IoT Lecture HS 2017
Group Exercises

UNIVERSITÄT BERN

Prof. Dr. Torsten Braun

Dr. Eryk Schiller

Jose Luis Carrera

University of Bern

Preliminaries I

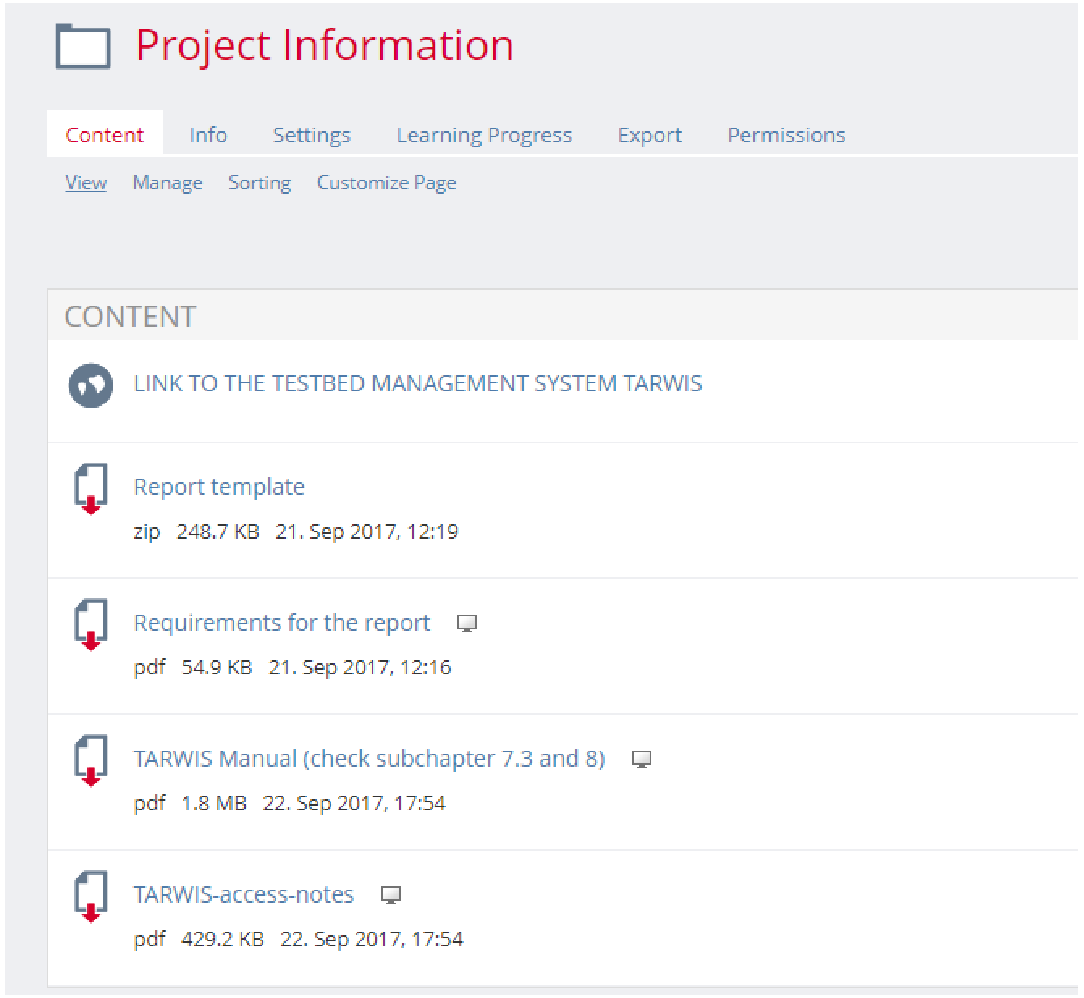
> Group of 4 students.

