Technical University Berlin

Telecommunication Networks Group

Batch Delivery Schemes for Wireless Sensor Networks

Federico Naldi and Andreas Willig

awillig@tkn.tu-berlin.de

Berlin, Dezember 2007

TKN Technical Report TKN-07-006

TKN Technical Reports Series

Editor: Prof. Dr.-Ing. Adam Wolisz

Abstract

In this paper we consider a situation where a transmitter has a batch of packets in its buffers and wants to transmit these in one go and in a semi-reliable fashion to an arbitrary subset of its neighbors over error-prone channels. We design schemes that let the transmitter node control the sleeping activities of the receivers (in order to save energy) and which adapt the sequence in which receivers are addressed to current channel conditions.

Contents

Chapter 1

Introduction

In this paper we consider situations where packets from one sensor node have to be delivered to multiple sinks within a multi-hop sensor network. The packets might either be generated by the sensor node itself or they could be packets that the sensor node has to forward on behalf of other nodes. Inspired by the approach presented in [10] and by the IEEE 802.11 power-save mode [8, 13], we assume that the sensor node (which we henceforth call the *transmitter*) does not attempt to transmit each packet individually. Instead, the transmitter waits until a certain minimum number of packets has been buffered, acquires the channel once (using some medium access control [MAC] protocol) and then transmits all packets in one go including retransmissions. We call this approach *batch delivery*. From the multi-sink assumption, the packets buffered in the transmitter might need to be transmitted to an arbitrary subset of its single-hop neighbors. For a single batch, we call the set of neighbors for which the transmitter has at least one packet buffered the *receivers*. The transmitter acquires the channel and then starts to control the batch-delivery process. It transmits the packets to the different receivers in a sequence that is governed by a *scheduling policy*. The packet transmission itself is subject to wireless channel errors and we assume that a semi-reliable automatic-repeat request (ARQ) protocol is employed to provide a certain degree of reliability. From the perspective of a receiver R it might take a random time before the transmitter starts to actually transmit packets to R, the spacing between different packets destined to R might be random as well. This is a result of the operation of the scheduling policy (which might give other receivers precedence over R) and of random channel errors. An important goal of a batch-delivery scheme is thus not only to find a good, channel-sensitive transmission schedule (which attempts to identify channels which are currently "bad" and avoids wasting energy on them), but in addition to give receivers information about the times where they can (potentially) receive packets, so that they can spent all the other times in sleep mode. In other words, a *signaling scheme* is required by which the transmitter controls the sleeping activities of the receivers and of all its other neighbors for which the transmitter has no packet buffered.

The batch-delivery approach adopted in this paper has two potential advantages: It is not required to invoke the MAC protocol separately for each packet to be delivered, and, by having the transmitter keeping the packets for a while, it allows its neighbors to sleep longer times and to concentrate their activities into small time windows. This fits very well with MAC protocols like S-MAC [14] where a neighborhood coordinates its wakeup times, exchanges data and goes back to sleep for long time. At the time of wakeup a number of packets might have queued up at a node. In some respects the setting assumed in the paper is similar to the one considered in wireless fair scheduling or channel-state dependent scheduling approaches [2, 9, 11], but instead of maximizing sum throughput subject to short- and long-term fairness constraints, the work presented here focuses on the energy consumption and sleeping activities of the neighbours (including the receivers), and, as the second-most important performance measure, the number of packets that are not successfully delivered to the receivers after exhausting their retransmission budget. The goal of this paper is to investigate how schemes for batch delivery could be organized in an energy-efficient and channel-adaptive manner, and to explore the tradeoffs between the extra energy spent by the transmitter for controlling the activities of its neighbours, and the energy spent by those neighbors. We present a protocol framework for batch-delivery, in which different scheduling policies and signaling schemes can be cast. We design a range of scheduling and signaling schemes and perform a simulation-based performance assessment. The results show that:

- It is important to use additional signaling to let neighbors that are initially empty or receivers that become empty learn about this as quickly as possible, so that they can adopt a relaxed sleeping schedule.
- For this signaling a “minimum description length” principle should be applied dynamically to the lists of non-empty and empty receivers
- Under such a signaling scheme SRQF-type (shortest request queue first) policies have intrinsic advantages: they reduce the size of the non-empty list as quickly as possible and therefore reduce the signaling load.
- The addition of different approaches to incorporate channel awareness to SRQF-type policies does not help much in terms of energy consumption, but can reduce the packet loss rates / number of failed packets.

The remaining paper is structured as follows: in the following Chapter 2 we specify in more detail the system under consideration. In Chapter 3 we describe the general protocol framework assumed for our batch-delivery schemes as well as the signaling schemes and scheduling policies used in this paper. In Chapter 4 we discuss the results of an extensive simulation study, and in Chapter 5 we put our work in perspective with related work. The paper is concluded in Chapter 6.

Chapter 2

System Model

We assume a single transmitter having a number M of stationary single-hop neighbours. Packets arrive in *batches* to the transmitter. While one batch is handled, no new batches arrive. Each packet can be destined to a different neighbour, which is then called a *receiver*. The set of receivers can be an arbitrary subset of the transmitters neighbours, and the number of packets destined to each receiver can be random as well. Conceptually, the transmitter maintains a separate queue for each receiver, see also Figure 2.1. There are two degrees of freedom for batch arrivals: one specifying the subset of receivers out of the set of neighbours, the other one the distribution of the number of packets per selected receiver. We assume that the transmitter has acquired the transmission right by some MAC protocol. The transmitter can use the medium as long as he wants, for example by using a mechanism similar to the IEEE 802.11 PCF where a NAV field is used to inhibit transmission of other stations. However, the precise operation of the MAC protocol is out of the scope of this paper.

We have oriented our work on the hardware characteristics of MicaZ motes. The physical layer is modeled after the characteristics of the ChipCon CC2420 IEEE-802.15.4-compliant wireless transceiver [3], using a bitrate of 250 kBit/sec and a maximum packet length of 127 bytes. With respect to the energy model we assume that the wireless transceiver is the dominant energy consumer, other consumers like a node's processor are disregarded. The transceiver can be switched between different modes: *transmit*, *receive*, and *sleep* mode. It is assumed that the transceiver consumes the same amount of energy when just listening on an idle channel for incoming packets and when actually receiving a packet. Hence, the idle state is not modeled explicitly. The energy consumption in the sleep state is much lower than in the transmit or receive state. Furthermore, the energy and time required for switching between states is considered as well in our model. The precise energy consumption values and switching times are given in Table 2.1. The energy consumed by any node is calculated by taking into account the total time it stays in each of the considered modes while handling a batch.

