Time Synchronization in Wireless Sensor Networks

Ren Fengyuan

July 26, 2005

Agenda

- **Need for time synchronization in sensor networks**
- **Definition of the time synchronization**
- **Common challenges for time synchronization**
- **Typical schemes and algorithms**
- **Our works**

Wireless Sensor Networks

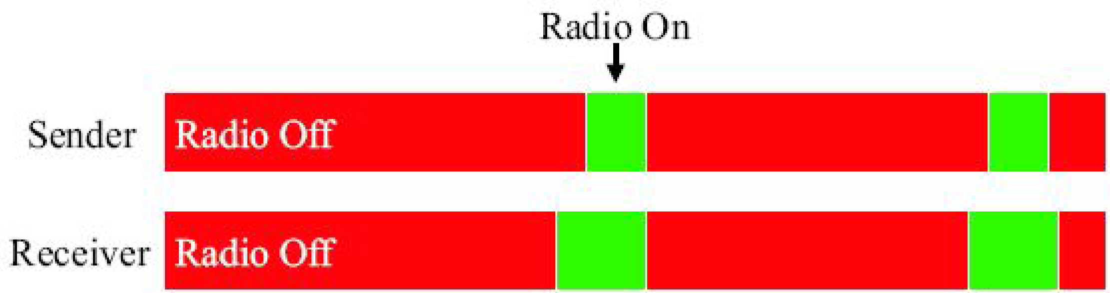
- **Functions**
  - Detect events
  - Relay information to users
- **Applications**
  - Monitoring of wildlife, intruders, machine conditions, earthquakes, fire, office environment etc.

Why Need for Time Synchronization

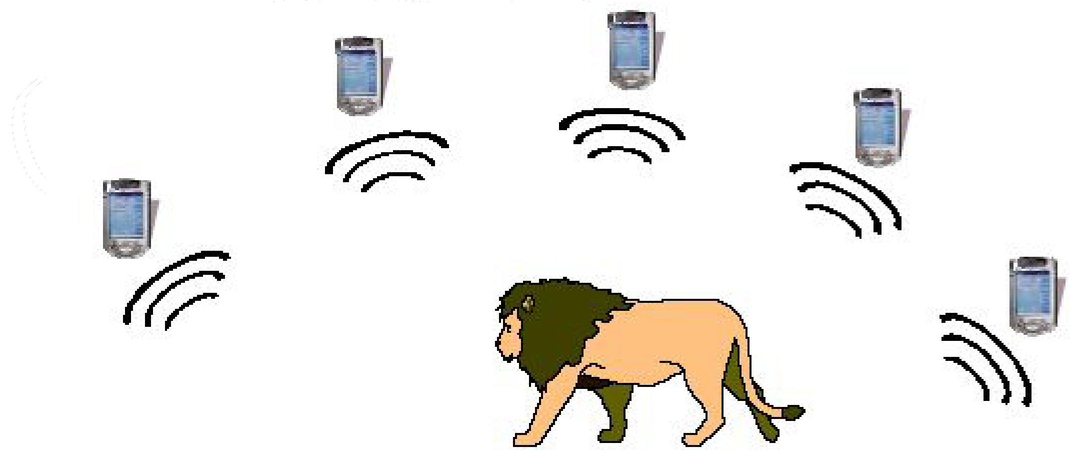
- **Link to the physical world!**
  - **When does an event take place?**
- **Key basic service of sensor networks**
  - **Fundamental to data fusion.**
- **Crucial to the efficient working of other basic services**
  - **Localization, Calibration, In-network processing etc.**
- **Several protocols require time synchronization**
  - **Cryptography, MAC, Topology management ….**

Examples

- **Energy-efficient Radio Scheduling**
  - **TDMA, Guard band**



- **Array Processing**



Computer Clocks

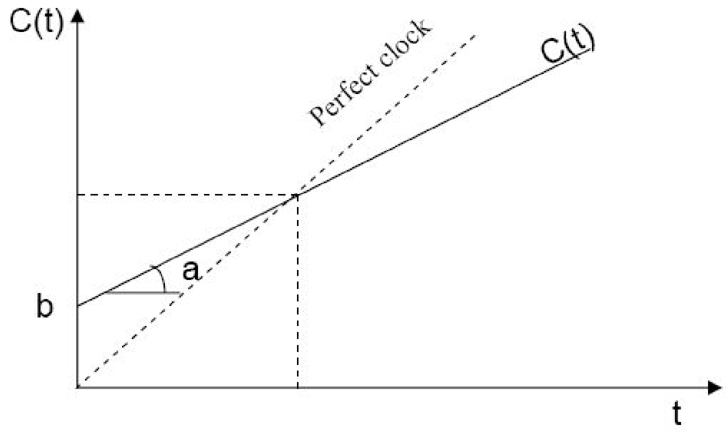
- **Clocks in computers**

$$C(t) = k\int_{t_0}^{t} \omega(\tau)d\tau + C(t_0)$$

ω is frequency of oscillator, $C(t_0)$ is initial value

- Time of the computer clock implemented based on a hardware oscillator

- **Computer clock is an approximation of a real time t**
  - C(t) = a*t+b
    - ✓ a is a clock drift (rate)
    - ✓ b is an offset of the clock



- **Perfect clock**
  - Rate = 1
  - Offset = 0

Definition of time synchronization

- Let $C(t)$ be a perfect clock

If $C_i(t) = C(t)$. A clock $C_i(t)$ is called correct at time t

If $d\,C_i(t)/dt = dC(t)/dt = 1$, a clock $C_i(t)$ is called accurate at time t

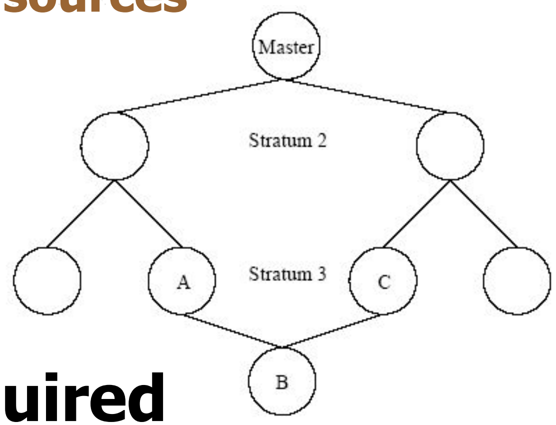
If $C_i(t) = C_k(t)$, two clocks $C_i(t)$ and $C_k(t)$ are synchronized at time t

Time Synchronization

requires knowing both offset and drift

NTP Overview

- **Most widely used time synchronization protocol**
  - Hierarchical: C/S model
    - ✓ *stratum* levels
    - ✓ each daemon can use several independent time sources
    - ✓ Daemon can pick the most accurate one



- **Perfectly acceptable for most cases**
  - coarse grain synchronization
- **Inefficient when fine-grain sync is required**
  - sensor networks applications:
    - ✓ localization, beamforming, TDMA scheduling etc

Why not Use NTP

- **Link**
  - Ratio of packet loss is very low in Internet (fiber ,cable)
  - Links are short range and short lived in sensor networks (wireless)
- **Topology**
  - Static, robust and configurable
- **Infrastructure-Supported**
  - Canonical sources (Stratum servers) are synchronized with each other via variety of "out of band" mechanisms (GPS, WWVB radio broadcast)
- **Energy Aware**
  - Frequent message exchange
  - Listening to the Network is free
  - Using the CPU in moderation is free

Why not Use GPS

- **Cost**
  - 300$ (achieve < 20ns phase error to UTC)
- **Practical Limitations**
  - Cannot be used under special environment where is no free line of sight to the GPS satellites
    - ✓ e.g. dense foliage or inside buildings
- **Policy Limitations**
  - Military Applications

Difficulties in Sensor Networks

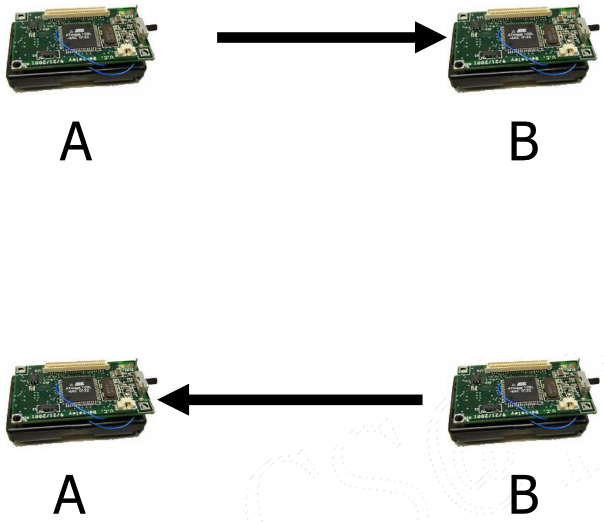
- **No periodic message exchange is guaranteed**
  - There may be no links between two nodes at all
- **Transmission delay between two nodes is hard to estimate**
  - The link distance changes all the time
- **Energy is very limited**
  - Nodes sleep most of the time to conserve power

