Wireless Sensor Networks

14th Lecture

12.12.2006

CoNe Freiburg

University of Freiburg
Computer Networks and Telematics
Prof. Christian Schindelhauer

Christian Schindelhauer
schindel@informatik.uni-freiburg.de

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

Overview

- The time synchronization problem
- Protocols based on sender/receiver synchronization
- Protocols based on receiver/receiver synchronization
- Summary

Clocks in WSN nodes

- **Often, a *hardware clock* is present:**
  - Oscillator generates pulses at a fixed nominal frequency
  - A counter register is incremented after a fixed number of pulses
    - Only register content is available to software
    - Register change rate gives achievable time resolution
  - Node i's register value at real time t is $H_i(t)$
    - Convention: small letters (like t, t') denote real physical times, capital letters denote timestamps or anything else visible to nodes

- **A (node-local) software clock is usually derived as follows:**

$$L_i(t) = \theta_i \, H_i(t) + \phi_i$$

  - (not considering overruns of the counter-register)
  - θ_i is the (drift) rate, ϕ_i the phase shift
  - Time synchronization algorithms modify θ_i and ϕ_i, but not the counter register

Synchronization accuracy / agreement

- **External synchronization:**
  - synchronization with external real time scale like UTC
  - Nodes i=1, ..., n are accurate at time t within bound δ when

$$|L_i(t) - t| < \delta \text{ for all } i$$

    - Hence, at least one node must have access to the external time scale
- **Internal synchronization**
  - No external timescale, nodes must agree on common time
  - Nodes i=1, ..., n agree on time within bound δ when

$$|L_i(t) - L_j(t)| < \delta \text{ for all } i,j$$

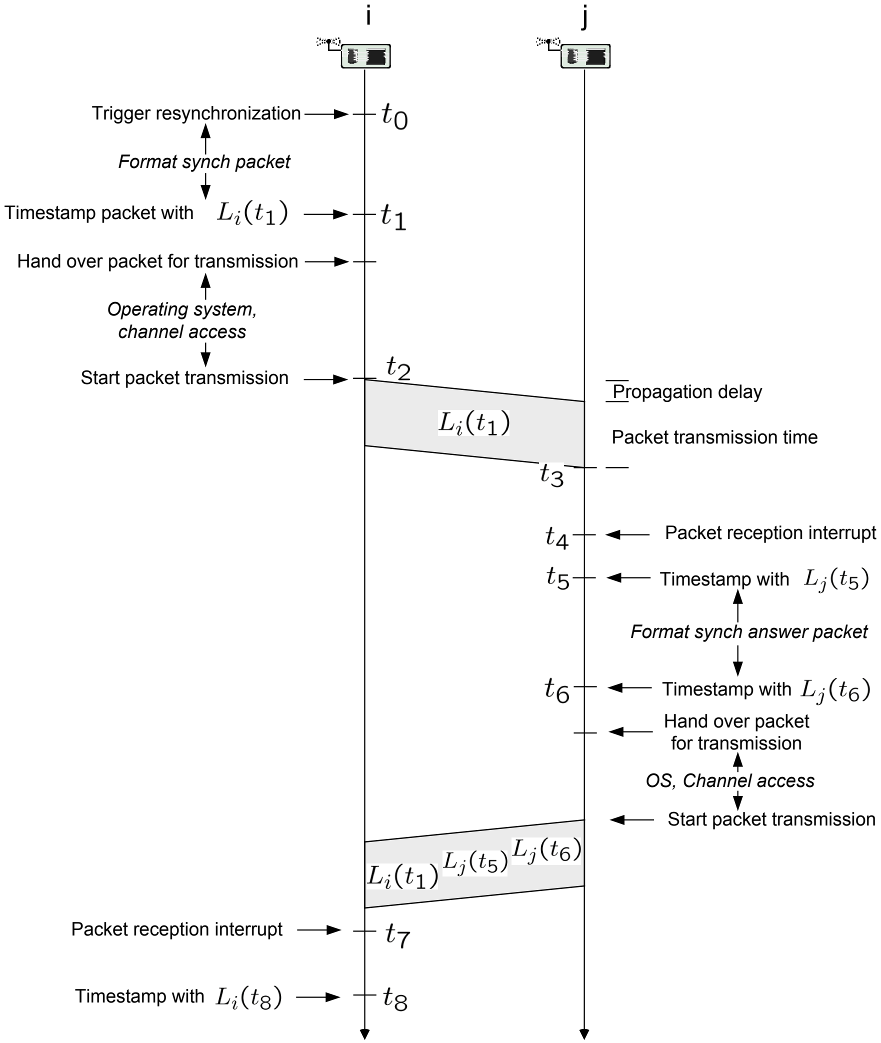
Overview

- The time synchronization problem
- Protocols based on sender/receiver synchronization
- Protocols based on receiver/receiver synchronization
- Summary

LTS – Lightweight Time Synchronization

- **Jana van Greunen, Jan Rabaey, WSNA 2003**
- **Overall goal**
  - synchronize the clocks of sensor nodes to one reference clock
  - e.g. equipped with GPS receiver
- **It allows to synchronize**
  - the whole network,
  - or parts of it
  - also supports post-facto synchronization
- **It considers only phase shifts**
  - does not try to correct different drift rates
- **Two components:**
  - pairwise synchronization: based on sender/receiver technique
  - networkwide synchronization: minimum spanning tree construction with reference node as root