> Each group of students solves 1 group project assignment

> Final evaluation on the 40-nodes UniBE testbed

Tabletop prototype demo:	November 6
Presentation of evaluation results:	December 11

Preliminaries II



Project Information

Content Info Settings Learning Progress Export Permissions

View Manage Sorting Customize Page

CONTENT

LINK TO THE TESTBED MANAGEMENT SYSTEM TARWIS

Report template
zip 248.7 KB 21. Sep 2017, 12:19

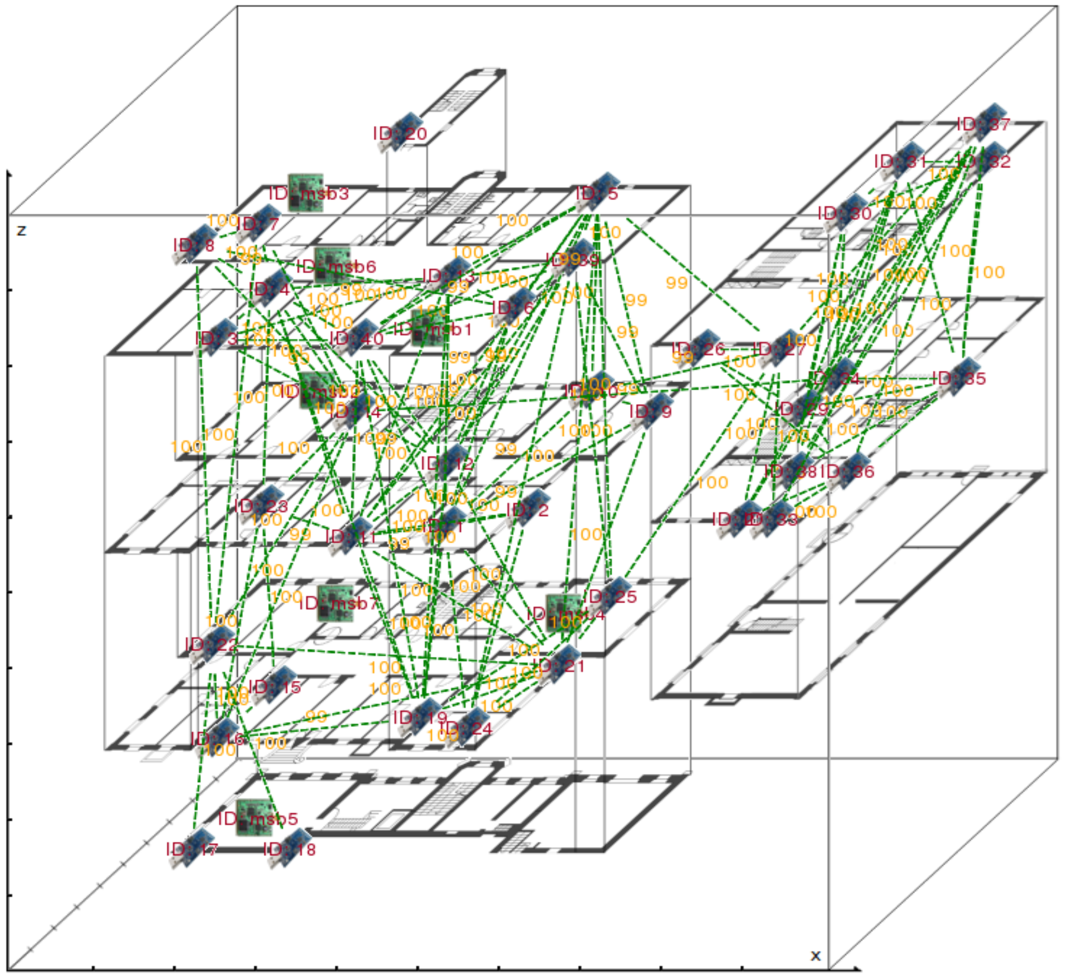
Requirements for the report
pdf 54.9 KB 21. Sep 2017, 12:16

