Rate-based Synchronous Diffusion

Internet of Things

Balz Aschwanden, David Boesiger, Jovana Micic, Raoul Norman Grossenbacher

University of Bern
{balz.aschwanden, david.boesiger, jovana.micic,
raoul.grossenbacher}@students.unibe.ch

1 Protocol Introduction

Very often time synhronization of all sensors is requred in Wireless Sensor Network (WSN). Since each node has its own clock, it is needed to synchronize clocks in order to support synhronized sleep and duty cycles among nodes.

Rate-Based Diffusion Protocol (RDP) aims to synchronize the nodes in the network to the average value of the clocks in the network. Rate-Based Diffusion Protocol has two main phases:

1. Neighborhood Discovery Phase

   In this phase, each node has to periodically broadcast a packet with its ID and sequence number to get to know neighbors. All recognized neighbords are saved in neighbor table. Additionaly, with each neighbor we have to save the time offset between the node's time and the neighbors times. Broadcast is determined by time the node waits after starting broadcasting. This parameter value will be discused later in further sections.
2. Convergence Phase

   In covergence phase, each node periodically go throught neighbors table and update own time using following formula:

   $$t_i = t_i - r * (t_i - t_j)$$

   Basic idea is to adapt time of the node to the neighbours node time using some r-value. R-value needs to be 0<r<1. Results of choosing different r-values will be discused in further sections. In this phase, unicast messages are used to determine the offset between the clocks.

Alghoritm 1 is showing the pseudo code for Rate-based Diffusion Protocol.

Algorithm 1 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 $t_i$-$r_ij$($t_i$-$t_j$)
7:    end for
8: end for
```

2 Methods

In the following part we will show implemented code for recieving and sending unicast messages.

3 Experimental setup

There were two phases of experiment. In the first phase we tested our program using Telos nodes and in the second phase we uploaded our code to TARWIS platform. We tried different values for*r-value* and r*unicast interval.* In addition, we run the code with different MAC protocols. In the first version of the program we used default NullMAC protocol and in the second we used X-MAC protocol.

By protocol alghoritm, *r-value* needs to be value from range of zero to one. We chose to test our code for five different r-values: 0.1, 0.25, 0.5, 0.75 and 0.9.

For *unicast interval* value we chose values from 1 to 5 seconds.

4 Results and Analysis

5 Conclusions

References

[RE1] Author: Article/Book: Other info: (date) page numbers.