LTS – Pairwise Synchronization



- **Assumptions:**
  - no drift
  - same hardware, same OS, same software
- **Goal: compute**

$$\Delta = L_i(t_1) - L_j(t_1)$$

- **Further assumptions**

$$\Delta = L_i(t_k) - L_i(t_k)$$

$$L_j(t_5) - L_j(t_1) = L_i(t_5) - L_i(t_1)$$

$$\approx$$

$$L_i(t_8) - L_i(t_6) = L_j(t_8) - L_j(t_6)$$

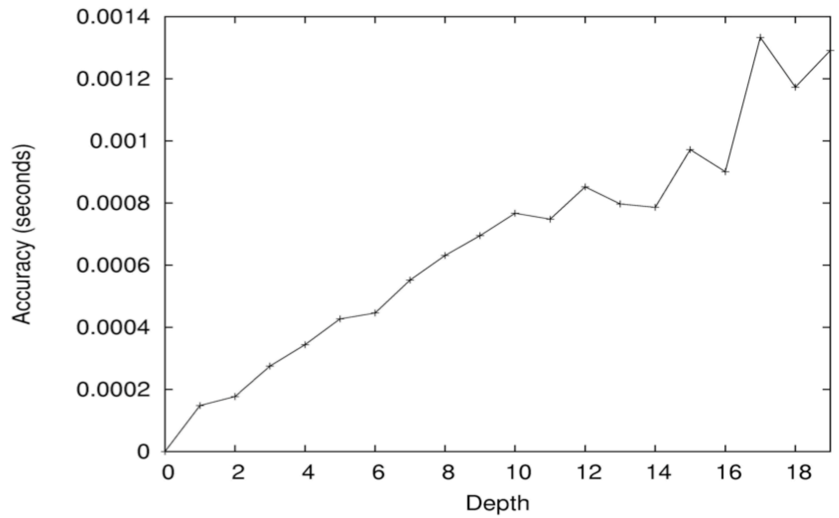
- **Solution:**

$$\Delta = \frac{L_i(t_8) - L_j(t_6)}{2} - \frac{L_j(t_5) - L_i(t_1)}{2}$$

LTS – Network-wide Synchronization

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

- **All nodes synchronize to a given reference node R**
  - R's direct neighbors (level-1 neighbors) synchronize with R
  - Two-hop (level-2) neighbors synchronize with level-1 neighbors
  - ....
- **Creates a spanning tree**
- **Problem: Error amplification**
  - Consider a node i with hop distance h_i to the root node
  - Assume that:
    - all synchronization errors are independent
    - all synch errors are identically normally distributed with zero mean and variance $4\sigma^2$
  - Then node i's synchronization error is a zero-mean normal random variable with variance $h_i \, 4 \, \sigma^2$
  - Hence, a tree with minimal depth minimizes synchronization errors



LTS
Centralized Multihop LTS

- **Reference node R**
  - triggers construction of a spanning tree
  - it first synchronizes its neighbors
  - then the first-level neighbors synchronize second-level neighbors
  - and so on

- **Different distributed algorithms for construction of spanning tree can be used**
  - e.g. Distributed Depth First Search (DDFS), Echo algorithm

- **Communication costs:**
  - Costs for construction of spanning tree
  - Synchronizing two nodes costs 3 packets, synchronizing n nodes costs 3n packets

Echo

- **Algorithm for tree exploration**
- **Less efficient:**
  - O(nm) time
  - n: nodes
  - m: edges
- **In practice:**
  - O(d) time
  - d: depth of tree

```
var rec_u       : integer init 0;
    father_u    : neighbor init undef;

Algorithm for the initiator:
    forall v ϵ Neigh_u do send <echo> to v ;
    while rec_u < |Neigh_u| do
      begin receive <echo> ; rec_u := rec_u + 1 end

Algorithm for other nodes:
    receive <echo> from w ; father_u := w ; rec_u := 1;
    forall v ϵ Neigh_u \ {w} do send <echo> to v ;
    while rec_u < |Neigh_u| do
      begin receive <echo> ; rec_u := rec_u + 1 end ;
    send <echo> to father_u
```

Distributed DFS
(Awerbuch 1985)

