Adaptive Control-Based Clock Synchronization in Wireless Sensor Networks

Kasım Sinan YILDIRIM*, Ruggero CARLI+, Luca SCHENATO+

* Department of Computer Engineering, Ege University, İzmir, TURKEY

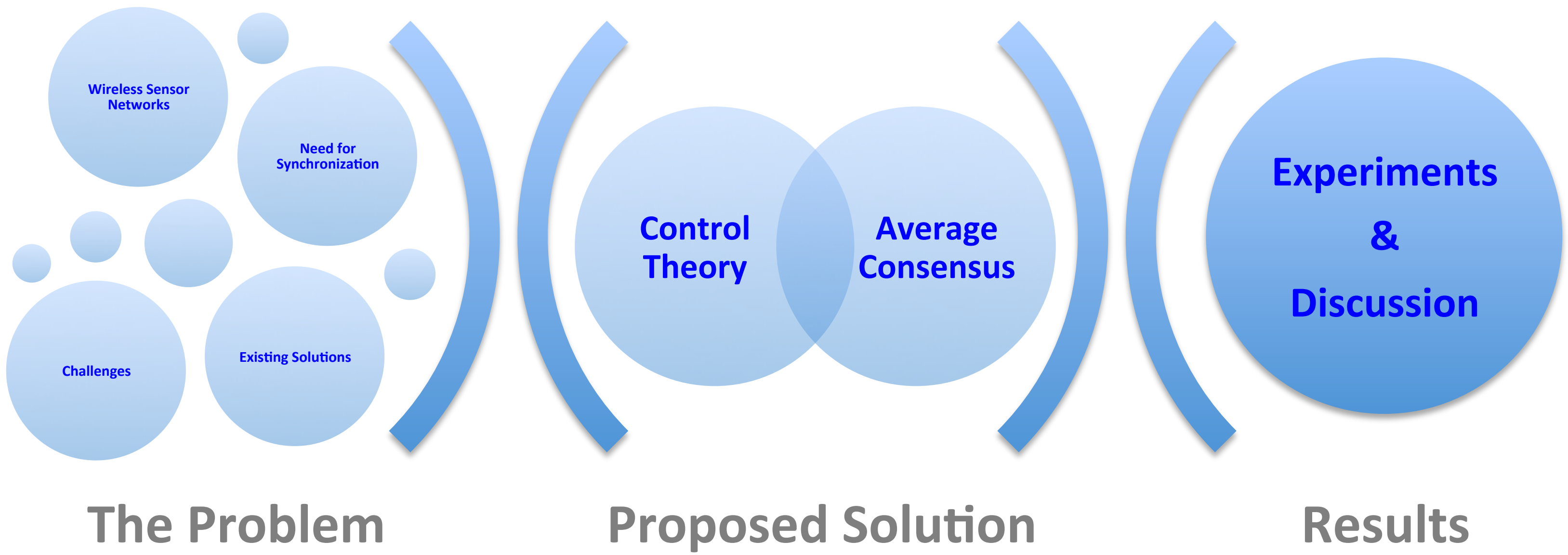
+ Department of Information Engineering, University of Padova, Padova, ITALY

Design and Analysis of Communication Systems (DACS)

University of Twente

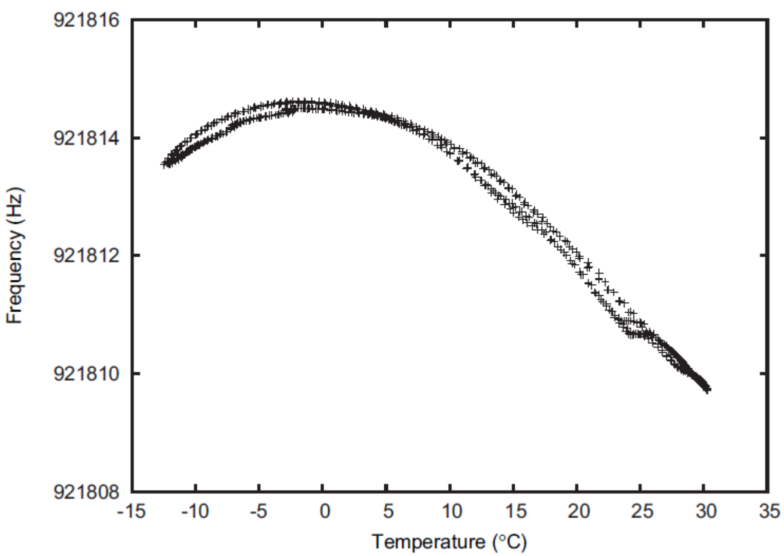
March 19, 2015

Outline


Wireless Sensor Networks
Need for Synchronization
Challenges
Existing Solutions
Control Theory
Average Consensus
Experiments
&
Discussion
The Problem
Proposed Solution
Results

Wireless Sensor Networks

- Hardware clock (built-in clock)
  - Counter register
    - Clocked with external crystal
      - 32kHz, 7.37 MHz nominal rate
    - Read-only
- Clock drift
  - Deviation from the nominal rate
    - 30-100 parts per million (ppm)
    - dependent on environmental factors such as:
      - temperature, power supply, aging...

