Technical University Berlin

Telecommunication Networks Group

Dynamic Adaptive Wake-up Scheduling Scheme for Supporting Periodic Traffic in WSNs

Osama Khader, Andreas Willig

{khader, awillig}@tu-berlin.de

Berlin, November 2008

TKN Technical Report TKN-08-012

TKN Technical Reports Series

Editor: Prof. Dr.-Ing. Adam Wolisz

Abstract

In many applications in wireless sensor network source nodes generate and send periodic traffic to the sink through a number of intermediate nodes (or *forwarders*). In such network forwarders have forwarding duties but want at the same time to spend as much time as possible in an energy-saving deep-sleep mode. In this work we explore the periodicity of traffic so that the forwarders wake up at "just the right time" to catch an incoming packet, forward it and go quickly back to sleep mode. A key assumption for this work is that the forwarders do not know the traffic period beforehand, but they have to *estimate* the period and maintain their estimate over time. A key difficulty is that the period estimation and the scheduling of wakeup times will have to deal with jitter in the packet inter-arrival times. If a packet arrives before the forwarder wakes up, it is lost. This opens up a tradeoff between loss rates and the sleeping activities of the forwarder: when the forwarder wakes up "early", the packet loss rate will be low but the forwarder spends more energy, and vice versa. The main contributions of this report are the following ones: (i) we design and implement local estimators for traffic period and jitter; (ii) we design and implement a scheduling scheme by which a forwarder locally decides when to sleep and when to wake up; and (iii) we adjoin mechanisms to this scheme that allow to update the period and jitter estimates and to react to changes in the locally observed loss rate. We use measurements and simulation experiments to evaluate our proposed algorithms.

Contents

Chapter 1

Introduction

In many wireless sensor network applications source nodes generate and send periodic traffic to a sink node [15], [16], [6] through a number of intermediate nodes. Nodes have usually only a limited amount of energy and thus achieving a long node lifetime is a major research concern. The sensed data should be transported reliably and in a timely fashion to the sink. At the same time the operation of the whole network and of individual nodes should be energy-efficient. One of the key approaches to achieve this is to let the forwarding nodes switch to an energy-conserving sleep state whenever possible. In this sleep state parts of the node hardware, especially the wireless transceiver, are switched off. This disables the communication ability of a node but leads to significant energy savings, since for most of the currently available sensor node platforms the wireless transceiver is the dominant source of energy consumption. The fraction of time where the node is awake is called its *duty cycle*, and from the perspective of energy-efficiency this duty cycle should be kept small. For a source node generating the periodic data there is no problem: the node wakes up, samples its sensor, transmits a packet and returns to sleep mode. However, in a multi-hop network other nodes are needed to forward the packet to a sink node. To be most energy-efficient, a forwarder should wakeup just before a periodic packet arrives, do the necessary forwarding work and enter sleep mode again. Even if the source node sends its date periodically, it may not appear to be periodic at the sink node due to time-varying cross traffic or transmission error handling in various network protocols. Furthermore, the amount of jitter (for example expressed as the average deviation from the period) is a function of the number of hops a packet traverses.

The goal of this work is to enable forwarder nodes to acquire knowledge of the traffic period and its jitter in a distributed way. We want to use this knowledge to let a forwarder node wake up and sleep "just at the right time". The "right time" to wake up is at the nominal arrival time of the next periodic packet minus some safety margin to avoid packet loss because of waking up too late. On the other hand, the "right time" to go back to sleep mode is at the nominal arrival time of the next periodic packet plus some safety margin to avoid packet loss because of sleeping too early. The size of this safety margin in relation to the duty cycle has a direct influence on the forwarders energy consumption as well as its packet losses: the larger it is the more energy it spends and less packet losses incurred and vice versa. The safety margin in general depends on the amount of jitter seen by a node and the percentage of sleeping-induced losses that can be tolerated.

Thus, instead of using explicit signaling of the period and furthermore maintaining a common time base through a time synchronization protocol to schedule the wakeup times, we adopt an approach in which the period and its jitter estimated and wakeup behavior is adapted to continuously controlled in terms of observed packet losses. A key goal of the estimation is not only to estimate the traffic period, but also to locally estimate certain quantiles of the jitter distribution, because only if these quantiles are known it is possible to choose the above mentioned safety margin so that not more than a prescribed percentage of packets is lost because of sleeping.

The efficiency of our approach, measured in terms of sleeping times and sleeping-induced loss rates, are evaluated on data obtained from real measurements using trace-driven simulation. To the best of our knowledge, the adaptation of sleep periods to the (estimated) period and jitter of periodic traffic subject to prescribed packet loss requirements has not been considered so far in the literature.

The rest of this report is structured as follows: the following Chapter 2 details the setup under consideration. In Chapter 3 we present the general framework of wake up times scheduling approach. Chapter 4 describes the experimental jitter distribution measurements. In Chapter 5 we present an estimation algorithms. In Chapter 6 we evaluate the proposed algorithms and discuss the results. Chapter 7 presents the related work and finally, Chapter 8 concludes the report with some future work.

Chapter 2

Setup under consideration

In this chapter we describe our experimental setup and performance metrics. 2.1.