TARWIS Manual (check subchapter 7.3 and 8)
pdf 1.8 MB 22. Sep 2017, 17:54

TARWIS-access-notes
pdf 429.2 KB 22. Sep 2017, 17:54

Indoor Testbed

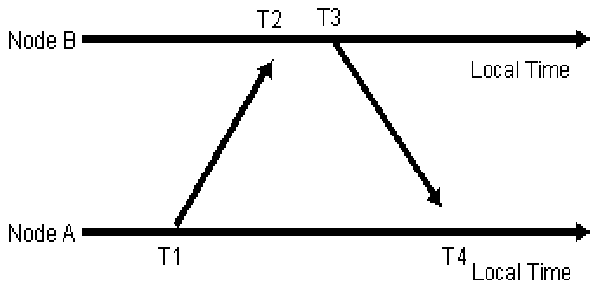
- 40 TelosB nodes distributed across the Institute building
- Web-Interface for reprogramming and output collection
- Develop your code by using the nodes we distributed to groups
- Use testbed only in the final stage, when you are confident that „your code works“
- Do not print more than one printf statement per second!
- Be fair! do not mono-polize the testbed!



1. Time Synchronization

Timing-Sync Protocol for Sensor Networks (TPSN)

> 1. Level Discovery Phase
(e.g. 2 x network-wide broadcast, 120s)



> 2. Synchronization Phase

> I) Prototype Implementation and Debugging with 2-3 Nodes

> II) Evaluation with TARWIS (using NullMAC, X-MAC)

[GKS] Ganeriwal, S., Kumar, R., Srivastava, M. (2003). "Timing-Sync Protocol for Sensor Networks.", The First ACM Conference on Embedded Networked Sensor Systems (SenSys)

1. Time Synchronization

Timing-Sync Protocol for Sensor Networks (TPSN)

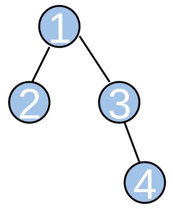
1. Level Discovery Phase

- Define one of your nodes as the root node. Let it periodically broadcast a packet with a) a unique identifier of the broadcast and b) a field defining the level in the tree

- When this packet arrives at another node, let the node check if it has already received this broadcast. If yes, discard the packet. If no, let the node re-broadcast the packet with the level increased by one.

- Each node, except for the root, shall select its “parent” node based on the level discovery phase. It shall select the node from which it has received a level discovery broadcast with the lowest level value. If there are more than one, let the node id decide conflicts (take the one with the lowest id).

- Test this mechanism with 4 of your distributed nodes, forming a tree with three levels



- Tip: Test the multi-hop capability by filtering out packets from nodes that are assumed not to be in each other's reach. E.g, if nodes 1, 3, 4 form a chain, discard packets received by 1 emitted by 4, and vice versa.

1. Time Synchronization

Timing-Sync Protocol for Sensor Networks (TPSN)

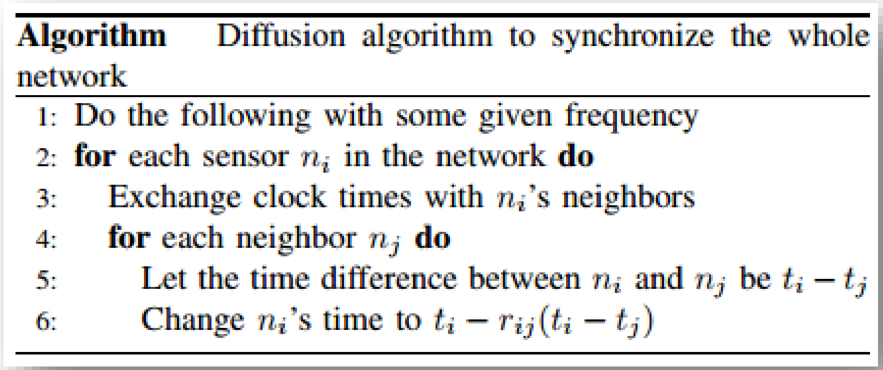
2. Time Synchronization Phase
- Define structs to implement the Synchronization phase. Re-visit the theory in the paper given on the previous slides and the slides of the lecture.
- Let each node in the network, except for the sink, periodically synchronize to its parent node (chosen in the level discovery phase). Use the unicast primitive for this task

3. Evaluation
- Assume that the timestamps that are generated by TARWIS are “synchronized”, i.e., that they represent a “global” time.
- Let your algorithm run for a certain time and calculate the “average” offset of your nodes times' versus the TARWIS timestamps.
- How much is the average difference between each node's offset?
→ this gives you a measure for the synchronization accuracy.
- How does this difference behave over time?
- Analyze the dependency from the variables (sync period, etc.)

