_{\text{jitter due to noise}}$$

Frequency offset $\Delta f\,(t)$ is due to different rates among the clocks and environmental effects (e.g. temperature, humidity)

Basics of time synchronization

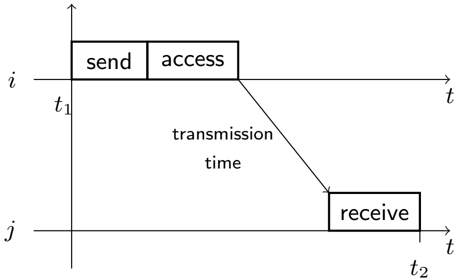
How do we synchronize the clocks?

Problem: Non-determinism of communication delay

Outline

- Basics of time synchronization
  - Hardware clock - Software clock
  - Message exchanges
- Synchronization protocols without drift model
- Time synchronization protocol with drift model
- Distributed clock synchronization

Message exchanges

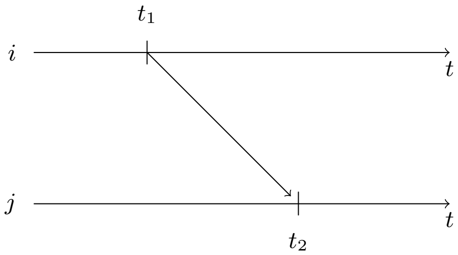


- Consider two nodes: node i and node j
- A message is sent to synchronize i with j
- But there may be a random access delay. E.g., IEEE 802.15.4 beaconless MAC delay of the previous lectures.

Outline

- Basics of time synchronization
  - Hardware clock - Software clock
  - Message exchanges
- Synchronization protocols without drift model
  - One way message exchange
  - Two ways message exchange
  - Receiver-receiver synchronization
- Time synchronization protocol with drift model
- Distributed clock synchronization

One way message exchange protocol



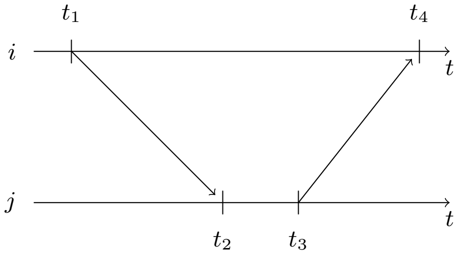
$$c_i\left(t_1\right) = t_1 + n_1$$

$$c_j\left(t_2\right) = t_2 = t_1 + \underset{\text{propagation delay}}{D} + \underset{\text{offset}}{\delta} + n_2$$

Node j makes an estimate of δ and D by assuming that t_1 is known and using the methods of Lecture 8.

Two way message exchange Protocol

A slightly more advanced estimation can be done as follows



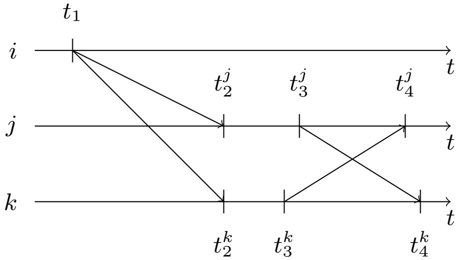
$c_j(t_2) = c_i(t_1) + D + \delta + n_2$

$c_i(t_4) = c_j(t_3) + D - \delta + n_4$

$c_i(t_1), c_j(t_2), c_j(t_3), c_1(t_4)$ measured $\Rightarrow$ estimate D and δ by methods of Lecture 8. In case $n_2 \approx n_4$

$$D = \frac{(c_j(t_2) - c_i(t_1)) + (c_i(t_4) - c_j(t_3))}{2} \qquad \delta = \frac{(c_j(t_2) - c_i(t_1)) - (c_i(t_4) - c_j(t_3))}{2}$$