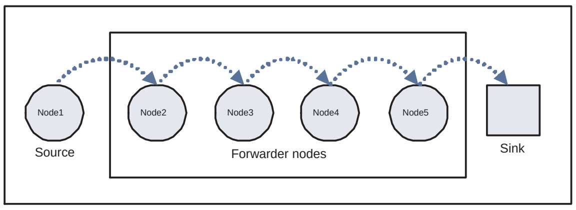


Figure 2.1: Experimental setup

2.1 Network setup

We consider our system model as a chain of one source, four forwarders and a sink node placed in our lab. The source node (node1) generates ad sends periodic traffic to the sink node through four forwarder nodes as shown in figure 2.1. The sink node is a laptop used to record the packets for offline evaluation. Packets are unicast along the chain. The sensor nodes are located close to each other to prevent packet losses from channel errors, allowing us to focus on sleeping-induced losses. There is sufficient space in the packet so that each node can append its local reception timestamp to the packet. Our evaluations use these local timestamps.

In our experiments we used Tmote sky nodes and TinyOs version 2.0 operating system [7], [5]. We have set the radio channel of IEEE 802.15.4 compliant CC2420 transceiver [2] to a channel that from preliminary measurements has been found to be relatively interference-free.

2.2 Protocols

We use the default protocol stack of Tinyos 2.0. The medium access control protocol is a CSMA/CA based protocol, so every node waits a random short amount of time before each transition and it goes into back off and pick another random time if the channel is still busy. The CC2420 radio stack can select one of three backoff periods: initialBackoff, congestionBackoff, and lplBackoff [12]: initialBackoff is the shortest backoff period, requests on the first attempt to transmit a packet. CongestionBackoff is a longer backoff period used when the channel is found to be busy and finally the LplBackoff is used for packet being delivered with low power listening. Because low power listening requires the channel to be modulated as continuously as possible while avoiding interference with other transmitters, the low power listening backoff period is intentionally short [12],[11]. For the purpose of our experiment we disabled the MAC layer acknowledgement and no retransmission was performed during the experiment.

The nodes did not sleep during the experiment but unconditionally try to capture all packets. The impact of the sleep/wakeup scheduling policy was evaluated offline, based on the timestamps obtained from each node.

2.3 Performance metric

The prime objective of this work is the relationship between the energy consumption of a node and the packet loss rate that results from either waking up too late or going back to sleep too early.

We have decided to use the duty cycle of a forwarder node as a measure for its energy consumption. In our opinion this is a reasonable approximation, since for the CC2420 transceiver that we have used for this work the power consumptions in the transmit, receive and idle (waiting to receive something) states are very similar [19], [14], [3].

The second important performance measure under consideration is the loss rate observed by an individual node when operated under a sleep/wakeup schedule.

Chapter 3

Wake-up times scheduling approach

In this chapter we explain our approach for controlling the wakeup and sleeping times of a forwarder node. Here we keep the discussion somewhat more general than in the later chapters of this report. The main difference is that here we do not make any particular assumptions on the underlying jitter distribution.

3.1 Estimators

Assume that there are N forwarder nodes. The forwarder node i, where $1 \leq i \leq N$ receives packets at times $(t_i n)_{n \geq 1} = (t_{i1}, t_{i2}, t_{i3}, \ldots)$. These packets carry link-local sequence numbers $(s_i n)_= (s_{i1}, s_{i2}, s_{i3}, \ldots)$ which for simplicity we assume monotonically increasing – to ease presentation we avoid here the issue of overruns, but it has been considered in our experimental implementation. The first building block of our scheduling approach is the design of appropriate estimators. The following estimators are only updated after each packet arrival (ordered pair), if there is a gap then we just drop it form the calculation. (i.e. when the packet with sequence number s_{in} has arrived at time t_{in}, the estimators uses the values $t_{i1}, t_{i2}, \ldots, t_{in}$ and $s_{i1}, s_{i2}, \ldots, s_{in}$):

- A period estimator $\widehat{p_i}(n)$ returns an estimate of the period. The general form of the period estimator is:
$$\widehat{p_i}(n) = P(t_{i1}, \ldots, t_{in}, s_{i1}, \ldots, s_{in})$$
- A quantile estimator returns the estimate α-quantile of the jitter distribution. The general form of the quantile estimator is
$$\widehat{q_i}(n; \alpha) = Q\left((t_{in_2} - t_{in_1}) - \widehat{p_i}(n), (t_{in_3} - t_{in_2}) - \widehat{p_i}(n), \ldots (t_{in_k} - t_{in_{k-1}}) - \widehat{p}(n); \alpha\right)$$
where the subsequence $(t_{n_k})_{k \geq 1}$ consists of those timestamps for which $t_{n_{2k}}$ and $t_{n_{2k-1}}$ belong to successive packets (i.e. for which $s_{n_{2k}} - s_{n_{2k-1}} = 1$ holds).

  It depends on the assumptions on the jitter distribution how many quantiles must be estimated simultaneously:

  - If the jitter distribution is assumed to be symmetric, only one quantile must be estimated. Given a prescribed allowed loss rate of $L_{\max}$, the quantile estimation would be applied with $\alpha = L_{\max}/2$ to account for the fact that losses can be incurred either because the forwarder awakes too late or goes back to sleep too early.
  - In the more general case of a non-symmetric distribution, two different quantiles would have to be estimated: $\alpha_1 = L_{\max}/2$ for the lower part, and $\alpha_2 = 1-L_{\max}/2$ for the upper part of the jitter distribution.