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

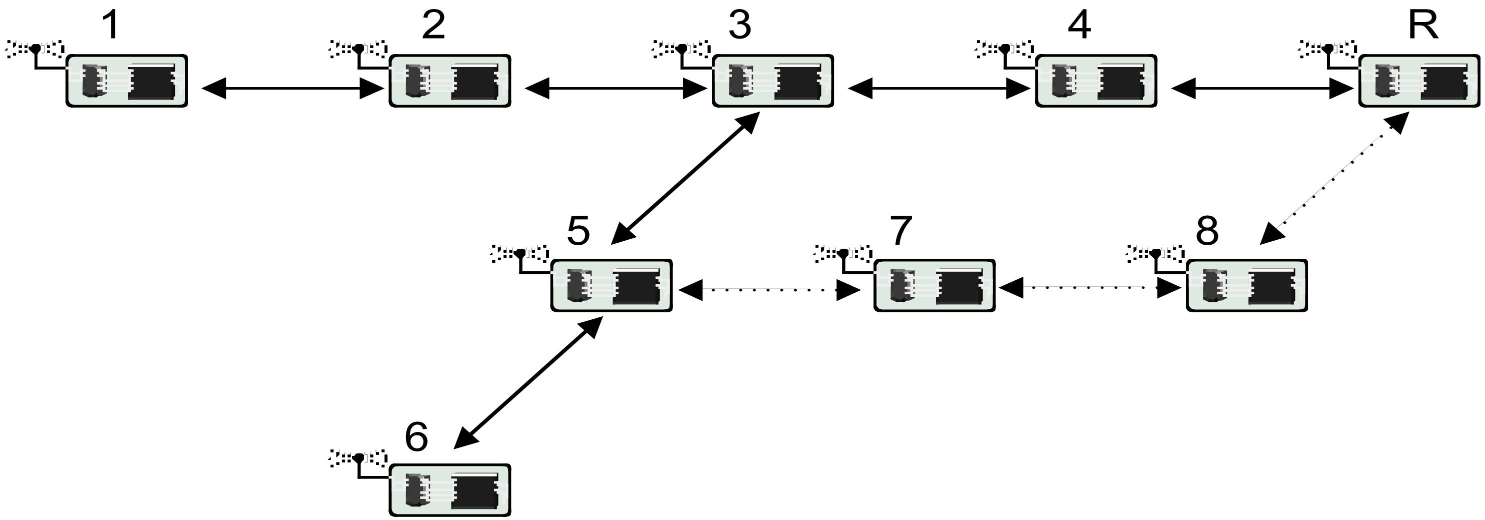
- **Performs DFS with 4 m messages and in time 4n-2**
  - m: number edges
  - n: time
- **BFS has higher complexity:**
  - algorithms known with
    - 10 n $m^{1/2}$
    - $O(n^{1.6} + m)$
  - messages
  - difficult to perform in a distributed manner
- **Hope:**
  - DDFS finds BFS-tree

```
Start the algorithm at node u the initiator:
visited_u := true ;
    for all x ϵ Neigh_u do send <visit> to x;
    for all x ϵ Neigh_u do receive <ack> from x;
    for some w ϵ Neigh_u do send <dfs> to w; statusu[w] := cal
    end

Upon receipt of <visit> from v:
    statusu[v] := done ; send <ack> to v

Upon receipt of <dfs> from v:
    if not visited_u then
      begin visited_u := true; status_u[v] := father;
          begin forall x ϵ Neigh_u \{v} do send <visit> to x;
          forall x ϵ Neigh_u \{v} do receive <ack> from x;
      end;
    if there is a w ϵ Neigh_u with status_u[w] = unused
      begin send <dfs> to w; status_u[w] := cal
    else if there is a w ϵ Neigh_u with status_u[w] = father
      begin send <dfs> to w end
    else (* initiator *) stop
```

LTS
Distributed Multihop LTS

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

- **No explicit construction of spanning tree needed, but each node knows identity of reference node(s) and routes to them**
- **When node 1 wants to synchronize with R, an appropriate request travels to R – following this, 4 synchronizes to R, 3 synchronizes to 4, 2 synchronizes to 3, 1 synchronizes to 2**
  - By-product: nodes 2, 3, and 4 are synchronized with R

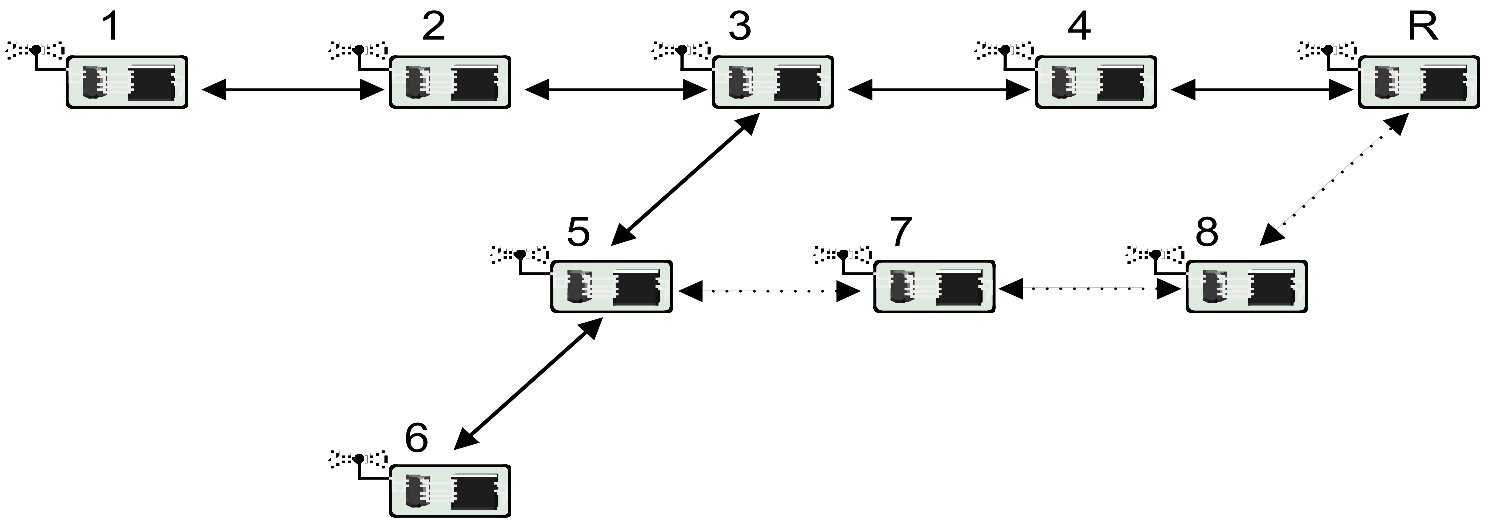


- **Small depth trees are constructed implicitly**
  - node 1 should know shortest route to the closest reference node