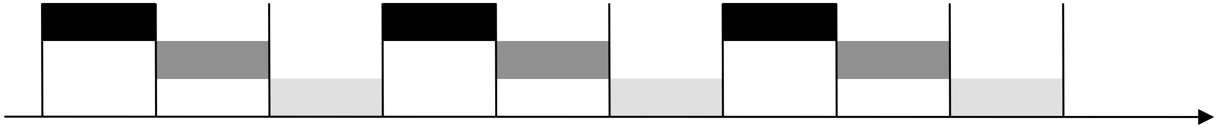


[Sommer et al. 2009]

Need for Synchronization

- Assigning global timestamps
  – sensed data and events
- Coordinated actions
  - Duty-cycling of the nodes for energy efficient operation

  - Low-power, TDMA-based MAC layer

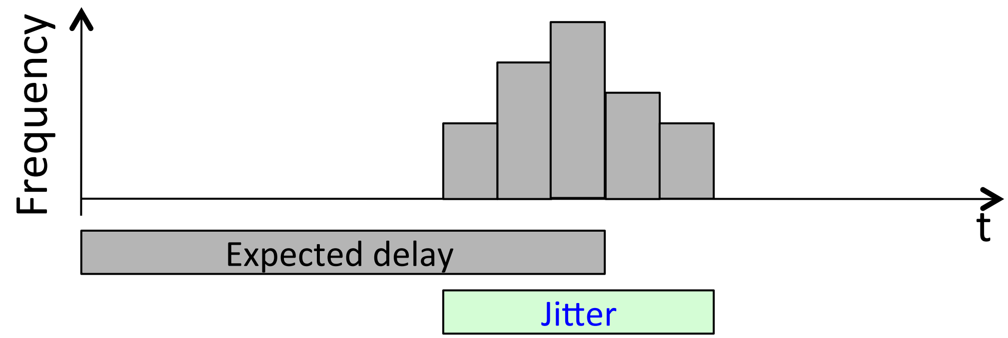


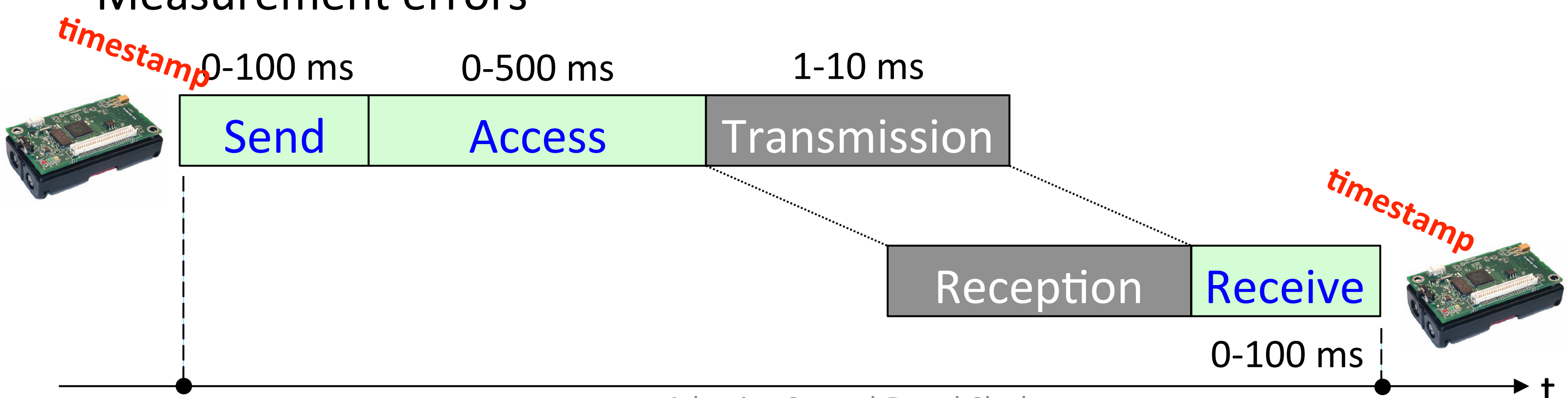
- Applications
  – Localization via time-of-flight measurements
  – Tracking of moving objects

Time Synchronization

- Communicate
  - Exchange current time information periodically
    - Hardware clock value
- Compute
  - Calculate a logical clock
    - Software function
    - represents the global time
- Challenge
  - Message delay
    - Deterministic and non-deterministic components
  - Measurement errors