- A loss-rate estimator computes the local loss rate between the forwarder and its successor. It mainly operates on the sequence numbers, but since the sequence number space is in general finite and ambiguities might occur, the packet arrival timestamps are also taken into account, i.e. the general form of the loss-rate estimator is

$$\widehat{l}(n) = L(t_1, \ldots, t_n, s_1, \ldots, s_n)$$

3.2 Node states

The second major building block is the introduction of two separate node states: the *acquisition state* and the *operational state.*

- In the *acquisition state* the forwarder node does not sleep but unconditionally tries to capture all packets in order to obtain reliable first estimates of the period, jitter variance and packet loss rate. This state is entered after the node has been switched on or too many losses have been incurred during the operational state. The latter can occur for example when the cross-traffic situation and therefore the actual jitter variance changes. All existing historical data is dropped upon entering the acquisition state. The end of the acquisition state is determined by a *stopping rule*, which in general can take the achieved accuracy of the period- and variance-estimator into account or can simply stop after recording a prescribed number of packets.

- In the *operational state* the sleep-/wakeup scheduling is applied. The forwarder alternates between sleep phases and activity phases.

  - The forwarder maintains three predicted times:
    * $t_w(n)$ refers to the wakeup time of the n-th activity phase (i.e. it denotes the start of the activity phase)
    * $t_s(n)$ refers to the sleep time (denoting the maximum end time of the n-th activity phase), and
    * $t_a(n)$ is the nominal packet arrival time for the n-th activity phase.

  For the n-th activity phase, the forwarder schedules wakeup for the time $t_w(n)$. The forwarder remains awake until either a packet is received and forwarded, or until time $t_s(n)$ at maximum. At the end of the activity phase the forwarder updates its estimates of the period, jitter, and loss rate. Based on this, the times for the $n+1$-st activity period are calculated as follows (assuming a symmetric jitter distribution):

$$\begin{aligned} t_a(n+1) &= t_a(n) + \widehat{p}(n) \\ t_w(n+1) &= t_a(n+1) - \widehat{q}(n) \\ t_s(n+1) &= t_a(n+1) + \widehat{q}(n) \end{aligned}$$

As the second major action at the end of the activity phase, the forwarder decides about a possible state transition back into the acquisition state. We refer to the rule used for this as the *transition rule.*

Please note that in this approach the forwarder continuously updates its period and jitter variance estimates in the operational state. We later on show results that confirm the necessity of these continuous updates. For later reference, we refer to the policy with updates as the *adaptive policy* and to the policy without updates as the *non-adaptive policy.*

Chapter 4

Experimental jitter distribution measurements

In this chapter we study and analyse the jitter distribution to get clear picture of its characteristics.

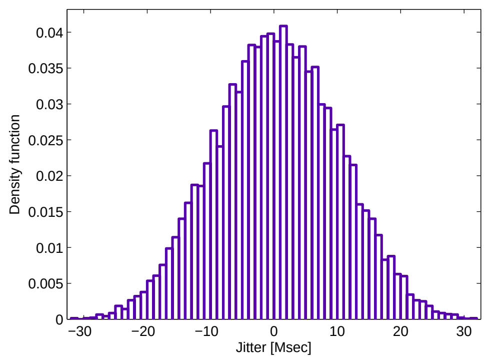
4.1 Experiment setup

The experiment set-up, has been described in detail in chapter 2. The experiment was conducted to measure the jitter in a linear multi-hop network.

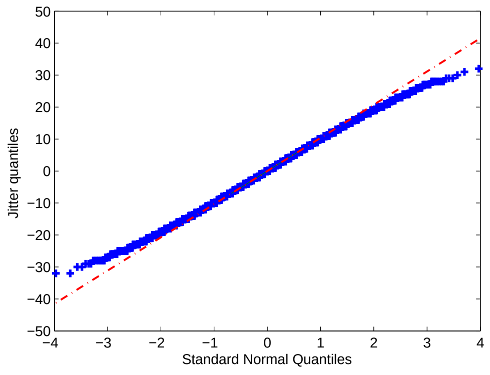
4.2 Discussion and result for jitter measurements

We conducted several experimental runs, measuring the jitter varying for example the number of hops in the multi-hop network and the packet generation period. As a matter of fact, in general the jitter distribution depends on a multitude of factors: the MAC and link-layer protocol, the local load situation, and the position of a particular forwarder in the forwarder chain. We do not investigate this relationship in general, but look at one particular measurement setup and set of protocols.

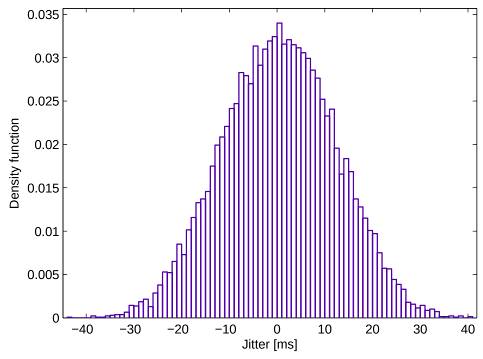
For these specific choices we find that the jitter distribution starrting from the second hop as shown in figures 4.1(a), 4.1(c), and 4.1(e) can be well modeled by a normal distribution as shown in the quantile-quantile plots in figures 4.1(b), 4.1(d), and 4.1(f).