Receiver-receiver synchronization



$$t_4^k = t_3^j + \delta_{jk} + n_4^k$$
$$t_3^j = t_2^j + \Delta_j = t_1 + \delta_{ij} + \Delta_j + n_3^j$$
$$t_4^j = t_3^k + \delta_{kj} + n_4^j$$
$$t_3^k = t_2^k + \Delta_k = t_1 + \delta_{ik} + \Delta_k + n_3^k$$

where Δ_j and Δ_k are known and δ_{ik}, δ_{kj}, δ_{jk}, δ_{kj} unknown

Outline

- Basics of time synchronization
- Synchronization protocols without drift model
- Time synchronization protocol with drift model
  - Estimation based on LS
  - Estimation based on MMSE
- Distributed clock synchronization

Time synchronization protocols with drift model

Consider two nodes, node i and node j, with different drifts and offsets

Let us define $c_i(t)$ as the clock of node i and $c_j(t)$ as the clock of node j

Consider the following clock synchronization model

$$c_j(t) = a_0 + a_1 \cdot c_i(t) + n_{ij}$$

For synchronization, measurements are required in order to determine a_0 and a_1

Time synchronization protocol

$$c_j(t) = a_0 + a_1 \cdot c_i(t) + n_{ij}$$

<u>STEP 1</u>

$c_i(t_0) \triangleq x_0$ (measured)

$c_j(t_0) = a_0 + a_1 \cdot c_i(t_0) + n_{ij,0} = a_0 + a_1 \cdot x_0 + n_{ij,0} \triangleq y_0 + n_{ij,0}$ (measured)

<u>STEP 2</u>

$c_i(t_1) \triangleq x_1$

$c_j(t_1) = a_0 + a_1 \cdot x_1 + n_{ij,1} \triangleq y_1 + n_{ij,1}$

$\vdots$

<u>STEP n</u>

$c_i(t_{n-1}) \triangleq x_{n-1}$

$c_j(t_{n-1}) = a_0 + a_1 \cdot x_{n-1} + n_{ij,n-1} \triangleq y_{n-1} + n_{ij,n-1}$

Time synchronization protocol

Putting all the measurements together, we end up in the following system of equations

$$A \cdot X + N = Y$$

where

$$A = \begin{bmatrix} x_0 & 1 \\ x_1 & 1 \\ \vdots & \vdots \\ x_{n-1} & 1 \end{bmatrix} \qquad X = \begin{bmatrix} a_1 \\ a_0 \end{bmatrix} \qquad N = \begin{bmatrix} n_{ij,0} \\ n_{ij,1} \\ \vdots \\ n_{ij,n-1} \end{bmatrix} \qquad Y = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{bmatrix}$$

Estimation based on LS (noise assumed absent)

Least Square Estimator

$$\widehat{X} = L \cdot Y$$

where

$$L = \left(A^T \cdot A\right)^{-1} A^T$$

$$A^T \cdot A = \begin{bmatrix} \sum_{i=0}^{n-1} {x_i}^2 & \sum_{i=0}^{n-1} x_i \\ \sum_{i=0}^{n-1} x_i & n \end{bmatrix} \qquad A^T \cdot Y = \begin{bmatrix} \sum_{i=0}^{n-1} x_i y_i \\ \sum_{i=0}^{n-1} y_i \end{bmatrix}$$

Estimation based on MMSE

MMSE Estimator

The MMSE estimator of X given that $Y = y$ is

$$P^{-1}\hat{X} = AR_N^{-1}y$$

with error covariance

$$P^{-1} = R_X^{-1} + A^T R_N^{-1} A$$

where R_N the covariance of zero mean Gaussian noise

In this specific case, $\hat{X} = PAR_N^{-1}y$ where

$$P^{-1} = \begin{bmatrix} \sum_{k=0}^{n-1} \frac{x_k^2}{\sigma^2_{n_{ij,k}}} & \sum_{k=0}^{n-1} \frac{x_k}{\sigma^2_{n_{ij,k}}} \\ \sum_{k=0}^{n-1} \frac{x_k}{\sigma^2_{n_{ij,k}}} & \sum_{k=0}^{n-1} \frac{1}{\sigma^2_{n_{ij,k}}} \end{bmatrix}$$