We consider two different channel models. In the first model, the *static model* all channels are independent binary symmetric channels (BSC), which in turn arise from assuming additive white Gaussian noise (AWGN) channels. To reflect different distances between the transmitter and receivers, each BSN channel can have its own bit error rate. In the second model, the AWGN channels are in addition subjected to flat fading. Specifically, each channel is varied according to a Gilbert-Elliot model [7, 6], i.e. a channel model where a single channel switches between two states "good" and "bad" according to a Markov chain, and in either state a BSC with a certain bit error probability is assumed such that in the bad state the bit error rate is much higher than in the good state. The Markov chains associated to different channels are stochastically independent and are assumed to have already

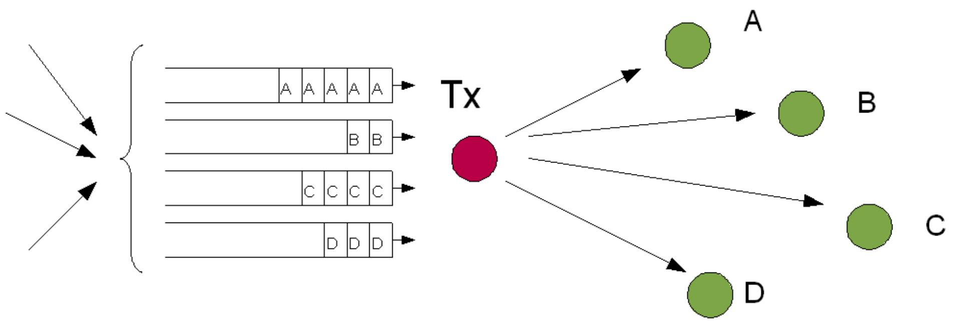


Figure 2.1: Generic system scenario

Operation	Telos	Mica2	MicaZ
Minimum voltage	1.8V	2.7V	2.7V
Mote Standby (RTC on)	5.1 μA	19.0 μA	27.0 μA
MCU Idle (DCO on)	54.5 μA	3.2 μA	3.2 μA
MCU Active	1.8 mA	8.0 mA	8.0 mA
MCU + Radio RX	21.8 mA	15.1 mA	23.3 mA
MCU + Radio TX (0dBm)	19.5 mA	25.4 mA	21.0 mA
MCU Wakeup	6 μs	180 μs	180 μs
Radio Wakeup	580 μs	1800 μs	860 μs

Table 2.1: Current consumption of Telos compared to Mica2 and MicaZ motes

reached their steady-state. We also assume that the batch interarrival time is large enough so that the channel has again reached steady state when the next batch arrives, and subsequent batches do not see correlated channels.

The independence assumption between the different channels is reasonable when errors are mostly due to fading and the receivers have a mutual distance of at least half a wavelength [12, Sec. 5.].

Finally, we assume that the transmitter and all its neighbours are time-synchronized, for example through a protocol external to the batch-delivery schemes considered in this paper.

Chapter 3

Protocol Framework, Scheduling Policies and Signalling schemes

3.1 Protocol Framework

The description of the protocol framework sets off in the moment where the transmitter has just acquired the transmission medium from executing some MAC protocol and is about to start the transmission of a batch. All neighbours are awake at this point of time. The transmitter has packets buffered for some of these neighbors, the *receivers*. We call those neighbors for which no packet is buffered or to which all buffered packets have already been successfully transmitted, the *empty receivers*. When starting to handle the batch the set of empty receivers is exactly the set of those neighbors for which no packet is buffered. However, the set of empty receivers grows as more and more receivers are successfully handled.

The protocol framework is *round-based* (compare Figure 3.1) – the handling of a whole batch can be subdivided into several of these rounds. A single round starts with a *control packet* broadcasted by the transmitter, followed by zero or more *packet slots* (the number of which is denoted as *round size*). The control packet contains information about the number of packet slots and their allocation to specific receivers – this information is the result of a *scheduling policy* executed at the transmitter. When multiple packet slots are allocated to a single receiver, these slots are allocated contiguously and

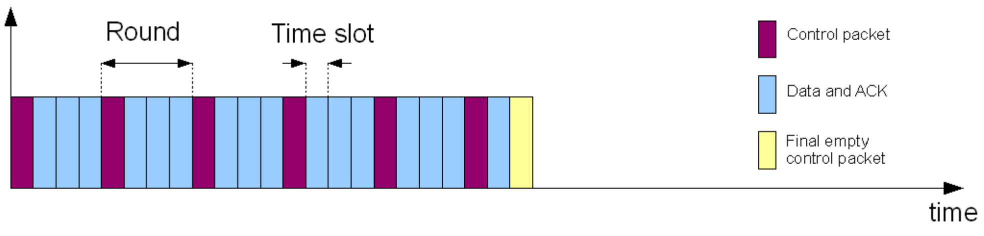


Figure 3.1: Batch delivery protocol framework

the allocation itself is run-length encoded. For simplicity, all data packet sizes are the same. A packet slot is large enough to accommodate a data packet (unicast from the transmitter to a certain receiver) and a subsequent immediate acknowledgement as well as all the required transceiver turnaround times. The control packet may contain further information by which the transmitter disseminates information about the set of empty or non-empty receivers (see below). We refer to this information as the *signaling scheme*.

The transmitters neighbors behave in the following way: at the beginning of a round they have to be awake to receive the transmitters control packet. If a receiver r fails to receive this packet, it has to stay awake for the remaining round, in order to possibly receive and acknowledge a packet destined to r. If r properly receives the control packet, it evaluates its contents. If the transmitter has allocated one or more data slots to r, the receiver r schedules its wakeup times properly to wake up just for these slots.[1] Node r remains in sleep mode during all other packet slots. When there are no slots allocated to r but r still is convinced to be a non-empty receiver, node r sleeps during all the packet slots and wakes up again immediately before the next control packet (which directly follows the last packet slot of the current round). Depending on the actual contents of the control packet (see below), a receiver r might be instructed by the transmitter that it now belongs to the set of empty receivers. It is hence not necessary for r to wake up for all the control packets of the remaining batch, and r is henceforth allowed to adopt another, more relaxed sleeping schedule. The details of r's new sleeping schedule after becoming an empty receiver are out of the scope of this paper. For simplicity, however, we assume that r sleeps until the arrival of the next batch. Without considering run-length coding, the size of the control packet increases linearly with the round size (since one address field is required per packet slot), not taking the additional signaling information into account (see below). As the size of the control packet increases, the probability that a receiver fails to successfully receive it increases as well, forcing him to stay awake for the whole round.

When the transmitter has emptied all his packet queues at the end of the batch, it broadcasts a special empty control packet, called the *sleep packet*. Upon receiving this packet a receiver now knows that it is not going to receive any further packets and that it can go back to the more relaxed sleep schedule.