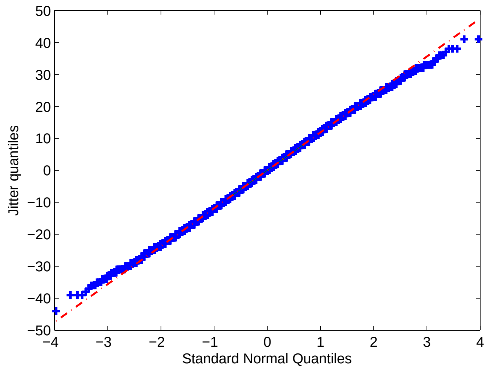
(a) Jitter histogram for 2nd hop



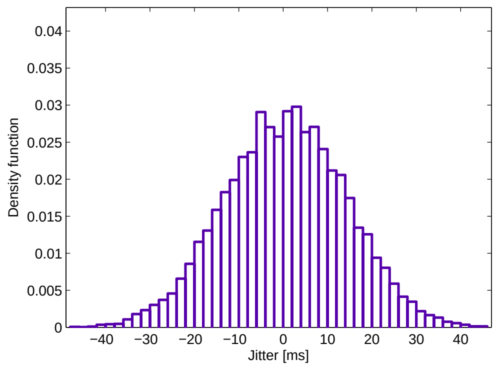
(b) QQ-plot for 2nd hop



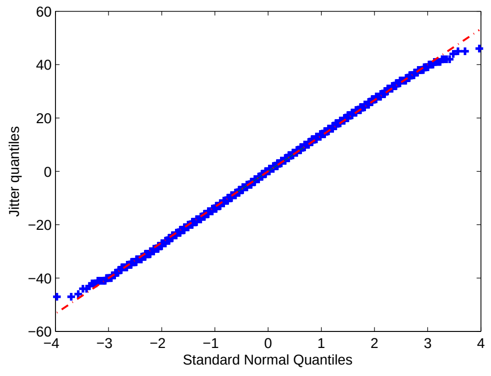
(c) Jitter histogram for 3rd hop



(d) QQ-plot for 3rd hop



(e) Jitter histogram for 4th hop



(f) QQ-plot 4th for hop

Figure 4.1: Jitter histogram and qq-plot based on 14000 samples

In addition, figure 4.2 summarizes the jitter histogram and it is probability density function for hop 2, hop 3 and hop 4 with their parameter values.

The results obtained from the above experiments are really motivating and it might

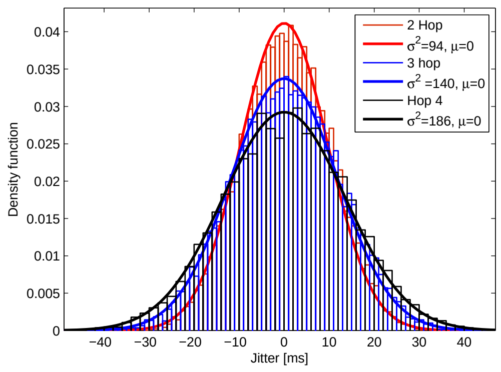


Figure 4.2: A Jitter histogram for 2,3 and 4 hops based on 14000 samples

be exploited to model and propose some solutions for several problems in wireless sensor networks. Moreover, this result allows to reduce the problem of quantile estimation (which is much harder and more memory-intensive than the estimation of averages [9, Sec. 9.5]) to the problem of estimating the variance of a normal distribution.

Chapter 5

Estimation

Recall from Section 4.2 that the random variable of the jitter is modeled as normal distribution and the parameters model of this distribution are determined by the mean and the variance. Thereby, we can exploit these interesting results and design an estimator for the parameters model. The following two estimators in which the current estimate depends on the previous estimate and the current measurement are called recursive estimator.

5.1 Mean estimator

Given k measurements of packet inter-arriving time X_i the sample mean is

$$\widehat{X}_k = \frac{1}{k}\sum_{i=1}^{k} X_i \tag{5.1}$$

Suppose that $\widehat{X}_k$ has been computed based on measurements X_i for i=1,....,k. Now one more measurement X_{k+1} is made. The new sample mean is computed as :

$$\widehat{X}_{k+1} = \frac{1}{k+1}\sum_{i=1}^{k+1} X_i \tag{5.2}$$

$\widehat{X}_{k+1}$ can be computed in terms of $\widehat{X}_k$ and X_{k+1} by proceeding as follows.

$$\begin{aligned}\widehat{X}_{k+1} &= \frac{k}{k+1}\left(\frac{1}{k}\sum_{i=1}^{k} X_i\right) + \frac{1}{k+1}X_{k+1} \\ &= \frac{k}{k+1}\widehat{X}_k + \frac{1}{k+1}X_{k+1}\end{aligned}$$

So that the estimator for the mean is:

$$\widehat{X}_{k+1} = \widehat{X}_k + \frac{1}{k+1}\left(X_{k+1} - \widehat{X}_k\right) \tag{5.3}$$

The random variable (packet inter-arrival time) is denoted by $x(k)$ and its estimated value of the mean is denoted by $\widehat{x}(k)$.