- Re-compile your code with X-MAC and do the same calculation again. How does the synchronization accuracy behave when using X-MAC?

2. Time Synchronization II

Rate-based Synchronous Diffusion



Algorithm Diffusion algorithm to synchronize the whole network

```
1: Do the following with some given frequency
2: for each sensor $n_i$ in the network do
3:     Exchange clock times with $n_i$'s neighbors
4:     for each neighbor $n_j$ do
5:         Let the time difference between $n_i$ and $n_j$ be $t_i - t_j$
6:         Change $n_i$'s time to $t_i - r_{ij}(t_i - t_j)$
```

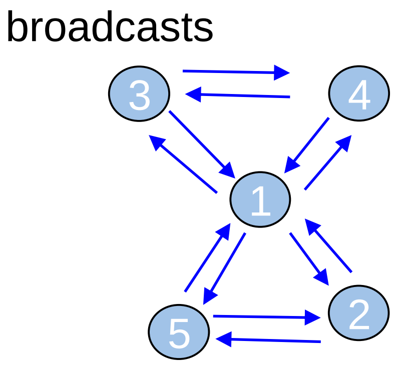
> I) Prototype Implementation and Debugging with 4 Nodes

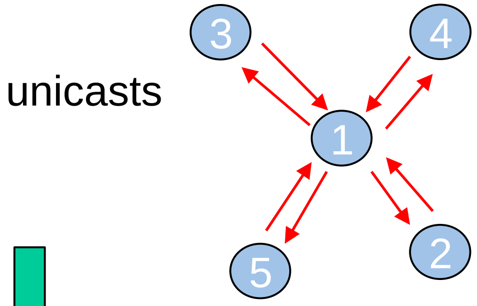
> II) Evaluation with TARWIS (using NullMAC, X-MAC)

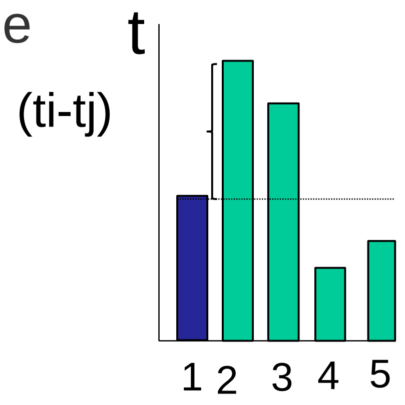
[GKS] Li, Rus: Global Clock Synchronization in Sensor Networks, IEEE Transactions on Computers, February 2006, pp. 214

2. Time Synchronization II
Rate-based Synchronous Diffusion

- Broadcast ID and Sequence Number to get to know neighbors
  - build up neighbor table
- Each sensor iterates over neighbors
  - determines offset via round-trip sync
  - saves offset to table
- Converge time to network-wide time
  - Set ti = ti – r * (ti-tj)
  - 0 < r < 1







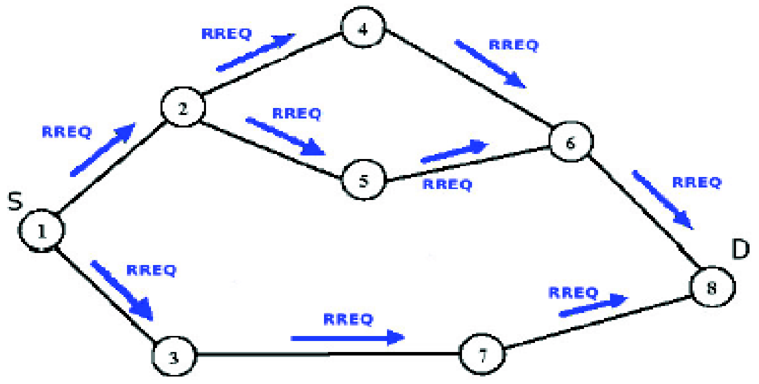
2. Time Synchronization II
Rate-based Synchronous Diffusion