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

Distributed Multihop LTS Variations

- **When node 5 wants to synchronize with R, it can:**
  - issue its own synchronization request using route over 3, 4 and put additional synchronization burden on them
  - ask in its local neighborhood whether someone is synchronized or has an ongoing synchronization request and benefit from that later on
  - Enforce usage of path over 7, 8 (path diversification) to also synchronize these nodes



Distributed Multihop LTS Variations

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

- **Discussion:**
  - Simulation shows that distributed multihop LTS needs more packets (between 40% and 100%)
    - when ***all*** nodes have to be synchronized, even with optimizations
  - Distributed multihop LTS allows to synchronize only the minimally required set of nodes

    → post-facto synchronization

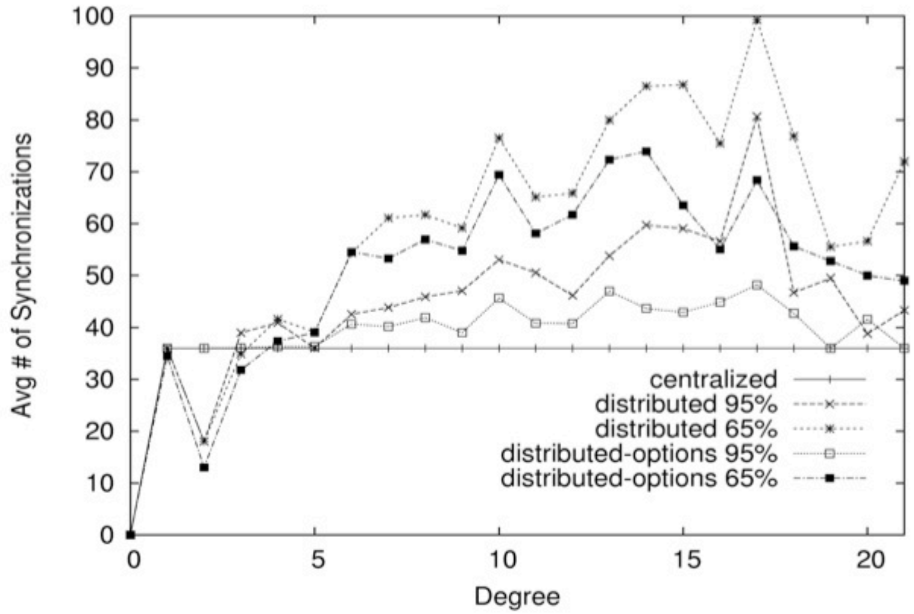


Figure 8: Average number of synchronizations as a function of node degree

Other Sender-/Receiver-based Protocols

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

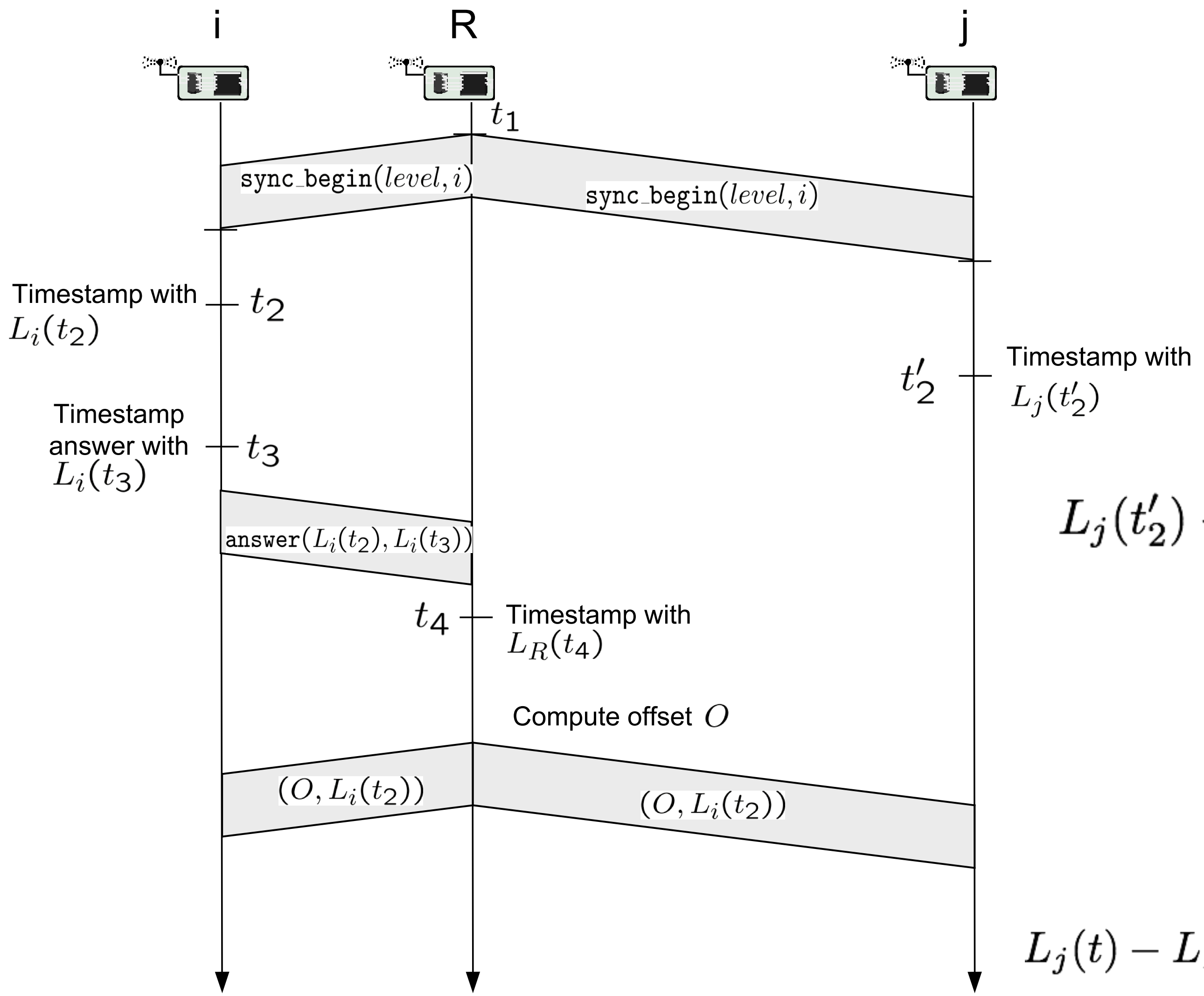
- **These protocols work similar to LTS, with some differences in:**
  - Method of spanning tree construction
  - How and when to take timestamps
  - How to achieve post-facto synchronization
- **One variant: TPSN (Timing-Sync Protocol for Sensor Networks)**
  - Ganeriwal, Kumar, Srivastava [SenSys 2003]
  - Pairwise-protocol similar to LTS
    - but timestamping at node i happens immediately before first bit appears on the medium
    - timestamping at node j happens in interrupt routine
  - Spanning tree construction based on level-discovery protocol:
    - root issues level_discovery packet with level 0
    - neighbors assign themselves level 1 + level value from level_discovery
    - neighbors wait for some random time before they issue level_discovery packets indicating their own level
    - Nodes missing level_discovery packets for long time ask their neighborhood

TSync

- **TSync combines:**