5.2 Variance estimator

A recursive method can also be found to computer an estimator for the variance:

$$\sigma_{k+1}^2 = \sigma_k^2 + \frac{1}{k+1}\left[\frac{k}{k+1}(X_{k+1} - \widehat{X}_k)^2 - \sigma_k^2\right] \tag{5.4}$$

Chapter 6

Evaluation and result

We performed several experimental runs to evaluate our proposed schemes. The same experiment configuration and setup as elucidated in section 2 is utilized. We use the following parameters to evaluate the proposed schemes:

1. The first two parameters are mean and variance. These two parameters are combined to determine an appropriate sleep and wakeup time windows depending on a given performance requirements.
2. Stopping rule, which decides whether to continue or stop the acquisition state based on a prescribed number of packets.
3. Maximum allowable packets loss for each individual node, which tradeoffs between energy usage and packet loss rate.

6.1 Simulation

In order to evaluate our approach, we conduct a trace driven simulation study based on real data collected from sensor nodes measurements. The evaluation consists of using real traces as an input to the simulation. These traces were taken from each forwarder node and contain the packets arrival times as well as the sequence number of the packets. Our approaches and algorithms are implemented and evaluated through Matlab version 7.1.

Figure 6.1 illustrates a simplified simulation model for non-adaptive and adaptive scheduling. For both, the acquisition state starts by initializing the mean and the variance to zeros. The initialization of model parameters are applied to each hop independently and each hop estimates the values of the parameters model. The end of this state is determined by a stopping rule (must be explicitly stated) as described in section 3.2. After the acquisition state is converged and before entering the operational state a value of allowable loss rate should be specified based on the application performance requirements. In the operational state the sleep and wakeup times scheduling is applied by selecting an appropriate windows for sleep and wakeup times as discussed in section 3.2. During this phase each node locally monitors its packet loss and reacts based on the specified policy. In the first scenario non-adaptive scheduling as shown in figure 6.1 if the policy equals ”A” then, the estimator algorithms

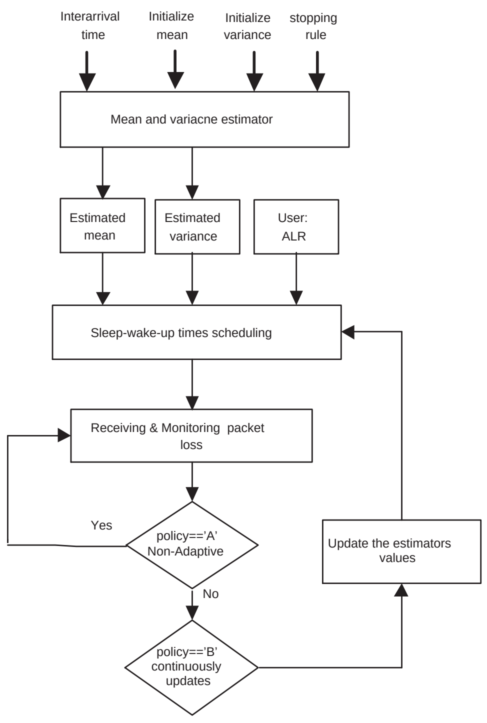

Interarrival time
Initialize mean
Initialize variance
stopping rule
Mean and variacne estimator
Estimated mean
Estimated variance
User: ALR
Sleep-wake-up times scheduling
Receiving & Monitoring packet loss
Yes
policy=='A' Non-Adaptive
No
Update the estimators values
policy=='B' continuously updates

Figure 6.1: Simulation model

run only once and no adaptation of the parameters model are performed, regardless of the packet loss rate. This policy provides a basis for evaluating the advantage of the other policy that does adapt. In the other hand if the policy equals "B" as shown in the same figure 6.1 then, the second scenario is applied, in which each forwarder continuously updates the estimate values and adapts its sleep and wakeup windows accordingly, without reentering the acquisition state again.

6.2 Results

In this section we present the simulation results of the proposed algorithms. We present our simulation result into the following scenarios: scheduling based on real parameters values, Non-adaptive scheduling, and adaptive scheduling.

6.2.1 Scheduling based on real parameter values

We start by examining the effect of using the real values of model parameters. The observed sample size of 9000 packets for each hop were taken independently from a real measurement with expected value of 1024ms for each hop and variance values of 31ms, 46ms, 62ms for hop 2 , hop 3 and hop 4, respectively as shown in table 6.1.

Table 6.1 shows the result of wakeup times based on requested allowable loss rate of 2% and 5%, respectively using the real values.

Hop	**Sample**	**Real mean**$[ms]$	**Real variance**$[ms]$
2	9000	1024	31
3	9000	1024	46
4	9000	1024	62

Table 6.1: Full statistics without consideration of sleeping activities

Real values			**Operational state**				**Fraction of**	
Hop	$\mu[ms]$	$\sigma[ms]$	$L_{\max}$	Sample	Loss	% loss	Wakeup	Sleep
2	1024	31	2%	9000	180	2.00	0.0253	0.9747
3	1024	46	2%	8820	111	1.26	0.0309	0.9691
4	1024	62	2%	8709	76	0.87	0.0358	0.9642
2	1024	31	5%	9000	310	3.44	0.0217	0.9783
3	1024	46	5%	8690	246	2.83	0.0266	0.9734
4	1024	62	5%	8444	223	2.64	0.0308	0.9692

Table 6.2: Results for wakeup scheduling based on real parameter values

The result achieved in this scenario could be used to evaluate the reliability of the following scenarios.