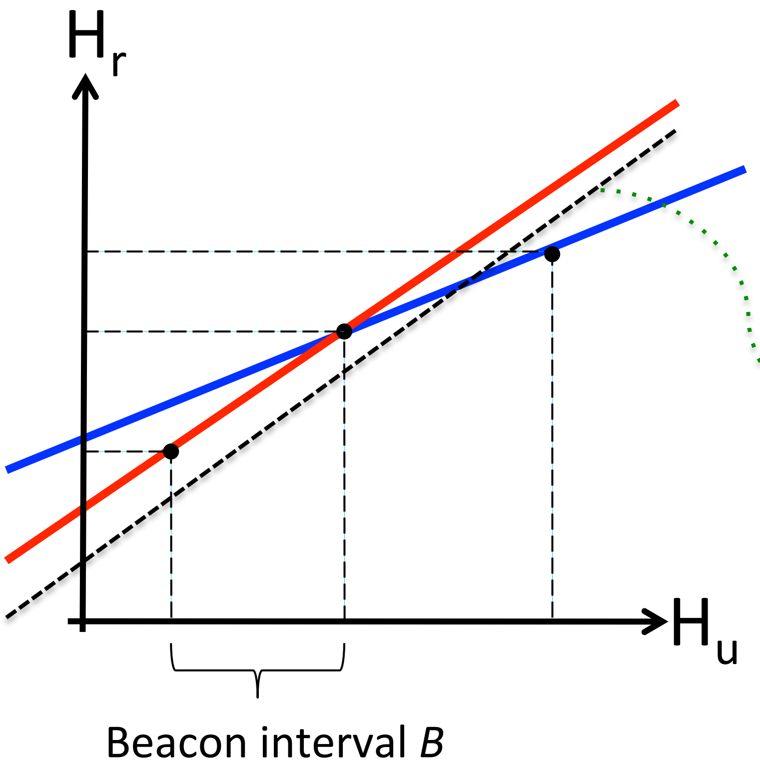
MAC Layer timestamping





Pairwise Synchronization in Practice

- Master – Slave Synchronization
  - Collect (H_u,H_r) timestamp pairs periodically
  - Employ least-squares regression



Reference Node r

Slave Node u

Linear relationship: $H_r(t)=\alpha+\beta H_u(t)$

clock offset

relative clock rate

Least-Squares with Two Pairs

Measurement errors

Slope: $\hat{\beta} = \frac{H_r(t_1) - H_r(t_0)}{H_u(t_1) - H_u(t_0)}$

...suremen...

...con...ted by ...sion delays

Error contributed by the transmission delays

Measure...

...curement errors

Poor multi-hop performance scalability

Intercept:

$\hat{\alpha} = \frac{H_r(t_1) + H_r(t_0) \ldots}{2} \ldots \frac{\ldots}{2}$

Errors ent... on-l...early to the equations!

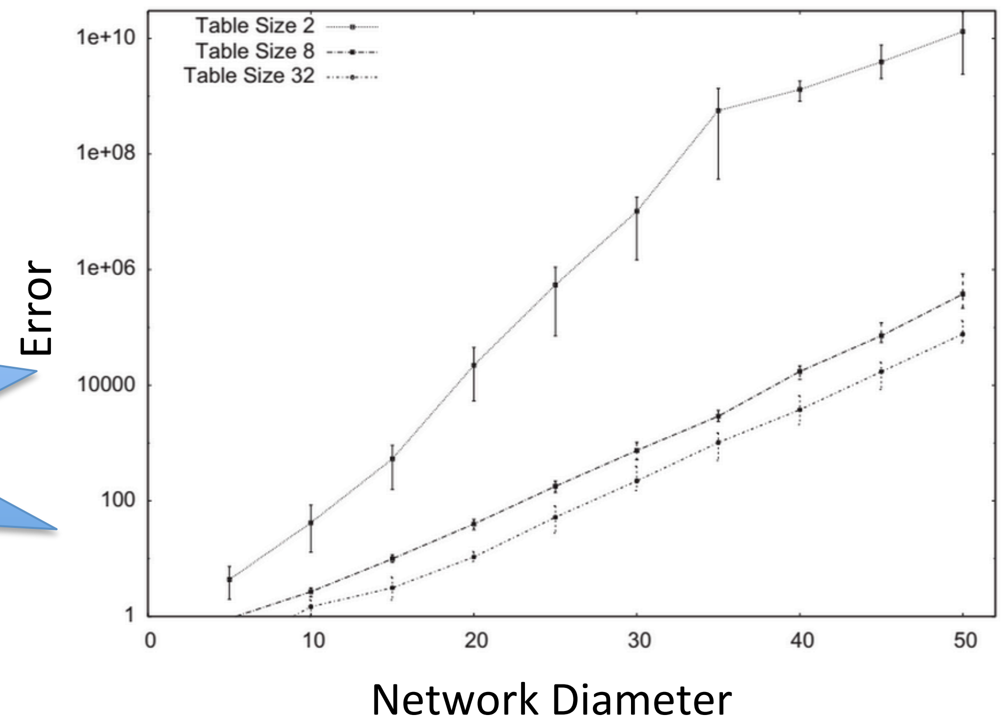
$$\hat{H}_r(t) = \hat{\alpha} + \hat{\beta} H_u(t)$$

The effect of various error sources appear as **multiplicative noise!**

Multi-hop Performance of Least-Squares

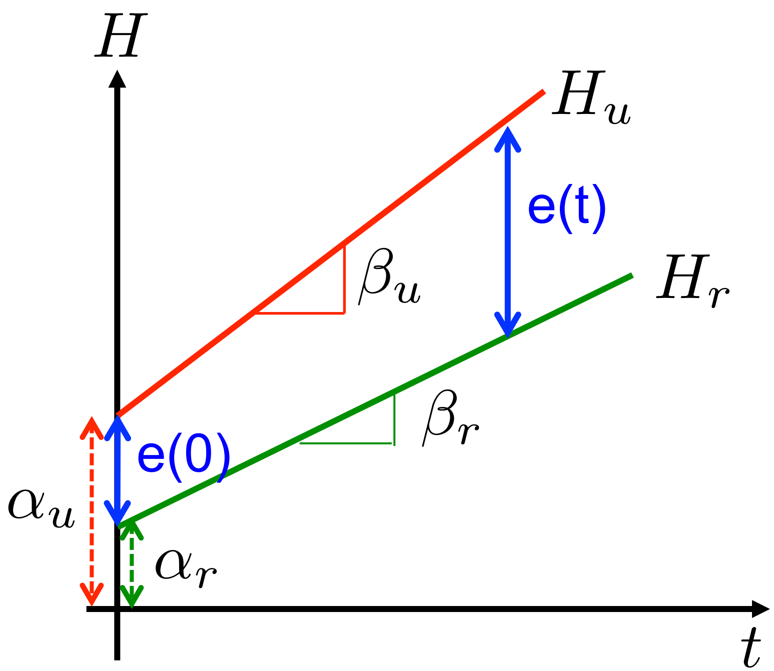
- Simulation of multi-hop least-squares based time synchronization with different regression tables sizes.
  - Amplification of errors at each hop
  - Multiplicative noise!