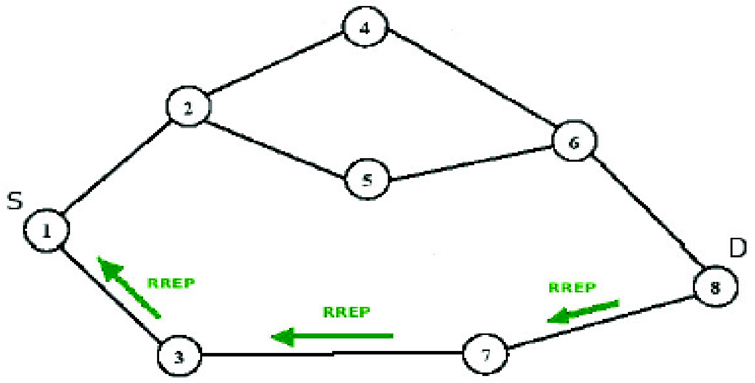
1. Neighborhood Discovery Phase
-Let each node periodically broadcast a packet with its id.
-Let each node set up a table of neighbors with the time offsets between the own time and the neighbors times (see slide before)

2. Convergence Phase
-After an initial setup phase, let each node periodically go through the table and adapt the own time according to line 6 in the algorithm noted in the slide before

3. Evaluation
- Assume that the timestamps that are generated by TARWIS are “synchronized”,
i.e., that they represent a “global” time.
- Let your algorithm run for and calculate the “average” offset of your nodes times' versus the TARWIS timestamps.
- How much is the average difference between this offset and each node's offset?
→ this gives you a measure for the synchronization accuracy.
- How does this difference behave over time?
- Analyze the dependency from the variables (sync period, etc.)
- Re-compile your code with X-MAC and do the same calculation again. How
Does the synchronization accuracy behave when using X-MAC?

3. Small AODV Routing





> Reactive Routing Protocol

> Implement network-wide Route Request (RREQ) & Route Reply (RREP) phase

> I) Prototype Implementation and Debugging with 2-3 Nodes

> II) Evaluation with TARWIS (using NullMAC, X-MAC)

AODV http://en.wikipedia.org/wiki/Ad_hoc_On-Demand_Distance_Vector_Routing

3. Small AODV Routing

1. Route Request (RREQ)
- Define one of your nodes as the source S, which wants to find a Route to the node D in order to send a data packet. Let this route broadcast a packet across the network to find S.

- When the RREQ broadcast packet arrives at another node, let the node check if it has already received this broadcast. If yes, discard the packet. If no, let the node re-broadcast the packet. Use the mechanism used in AODV that can uniquely identify a packet by the source sequence number and the source id.

- When the RREQ is distributed across the network, let each node save the path information in the AODV table. E.g., if node A receives an RREQ via node B, with the RREQ having been originated by node S, let node A save this path and the number of hops required to reach S.

- When the RREQ finally reaches the targeted destination node D, let node D, let node D first wait some seconds to get the RREQ from multiple nodes. Update the AODV table when multiple paths are advertised. Choose the shortest path.

3. Small AODV Routing

2. Route Reply (RREP)
- Let node D send a packet towards S using the reliable unicast. The packet shall be forwarded by all the nodes in the middle based on the AODV routing tables.

3. Data Exchange
- If the packet reaches S, let node S send the data packet towards D using the reliable unicast

4. Evaluation

- Let your routing protocol run on TARWIS and choose sources and destinations.
Each couple of seconds, let one source search a path to the destination.