The simple round-based scheme described so far is integrated with an error-control/ARQ protocol in the following way. When the transmitter schedules a round of, say, five packet slots, it picks five packets destined to arbitrary receivers at its discretion. Nothing prevents the transmitter from picking multiple (distinct!) packets towards the same receiver. The packets are assumed to carry the receivers address and a sequence number that is unique for the transmitter-receiver pair. All the five packets are transmitted by the transmitter in their respective packet slot, and for each of these packets the transmitter obtains binary feedback indicating whether an acknowledgement has been successfully received or not. Packets for which positive feedback is obtained are removed from the transmitters queues, the remaining packets are retransmitted later. A retransmission can never happen in the same round, but has to wait until later rounds. The maximum number of retransmissions that a packet can have is bounded. This arrangement can lead to out-of-sequence reception of packets at a certain receiver. The receiver has to buffer incoming packets until they can be delivered in-sequence to its higher layers or all retransmissions of the missing packets have been exhausted. It should be noted that this introduces additional delays for already received packets when they are blocked by

[1] If r has multiple contiguous slots allocated, it depends on the actual energy costs for switching the transceiver states whether it makes sense to go sleeping between two neighboured slots.

missing predecessor packets in the receivers buffer. In future work it could also be considered to replace the individual acknowledgements of each packet destined to a certain receiver by summary acknowledgements, which potentially save bandwidth but whichs loss means also the loss of more channel feedback information.

The rationale behind the design of this protocol framework is to let the transmitter explicitly control the sleeping activities of the receivers, based on its own observations of the buffer contents and channel feedback. It, however, has the additional costs of control packet transmission, and hence the round size (number of packet slots in a round) as well as the actual scheduling policies and signaling schemes are critical design parameters of our framework.

3.2 Signaling schemes

With the help of signaling schemes the transmitter informs each of its neighbors whether there are still outstanding packets destined to it (i.e. whether it is an *non-empty* neighbor) or not (*empty* neighbor). A neighbor that learns about being an empty neighbor (especially one for which no packet arrived at all in the batch) can sleep until the next batch arrival and save a lot of energy, whereas a non-empty neighbor (a *receiver*) needs to wake up for every subsequent control packet and whenever there are packet slots allocated to him.

The relevant information is piggybacked onto the control packets. There are fundamentally two different options:

- The transmitter can specify the list of empty receivers. Depending on the load characteristics, this set can initially be small, but it grows as more and more receivers are finishing their service.
- The transmitter can specify the set of non-empty receivers. This set shrinks as more and more receivers are finishing their service.

There are different options to transmit an empty-receiver list. In the **complete-list notification scheme** this list is transmitted fully once every `notification-interval` control packets, where `notification-interval` is a protocol parameter. A drawback of this scheme is that it possibly produces very long control packets with increased susceptibility to channel errors. In a variation of this scheme, the **partial-list notification scheme**, the empty list is splitted over multiple control packets within a period of `notification-interval` control packets. The `distribution-factor` is a parameter between zero and one describing how many control packets are used: the higher the distribution factor, the higher the number of control packets used and hence the smaller each individual control packet. On the other hand, the empty receivers mentioned in later control packets have to receive more control packets until they can go to sleep mode.

For the transmission of non-empty receivers lists one firstly has to take into account that the control packet already contains an allocation of packet slots to receivers, which naturally are non-empty. Hence, a list of non-empty receivers that is added to a control packet does not need to include the receivers having a packet slot, but only the remaining ones. For a non-empty receivers list it is not an option to include only a partial list into a control packet, since an empty receiver could not gain any useful information here – he cannot know whether it is an empty receiver or whether it has simply not been mentioned yet. Hence, in the **nonempty-list notification scheme** the transmitter repeats the full non-empty list every `notification-interval` packets.

It is in general of utmost importance to keep the control packets as short as possible to reduce the risk of losing them due to channel errors. As mentioned above, the empty-list grows and the non-empty list shrinks in the course of handling a batch and it is therefore appropriate to dynamically pick the one having the shorter description length whenever a control packet with signaling information is to be transmitted. This approach is picked up in the **dynamic notification scheme**. Specifically, every `notification-interval` control packets the sizes of the empty list and non-empty lists are checked and the description leading to fewer neighbour addresses to be included in the control packet is chosen. In addition, a partial-list scheme is applied to the receivers that become empty in the meantime, i.e. before the `notification-interval` control packets have been finished.

3.3 Scheduling policies

In this paper we investigate a number of different scheduling policies. Some of them can be considered as baseline policies, others are designed to incorporate channel knowledge.

3.3.1 Baseline policies

We first describe the baseline policies. To ease explanation, we use an example. Specifically, we assume a configuration with three receivers A, B and C, a constant round size of three packet slots and no channel errors. The packets destined to A are denoted as a, b, c and d, the packets to B are called α, β, γ, δ and ϵ, and the packets to C are called 1, 2, and 3.

- **Round-robin (RR)**: the receivers are served in classical, packet-wise round-robin fashion. The resulting transmission order is $a, \alpha, 1, b, \beta, 2, c, \gamma, 3, d, \delta, \epsilon$.
- **Exhaustive round-robin (ExRR)**: the receivers are handled in turn, but each one is handled exhaustively before service for next receiver starts. the transmission order is hence $a, b, c, d, \alpha, \beta, \gamma, \delta, \epsilon, 1, 2, 3$.
- **Shortest-Receiver-Queue-First (SRQF)**: the receivers are served exhaustively, but they are served in increasing order of their queue lengths. The transmission order would hence be $1, 2, 3, a, b, c, d, \alpha, \beta, \gamma, \delta, \epsilon$. It is well-known [4, Sec. 3.2] that, in the absence of channel errors, this order minimizes the mean job delay experienced by the receivers for finishing their respective service.

3.3.2 Channel-adaptive policies

The baseline policies ignore the available feedback from the channel. However, it is well-known that on fading channels the feedback can be fruitfully used to avoid transmissions during bad channel states in favor of transmitting over other wireless channels [2, 9, 11]. In our setting, the feedback is of binary nature and obtained from the presence or absence of acknowledgement packets in packet slots. It is used in different ways. One way is to compute a *packet error rate* (PER) for a specific receiver, defined as the fraction of unacknowledged packet slots to the total number of packet slots so far allocated to this receiver. Since scheduling decisions are made only once per round, the round length has impact on the “freshness” of the PER estimates. This is an important tradeoff: the shorter the round length, the more overhead, but also the more recent the PER estimates, at least for the receivers addressed in this round.

One of the basic assumptions in the design of our batch delivery schemes is that a transmitter collects packets for a time significantly longer than the channel coherence time before starting the next batch transmission. This means that for a new batch all channel feedback collected from previous batches is outdated and is therefore discarded.

We now describe the different adaptive policies in more detail. The first two take SRQF as their starting point. By modifying SRQF with channel-awareness we hope to conserve some of its favorable properties in terms of job delays.

In the **TestWindow (TW)** policy the transmitter maintains for each non-empty receiver a dynamic *window*, which is initialized according to the parameter `test-window`. The current window limits the number of packet slots that can be allocated to a receiver within a round. By keeping `test-window` smaller than the round size more than one receiver can be tested per round. In this policy the first rounds are used to acquire an initial estimate of the PER.[2] The number of packet slots dedicated to acquire an initial PER estimate for a fixed receiver is a parameter called `learning-slots`. During these first rounds the unmodified SRQF policy is used, subject, however, to the window. After the initial rounds the TW policy modifies the windows based on the (frequently updated) PERs: when the PER is below a certain threshold, the transmitter increases the window by a prescribed increment. When a receiver has acknowledged all packets during a round, the window is doubled. The scheduler selects the receivers having the largest window sizes and allocates as much packet slots as possible, either until the window or the round size is exhausted. This policy therefore favors good channels.