  - HRTS (Hierarchy Referencing Time Synchronization): a protocol to synchronize a broadcast domain to one of its members
  - ITR (Individual-based Time Request): a sender-/receiver protocol similar to LTS/TPSN
  - A networkwide synchronization protocol
- **HRTS provides a technique to synchronize a group of nodes to a reference node with only three packets!**

HRTS
Hierarchy Referencing Time Synchronization

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer



- **i and j**
  - synchronize to R'
  - cannot hear each other
- **Assumptions:**
  - no drift

$$O = L_i(t) - L_R(t)$$

$$L_j(t'_2) - L_j(t_1) \approx L_i(t_2) - L_i(t_1)$$

- **Compute**

$$\begin{aligned} \Delta &= \\ L_j(t) - L_R(t) &= O - (L_i(t_2) - L_j(t'_2)) \end{aligned}$$

HRTS - Discussion

- **Node j is not involved in any packet exchange**
  - ➔ by this scheme it is possible to synchronize an arbitrary number of nodes to R’s clock with only three packets!!
- **The synchronization uncertainty comes from:**
  - The error introduced by R when estimating $O_{R,i}$
  - The error introduced by setting $t_2 = t_2'$
    - This makes HRTS only feasible for geographically small broadcast domains
- **Both kinds of uncertainty can again be reduced by:**
  - timestamping outgoing packets as lately as possible (relevant for t_1 and t_3)
  - timestamping incoming packets as early as possible (relevant for t_2, t_2', t_4
- **The authors propose to use extra channels for synchronization traffic**
  - when late timestamping of outgoing packets is not an option
  - Rationale: keep MAC delay small

TSync – Networkwide Synchronization

- **It is assumed that some reference nodes are present in the network, e.g. having a GPS receiver**
- **Initialization:**
  - Reference nodes assign themselves a level of 0
  - All other nodes assign themselves a level of ∞
  - The reference node becomes a root node and synchronizes its neighbors
- **Whenever any node receives a sync_begin packet with a smaller level x than its current level y:**
  - It synchronizes to the issuing node
  - It assigns itself a level y := x+1
  - It synchronizes its neighbors
- **This way a minimal spanning tree is constructed**