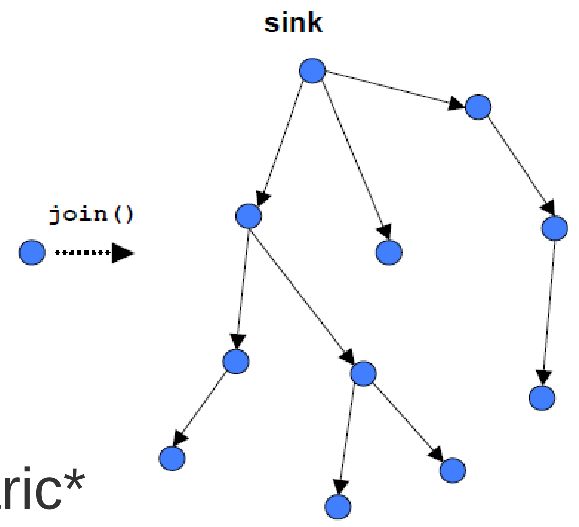
- Using the TARWIS timestamps, measure how long on average it takes for
a) the RREQ to find the destination
b) the time from the first RREQ to the incoming of the first RREP
(path establishment time)

- Switch your MAC layer to X-MAC and do the same evaluation again. Compare the results and describe them in the report. Explain the differences.

4. Tree-based Routing

> Proactive Routing Protocol

> Periodic Network-wide Broadcast to establish Tree Structure



- Tasks: a) use hopcount as metric
  b) use hopcount & RSSI as metric*

> I) Prototype Implementation and debugging with 4 Nodes

> II) Evaluation with TARWIS (using NullMAC, X-MAC)
- how is latency, packet delivery rate with RSSI as metric
- how is latency, packet delivery rate with hopcount as metric

* nodes choose the parent with the shortest hopcount and the strongest signal

4. Tree-based Routing

1. Tree establishment mechanism:

- Define one of your nodes as the root (sink) node. Let it periodically broadcast a packet with a) a unique identifier of the broadcast and b) a field defining the depth level in the tree. Periodically means every 2 minutes.

- When this packet arrives at another node, let the node check if it has already received this broadcast. If yes, discard the packet. If no, let the node re-broadcast the packet with the level increased by one.

- Each node, except for the root, shall select its “parent” node based on the level discovery phase. For task a) each node selects the parent node by determining the
node from which it has received a broadcast with the lowest level value. If there are more than one, let the node id decide conflicts (take the one with the lowest id).

- For task b) each node selects the parent node by determining the node from which it has received a broadcast with the best signal quality level – find out how to determine signal quality using the RSSI value provided by Contiki with each received packet. If there are more than one node offering the same link quality, let the node id decide conflicts (take the one with the lowest id).

4. Tree-based Routing

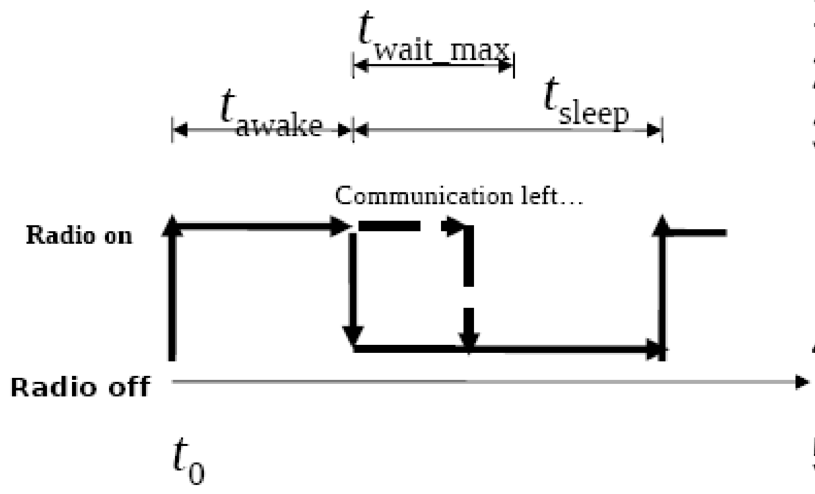
2. Periodic messages to the sink:
- When a node has chosen a parent, it shall periodically (e.g., every 5 minutes) send a packet towards the sink with its temperature value. The packet shall be forwarded by the intermediate nodes in the tree until the sink is reached.