In the **Effective-Dynamic (ED)** policy the SRQF policy is modified. For each receiver the PER estimate is used to determine the expected number of retransmissions required per packet, and from this the expected number of packet slots that are needed to empty the queue under the given PER estimate (the *effective queue length*) is determined. To these effective queue lengths then SRQF is applied. The initial PER estimate is obtained similarly as in the TW policy using a window, but afterwards the window is not maintained anymore. The PER estimate and the effective queue lengths are updated after each round.

The **Avoid-Bad-Channels (ABC)** policy and the postponing policies follow the same idea: once there is evidence that the channel towards a certain receiver becomes bad, its use is postponed. In the ABC policy the service for the bad receiver is postponed until the end of the batch when the PER is above a pre-specified threshold. In the postponing policy the service for a receiver is inhibited for a certain number of rounds when a prespecified number of consecutive packets (either in the same round or in subsequent rounds) fail. At the end of the inhibition period two different strategies are used. In the **Postponing** policy the transmitter continues in normal fashion, whereas with the **Postponing-Probe** mode the transmitter uses only a single packet (the head-of-queue packet) as a probing packet to test the channel. If the probing packet is successful, normal operation is resumed, otherwise another inhibition period follows, which is longer (linear increase). This approach has the drawback that it exhausts the retransmission budget of the head-of-queue packet. Methods to overcome this problem are a possible subject of future work.

[2] At this place an important tradeoff concerns the amount of feedback that is used to create an initial estimate of a PER before it is being used for scheduling decisions: as more feedback is collected, the initial PER estimate becomes more precise, but takes more time to acquire and faces the risk of becoming outdated when the acquisition time becomes larger than the channel coherence time. It should be noted, that channel adaptation in general is only useful when the channel coherence times are significantly longer than an individual round, since otherwise a decision is made for a channel that has likely changed since the last observation.

3.3.3 Round optimization

All the scheduling policies described so far operate with fixed round-sizes. Round optimization adds the following rule: when, after applying the scheduling algorithm, there remains a receiver to which packets slots have been allocated in this round and which furthermore has only a single outstanding (i.e. unscheduled) packet, then the outstanding packet is included as well and the round is prolonged.

3.4 Performance measures

The major performance measures considered in this paper are the following ones:

- *Energy consumption* refers to the sum energy spent per batch by all receivers. Only transceiver modes (transmit, receive, sleep) and mode-switching operations are taken into account. The energy is expressed in energy units. The smaller this value is, the better.
- *Overhead*: this denotes especially the number of control packets and charactarizes the (additional) energy spent by the transmitter for a certain scheme. Smaller values are preferrable.
- *Energy efficiency* indicates the number of successfully received bits per energy unit for a non-empty receiver and characterizes a relationship between transmission reliability and required energy investment. Higher values are preferrable.
- *Job delay* for a certain receiver. This denotes the time between batch arrival and the point in time where the receiver queue becomes empty. Smaller values are preferrable.
- *Packet Loss Rate* gives the fraction of packets within a batch that are not received successfully by the intended receiver. In the below figures we represent the packet loss rate by giving the average total number of packets that could not be successfully transmitted. This measure is equivalent to the packet loss rate and also highlights the relevant trends.
- *Fraction of awake receivers* at the end of a batch (i.e. after transmitting the sleep packet). This parameter characterizes a signaling schemes ability in informing the empty receivers such that they can sleep until the next batch arrival. Smaller values are preferrable.

Chapter 4

Simulation results

In this section we present the results of a simulation study investigating different signaling and scheduling policies. We study signaling schemes and scheduling policies separately because the parameter space that would have to be considered for a joint characterization of these two aspects would have been too large. We start our discussion with the investigation of signaling schemes, keeping the scheduling policy fixed to the round-robin (RR) strategy. In a second step, we keep the signaling scheme fixed (a dynamic notification scheme with notification interval equal to 3 has been selected because of its reasonable performance compared to the other investigated strategies) and investigate the different scheduling policies.

The results have been obtained with the help of a simulation tool developed with OMNet++ [1] and the TKN mobility framework providing, among others, different wireless channel models [5]. Please note that our system, load and channel assumptions imply that our simulation is of regenerative type.

4.1 Performance of signaling schemes

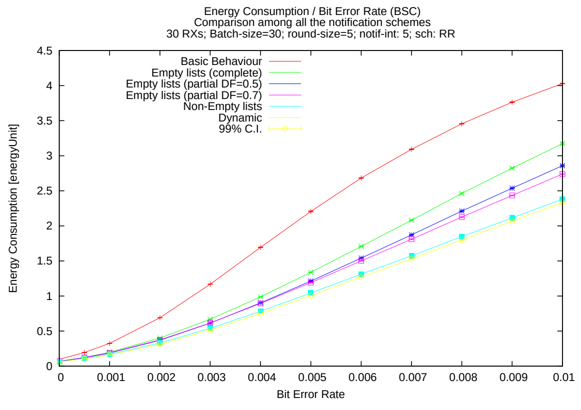
The performance of signaling schemes is mainly related to its ability to inform empty receivers about their state, and in the second place to control the future sleeping activities of non-empty receivers. Both abilities are related to the contents and size (packet losses!) of control packets and require the transmitter to spend additional overhead and energy to transmit them.

The following experiment setup has been used for comparison of signaling schemes. The scheduling policy has been fixed to round-robin. For the channel model we assume that all the channels (one channel per receiver) are independent BSC channels of the same bit error rate. This common bit error rate is varied in the experiments. The assumption of having stochastically identical channels is harmless, since each receiver is influenced separately by the signaling. The traffic load is defined as follows: the batch size is kept fixed to 30 packets (having a size of 29 bytes user data), but the number of (empty or non-empty) receivers is varied and either 10 or 30. To each non-empty receiver a random number of packets uniformly distributed between one and four is directed.

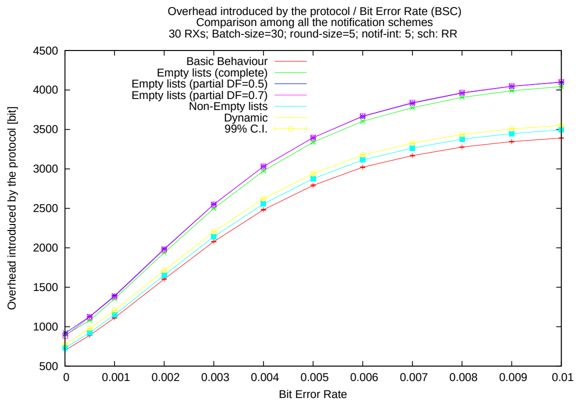
In all the following curves we show averages over all the (empty or non-empty) receivers, which is valid since all receivers have the same BER. For each parameter set a large number of batches has been simulated, resulting in very tight 99% confidence intervals for the averages.