- **Node need to be small and cheap**
  - No expensive clock circuitry

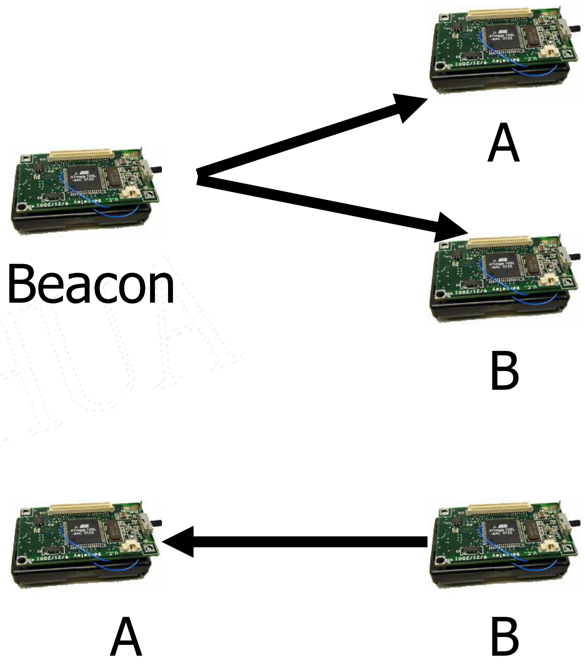
Basic Approach

- **Collaboration among sensor nodes**
  - **Establish pairwise relationship between nodes.**
  - **Extend this to network level**
- **Two approaches for collaboration**
  - **Receiver – Receiver**
  - **Sender – Receiver**

S-R vs. R-R



S – R

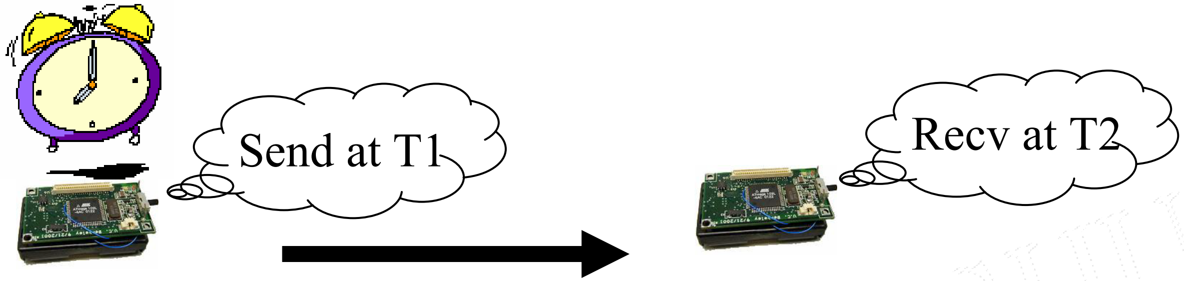


R – R

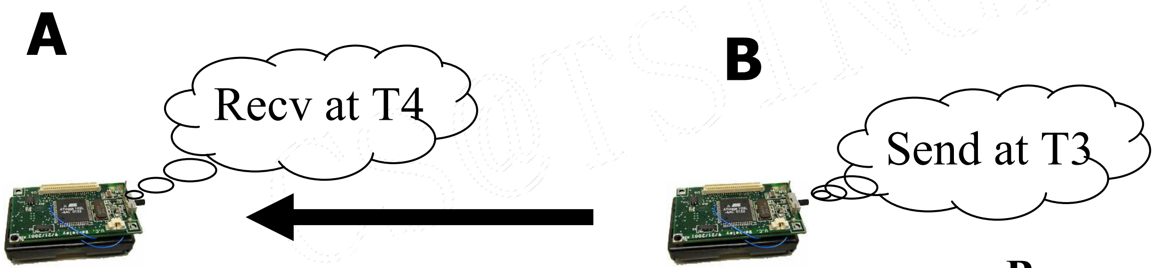
- Sources of error – variation in packet delays and clock drift

Basic Mechanism

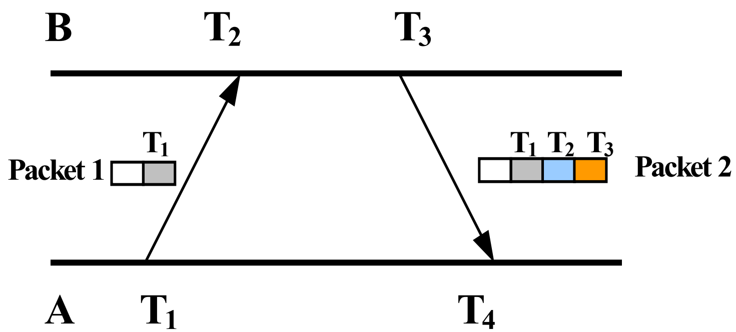
Pair-wise Synchronization



T2 = T1 + DELAY + OFFSET



T4 = T3 + DELAY- OFFSET



OFFSET = {(T2-T1)-(T4-T3)}/2

DELAY = {(T2-T1)+(T4-T3)}/2

Sources of Errors

- Send time
  - Kernel processing
  - Context switches
  - Interrupt Processing
- Access time
  - Specific to MAC protocol
    - ✓ E.g. in Ethernet, sender must wait for clear channel
- Transmission Time
- Propagation time
  - Very small in WSNs, can be omitted
- Reception time
- Receive time

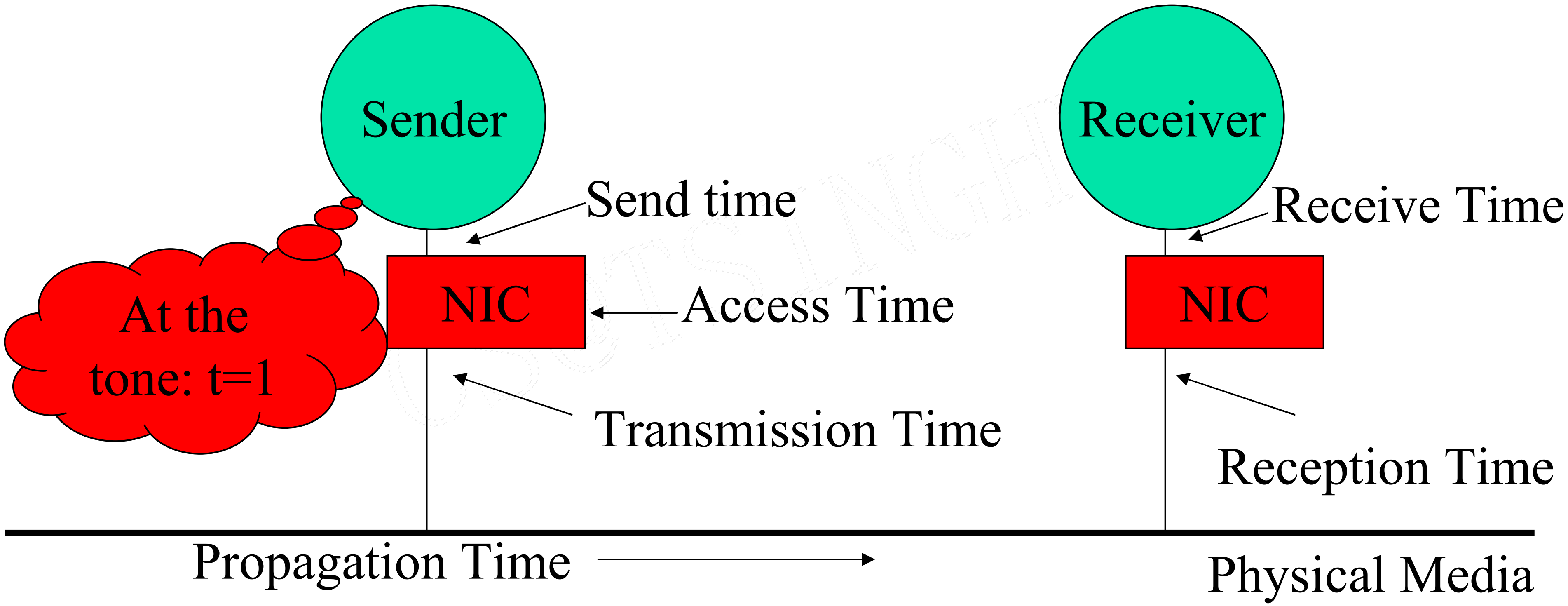
Common denominator:

(1) non-determinism

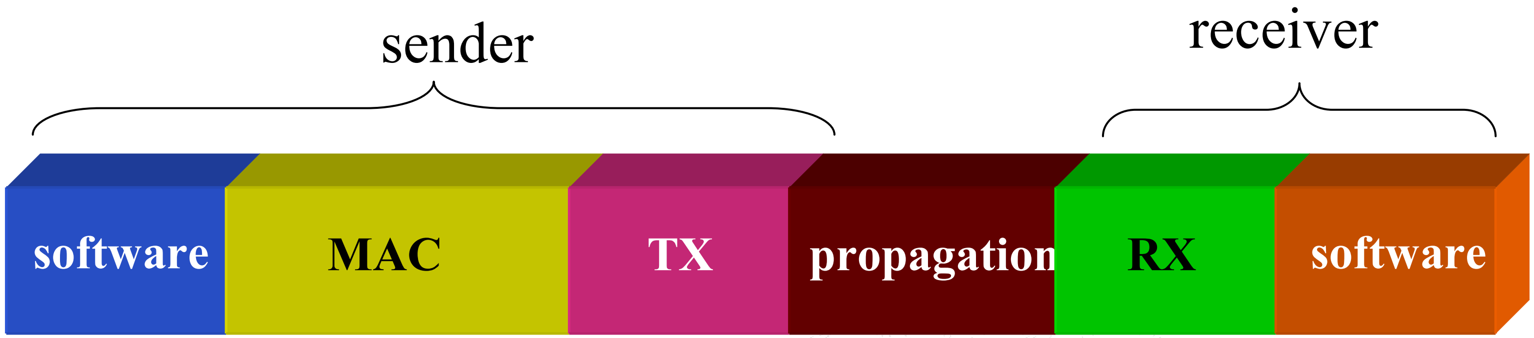
(2) difficult to estimate

Send, access, receive

Illustration



Variability in Packet Delay



ALL DELAYS ARE VARIABLE !

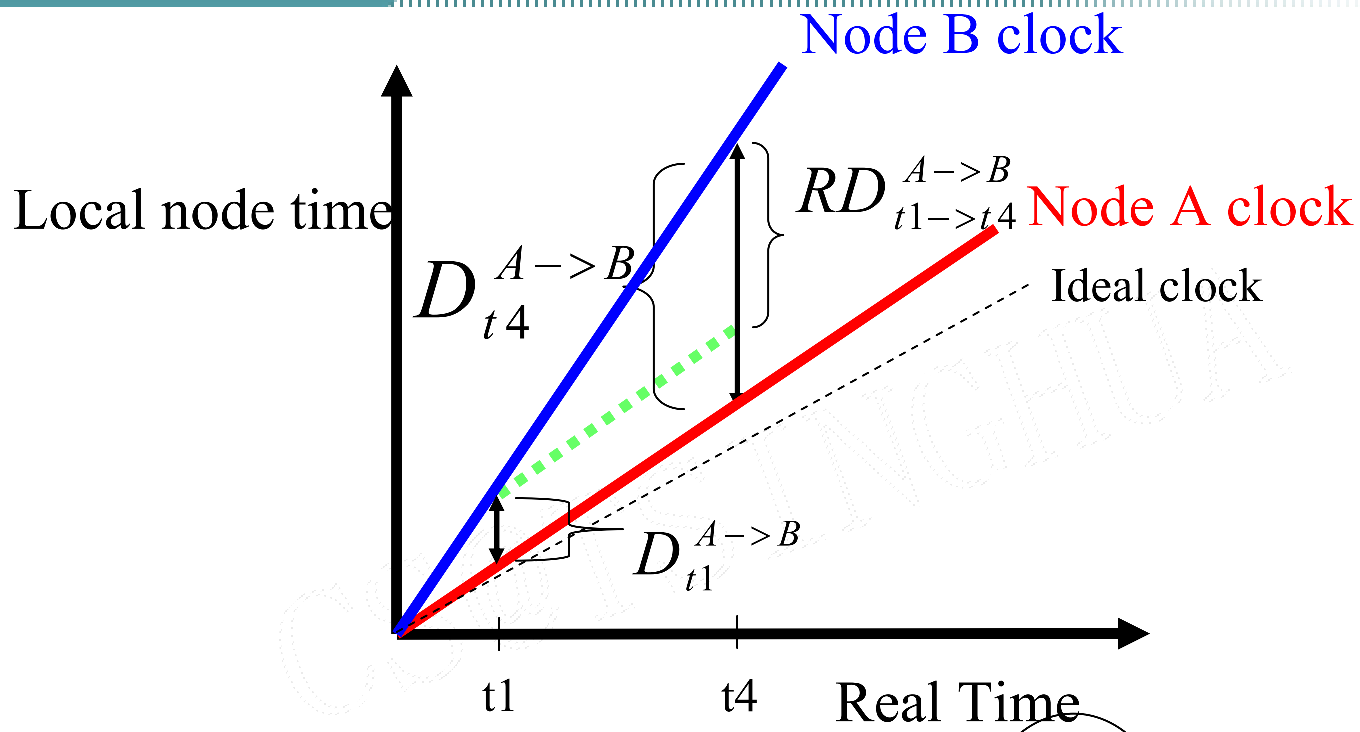
$$Error = \frac{S^{UC}}{2} + \frac{P^{UC}}{2} + \frac{R^{UC}}{2}$$

Sender uncertainty

Propagation uncertainty

Receiver uncertainty

Variability in Clock Drift



$$Error = \frac{S^{UC}}{2} + \frac{P^{UC}}{2} + \frac{R^{UC}}{2} + \frac{RD_{t1->t4}^{A->B}}{2}$$

Relative drift

Requirements for Synchronization Schemes

- **Energy efficiency**
  - Sensor nodes have very limited energy resources
  - Each transferred bit equals to the energy of hundreds of executed instructions
- **Scalability**
  - Large number of nodes
- **Precision**
  - Need for accuracy differs significantly
- **Robustness**
  - Synchronization scheme should remain functional even when part of the nodes are not functional

Requirements (cont.)

- **Lifetime**
  - The synchronized time can be valid either for a single instant or a longer period
    - ✓ e.g. some applications may require only momentary synchronization
- **Scope**
  - Global time for all nodes in the network
  - Local synchronization among spatially close nodes
- **Convergence Time**
  - Some applications may require that event is communicated immediately to the sink node (e.g. Emergency applications)
  - Some apps requires pre-synchronization of the nodes
  - post-synchronization is feasible for other apps.

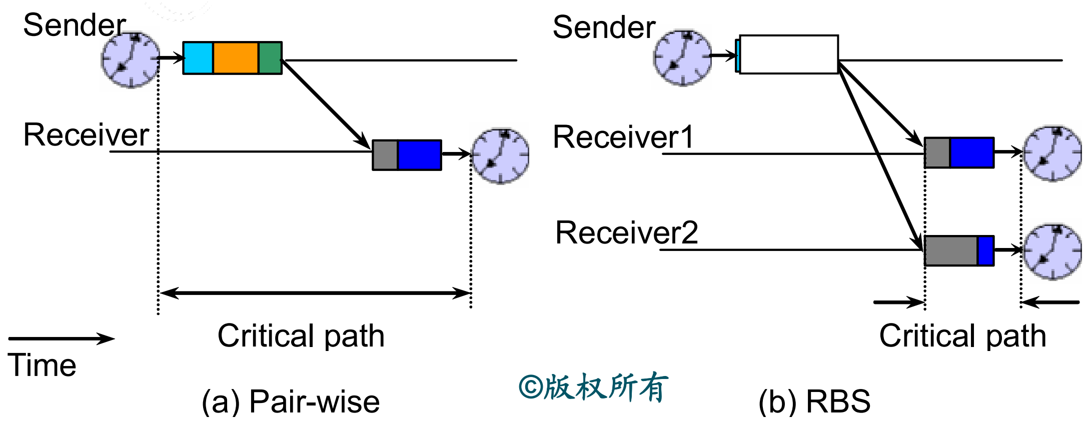
Typical Schemes and Algorithms

- RBS (Reference Broadcast Synchronization)
- TPSN (Timing-Sync Protocol for Sensor Networks)
- DMTS (Delay Measurement Time Synchronization)
- LTS (Lightweight Time Synchronization)
- FTSP (Flooding Time Synchronization Protocol)
- TS/MS (Tiny-Sync and Mini-Sync)
- TSync
- AD (Asynchronous Diffusion)
- ……

RBS[1]

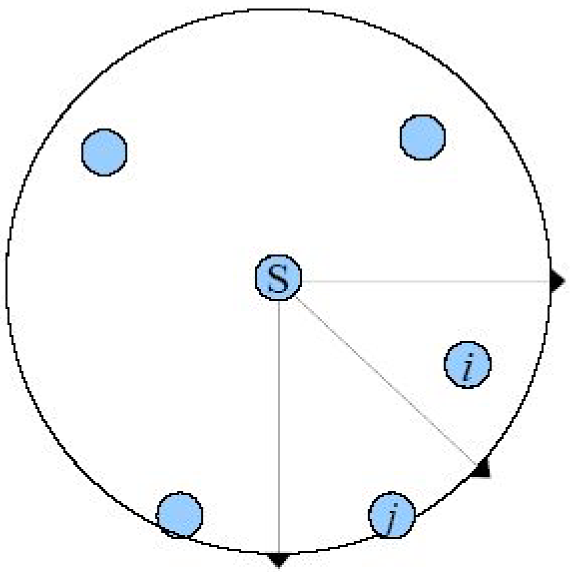
➢ Properties

- **Originally based on the idea of R-R synchronization**
- **Nodes send reference beacons to their neighbors**
  - ✓ **Beacon does not include a timestamp,**
  - ✓ **Time of arrival is used by receiving nodes as a reference for comparing clocks**
- **Removes completely non-determinism caused by the sender**
  - ✓ **only one source of error – receiver**