- Use the unicast primitive for the data packets. Use a counter to count the sent data packets, such that the packet-delivery rate at the sink can be calculated

3. Evaluation
- Let your routing protocol run on TARWIS with a) the hop-count metric and b) the rssi-metric. Set a maximum number of data packets to be sent by each node after discovery of a parent (e.g., 20).

Evaluate
i) what is average number of hops that data packets coming in a the sink have to travel? With the
 a) routing metric (hopcount) and
 b) routing metric (rssi).
ii) what is the packet delivery rate with the
 a) hopcount routing metric
 b) routing metric.
Iii) same evaluation with X-MAC on the MAC layer.

5. S-MAC Protocol



1. Turn radio on.
2. Wait until *t* = *t_0* + *t_awake*.
3. If communication has not completed, wait until it has completed or *t* = *t_0* + *t_awake* + *t_wait_max*.
4. Turn the radio off. Wait until *t* = *t_0* + *t_awake* + *t_sleep*.
5. Repeat from step 1.

> I) Prototype Implementation and debugging with 4 Nodes

> II) Evaluation with TARWIS

- how is latency over 1, 2, 3, 4 and 5 hops?
- how is the packet delivery rate over 1, 2, 3, 4 and 5 hops?

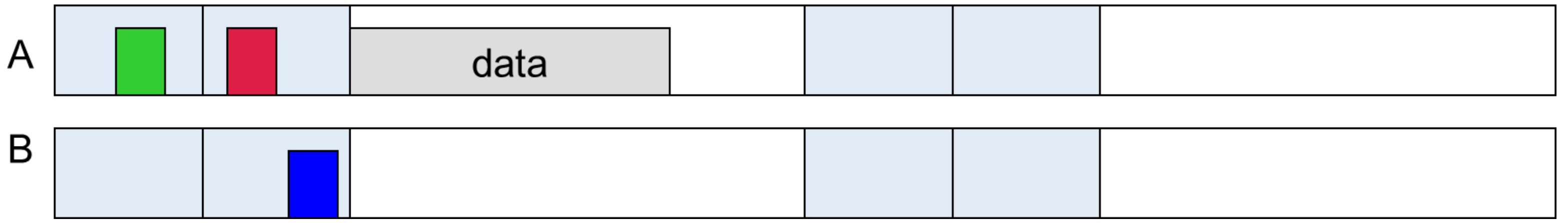
5. S-MAC Protocol

> Each node has alternating sleep and listen states.

- Sleep: Node turns off its radio and sets a wakeup timer.
- Listen: Node listens whether other nodes want to talk with it.

> Coordinated Sleeping and Synchronization

- Each node selects its own listen/sleep schedule.
- Neighbor nodes synchronize by periodic SYNC broadcasts describing the nodes' schedule
- Collision avoidance similar to IEEE 802.11: RTS / CTS



Wei Ye, John Heideman, D. Estrin: Medium Access Control With Coordinated Adaptive Sleeping for Wireless Sensor Networks, IEEE/ACM Transactions on Networking, Vol. 12, No. 3, June 2004, pp. 495

5. S-MAC Protocol

1. Synchronization Mechanism:

- Choose an interval of 1s for the entire sleep wake interval, and wake time of 200ms (20%). The first 30ms shall be reserved for the SYNC messages.

- Each node shall randomly send out SYNC messages every 30-40 intervals, in order to synchronize the network to a common schedule. Each node updates its tact after reception of the first SYNC, and chooses a new value for the next interval when it would intend to send a SYNC.

- Only synchronize the nodes to wake up and sleep in each 1s interval. Hence, do no synchronize the “global time” -> do only synchronize the milliseconds of the clock.