6.2.2 Non-adaptive scheduling

Tables 6.3 and 6.4 summarize the results of the non-adaptive scheduling scenario for 2% and 5% allowable loss rates, respectively. The first column lists the hop number being examined, the second column shows the stopping rule based on prescribed number of packets and the last two columns in the acquisition state show the estimate of mean and the variance parameters. In the operational state the $L_{\max}$ column specifics the allowable loss rate, the subsequence columns lists the number of packets being transmitted for each hop, the frequency of packet losses, and the packets loss rate, respectively.

It is easy to see from the below tables that (a) from one hop to the next hop the fraction of wakeup times do not vary too much. (b) The first hop has the highest fraction of sleeping times and (c) within every hop you can see the trade off between acquisition state and the energy consumption. It is also clear that the fraction of sleep times is increasing tell 30 packets in the stopping rule then it goes slightly down and it behaves almost the same for all the hops.

Acquistion state				Operational state				Fraction of	
Hop	Stopping rule	$\widehat{\mu}[ms]$	$\widehat{\sigma}[ms]$	$L_{\max}$	Sample	Loss	% loss	Wakeup	Sleep
2	10	1024.2	24.524	2%	9000	310	3.44	0.0225	0.9775
2	20	1024.1	30.428	2%	9000	176	1.96	0.025063	0.9749
2	30	1024	29.045	2%	9000	180	2.00	0.024487	0.9755
2	50	1024.1	30.522	2%	9000	177	1.97	0.025102	0.9749
2	70	1024.1	30.832	2%	9000	139	1.54	0.025229	0.9748
2	100	1024.1	31.476	2%	9000	138	1.53	0.025491	0.9745
3	10	1024.3	29.501	2%	8690	396	4.56	0.024678	0.9713
3	20	1024	44.582	2%	8824	110	1.25	0.030337	0.9697
3	30	1023.9	44.562	2%	8820	110	1.25	0.03033	0.9697
3	50	1024	44.586	2%	8823	110	1.25	0.030339	0.9697
3	70	1024.1	45.725	2%	8861	120	1.35	0.030724	0.9693
3	100	1024.1	46.733	2%	8862	120	1.35	0.031061	0.9689
4	10	1024.1	45.023	2%	8294	176	2.12	0.030487	0.9692
4	20	1024	46.154	2%	8714	318	3.65	0.030867	0.9691
4	30	1023.9	50.597	2%	8710	203	2.33	0.032319	0.9677
4	50	1024	55.467	2%	8713	121	1.39	0.033839	0.9662
4	70	1024	54.951	2%	8741	122	1.40	0.033681	0.9663
4	100	1024	55.921	2%	8742	130	1.49	0.033977	0.966

Table 6.3: Non-Adaptive scheduling for 2% allowable loss rate

Acquistion state				Operational state				Fraction of	
Hop	Stopping rule	$\widehat{\mu}[ms]$	$\widehat{\sigma}[ms]$	$L_{\max}$	Sample	Loss	% loss	Wakeup	Sleep
2	10	1024.1	24.524	5%	9000	638	7.09	0.019344	0.9807
2	20	1024.1	30.428	5%	9000	400	4.44	0.021547	0.9785
2	30	1024	29.045	5%	9000	503	5.59	0.021052	0.9789
2	50	1024.1	30.522	5%	9000	401	4.46	0.021581	0.9784
2	70	1024.1	30.832	5%	9000	406	4.51	0.02169	0.9783
2	100	1024.1	31.476	5%	9000	307	3.41	0.021915	0.9781
3	10	1024	29.501	5%	8362	596	7.13	0.021217	0.9788
3	20	1024	44.582	5%	8600	218	2.53	0.026082	0.9739
3	30	1023.9	44.562	5%	8497	198	2.33	0.026076	0.9739
3	50	1024	44.586	5%	8599	218	2.54	0.026083	0.9739
3	70	1024.1	45.725	5%	8594	218	2.54	0.026414	0.9736
3	100	1024.1	46.733	5%	8693	243	2.80	0.026704	0.9733
4	10	1023.9	45.023	5%	7766	261	3.36	0.026211	0.9738
4	20	1024	46.154	5%	8382	456	5.44	0.026538	0.9735
4	30	1023.9	50.597	5%	8299	302	3.64	0.027786	0.9722
4	50	1024	55.467	5%	8381	318	3.79	0.029092	0.9709
4	70	1024	54.951	5%	8376	319	3.81	0.028957	0.971
4	100	1024	55.921	5%	8450	333	3.94	0.029211	0.9708

Table 6.4: Non-Adaptive scheduling for 5% allowable loss rate

6.2.3 Adaptive scheduling

Tables 6.5 and 6.6 list the results obtained by continually updating the parameters estimate for 2% and 5% allowable loss rate, respectively. The first column lists the hop number being examined, the second column shows the stopping rule based on prescribed number of packets and the last two columns in the acquisition state show the estimate of mean and the variance parameters. In the operational state the $L_{\max}$ column specifics the allowable loss rate, the subsequence columns lists the number of packets being transmitted for each hop, the frequency of packet losses, and the packets loss rate, respectively.