RBS Operation

- Node S sends reference beacon
- Reception time of the beacon is recorded according to it's local clock
- Nodes exchange their observations
- Any two nodes i and j can calculate the offset of their local clocks
- Also clock drift can be calculated with multiple samples using linear regression



Offset

- **Estimation**
  - **n**: number of receivers
  - **m**: number of reference broadcasts
  - $T_{r,b}$: r′s clock when it receives broadcast b

  $\forall\ i \in n,\ j \in n : Offset[i, j] = (\sum(T_{j,k} - T_{i,k}))/m \quad (k=1,\ldots,m)$

- **Precision**
  - Berkeley Mote can be synchronized with an average error of 11us by using 30 broadcasts

Extension to Multi-hop

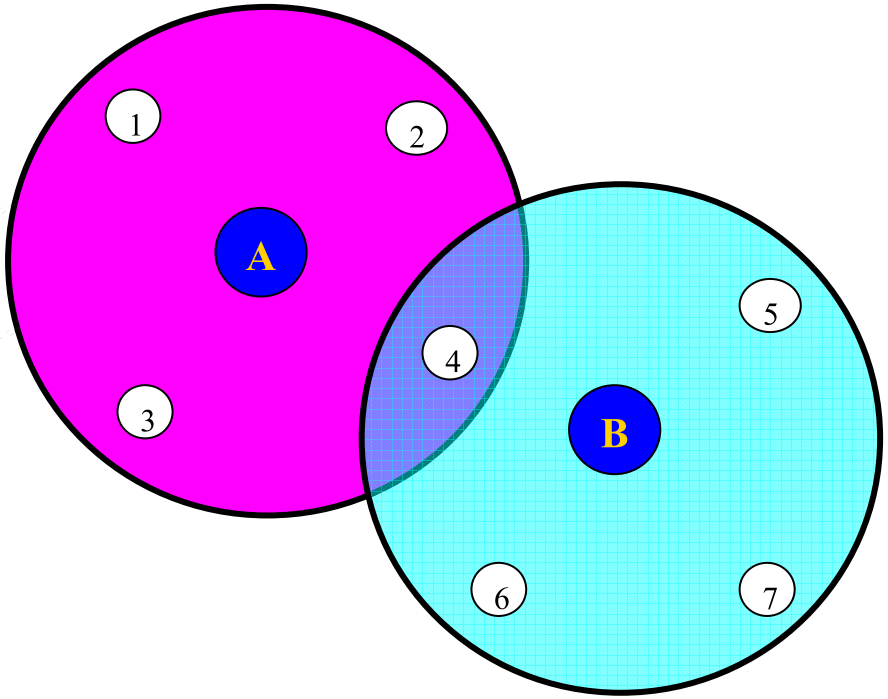
- Nodes A and B send synchronization pulses at distinct times P_A , P_B

If $E_1 = P_A + 2$

$P_B = E_7 + 4$

$P_A = P_B + 10$

then $E_1 = E_7 + 16$



Precision

- The average error grows with hops

$$O(\sqrt{n})$$

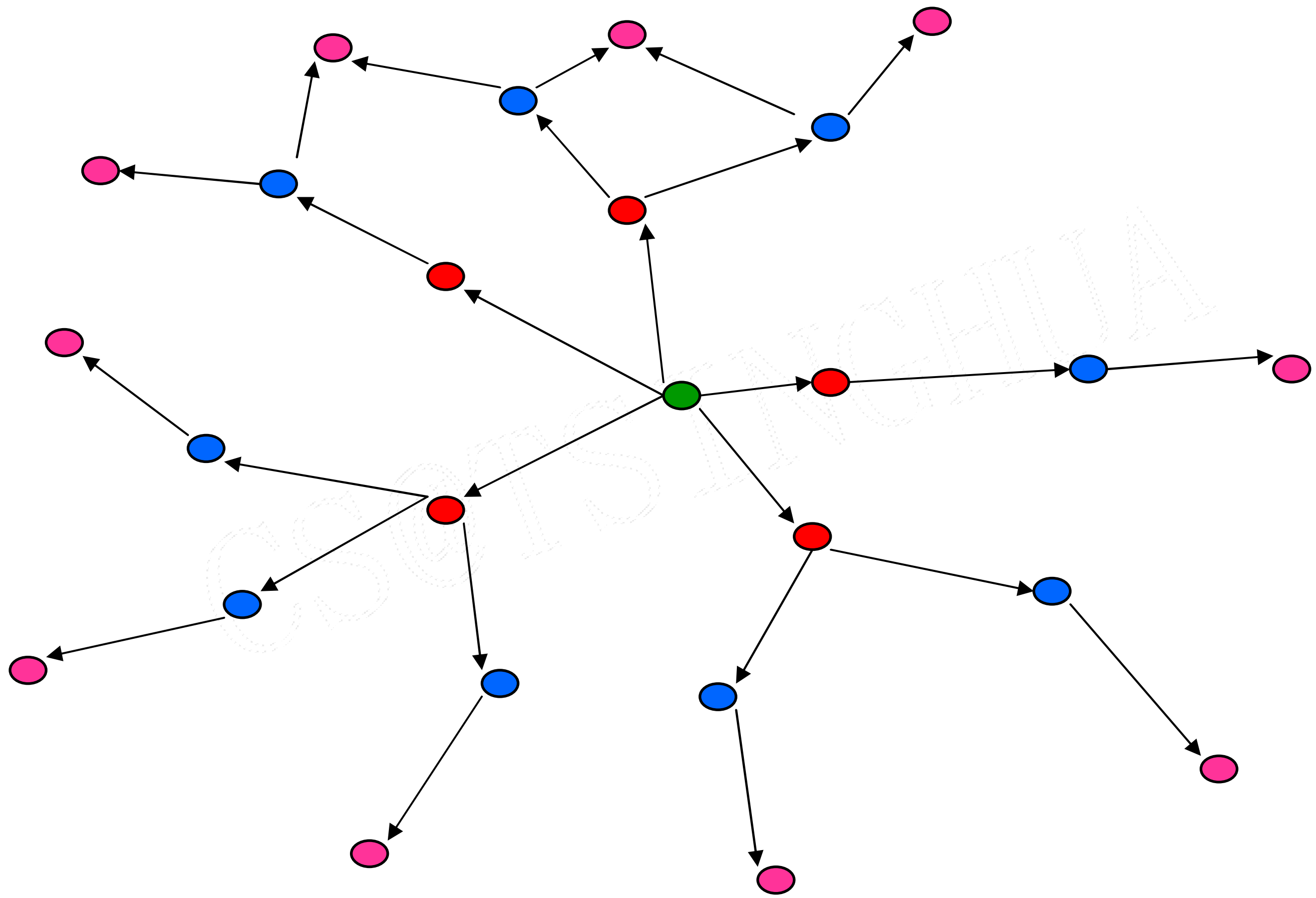
TPSN[3]

➢ Features

- Sender-receiver bidirectional mechanism
- Two phases of operation:
  - ✓ level discovery and synchronization
- Level discovery phase
  - ✓ Root node initiates level discovery
  - ✓ A node on receiving its level broadcasts it
  - ✓ Any node receiving multiple level packets takes the first one and ignores the others
  - ✓ Any node not receiving a level packet times out and sends a request for level packet

