Poster Abstract: Message Position Modulation for Power Saving and Increased Bandwidth in Sensor Networks

Johannes Schneider
Computer Engineering and Networks Laboratory
ETH Zurich
schneider@tik.ee.ethz.ch

Roger Wattenhofer
Computer Engineering and Networks Laboratory
ETH Zurich
wattenhofer@tik.ee.ethz.ch

Abstract

We use a form of pulse-position modulation, i.e. message-position modulation (MPM), to reduce the payload of a message by encoding parts of the message through its transmission time. MPM allows to conserve energy and channel usage, thus reducing the risk of collisions and increasing available bandwidth. We evaluate our ideas on the TinyNode platform showing a notable reduction of transmission energy and channel access without requiring significant usage of additional resources such as memory.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design,Wireless communication

General Terms

Algorithms

Keywords

Low power,Medium Access Control(MAC), Time coding

1. INTRODUCTION

Low duty cycles, e.g. infrequent transmissions, is a must for Medium Access Control(MAC) for low power sensor networks. For many applications information, e.g. measured data, reaches the base station only after tens of seconds. Thus, a small variation of the data transmission time of a (sensing) node is generally of no concern. The base station itself usually has much larger storage, processing and energy means than a sensing node. MPM exploits this asymmetry by keeping the base station awake and letting the neighbors of the base transmit some information essentially without energy consumption, i.e. through passed time between the earliest possible transmission and the actual transmission. MPM also results in less channel usage, since message-position modulated messages are shorter than ordinary messages.

Pulse-position modulation is a well-known signal modulation technique where m bits are encoded using 2^m time-shifts, yielding a transmission ratio per time shift of $m/2^m$. In (only) one of the m time-shifts (or time slots) a pulse, carrying itself little or no information, is transmitted, say in time-slot $x \in [0, 2^m - 1]$, which is decoded by the receiver as value x. We suggest using pulse-position modulation to code only parts of the data to transmit using time-shifts. The technique is sensitive to multipath effects for high data rates, i.e. short time-shifts, and clock drift since a signal running on a long path and arriving at a later time-shift than intended can cause the decoding of wrong data. We avoid the danger of faulty data decoding due to multipath effects by using time-shifts that are long enough such that a signal must cover long distances. Therefore, it attenuates below the threshold for reception when reaching the receiver in the wrong shift. Differential position modulation, i.e. measuring the time between transmissions, rather than absolute message arrival times at the base station, helps to deal with clock drift.

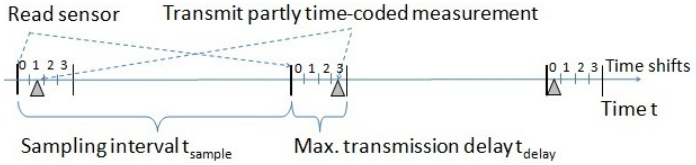

Figure 1: Basics of MPM

2. MESSAGE PASSING MODULATION (MPM)

We describe all important parameters for MPM for a low power sensor network such as a network measuring temperature, humidity, light etc. using exemplary values (We discuss settings in general in Section 4). A node should read its sensor value every $t_{sample} = 100$ seconds. The transmission of a sample can be postponed by $t_{delay} = 4$ seconds.

To account for multipath effects we assume that the longest path a signal might travel and still be received takes $t_{multi} =$ 0.1ms longer to follow than the shortest distance between base station and the sensing node. This corresponds to an additional length of 30 km. Typical sensor nodes can communicate up to a few kilometers given very good conditions (e.g. line of sight) and thus any signal that traveled an extra 30 km is too weak to be decoded.

We assume a clock drift d_{ppm} of about 45 ppm between the base and a sensing node. Given that data is transmitted reliably (i.e. the base station acknowledges each message and the sensing node retransmits if it misses an ACK), the longest time between two messages transmitted via MPM from a sensing node is in $[t_{sample}, t_{sample} + t_{delay}]$. If the system works in a best effort manner, i.e. lost messages im-

IPSN'11, April 12–14, 2011, Chicago, Illinois.
.

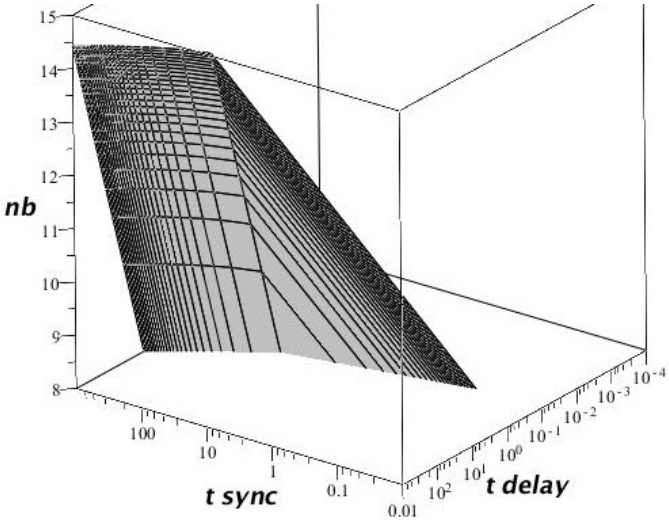

Figure 2: Number of saved bits n_b per transmission (of at least one byte) depending on the allowed (maximum) delay of a transmission (t_{delay} in [s]) and the maximum time between two message receptions t_{sync} (in [s]) by the base, i.e. the synchronization interval.