We compare the different signaling schemes described in Section 3.2 with each other and with a

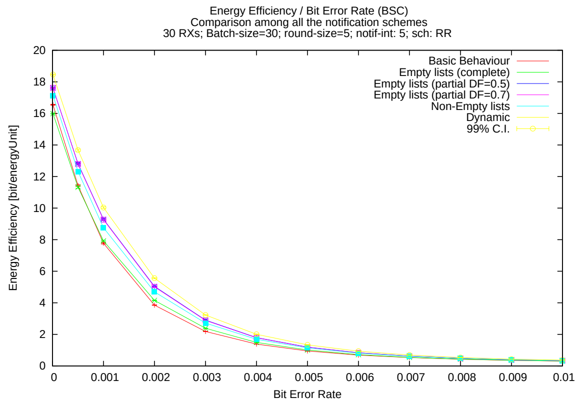
basic behaviour, in which no signaling scheme is used at all and each control packet only contains the packet slot allocation. To save space, we show in Figure 4.1 only results for the case of 30 receivers, comparing the dynamic notification scheme with the basic behavior, the complete-list notification scheme, two instances of the partial-list notification scheme (with distribution factors of 0.5 and 0.7, respectively), and the non-empty list notification scheme. The round size is fixed to five, the notification interval is also set to five. We have varied the common bit error rate (BER) in the curves. Please note that on average an increased BER results in longer times (and more control packets) needed to handle a batch.



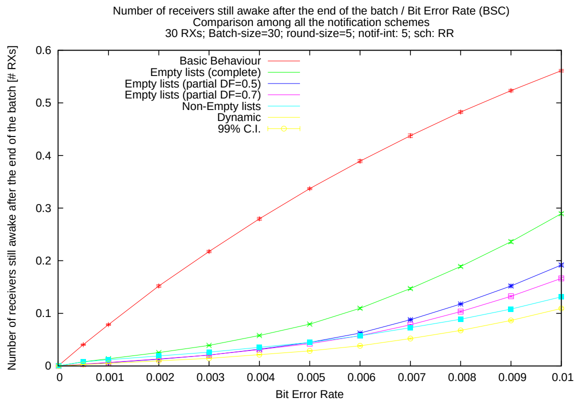
(a) Energy consumption



(b) Overhead



(c) Energy efficiency



(d) Percentage of awake receivers

Figure 4.1: Ccomparison among all the schemes

The following points are remarkable:

- Except for the overhead, the basic behavior has the worst performance of all considered schemes considering energy consumption, energy efficiency and fraction of awake receivers. This shows that the use of signaling schemes indeed has significant energy benefits for the receivers over the basic behavior. While not explicitly investigated, we believe that avoiding at all the control packets would even be worse for the receivers, since they would have to stay awake *all* the time until they receive their last packet.

- The dynamic notification scheme and the non-empty lists notification scheme have very similar performances at approximately the same overhead. This can essentially be attributed to the fact that for not too large initial numbers of non-empty neighbors both schemes are identical most of the time, perhaps with the exception of the first few rounds. The dynamic notification scheme has a slightly higher overhead (due to the additional partial empty-list approach applied to receivers that become empty in the middle of a notification interval), but a slightly better energy efficiency (emptying receivers can go back to sleep earlier).

- With respect to the overhead, the "minimum-description length" approach of dynamic notification leads (not surprisingly) to having lowest overhead among all signaling schemes except the basic behavior.

- The non-empty list notification scheme tends to outperform all empty-list based schemes with respect to all the considered performance measures. This can be explained by the systematic advantage that the former has over the latter, coming from the fact that the non-empty lists are already partially specified by the packet slot allocations and less additional information is needed. Hence, control packets tend to be smaller than for empty-list schemes, which in turn is beneficial since smaller control packets are less prone to channel errors and fewer receivers have to be awake for a whole round due to loss of control packets. It is not shown in the figures but confirmed by simulation results that partial empty-list notification schemes can have energy consumption advantages over non-empty list notification schemes when the initial number of receivers and the fraction of non-empty receivers is high.

- The dynamic notification scheme has also the best performance in terms of the fraction of awake receivers. This can be partly explained by the comparably short control packets of dynamic notification (giving higher probability that empty receivers know their status), and partly by the fact that an initially non-empty receiver is actually signaled *more often than elsewhere* about becoming empty: one time when it is included in the partial empty-list between two control packets carrying a non-empty notification, and the second time in the subsequent and all following control packets.

We mention some further findings without showing results. The round size and the notification interval influence the protocol performance. For example, we have chosen the round size from the set $\{5, 10, 20, 30\}$ for the dynamic notification scheme. The results show that smaller round sizes lead to less energy consumption (at the receivers!) while having a higher overhead. The advantage of smaller round sizes can be attributed to the less severe consequences that a lost control packet has for a receiver. The influence of the notification interval on the dynamic scheme has also been investigated with the notification interval chosen from the set $\{3, 5\}$. The results show that shorter notification intervals lead to reduced energy consumption and (not surprisingly) higher overhead. A possible explanation of the better energy consumption for shorter intervals is that a receiver who has been emptied within a notification interval and who has missed the emptying notification from the partial empty-list scheme, can recover earlier from having incomplete knowledge when the notification interval is shorter.

Please note that the non-empty list and the dynamic notification scheme fit very well together with SRQF-type scheduling policies. This is for two reasons. Firstly, SRQF-policies have the tendency to reduce the size of the non-empty list as quickly as possible. This, together with the above mentioned optimality property of SRQF means that emptied receivers can go back to sleep mode as early as possible, leading to energy savings. Furthermore, the regime where transmission of a non-empty list is more energy-efficient than transmitting empty-lists is reached quickly. Secondly, these scheduling policies serve the selected receivers exhaustively, which often means that multiple packet slots are allocated to a receiver within a single round. In this case there is the additional benefit of a size reduction of the control packet due to the run-length encoding of the packet allocation list and due to addressing fewer receivers in the list.

4.2 Performance of scheduling schemes

For investigating the performance of the scheduling policies a more elaborate channel model is used, mimicking flat fading channels. Specifically, each channel is an independent Gilbert-Elliot (GE) channel, and the channels of different receivers can be of different average quality. A GE channel can be characterized by four quantities: bit error rate (BER) in the good state e_g, BER in the bad state e_b, average state holding time in good state T_g and average state holding time in bad state T_b. We have fixed $e_b = 10^{-2}$, $T_g = 200$ ms and $T_b = 25$ ms, but e_g is a random variable, differing among receivers. This random variable is drawn as $\alpha \cdot 10^{-6}$ with α taken from a truncated exponential distribution with range $[0, \alpha_{\text{max}}]$ and parameter $\alpha_{\text{max}}/2$. The parameter α_{max} is varied between 100 and 10.000 and is referred to as the *channel variability*. It can roughly be regarded as a measure of the variability among the different wireless channels, and it is also a measure of the average channel quality: the higher α_{max}, the larger the average bit error rates in the good state. The batch interarrival times are large enough to let the channels have reached their steady-state again after finishing one batch, so that subsequent batches do not see correlated channels.

The traffic model is chosen as follows. The number of receivers is fixed to 30, and in each batch there are packets to half of the receivers. The number of packets directed to each non-empty receiver is chosen uniformly from one to five.