Level Discovery Phase

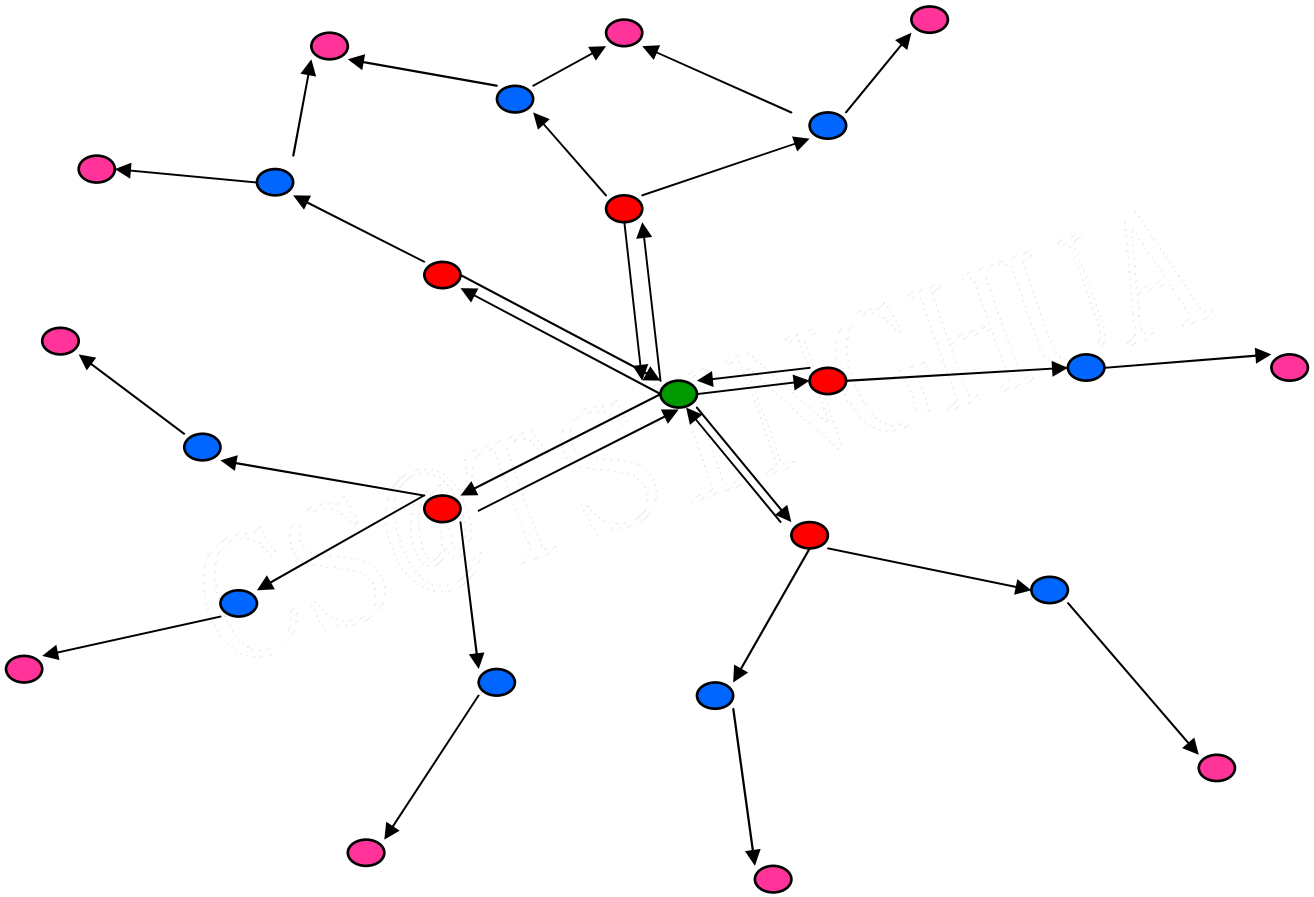
Tsinghua University



TPSN (cont.)

- **Synchronization phase**
  - Root node sends a start synchronize packet
  - All nodes of level *1* synchronize themselves to root node
  - For every level *i* , the nodes of that level synchronize to the nodes of level *i -1*
  - If a node in level *i-1* is not synchronized, then it does not respond to synchronization requests from level *i*

Time Synchronization Algorithm



Performance

➢ Precision

- TPSN achieves two times better precision than RBS
  - ✓ Berkeley Mote
  - ✓ RBS achieves 29.13 μs, while TPSN results 16.9 μs average error
- Sender uncertainty contributes very little to the total synchronization error due to low level time stamping (MAC layer)

	TPSN	RBS
Average error (in μs)	16.9	29.13
Worst case error (in μs)	44	93
Best case error (in μs)	0	0
Percentage of time error is less than or equal to average error	64	53

LTS[4]

- **Features**
  - Minimize synchronization complexity rather than maximizing accuracy
    - ✓ Authors claim that wireless sensor networks need quite low synchronization accuracy
  - Sender-receiver bidirectional mechanism
  - Two LTS algorithms
    - ✓ Centralized
      - Node sends a synchronization request to a closest reference node by any routing mechanism
    - ✓ Distributed
      - Requires a spanning tree to be constructed firstly

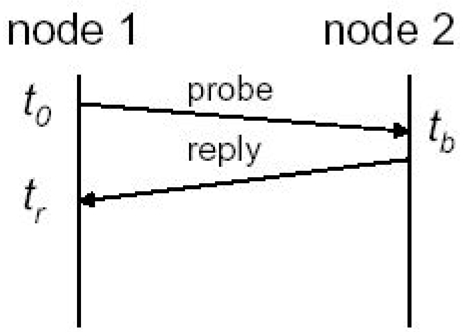
LTS (cont.)

- **LTS optimizes synchronization frequency with required precision**
  - ✓ **The synchronization frequency is calculated from the requested precision, from the depth of the spanning tree, from the drift bound**
- **Simulation results**
  - ✓ **500 nodes (120m x 120m)**
  - ✓ **Target precision: 0.5**
  - ✓ **Duration : 10hrs**
  - ✓ **Centralized : 65% of all nodes request Synchronization**
    - **4-5 synchronization operation on average per node**
  - ✓ **Distributed:**
    - **average 36 pairwise synchronization per node**

TS/MS

Features

- Determining relative offset and drift between two nodes
- Sender-receiver bidirectional scheme
  - ✓ Node 1 sends a probe message to node 2 timestamped with t_0
  - ✓ Node 2 generates timestamp t_b and responds immediately
  - ✓ Node 1 generates timestamp t_r
  - ✓ The 3-tuple of (t_0, t_b, t_r) is called a *data point*

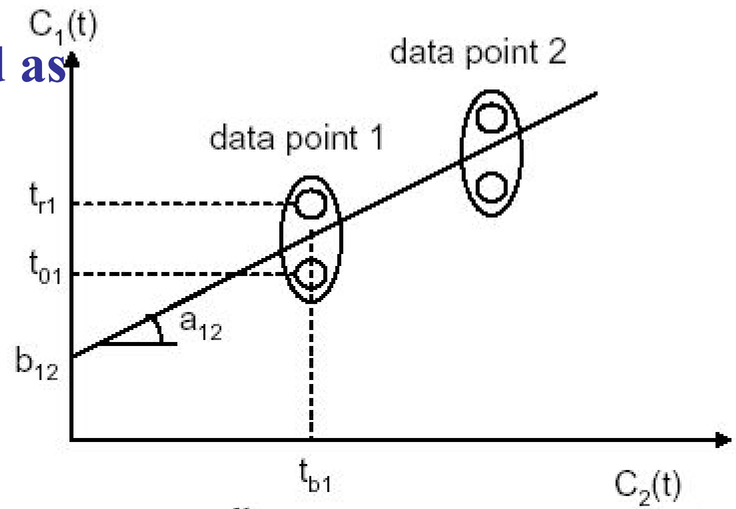


- Two clocks $C_1(t)$ and $C_2(t)$ are linearly related as

$$C_1(t) = a_{12}C_2(t) + b_{12}$$

- The following inequalities are held:

$$t_0 < a_{12}\, t_b + b_{12} < t_r$$



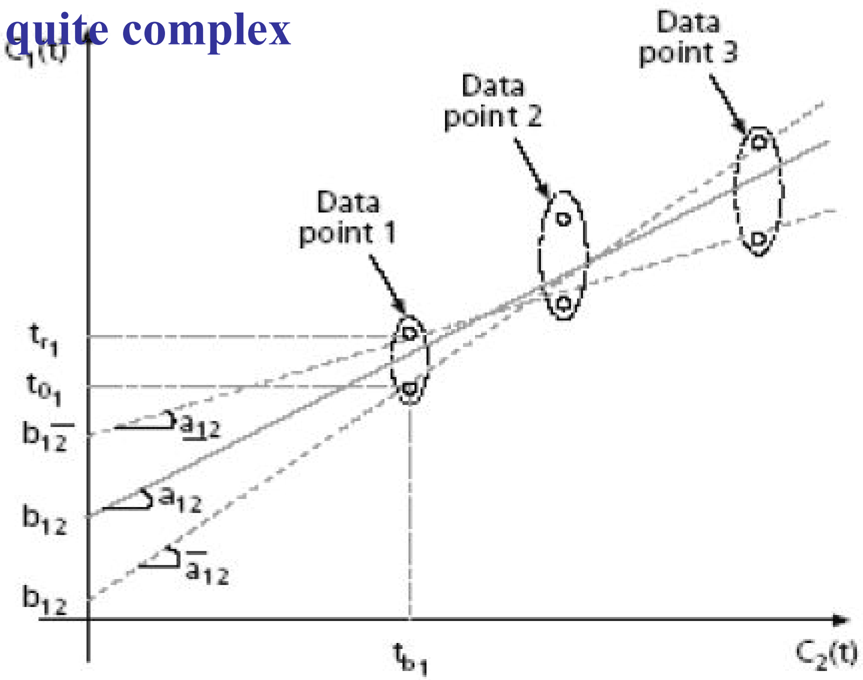
Parameters Estimation

- **Relative drift a_{12} and offset b_{12}**

$$\underline{a_{12}} \le a_{12} \le \overline{a_{12}}$$

$$\underline{b_{12}} \le b_{12} \le \overline{b_{12}}$$

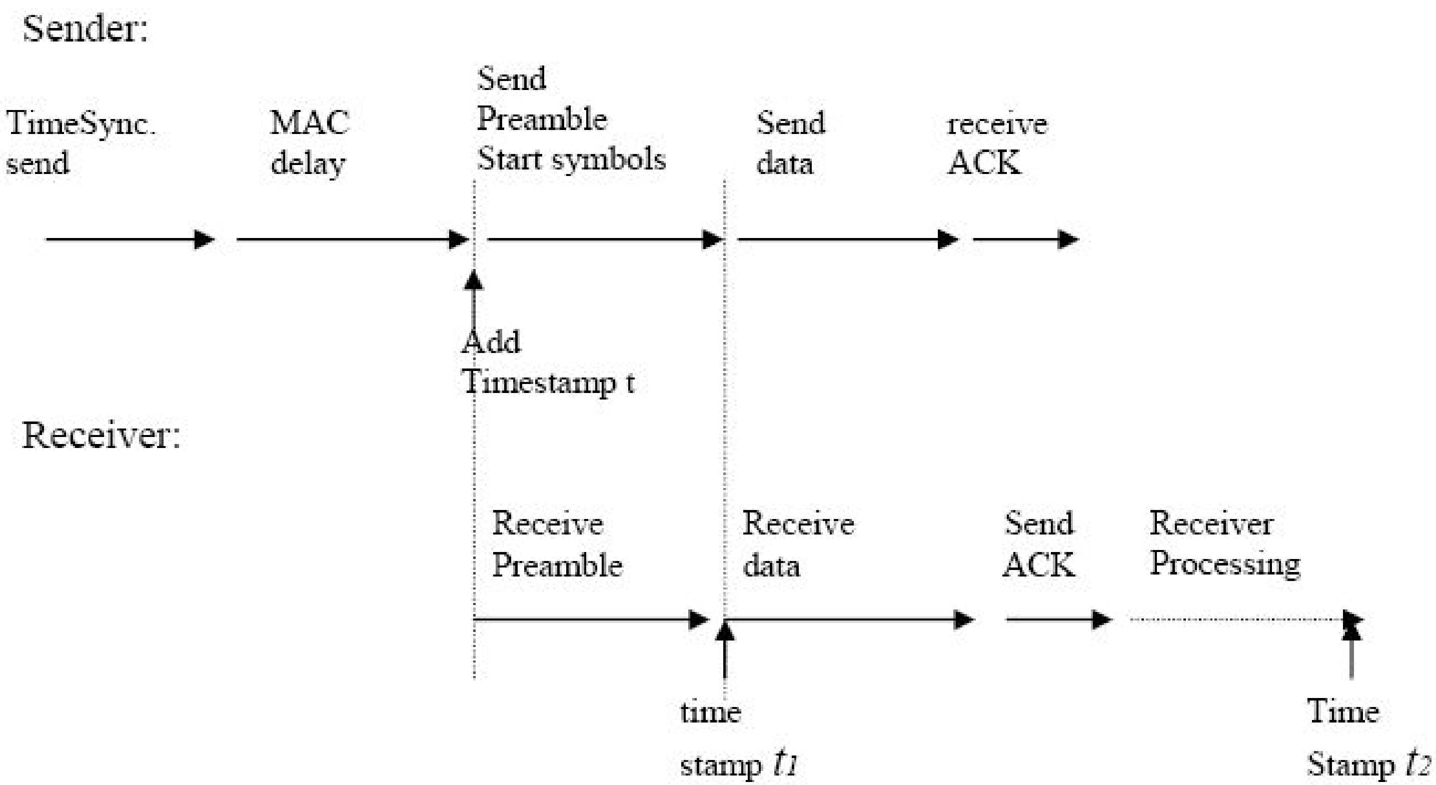
  - The tighter the bounds get, the higher is the synchronization precision
  - Requires high amount of data points and quite complex
  - Algorithm precision increases with the increased number of data points



Differences Between TS and MS

- **Different methods in selecting useful data points**
  - Tiny-Sync
    - ✓ Keeps only four constrains of all data points
    - ✓ Does not always give the best solution for the bounds
  - Mini-Sync is an extension of Tiny-Sync
    - ✓ More optimal solution with increased complexity
    - ✓ Keeps also the data points which may be useful by some future data points to give tighter bounds
    - ✓ A data point is discarded only if it is definitely useless
    - ✓ Quite complex selection criterion

DMTS [5]



$$t_r = t + nv + (t_2 - t_1)$$

$$t_e = nv$$

Other Algorithms

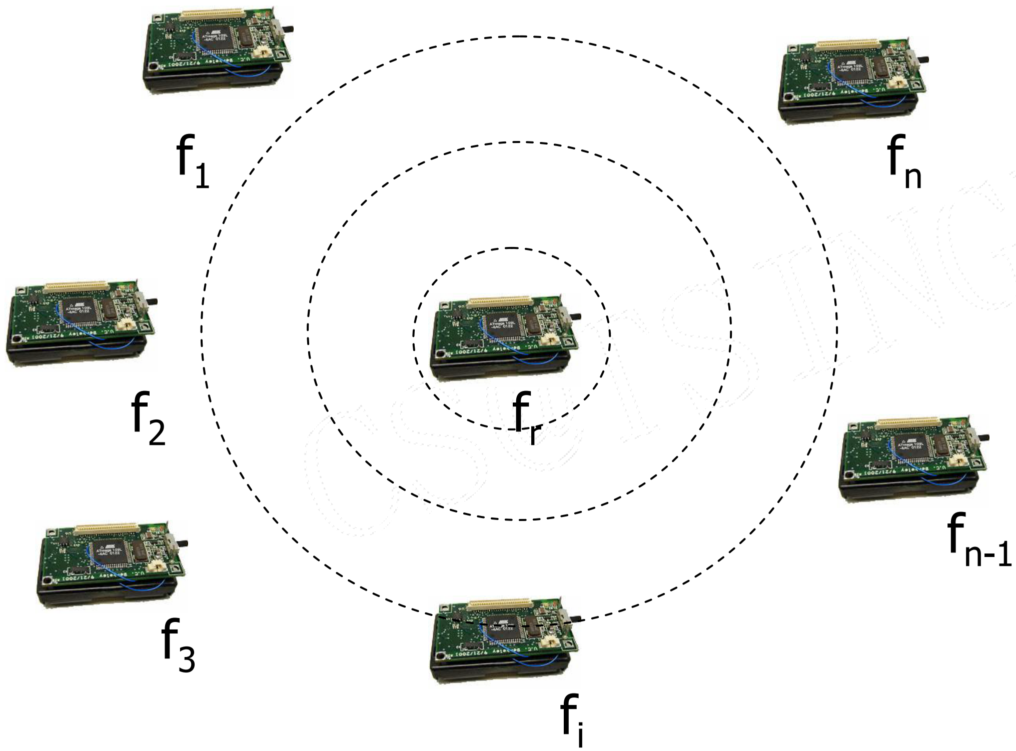
- **FTSP**
  - Sender-receiver uni-directional mechanism
  - Similar with DMTS, but timestamp after SYNC
  - Linear regression (time, offset)
- **TSync**
  - Special channel for time synchronization
- **AD**
- ……

Summary

Class	RBS	TPSN	TS/TM	LTS	TSync	FTPS	AD
Internal vs. External	I	E	I	E	E	I	I
Cont. vs. On-demand	O	C	C	O	C	C	C
All nodes vs. Subset	S	A	S	A/S	A	A	A
Rate vs. Offset	RO	O	RO	O	O	RO	O
Assumption							
Broadcast	X	X			X	X	X
Uni vs. Bidirectional	U	B	B	B	B	U	B
Constant rate			X				
Bound Drift				X			
Multichannel					X		
MAC access		X				X	