2. RTS / CTS exchange

- Implement the RTS/CTS message exchange in order to reserve the current time interval.

- A node receiving an RTS shall go to sleep if it is not the targeted receiver.

5. S-MAC Protocol

3. Static Route

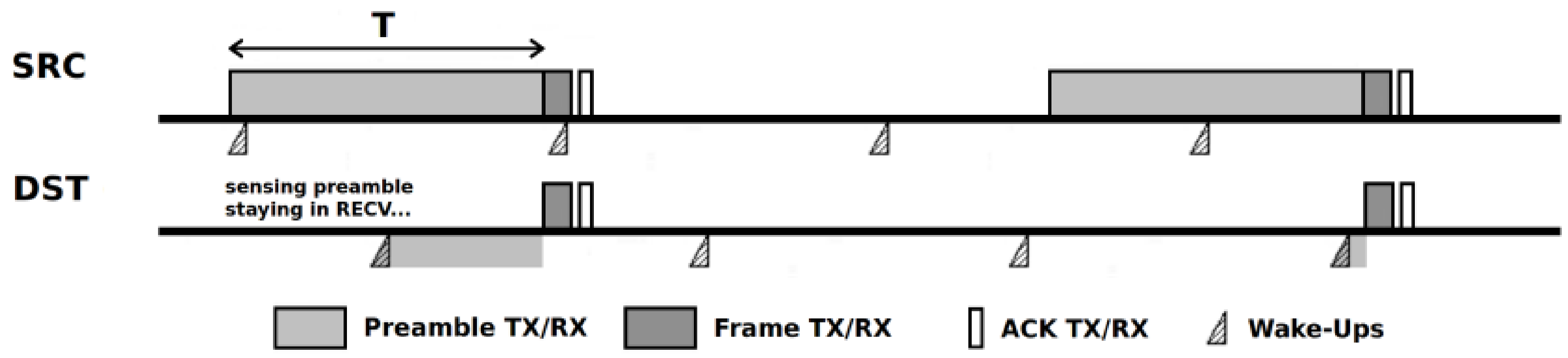
- Choose routes in the testbed in order to test out the behavior of the MAC Protocol in the distributed testbed.

4. Evaluation

- Evaluate packet delivery rate and latency across 1-5 hops
- Evaluate how the packet delivery rate and latency behave for different rates of traffic
  - 1 packet each 10 seconds
  - 1 packet each 5 seconds
  - 1 packet each 3 seconds
  - 1 packet each second
- Compare the results against NullMAC

6. B-MAC Protocol

UNIVERSITÄT BERN



> Clear channel assignment and packet back-offs for channel arbitration

- Transmission of long preambles
- Low power listening: Periodic wakeups for channel sampling based on noise estimation

> I) Prototype Implementation and debugging with 4 Nodes

> II) Evaluation with TARWIS

- how is latency over 1, 2, 3, 4 and 5 hops?
- how is the packet delivery rate over 1, 2, 3, 4 and 5 hops?

6. B-MAC Protocol

1. Sleep Wake Pattern

- Choose an interval of 1s for the entire sleep wake interval T, and wake time of 20-30ms (2-3%)
- Implement the functionality as protothreads. You can use NullMAC with CSMA on the Contiki MAC Layer in order to implement the Backoff mechanism

2. Transmission of Preambles

- Continuously send out empty frames with the targeted receiver for T=1s in order to catch the destination in its wake-up.

3. Channel Polling

- Check the implementation of the polling mechanisms in X-MAC and ContikiMAC to implement channel Polling

6. B-MAC Protocol

3. Static Route

- Choose routes in the testbed in order to test out the behavior of the MAC Protocol in the distributed testbed.

4. Evaluation

- Evaluate packet delivery rate and latency across 1-5 hops
- Evaluate how the packet delivery rate and latency behave for different rates of traffic
    - 1 packet each 10 seconds
    - 1 packet each 5 seconds
    - 1 packet each 3 seconds
    - 1 packet each second
- Compare the results against NullMAC