Exponentially Increasing Synchronization Error!



[Lenzen et al. 2009]

Time Synchronization as a Control Problem

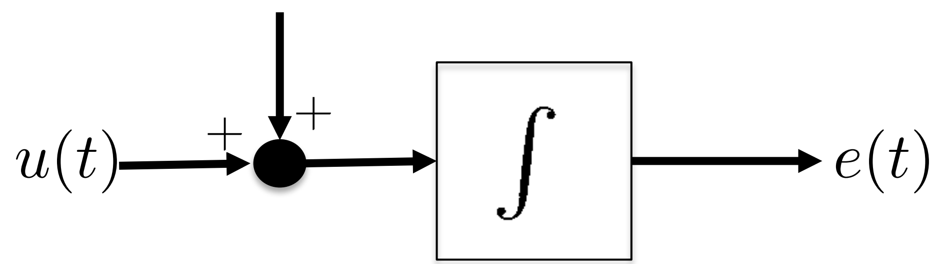


$$e(t) = H_u(t) - H_r(t)$$

$$e(t+B) = e(t) + \underbrace{(\beta_u - \beta_r)}_{\text{Speed difference}}\underbrace{B}_{\text{input}}$$

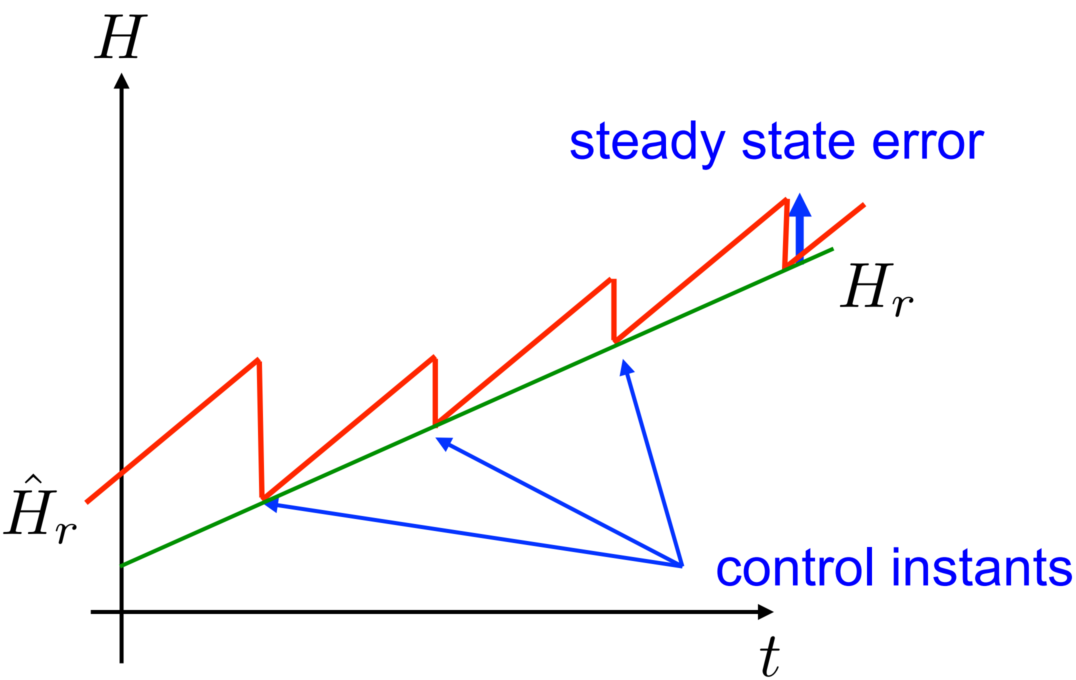
constant disturbance

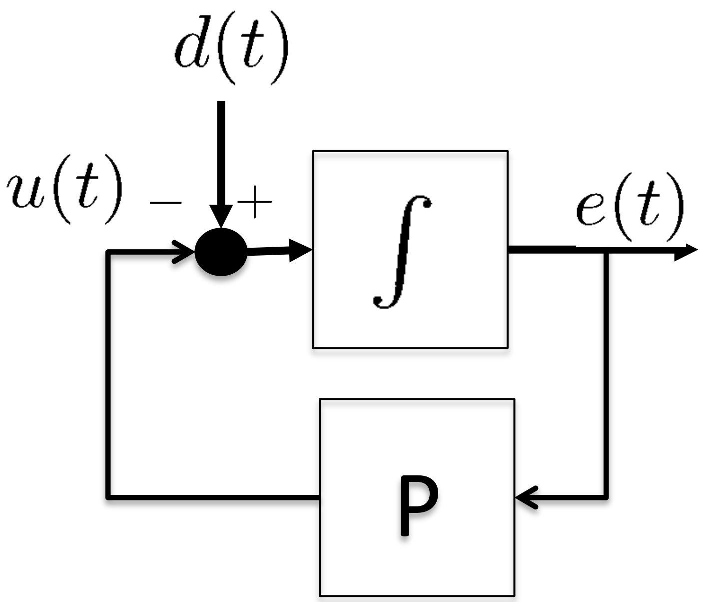
$$d(t) = (\beta_u - \beta_r)$$



Proportional Control

$$u(t) = -k_P e(t)$$

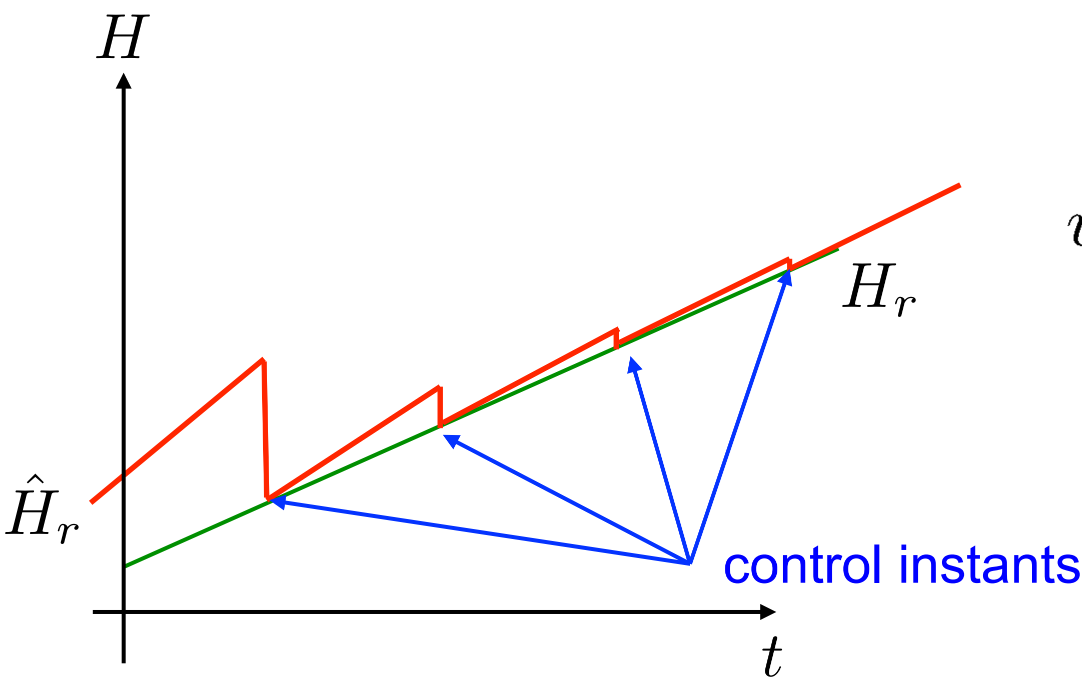


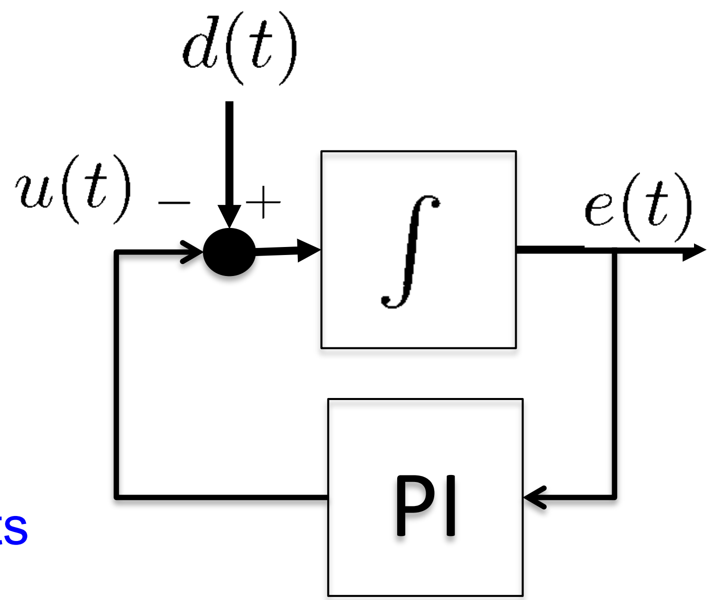


Compensates only initial offset differences

Proportional-Integral Control

$$u(t) = -k_P e(t) - k_I \int e(t) dt$$





Compensates both offset and clock speed differences

PISync Algorithm - Pairwise Synchronization

Reference Clock Estimate

$$\hat{H}_r(t) = \underbrace{\hat{H}_r(t_{up})}_{\text{Offset}} + \underbrace{\Delta(t_{up})}_{\text{Spe}}\underbrace{(H_u(t) - H_u(t_{up}))}_{\text{Local time passed since update}}$$

Update Rule

- Receive $<H_r(t_{up}$

  Calculate Error

  Update Offset

  Update Speed

$$\ldots up) - H_r(t_{up})$$

$$\ldots - k_P e(t_{up})$$

$$\ldots (t_{up}) - k_I e(t_{up})$$

No need to store received timestamps!

No regression table!

Very simple arithmetic operations!

Convergence Conditions

$$0 < k_P < 2$$

$$0 < k_I < \frac{2(2-k_P)}{f_{nom}B}$$

ptimal onvergence Rate

$$k_I^* = \frac{1}{f_{nom}B} \text{ for } k_P = 1$$

Not the smallest steady-state error!

Integral gain should be adjusted adaptively!