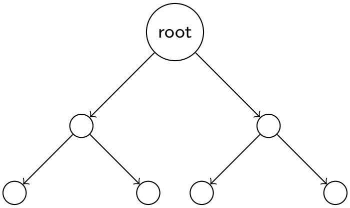
$$R_N^{-1} = \begin{bmatrix} \frac{1}{\sigma^2_{n_{ij,0}}} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sigma^2_{n_{ij,1}}} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \frac{1}{\sigma^2_{n_{ij,n-1}}} \end{bmatrix}$$

Outline

- Basics of time synchronization
- Synchronization protocols without drift model
- Time synchronization protocol with drift model
- Distributed clock synchronization

Distributed clock synchronization

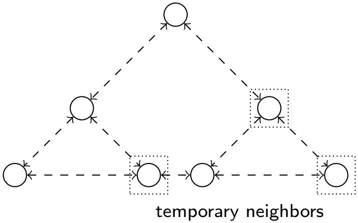
Previous part of lecture



- Nodes synchronize with the root
- Synchronization tends to deteriorate as we move away from the root

Distributed clock synchronization

Now, we would like to have a real common synchronization across the network



No root for synchronization - only P2P

Distributed clock synchronization

Consider the following hardware clock model at node i:

$$H_i(t) = \int_{t_0}^{t} h_i(\tau)\, d\tau + \phi_0(t_0)$$

where $h_i(\tau)$ is the hardware clock rate and $\phi_0(t_0)$ is the hardware clock offset at time t_0.

Assume that the clock rate has a bounded drift ρ, i.e.,

$$1 - \rho_{\max} \leq h_i(t) \leq 1 + \rho_{\max}$$

Distributed clock synchronization

The software clock of node i at time t can be expressed as

$$c_i(t) = \int_{t_0}^{t} h_i(\tau)\, l_i(\tau)\, d\tau + \theta_i(t_0)$$

where $l_i(\tau)$ the relative logical clock rate which can be properly tuned to achieve synchronization and $\theta_i(t_0)$ a clock offset.

We can then define the absolute logical clock rate of node i at time t as

$$x_i(t) \triangleq h_i(t) \cdot l_i(t)$$

Then, nodes are synchronized if $x_i(t)$ converge to the same value. How to do so?

Distributed clock synchronization

Define the following update rules for all nodes

$$x_i\left(k+1\right) \triangleq \frac{\sum\limits_{j\in N_i(k)} x_j\left(k\right) + x_i\left(k\right)}{\left|N_i\right| + 1}$$

where

- $N_i\left(k\right)$ the set of neighbors of node i at time k
- $\left|N_i\right|$ the cardinality of neighbor nodes

Distributed clock synchronization

By setting $X(k) = \begin{bmatrix} x_1(k) \\ \vdots \\ x_N(k) \end{bmatrix}$

we obtain the compact form

$$X(k+1) = A(k) \cdot X(k)$$

where

$$A(k) = [a_{i,j}(k)] = \begin{cases} \frac{1}{|N_i|+1} & \text{if } i,j \text{ are connected,} \\ 0 & \text{otherwise.} \end{cases}$$

and $A(k) \cdot \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$ (raw stochastic matrix)

Note that the matrix dimensions of $A(k)$ depend on the topology and change every time the neighbors of node i change.

Distributed clock synchronization

Proposition

Suppose the communication graph is strongly connected, then all the logical clock rates converge to a steady state value, i.e.,

$$\lim_{k\to\infty} X(k) = x_{ss} \cdot \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

Practical implementation

1. Each node periodically broadcasts a synchronization beacon (to all neighbors) containing its current measured software time $c_i(t)$ and the relative logical clock rate $l_i(t)$.
2. The information received by neighbors is used to estimate the absolute logical clock rate $x_j(t)$ of those neighbors.
3. Each node updates its own relative clock rate $l_i(t)$ as

$$l_i\left(k+1\right) \triangleq \frac{\sum\limits_{j \in N_i(k)} \frac{x_j(k)}{h_i(t)} + l_i\left(k\right)}{|N_i| + 1}$$

4. The procedure is repeated.

Summary

- We have studied the basic of synchronization for sensor networks
- Synchronizing the nodes consists in applying estimation techniques

Next lecture

- The fourth and last part of the course starts: control over WSNs