Our scheme

- **Reference broadcast synchronization**

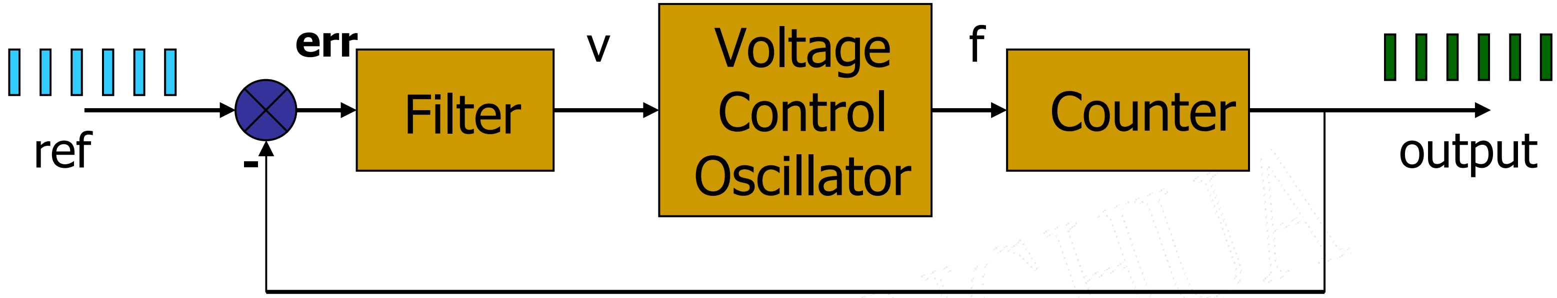


$$f_r=f_1=f_2=\ldots=f_n$$

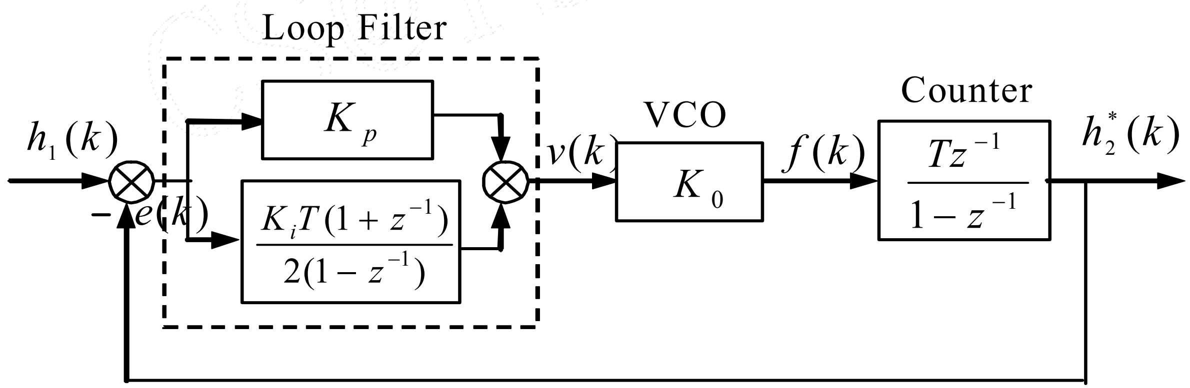
$$o_r=o_1=o_2=\ldots=o_n$$

Phase Locked Loop (PLL)

- **Principle**



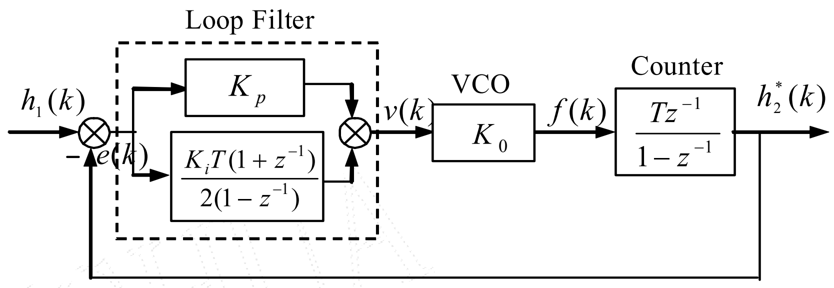
- **Discrete System**



Digital PLL

Avoid the extra hardware to reduce cost

$$h_2^*(k+1) = h_2^*(k) + K_0 v(k) T$$

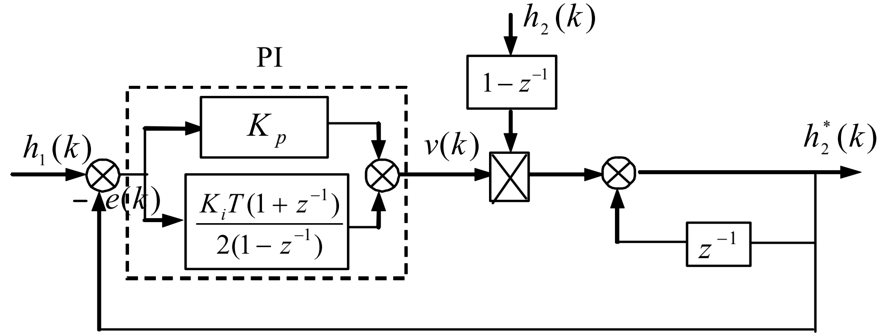


K_0 is physical frequency of VCO

$$K_0\{t(k+1) - t(k)\} = h_2(k+1) - h_2(k)$$

t(k) is real time, $h_2(k)$ is local clock of node

$$h_2^*(k+1) = h_2^*(k) + v(k)[h_2(k+1) - h_2(k)]$$



Tuning the parameter of filter

$$G_o(z)=\left[K_p+\frac{K_iT(1+z^{-1})}{2(1-z^{-1})}\right]\frac{K_0Tz^{-1}}{1-z^{-1}}=\frac{K_0T\left[(2K_p+K_iT)z+K_iT-2K_p\right]}{2(z-1)^2}$$

Let zero point at 0.5

$$z=\frac{2K_p-K_iT}{2K_p+K_iT}=\frac{1}{2} \Rightarrow 2K_p=3K_iT$$

Characteristic Equation of closed loop

$$2(z-1)^2+K_0T(2K_p+K_iT)z+K_0T(K_iT-2K_p)=0$$

$$\Downarrow$$

$$z^2-2(K_0K_iT^2-1)z+1-K_0K_iT^2=0$$

Let the closed system be overdamp

$$\Downarrow$$

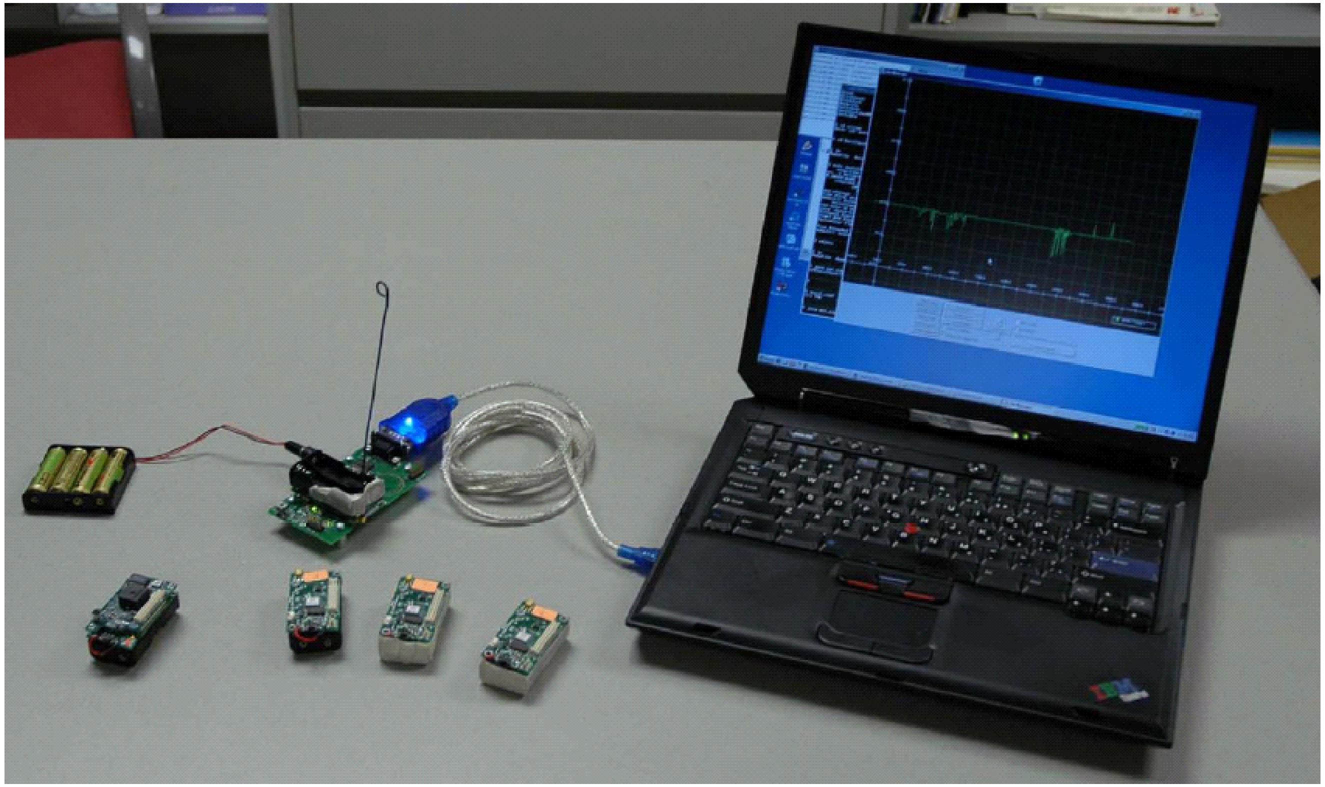
$$4(K_0K_iT^2-1)^2-4(1-K_0K_iT^2)=0 \Rightarrow K_i\ K_0T^2=1$$