PISync Algorithm – Error Dynamics

For $k_P=1$, PISync algorithm becomes

$$\hat{H}_r(t_{up}^+) = H_r(t_{up}) + \underbrace{q_r(t_{up})}_{\text{Measurement errors}} + \underbrace{\varepsilon(t_{up})}_{\text{Error contributed by the transmission delays}}$$

$$\Delta(t_{up}^+) = \Delta(t_{up}) - k_I(\hat{H}_r(\ldots) - H \ldots q_u(t \ldots) - q_r(t_{up}) + \overbrace{\varepsilon(t_{up})}^{\text{Error contributed by the transmission delays}})$$

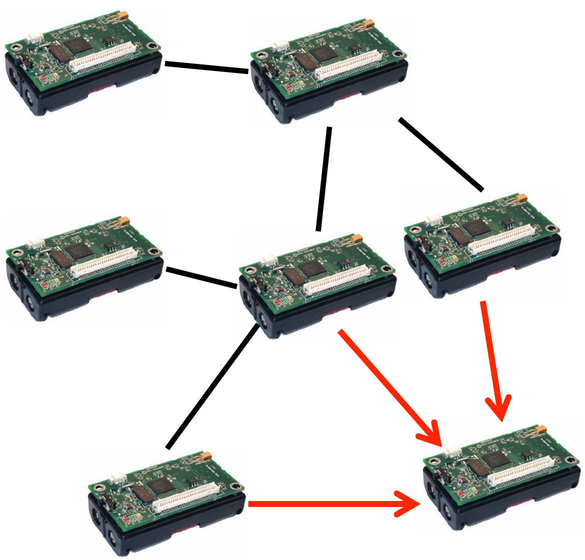
...ement errors

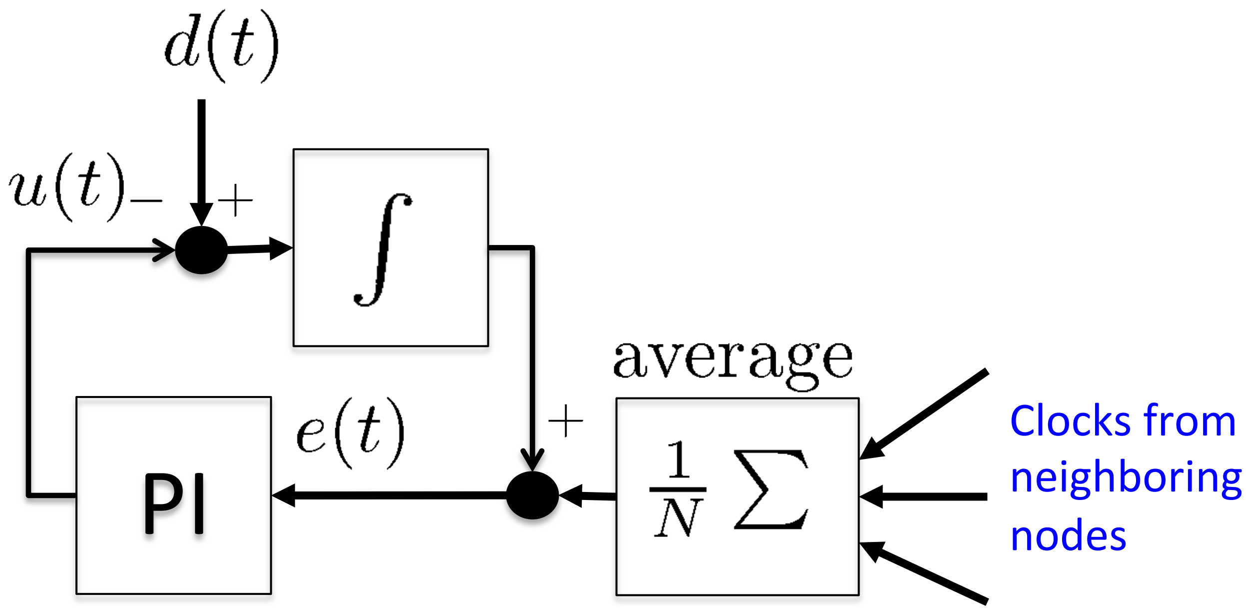
Scalable multi-hop performance

Erro... ations!

The effect of various error sources appear as **additive noise!**

PI control + Average Consensus



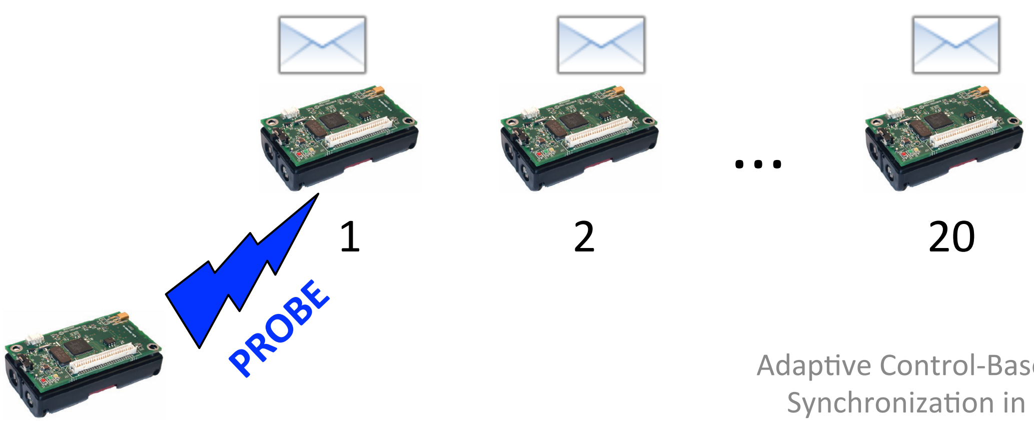


No special reference node!