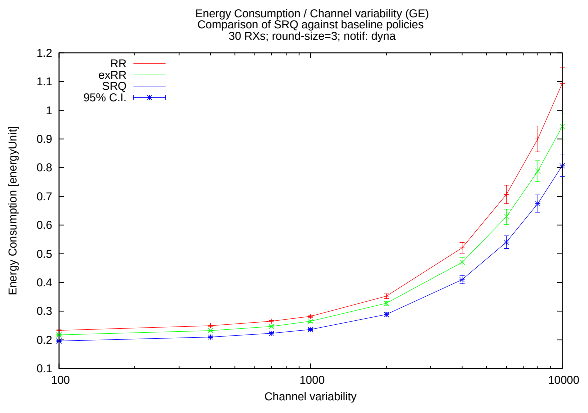
The signaling scheme has been fixed to dynamic signaling with a notification interval of 3.

For the averages mentioned in this section we indicate their 95% confidence interval in the figures, obtained over 30 independent instantiations of channel error rates for a fixed parameter setting.

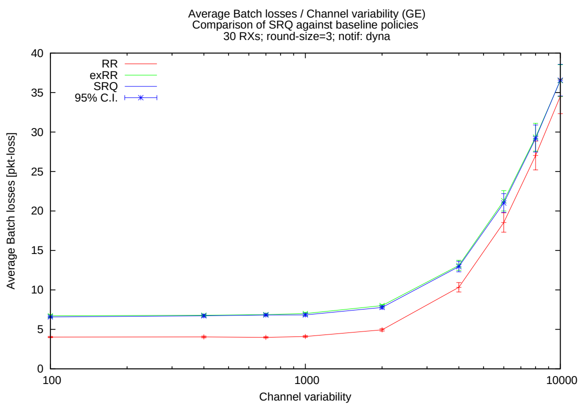
4.2.1 Insights about baseline policies

In Figure 4.2 we first evaluate major performance indicators for the three baseline policies RR, exRR and SRFQ over the above fading channels (similar results hold also for BSC channels). Here and in the subsequent figures the packet loss rate is expressed as the average number of packets which exhausted their maximum number of retransmissions (being four).

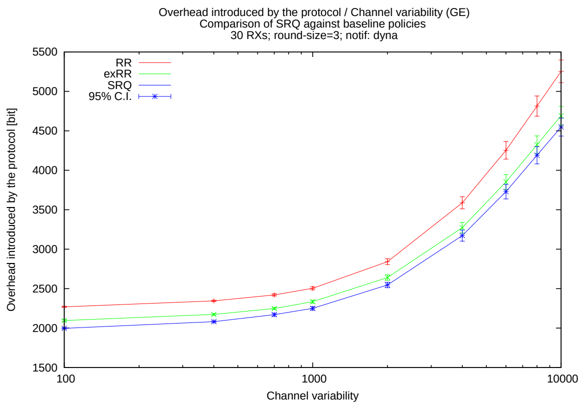
It can be seen that SRFQ has the best energy consumption and the smallest overhead over all considered channel variabilities, but, together with the other exhaustive strategy exRR, the worst packet loss rate. The latter points to a general problem of non channel-aware exhaustive policies. These can allocate multiple packets to channels currently in a deep fade (bad channel state) and doing so multiple rounds in succession, thus wasting both energy and the retransmission budget of packets. The superior energy consumption performance of SRQF results from two influences: (i) its above discussed tendency to quickly reduce the number of non-empty receivers and thus to reduce the signaling load for dynamic notification. (ii) its tendency to allocate many slots to a single receiver, allowing to benefit from the run-length coding scheme.



(a) Energy Consumption



(b) Number of failed packets



(c) Overhead

Figure 4.2: Performance of baseline policies on fading channels

4.2.2 Comparing channel-adaptive policies and baseline policies

The previous discussions indicate that a fruitful approach might indeed be to endow SRQF with some channel-awareness in order to maintain (more or less) its optimality properties while improving on its packet loss properties by avoiding fading channels for some time.

In Figure 4.3 we compare the energy consumption and the packet loss rate for the three strategies Test-Window (TW, with the initial window size set to three), Dynamic-Effective (DE) and Avoid-Bad-Channels (ABC) with the three baseline policies, and in Figure 4.4 we do the same for policies based on postponing (Postponing[1] and Postponing with additional DE scheduling). The chosen round size is three. In all the figures round optimization has been enabled, since our results indicate the usefulness of this especially for smaller round sizes (not shown here).

The results show two different types of behavior. On the one hand, the TW policy shows practically the same performance in terms of energy consumption (at the receivers!) and packet loss rate as the SRFQ policy. However, not shown here, the TW policy requires less overhead.

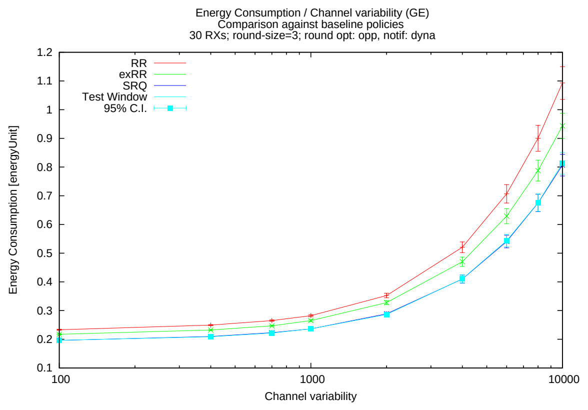
[1]Postpone-Probing shows no noticeable difference in performance to normal postponing and is therefore not explicitly shown.

On the other hand, the other policies (which are more responsive to channel variations within a batch than the TW policy is) all show a tradeoff between energy consumption and packet loss rate. All the four policies (DE, ABC and postponing policies) achieve practically the same packet loss rate as the baseline round-robin scheme, but have a slightly higher energy consumption than SRFQ for higher channel variabilities, whereas for lower variabilities the energy consumption is practically the same as for SRFQ. Visually, the postponing scheme appears to be the one that comes closest to the energy consumption of SRFQ over the widest range of channel variabilities.

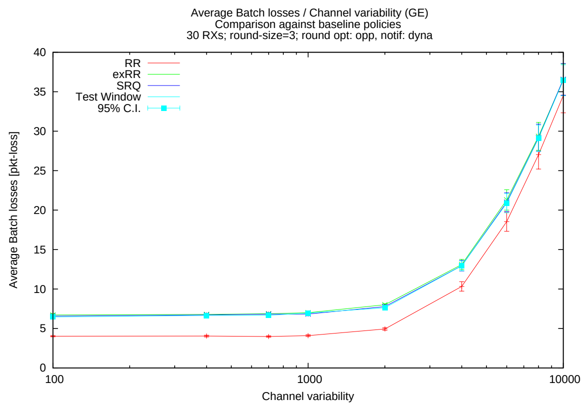
A possible explanation for the losses in terms of energy consumption that the four schemes DE, ABC and postponing have for higher variabilities are the difficulties in acquiring reasonable channel estimates. These are always present, but for low channel variability the channels are very similar and influenced in the same way by estimation errors. For higher channel variability the estimation errors differ as well, opening more space for inappropriate decisions. Another phenomenon, which we attribute to the same causes but which we do not show here, is that for low channel variabilities the DE, Postponing, Postponing with additional DE and ABC policies the job delay is truly better than for SRFQ, while for higher variabilities SRFQ wins also in this measure (the crossover point is slightly different for the different policies, though). However, the impact of estimation errors needs to be investigated more closely in future work.