ply data loss, the time between two received messages might be (much) larger, which increases the (maximum possible) clock drift. To deal with this problem and allow for regular resynchronization, a sensing node might demand an ACK from time to time to make sure that the time between received messages is not too large. Assuming a maximal time span of $t_{sync} = t_{sample} + t_{delay}$ between two received transmissions from a sensing node implies that the clock of the base station and the sensing node drift up to $t_{drift} = (t_{sample} + t_{delay})/10^6 \cdot d_{ppm} = 4.7$ms for this time interval.[1] If the sensing node's clock shows time t then the clock of the base station might have any value from $[t - t_{drift}, t + t_{drift}]$. Additionally to the clock drift, the time between execution of the "send" command and the actual transmission might vary. For example, the delay might depend on the amount of data to copy to the radio buffer after the "send" command is called. We assumed a random transmission (plus reception) jitter, i.e. delay, of up to $t_{trans} = 2$ ms.

Therefore, the length of a time shift is given by $t_{shift} := t_{drift} + t_{multi} + t_{trans} = 6.8$ms. Using MPM we can code $n_b := \lfloor \log(t_{delay}/t_{shift}) \rfloor = \lfloor \log(4000/6.8) \rfloor = 9$ bits through time-coding per transmission. From the view of the base station the transmission by the sensing node should occur in the middle of a time-shift, i.e. with an offset of $t_{shift}/2$ to account for a positive or negative clock drift. The sensing node can compute the time shift t_{shift} to transmit as follows $t_{shift} := (n_b + \frac{1}{2}) \cdot t_{shift}$. Thus after the sample has been obtained and the time shift t_{shift} is computed the sensing node sleeps for time span t_{shift}, wakes up and transmits.

[1]The number of clock ticks for $t_{sync} = t_{sample} + t_{delay}$ is $t_{sync} \cdot f$, where f is the frequency of the oscillator. The number of drifted clock ticks is $t_{sync} \cdot f/10^6 \cdot d_{ppm}$, dividing by f yields the drift $t_{drift} := t_{sync}/10^6 \cdot d_{ppm}$.

3. EVALUATION

Our test network consisted of a base station, two sensing nodes and a jammer node, all positioned within one room. The most interesting test case is a system where data is transmitted in a best effort way (without ACK), since lost messages prolong the time between two receptions of a message. We intentionally used a jammer node close to the base station to create interference by transmitting randomly every 8 ms on average. Therefore, a longer time shift t_{shift} as in Section 2 of 14ms was used to correctly decode a message and still save 1 byte per transmission. The other parameters, i.e. $t_{sample} = 100$s and $t_{delay} = 4$s, are as in Section 2. For implementation the TinyNode 584 sensor platform running TinyOS 2.1.1 was taken. We only relied on standard TinyOS commands for transmission. TinyOS does not guarantee that the point of transmission is the same as the time when the send procedure was called due to other (long-running) tasks. This is usually of minor concern, since low-power networks generally do not perform long and heavy computations requiring lots of energy. We did not use any synchronization mechanism (e.g. [1]) or temperature clock drift compensation[2] but employed the milli seconds timer provided by TinyOS. Our implementation uses only little extra system resources, e.g. less than 20 bytes of RAM. The system was ran until 500 messages were received by the base station. In theory, due to multiple missed messages the synchronization might deteriorate and data might be decoded wrongly. Still, all received time coded data was decoded correctly, the maximum clock drift between two receptions of a message by the base was 5ms.

4. DISCUSSION AND IMPROVEMENTS

The usefulness of MPM depends on the (worst case) drift between both clocks. A constant offset or slowly varying drift can be compensated to a large extend, e.g. by using temperature measuring capabilities of the node[2] or special algorithms, e.g.[1]. Either way, to maintain sufficient synchronization messages must reach the base station (or the sensing node) from time to time. For all applications, where the message loss is low or only a certain (or no) data loss is acceptable, this requirement is automatically fulfilled.

Figure 2 shows the time-coded bits per transmission $n_b := \lfloor \log(t_{delay}/t_{shift}) \rfloor$, where $t_{shift} := t_{drift} + t_{multi} + t_{trans} = t_{sync}/10^6 \cdot d_{ppm} + t_{multi} + t_{trans}$. The clock drift d_{ppm} is 45 ppm. The (unknown) random transmission (plus reception) delay t_{trans}, depending on the hardware is 2ms. The time used for multipath compensation t_{multi} is 0.1ms.

Roughly speaking, the number of "saved" bits grows by 1 when doubling t_{delay} (up to at most t_{sync}) and decreases by 1 when doubling t_{sync} or the clock drift rate d_{ppm}. Thus, the number of "saved" bits is relatively insensitive to changes in the setup. In many scenarios it is straight forward to save one byte and with better clocks (or more advanced synchronization) it is not hard to save two bytes, but saving three bytes requires special hardware, e.g. a clock drift of less than one microsecond for time interval t_{sync}.

5. REFERENCES

[1] C. Lenzen, P. Sommer, and R. Wattenhofer. Optimal Clock Synchronization in Networks. In *SenSys*, 2009.

[2] T. Schmid, P. Dutta, and M. B. Srivastava. High-resolution, low-power time synchronization an oxymoron no more. In *IPSN*, 2010.