Nodes synchronize directly to their neighbors!

Testbed

- Testbed setup
  - 20 MICAz sensor nodes
  - FTSP (Flooding Time Synchronization Protocol [Marotti et al. 2004]) and PISync implemented in TinyOS 2.1
- Network topology
  - Single-hop setup
    - Nodes are communication range of each other
  - Virtual line and grid topologies (in software)



Base Station

Flooding Time Synchronization Protocol Marotti et al. 2004

- In every 30 seconds, each node
  - Receives a new reference clock estimate
  - Stores the timestamp pair on the regression table of size 8
  - Employs least-squares regression on the pairs in the regression table
  - Forwards its estimate of the reference clock
    - each estimate is affected by
      - Message delays
      - Measurement errors
      - Estimation errors

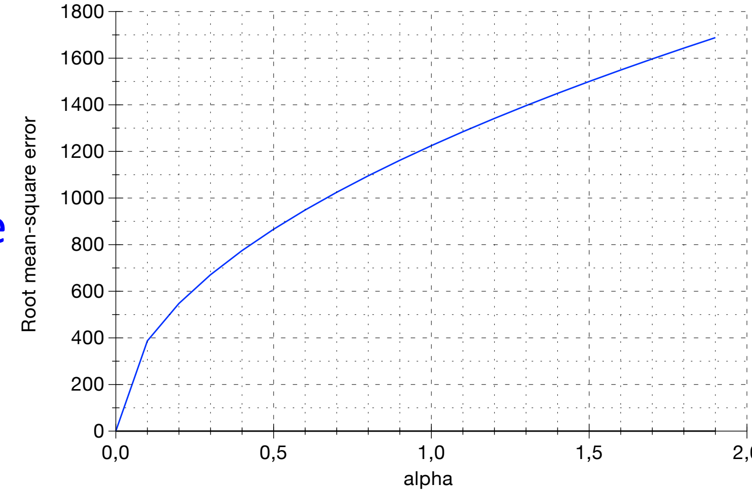
reference

1 2 … 20

PISync - Integral Gain Adaptation

$$k_I^* = \frac{1}{f_{nom} B} \text{ for } k_P = 1$$

As k_I gets smaller, we obtain smaller steady-state error but longer convergence time!



$$e_{max} = 2B\underbrace{\Delta f}_{\text{Maximum frequency } \textit{error}}/f_{nom}$$

Adaptation Rule

If $e(h) > e_m$ ... **then** ... $= 0$

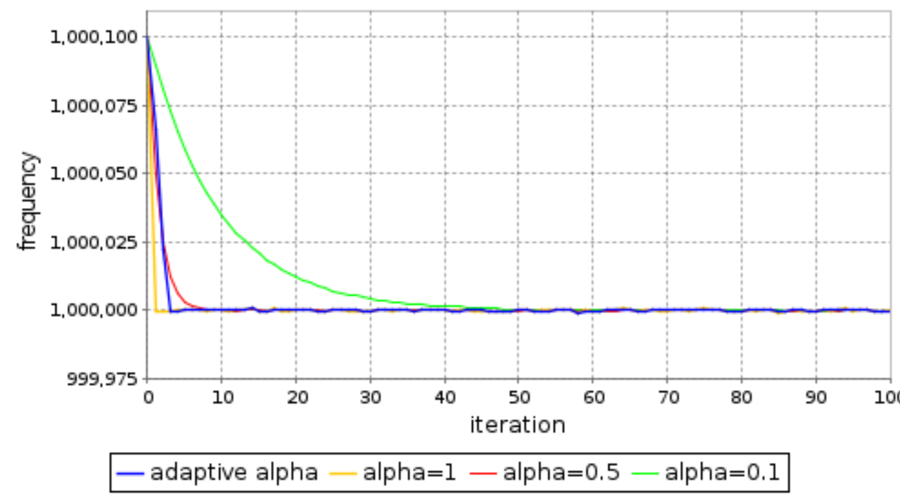
Els... k ... $=$ ...

El... ...$\max\{2k_I, k_I^*\}$

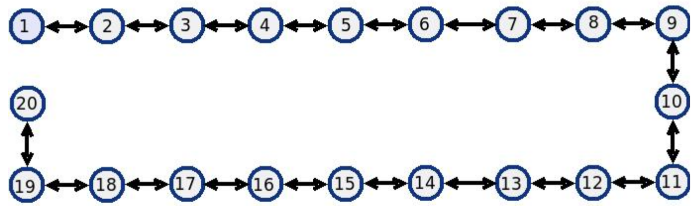
Els...