Why are there no significant gains in terms of energy consumption as compared to SRFQ for bursty channels? One possible explanation relates to the fact that during bad channel periods a receiver fails not only to receive data packets, but also control packets, and this means that the receivers have to stay awake during bad channel periods. Hence, only the transmitter overhead and the retransmission budget of packets benefit in the channel-aware schemes.

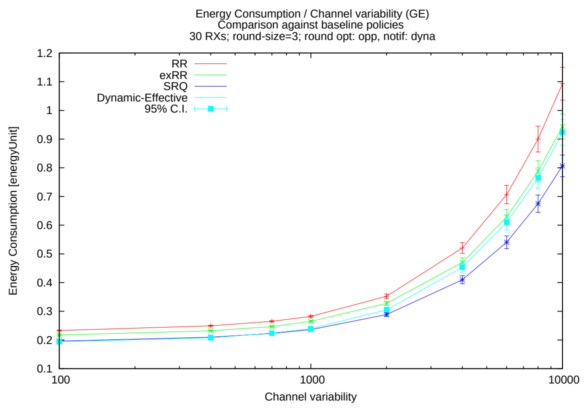
To summarize, most of the schemes presented here actually offer a tradeoff between energy consumption on the one hand (where SRFQ is actually hard to beat) and packet loss performance and overhead on the other hand. In the investigated scenarios it appears that a simple strategy where SRFQ is endowed with postponing is a robust choice, approaching the packet loss performance of round-robin and being reasonably close in energy consumption performance to SRFQ over a wide range of variabilities.



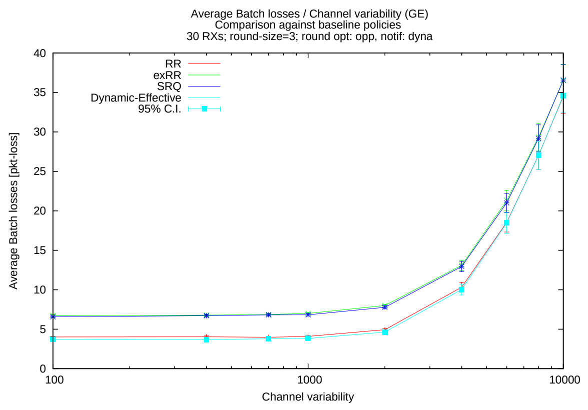
(a) Energy Consumption (TestWindow)



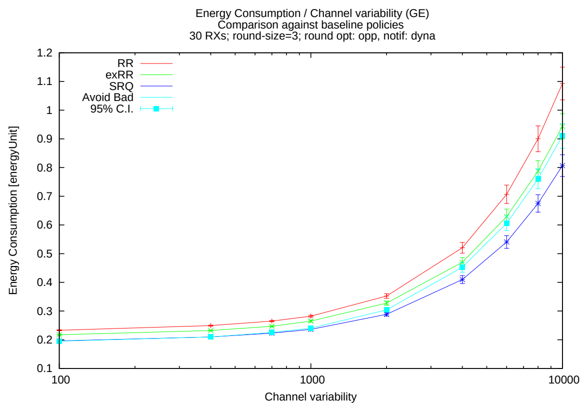
(b) Number of failed packets (TestWindow)



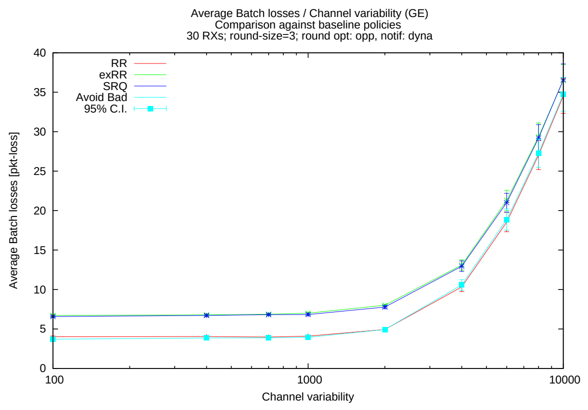
(c) Energy Consumption (Dynamic Effective)



(d) Number of failed packets (Dynamic Effective)

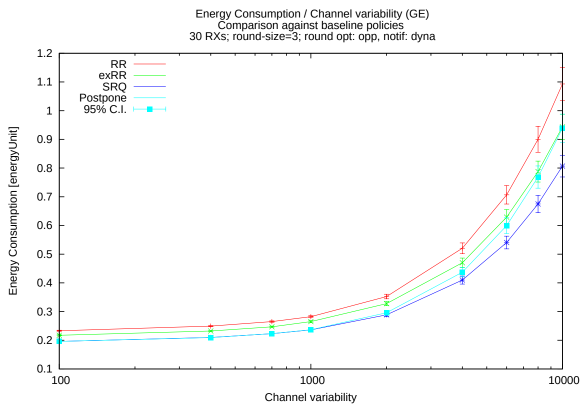


(e) Energy Consumption (Avoid Bad Channels)

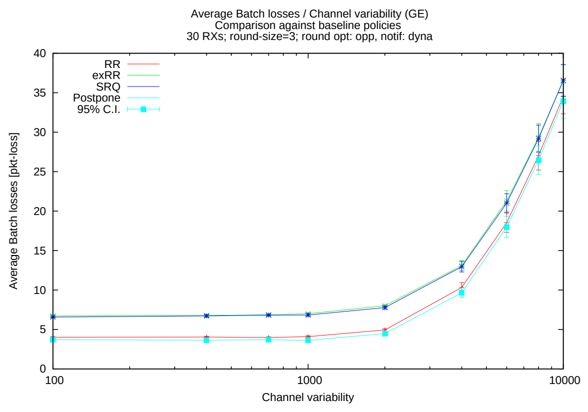


(f) Number of failed packets (Avoid Bad Channels)

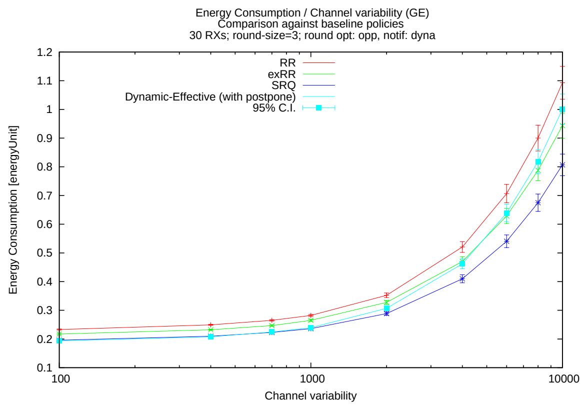
Figure 4.3: Performance comparison of channel-adaptive polices and baseline policies on fading channels with round optimization