Overview

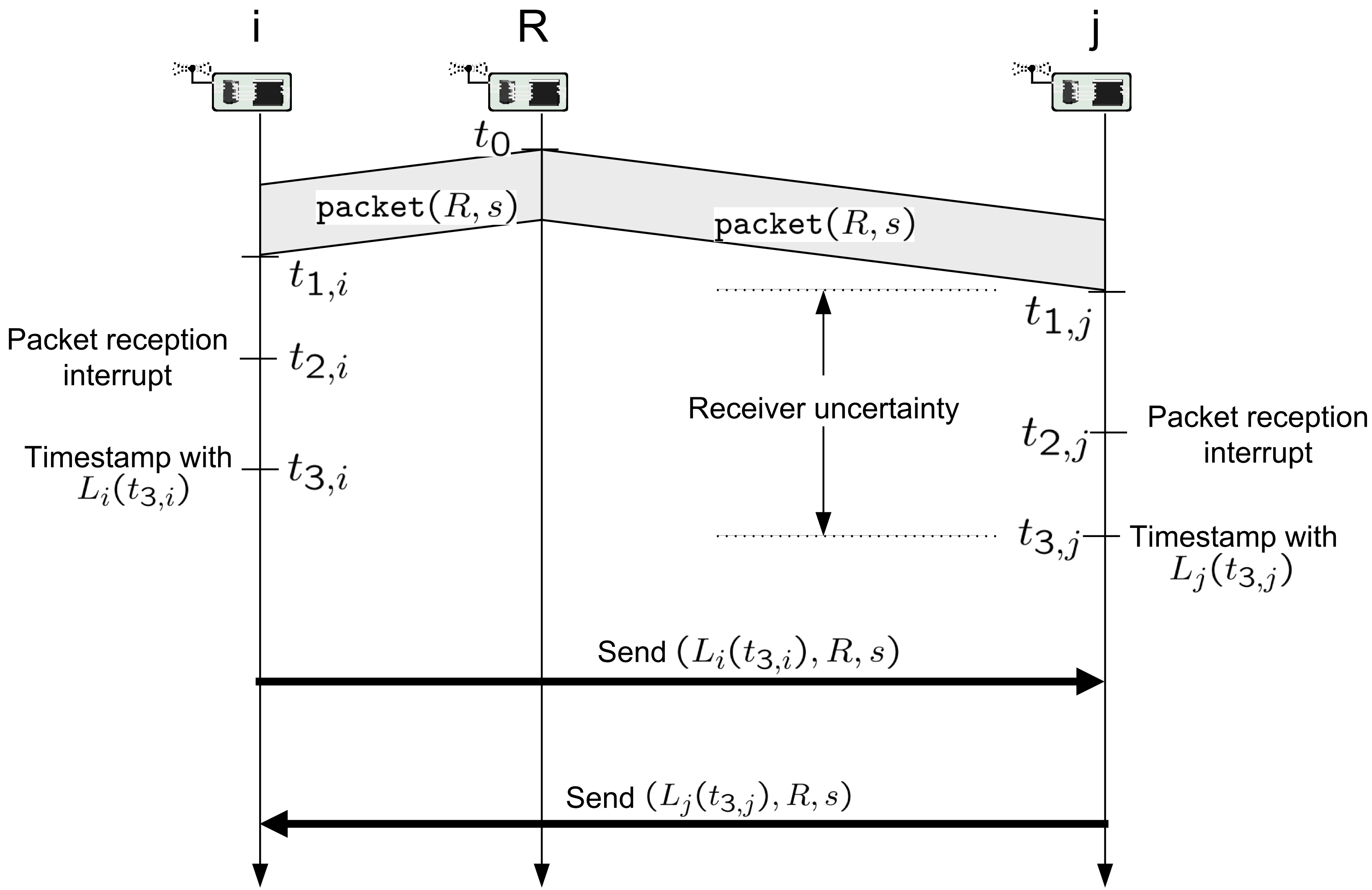
University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

- The time synchronization problem
- Protocols based on sender/receiver synchronization
- Protocols based on receiver/receiver synchronization
- Summary

Protocols based on receiver/receiver synchronization

- **Receivers of packets synchronize among each other**
  - not with the transmitter of the packet
- **RBS: Reference Broadcast Synchronization (Elson, Girod, Estrin, OSDI 2002)**
  - Synchronize receivers within a single broadcast domain
  - A scheme for relating timestamps between nodes in different domains
- **RBS**
  - does not modify the local clocks of nodes
  - but computes a table of conversion parameters for each peer in a broadcast domain
  - allows for post-facto synchronization

RBS – Synchronization in a Broadcast Domain



RBS – Synchronization in a Broadcast Domain

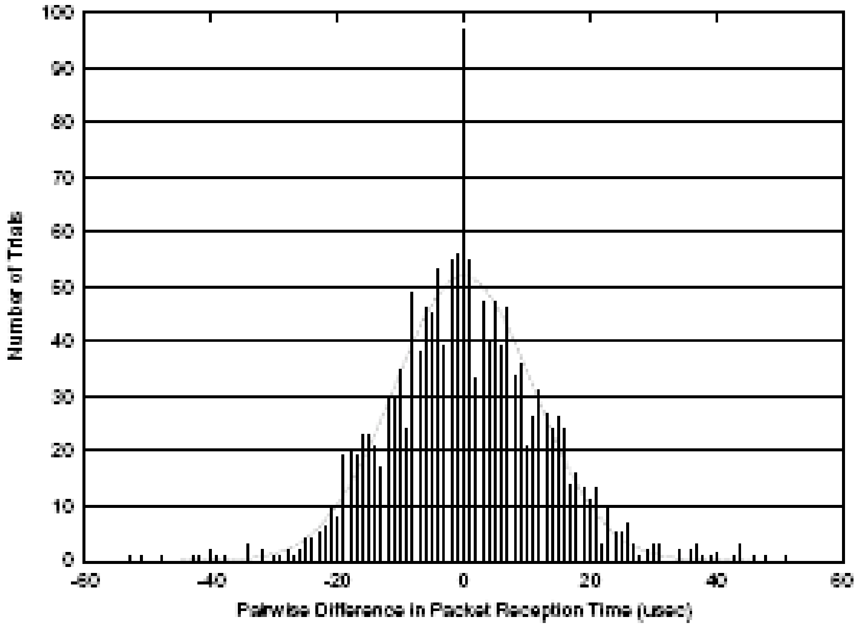
- **The goal is to synchronize i's and j's clocks to each other**
- **Timeline:**
  - Reference node R broadcasts at time t_0 some synchronization packet carrying its identification R and a sequence number s
  - Receiver i receives the last bit at time $t_{1,i}$, gets the packet interrupt at time $t_{2,i}$ and timestamps it at time $t_{3,i}$
  - Receiver j is doing the same
  - At some later time node i transmits its observation ($L_i(t_{3,i})$, R, s) to node j
  - At some later time node j transmits its observation ($L_j(t_{3,j})$, R, s) to node i
  - The whole procedure is repeated periodically, the reference node transmits its synchronization packets with increasing sequence numbers
    - R could also use ordinary data packets as long as they have sequence numbers ...
- **Under the assumption $t_{3,i} = t_{3,j}$ node j can figure out the offset $O_{i,j} = L_j(t_{3,j}) - L_i(t_{3,i})$ after receiving node i's final packet – of course, node i can do the same**