Let T=1, K_0=62.5KHz for MICA2

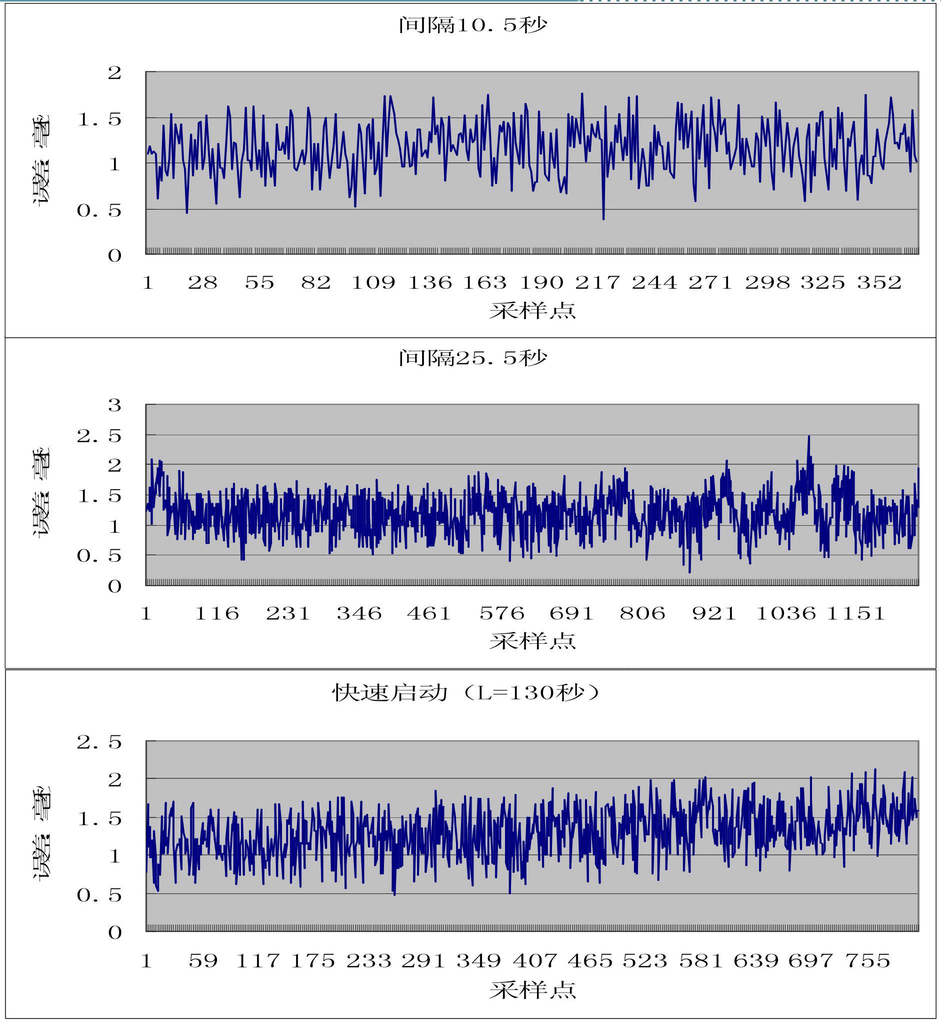
$$K_p=1.5K_i=2.4\times10^{-5} \qquad K_i=1.6\times10^{-5}$$

Experiment Configuration

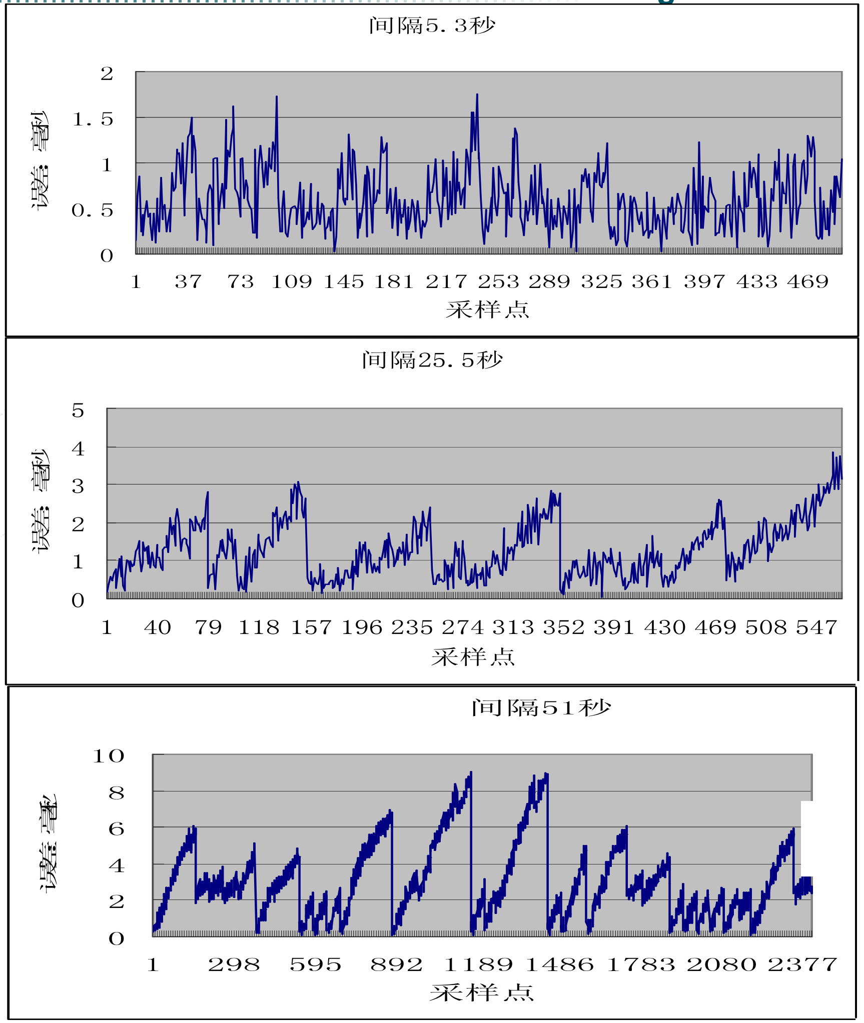
- Crossbow's Mote Kit 5152
- 5 Mica2 nodes, MIB 510 base station connected to PC by serial port



Experiment Results



PLL



TPSN

Conclusion & Future Works

- The time synchronization scheme based on PLL(TS-PLL) can reach millisecond resolution
- TS-PLL can compensate drift and offset simultaneously
- TS-PLL is simple, the overhead of communication is very small, and the space complexity is low
- Need to extend the multi-hop, including experiment.
  - Use the scheme developed in TPSN
- Improve the resolution

References

[1] J. Elson, L. Girod, and D. Estrin, "Fine-Grained Network Time Synchronization using Reference Broadcasts," In Proc. of the 5th Symposium on Operating systems Design and Implementation, Boston, MA. December 2002.

[2]Mihail L. Sichitiu and Chanchai Veerarittiphan, "Simple, Accurate Time Synchronization for Wireless Sensor Networks", In IEEE Wireless Communications and Networking Conference (WCNC'03), March 2003.

[3]Saurabh Ganeriwal, Ram Kumar, and Mani B. Srivastava, "Timing-Sync Protocol for Sensor Networks". In First ACM Conference on Embedded Networked Sensor Systems (SenSys), November 2003.

[4]J. Greunen, J. Rabaey, "Lightweight Time Synchronization for Sensor Networks," In Proc. of WSNA'03, San Diego, California, USA.

[5]Su Ping, "Delay Measurement Time Synchronization for Wireless Sensor Networks", Intel Research Berkeley Lab, 2003.

[6]Qun Li and D. Rus, "Global Clock Synchronization in Sensor Network", In Proc. of INFOCOM 2004, Hong Kong, China, March, 2004

[7]Hui Dai and Richard Han, "TSync: A Lightweight Bidirectional Time Synchronization Service for Wireless Sensor Networks", ACM SIGMOBILE Mobile Computing and Communications Review, 8(1):125–139, January 2004

[8]Maroti M., Kusy B., Simon G., Ledeczi A., "The Flooding Time Synchronization Protocol", In 2nd ACM Conference on Embedded Networked Sensor Systems(SenSys 04), Baltimore, November 2004

[9] S. Pal Chaudhuri, A. K. Saha and D. B. Johnson, "Adaptive Clock Synchronization in Sensor Networks", Proceedings of the Third International Symposium on Information Processing in Sensor Networks (IPSN 2004), pp. 340-348, IEEE, Berkeley, CA, April 2004

Thank You