Acquistion state				**Operational state**				**Fraction of**	
Hop	Stopping rule	$\widehat{\mu}[ms]$	$\widehat{\sigma}[ms]$	$L_{\max}$	Sample	Loss	% loss	Wakeup	Sleep
2	10	1024.1	24.524	2%	9000	141	1.57	0.025291	0.9747
2	20	1024.1	30.428	2%	9000	138	1.53	0.025291	0.9747
2	30	1024	29.045	2%	9000	141	1.57	0.025291	0.9747
2	50	1024.1	30.522	2%	9000	139	1.54	0.025291	0.9747
2	70	1024.1	30.832	2%	9000	141	1.57	0.025291	0.9747
2	100	1024.1	31.476	2%	9000	140	1.56	0.025291	0.9747
3	10	1024	29.501	2%	8859	98	1.11	0.030875	0.9691
3	20	1024	44.582	2%	8862	96	1.08	0.030875	0.9691
3	30	1023.9	44.562	2%	8859	96	1.08	0.030875	0.9691
3	50	1024	44.586	2%	8861	96	1.08	0.030875	0.9691
3	70	1024.1	45.725	2%	8859	96	1.08	0.030875	0.9691
3	100	1024.1	46.733	2%	8860	96	1.08	0.030875	0.9691
4	10	1023.9	45.023	2%	8761	93	1.06	0.035749	0.9643
4	20	1024	46.154	2%	8766	92	1.05	0.035749	0.9643
4	50	1023.9	50.597	2%	8763	90	1.03	0.035749	0.9643
4	30	1024	55.467	2%	8765	88	1.00	0.035749	0.9643
4	70	1024	54.951	2%	8763	89	1.02	0.035749	0.9643
4	100	1024	55.921	2%	8764	90	1.03	0.035749	0.9643

Table 6.5: Adaptive scheduling for 2% allowable loss rate

It is clear from the adaptive scheduling results that target packet loss rate has been improved and has never exceed the requested allowable loss rate however, this at the cost of wakeup times.

Acquistion state				**Operational state**				**Fraction of**	
Hop	Stopping rule	$\widehat{\mu}[ms]$	$\widehat{\sigma}[ms]$	$L_{\max}$	Sample	Loss	% loss	Wakeup	Sleep
2	10	1024.1	24.524	5%	9000	328	3.64	0.021743	0.9783
2	20	1024.1	30.428	5%	9000	317	3.52	0.021743	0.9783
2	30	1024	29.045	5%	9000	322	3.58	0.021743	0.9783
2	50	1024.1	30.522	5%	9000	318	3.53	0.021743	0.9783
2	70	1024.1	30.832	5%	9000	323	3.59	0.021743	0.9783
2	100	1024.1	31.476	5%	9000	319	3.54	0.021743	0.9783
3	10	1024	29.501	5%	8672	237	2.73	0.026544	0.9735
3	20	1024	44.582	5%	8683	234	2.69	0.026544	0.9735
3	30	1023.9	44.562	5%	8678	234	2.70	0.026544	0.9735
3	50	1024	44.586	5%	8682	234	2.70	0.026544	0.9735
3	70	1024.1	45.725	5%	8677	234	2.70	0.026544	0.9735
3	100	1024.1	46.733	5%	8681	234	2.70	0.026544	0.9735
4	10	1023.9	45.023	5%	8435	218	2.58	0.030734	0.9693
4	20	1024	46.154	5%	8449	218	2.58	0.030734	0.9693
4	50	1023.9	50.597	5%	8444	217	2.57	0.030734	0.9693
4	30	1024	55.467	5%	8448	214	2.53	0.030734	0.9693
4	70	1024	54.951	5%	8443	215	2.55	0.030734	0.9693
4	100	1024	55.921	5%	8447	216	2.56	0.030734	0.9693

Table 6.6: Adaptive scheduling for 5% allowable loss rate

Chapter 7

Related work

Wireless sensor networks are typically battery powered devices, therefore minimizing the energy usage is one of the main issues in WSN. In reality, the network lifetime depends on energy consumption performed at each sensor node. Thus, to extend the lifetimes of the WSN an efficient power management protocols must be considered in the design process of both hardware and software. Since the transceiver consumes the most energy and almost the same level of energy in idle, transmit and receive modes [19], [14], [3], therefore it is useful to enable the nodes to operate in low duty cycle. There are a great deal of research has been done in scheduling the duty cycle of wireless sensor nodes. Depending on the type of applications, difference approaches often have different assumptions.

Researchers may have different assumptions when developing wake up scheduling algorithms, this includes but not limited to, type of traffic, transmission range, time synchronization, location information, distance information, also there are different assumption about network topology, deployment strategy, and density of the multi-hop network. Despite the fact that all of the approaches of designing power management protocols have a common objective which is to maximize network lifetime they may have different objectives determined by some tradeoffs. Most power management protocols typically done in the MAC layer ([4], [20],[17], [19], [13], etc), few of them in both network layer ([8], [19],[18], etc), and application layer ([1], [10], etc). Our approach focuses of enabling the forwarder nodes to estimate the traffic period and its jitter. By obtaining this information, we design a distributed sleep-/wakeup scheduling algorithm that enable the forwarder nodes locally wakeup or sleep just at the right time. Moreover, our approach is independent from any MAC protocols. It is a general approach that could performs on top of any MAC protocol. We have identified our approach to be MAC independent, adaptive and decentralized wakeup scheduling to enable the wireless sensor network to be adaptive to the periodic traffic characteristics and flexible in term of controlling and balancing between packet loss and energy consumption depending upon the performance requirements.