RBS – Synchronization in a Broadcast Domain

- **The synchronization error in this scheme can have two causes:**
  - There is a difference between $t_{3,i}$ and $t_{3,j}$
  - Drift between $t_{3,i}$ and the time where node i transmits its observations to j
- **But:**
  - In small broadcast domains and when received packets are timestamped as early as possible the difference between $t_{3,i}$ and $t_{3,j}$ is very small
    - As compared to sender-/receiver based schemes the MAC delay and operating system delays experienced by the reference node play no role!!
  - Drift can be neglected when observations are exchanged quickly after reference packets
  - Drift can be estimated jointly with Offset O when a number of periodic observations of $O_{i,j}$ have been collected
    - This amounts to a standard least-squares line regression problem

RBS – Synchronization in a Broadcast Domain

- **Elson et al**
  - measured pairwise differences in timestamping times at a set of receivers
  - when timestamping happens in the interrupt routine (Berkeley motes)
- **This is just the distribution of the differences $t_{3,i}$-$t_{3,j}$**



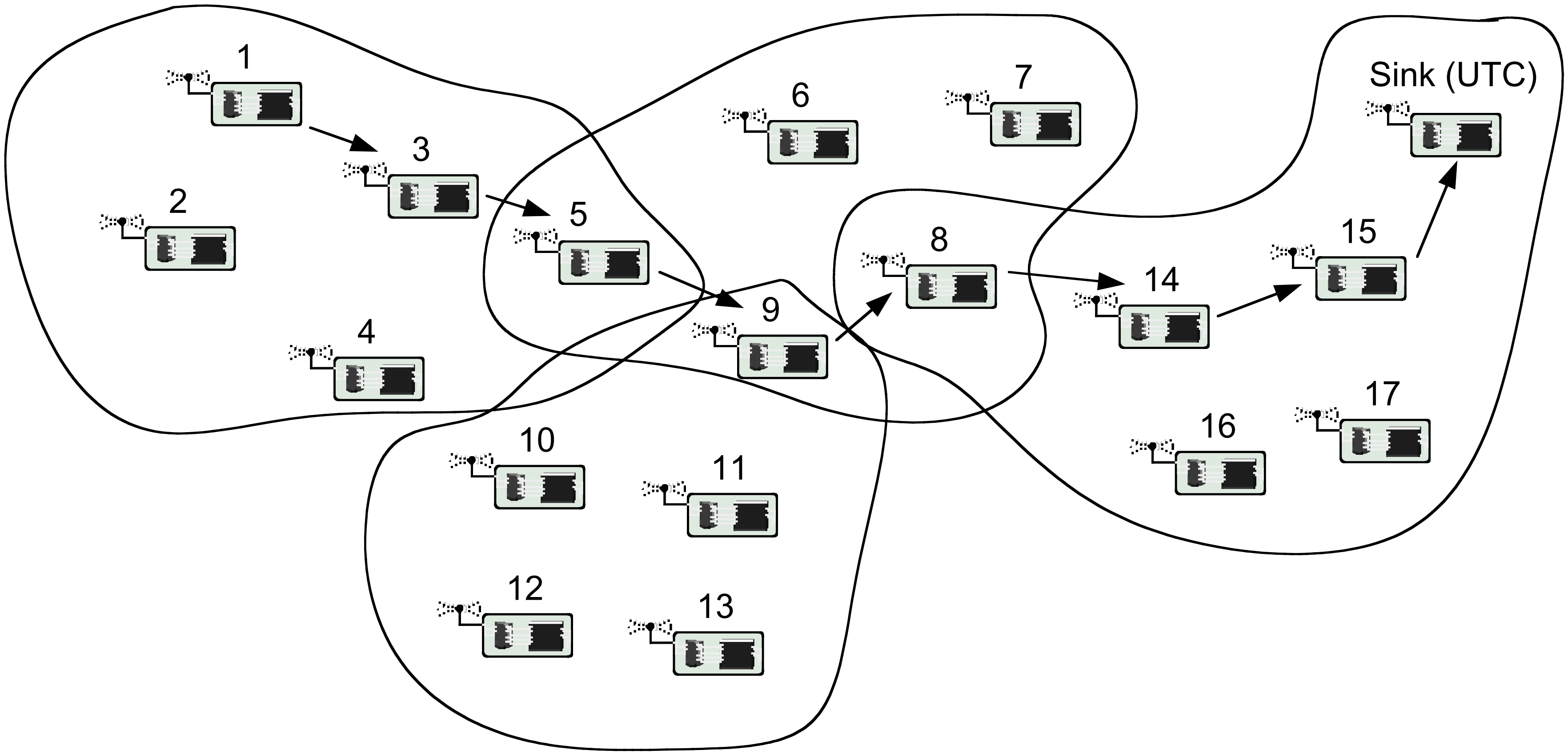
RTS – Synchronization in a Broadcast Domain