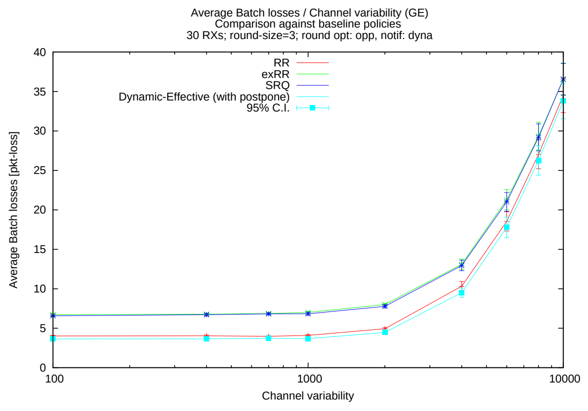
(a) Energy Consumption (Postpone)



(b) Number of failed packets (Postpone)



(c) Energy Consumption (Postponing with DE Scheduling)



(d) Number of failed packets (Postponing with DE Scheduling)

Figure 4.4: Performance comparison of postponing-based channel-adaptive polices and baseline policies on fading channels with round optimization

Chapter 5

Related Work

The general approach of collecting larger packet batches in a node and transmitting them in one go is well-known and followed for example in the IEEE 802.11 power save mode (ATIM mechanism). Collecting packets in general helps to reduce the duty cycle, and transmitting them in one batch avoids executing a MAC protocol multiple times. The usage of this approach in wireless sensor networks has for example been suggested in [10].

The work presented here has some similarities to previous works on channel-aware and fair scheduling from wireless access points to a number of wireless stations (see [11] for a survey, see furthermore [9, 2]). The main focus of these algorithms is to provide high sum throughput and fair scheduling to a number of wireless stations that are continuously backlogged in such a way that (location-dependent) wireless fading channels that are currently in a bad state are not used for a while in favor of better channels. To achieve long-term fairness, a compensation scheme is applied when the channel turned back into a good state. Numerous schemes have been devised, many of them use (wireline) fair queueing approaches as their starting point and endow it with channel estimation and compensation schemes. Our work differs from wireless fair scheduling in important aspects. Firstly, our goal is to optimize for energy consumption of the receivers (and to a lesser degree of the transmitter) and packet loss rates, whereas fairness and sum throughput are not a primary concern. Secondly, all the wireless fair scheduling schemes are centered around flows and essentially assume that there is always something to transmit. This assumption is not realistic, however, for many types of wireless sensor networks.

Chapter 6

Conclusions

In sensor networks where neighbored nodes wake up in a synchronized fashion it might well happen that multiple packets accumulate in a node and need to be delivered to several neighbors. In such a setup the approach taken by our batch delivery schemes, namely, to deliver all packets in one go, can be a very attractive approach. In this paper we have presented a protocol framework for batch delivery, which allows the transmitter to explicitly control the sleeping activities of its neighbours through control packets and signaling schemes, and to schedule the order of packet transmissions according to its own policy.

Our results from investigating different signaling schemes hint to the fact that scheduling policies in the best case arrange their transmissions so that as many stations can be emptied as quickly as possible so that they can sleep for the remaining batch duration. This in turn fits very well with shortest-request-queue-first-type policies, for which it is well-known that they achieve the above mentioned goal under the assumptions of no channel errors. A very useful result is that, when applied to fading channels and in situations where different receivers have different channel qualities, SRQF policies are hard to beat in terms of energy consumption of receivers, but can be improved upon in terms of reliability (packet loss rates). The channel-aware policies investigated in this paper give the application designer to gain additional reliability at comparably low losses in energy consumption, especially for cases where the channels to all receivers are relatively homogeneous (low variability).

There are several interesting topics for future research. One is the assessment of the influence of estimation errors on the performance of the channel-aware policies. Secondly, the design of additional protocol mechanisms that allow receivers currently suffering from a bad channel state to sleep even when they do not receive the control packets from the transmitter. Such a mechanism could for example be based on cooperative transmission techniques. Furthermore, it will also be interesting to compare batch delivery with individual delivery of each packet in order to check whether the fundamental design assumption that it is preferrable to transmit all packets in one go in a controlled fashion is really better energy-wise. However, this would depend crucially on the specific MAC (and related costs like time synch for TDMA) used for delivery of individual packets.

Bibliography

[1] *OMNet++ V. 3.3 Simulation Package*, 2006.

[2] Pravin Bhagwat, Partha Bhattacharya, Arvind Krishna, and Satish K. Tripathi. Using channel state dependent packet scheduling to improve TCP throughput over wireless LANs. *Wireless Networks*, 3(1):91–102, March 1997.

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

[4] Richard W. Conway, William L. Maxwell, and Louis W. Miller. *Theory of Scheduling*. Addison-Wesley, Reading/Massachusetts, 1967.

[5] W. Drytkiewicz, S. Sroka, V. Handziski, A. Koepke, and H. Karl. A mobility framework for omnet++. In *3rd International OMNeT++ Workshop*, Hungary, January 2003.

[6] E. O. Elliot. Estimates of error rates for codes on burst-noise channels. *Bell Systems Technical Journal*, 42:1977–1997, September 1963.

[7] E. N. Gilbert. Capacity of a burst-noise channel. *Bell Systems Technical Journal*, 39:1253–1265, September 1960.

[8] LAN/MAN Standards Committee of the IEEE Computer Society. *Information technology – Telecommunications and Information Exchange between Systems – Local and Metropolitan Area Networks – Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 1999.

[9] Songwu Lu, Vaduvar Bharghavan, and Rayadurgan Srikant. Fair queueing in wireless packet networks. In *Proc. of ACM SIGCOMM'97 Conference*, pages 63–74, Cannes, France, September 1997.

[10] Matthew J. Miller and Nitin H. Vaidya. A MAC Protocol to Reduce Sensor Network Energy Consumption Using a Wakeup Radio. *IEEE Transactions on Mobile Computing*, 4(3):228–242, May 2005.

[11] Thyagarajan Nandagopal and Xia Gao. Fair scheduling in wireless packet data networks. In Ivan Stojmenovic, editor, *Handbook of Wireless Networks and Mobile Computing*, pages 171–194. John Wiley & Sons, New York, 2002.

[12] Theodore S. Rappaport. *Wireless Communications – Principles and Practice*. Prentice Hall, Upper Saddle River, NJ, USA, 2002.

[13] Christian Röhl, Hagen Woesner, and Adam Wolisz. A Short Look on Power Saving Mechanisms in the Wireless LAN Standard IEEE 802.11. In J. M. Holtzmann and M. Zorzi, editors, *Advances in Wireless Communications*, pages 219–226. Kluwer Academic Publishers, April 1998.

[14] Wei Ye, John Heidemann, and Deborah Estrin. Medium access control with coordinated, adaptive sleeping for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 12(3):493–506, June 2004.