To the best of our knowledge, the adaptation of sleep periods to the estimated traffic period and jitter subject to prescribed packet loss requirements has not been considered so far in the literature.

Chapter 8

Conclusion and future work

In this research report we proposed a new scheme of extending the sleep times of wireless sensor nodes online and in decentralized way to save energy usage and consequently prolonging the network lifetimes. This approach is suitable for a number of applications, i.e. data collection and monitoring applications in which nodes have to send their data periodically to sink node thought a number of intermediate nodes. In our approach, the forwarder nodes compute a local estimate of both the traffic period and the jitter. The extra work of period estimation that forwarders do could save the source node from explicitly signaling the period to forwarders, and furthermore it saves the whole network from maintaining a common time base through a time synchronization protocol that lets all forwarders interpret the signaled period in a consistent way.

The novelty of our approach is the adaptation of sleep periods to the estimated traffic period and jitter subject to prescribed packet loss requirements. Moreover, we adjoin mechanisms to the proposed scheme that allow to update the period and jitter estimates and to react to packet losses.

We also studied the jitter distribution by conducting several experimental runs, measuring the jitter varying for example the number of hops in the multi-hop network and the packet generation period. Then we hit upon a theoretical distribution that fit the random variables of the jitter and exploited the characteristic of the jitter distribution to design and implement estimation algorithms. We used theoretical analysis, real raw measurements and simulation experiments to evaluate our proposed algorithms. Our proposed solution showed that by adapting the sleep cycles into the observation loss rate would indeed result in a significant energy saving and thus achieving a long network lifetimes. Furthermore, we demonstrated the results of the tradeoff between energy consumption and packet loss rate.

We are currently working one improving the estimation algorithms and testing the algorithms under different conditions.

Bibliography

[1] Phil Buonadonna, David Gay, Joseph M. Hellerstein, Wei Hong, and Samuel Maddent. Task: Sensor network in a box. In *European Workshop on Wireless Sensor Networks 2005*, Istanbule, Turkey, February 2005.

[2] Chipcon. *2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver.* Chipcon Products from Texas Instruments, 2004.

[3] M. Dhanaraj, B. S. Manoj, and C. Siva Ram Murthy. A new energy ef?cient protocol for minimizing multi-hop latency in wireless sensor networks. In *Proc. 3rd IEEE Intl Conf. on Pervasive Computing and Communications (PerCom 2005)*, 2005.

[4] A. El-Haiydi. Wisemac, an ultra low power mac protocol for the wisenet wireless sensor network. In *ACM SenSys*, Los Angeles, Nov 2003.

[5] D. Gay, P. Levis, R. V. Behren, M. Welsh, E. Brewer, and D. Culler. The nesC Language: A Holistic Approach to Networked Embedded Systems. In *Proc. ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation (PLDI)*, San Diego, California, USA., June 2003.

[6] S. D. Glaser. Some real-world applications of wireless sensor nodes. In *Proc. (SPIE) Symposium on Smart Structures and Materials/ NDE 2004*, San Diego, CA, USA, March 2004.

[7] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *Proc. ACM of the 9th Intl. Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS),*, pages 83–104, 2000.

[8] B. Hohlt, L. Doherty, and E. Brewer. Flexible power scheduling for sensor networks. In *Proc. of IPSN*, Berkeley, California, USA., April 2004.

[9] Averill M. Law and W. David Kelton. *Simulation Modeling and Analysis.* McGraw-Hill, third edition, 2000.

[10] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *Proc. ACM SIGMOD international conference on Management of data.*, pages 491–502, New York, USA, 2003.

[11] S. Moon, T. Kim, and H. Cha. Enabling low power listening on ieee 802.15.4-based sensor nodes. In *The 5th IEEE Wireless Communications & Networking Conferencen (WCNC)*, Hong Kong, China, March 2007.

[12] David Moss, Jonathan Hui, Philip Levis, and Jung Il Choi. *TEP 126: CC2420 Radio Stack. [Online]. Available: http://www.tinyos.net/tinyos-2.x/doc/html/tep126.ht.*

[13] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proc. ACM SenSys '04*, Baltimore, Maryland, USA., November 2004.

[14] C. Schurgers, V. Tsiatsis, and M. B. Srivastava. Stem: Topology management for energy efficient sensor networks. In *Proc. IEEE Aerospace Conference 2002*, volume vol.3, pages pp. 1099–1108, March 2002.

[15] R. Szewczyk, A. Mainwaring, J.Polastre, J. Anderson, and D. Culler. An analysis of a large scale habitat monitoring application. In *Proc. ACM SenSys '04*, Maryland, USA., November 2004.

[16] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong. A macroscope in the redwoods. In *Proc. ACM SenSys '05*, pages 51–63, San Diego, California, USA., November 2005.

[17] T. van and D. K. Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proc. ACM SenSys '03'*, Los Angeles, California, USA., November. 2003.

[18] Y. Yao, S. Alam, J. Gehrke, and S. D. Servetto. Network scheduling for data archiving applications in sensor networks. In *In Proceedings of the International Workshop on Data Management for Sensor Networks (DMSN 2006)*, Seoul, Korea, September 2006.

[19] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *Proc. IEEE Infocom, 2002*, volume vol.3, pages pp.1567–1576, June 2002.

[20] Wei Ye, John Heidemann, and Deborah Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. In *proc. ACM Transactions on Networrking*, volume 12, pages 493–506, June 2004.