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

- **Communication costs:**
  - Be m the number of nodes in the broadcast domain
  - First scheme: reference node collects the observations of the nodes, computes the offsets and sends them back ➔ 2 m packets
  - Second scheme: reference node collects the observations of the nodes, computes the offsets and keeps them, but has responsibility for timestamp conversions and forwarder selection ➔ m packets
  - Third scheme: each node transmits its observation individually to the other members of the broadcast domain ➔ m (m-1) packets
  - Fourth scheme: each node broadcasts its observation ➔ m packets, but unreliable delivery
- **Collisions are a problem:**
  - The reference packets trigger all nodes simultaneously to tell the world about their observations
- **Computational costs: least-squares approximation is not cheap!**

RBS – Network Synchronization

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer



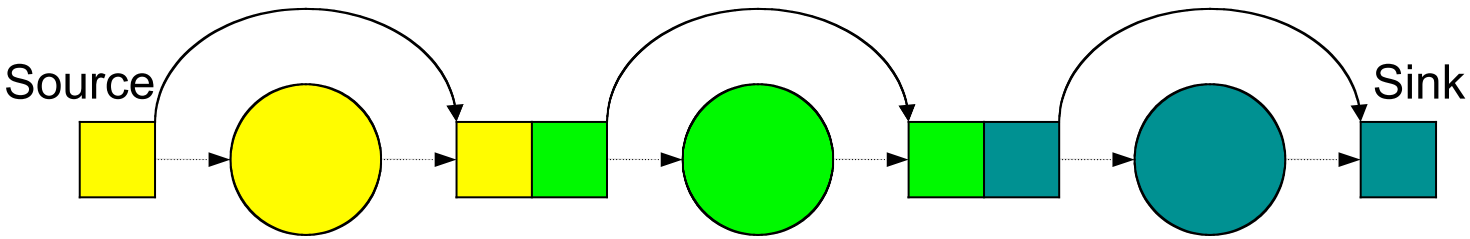
RBS – Network Synchronization

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

- **Suppose that:**
  - node 1 has detected an event at time $L_1(t)$
  - the sink is connected to a GPS receiver and has UTC timescale
  - node 1 wants to inform the sink about the event such that the sink receives a timestamp in UTC timescale
  - Broadcast domains are indicated by “circles”
- **Timestamp conversion approach:**
  - Idea: do not synchronize all nodes to UTC time, but convert timestamps as packet is forwarded from node 1 to the sink ➔ avoids global synch
  - Node 1 picks node 3 as forwarder – as they are both in the same broadcast domain, node 1 can convert the timestamp $L_1(t)$ into $L_3(t)$
  - Node 3 picks node 5 in the same way
  - Node 5 is member in two broadcast domains and knows also the conversion parameters for the next forwarder 9
  - And so on ...
  - Result: the sink receives a timestamp in UTC timescale!
  - Nodes 5, 8 and 9 are gateway nodes!

RBS – Network Synchronization

- **Forwarding options:**
  - Let each node pick its forwarder directly and perform conversion, the reference nodes act as mere pulse senders
  - Let each node transmit its packet with timestamp to reference node, which converts timestamp and picks forwarder
    - This way a broadcast domain is not required to be fully connected
  - In either case the clock of the reference nodes is unimportant



- **How to create broadcast domains?**
  - In large domains (large m) more packets have to be exchanged
  - In large domains fewer domain-changes have to be made end-to-end, which in turn reduces synchronization error
  - This is essentially a clustering problem, forwarding paths and gateways have to be identified by routing mechanisms

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

Overview

- The time synchronization problem
- Protocols based on sender/receiver synchronization
- Protocols based on receiver/receiver synchronization
- Summary

Summary

- **Time synchronization**
  - important for both WSN applications and protocols
  - Using hardware like GPS receivers is typically not an option, so extra protocols are needed
- **Post-facto synchronization**
  - allows time-synchronization on demand
  - otherwise clock drifts would require frequent re-synchronization
    - constant energy drain
- **Some of the presented protocols take significant advantage of WSN peculiarities like:**
  - small propagation delays
  - the ability to influence the node firmware to timestamp outgoing packets late, incoming packets early
- **More schemes exist....**

Thank you

(and thanks go also to Andreas Willig for providing slides)

CoNe Freiburg

University of Freiburg
Computer Networks and Telematics
Prof. Christian Schindelhauer

Wireless Sensor Networks
Christian Schindelhauer
schindel@informatik.uni-freiburg.de

14th Lecture
12.12.2006