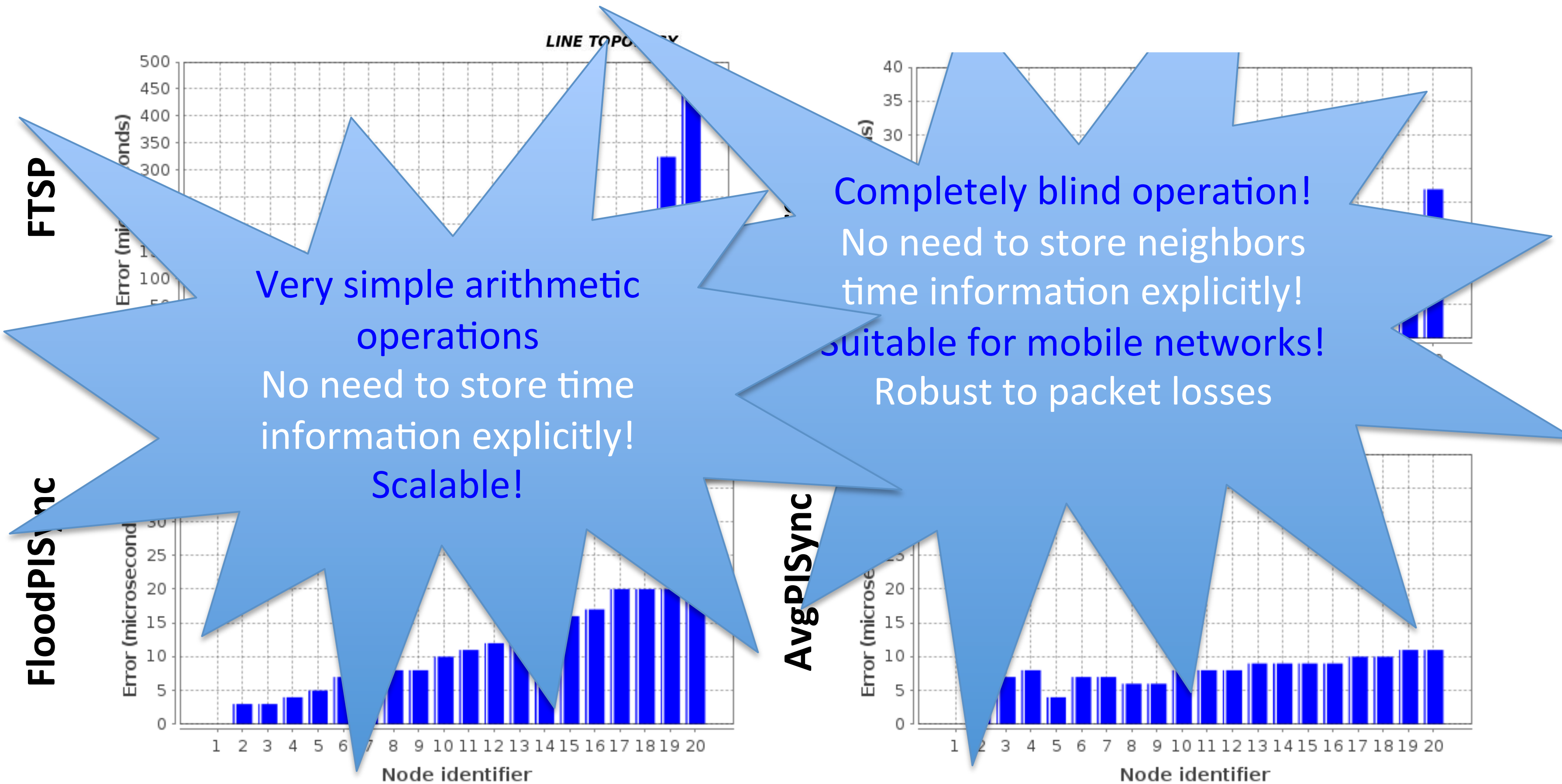
Gradually increase
or decrease

Inspired fro... ild... and ...urcan et al. 2014]



FTSP vs PISync





Very simple arithmetic operations
No need to store time information explicitly!
Scalable!

Completely blind operation!
No need to store neighbors time information explicitly!
Suitable for mobile networks!
Robust to packet losses

PISync Complexity

```
void synchronize(TimeSyncMsg *msg)
{
    int32_t timeError;
    float newSkew = skew;

    /* calculate offset differe
    timeError = msg->localTime;
    call GlobalTime.local2Global((u                    (&timeError
    timeError = msg->globalTime - tim

    /* adjust the speed of the logical c
    if( timeError < E_MAX && timeError > -
      /* calculate adaptive alpha */
      if(lastError != 0 &&
        currentAlpha *= ((f
      }

      currentAlpha = fabs(currentAlph

      if(currentAlpha > ALPHA_MAX) c
      /* adjust rate multiplier *
      newSkew += currentAlpha*(
    }

    lastError = timeError;

    /* update logical clock parameters */
    atomic{
      skew = newSkew;
      clock  = msg->globalTime;
      lastUpdate = msg->localTime;
    }
}
```

Very easy to implement & code

20 times better performance!
50 times fewer operations!
4 times less RAM requirements!
Low power consumption!

		PISync
		20 microseconds
		16 bytes
	s	15432 bytes
	5.5 m	145 microseconds

Just 15 Lines of TinyOS code!

Conclusions

- We considered time synchronization as a control problem
- We solved this problem with a very simple and practical technique
  - Proportional-Integral Controller
- We introduced a new time synchronization protocol
  - PISync
- We observed
  - Better performance scalability
  - Less resource consumption
    - Lower CPU and main memory overhead
    - Lower power consumption

Future Research Directions

- Performance evaluation on real mobile networks
- Implementation & evaluation of time-synchronized MAC layer
  - Adaptive receiving window
  - Evaluation of power consumption
- Other algorithmic techniques?
  - Better steady-state error & convergence
  - Even lower resource requirements?