Bit Reversal Broadcast Scheduling for Ad Hoc Systems

Marcin Kik,
Maciej Gebala, Mirosław Kutyłowski

Wrocław University of Technology, Poland

IDCS 2013, Hangzhou

How to broadcast efficiently?

Broadcasting
ad hoc systems

Broadcasting in ad hoc systems

- a shared radio channel for all receivers
- time division
- unpredictable what is to be sent to to whom
- the receivers may come and go

Targets

- robustness
- no a priori knowledge necessary
- easy adjustment
- energy efficiency

Remote management of a sensor network
application scenario

Kik, Gebala, Kutyłowski

Wireless sensor network

- sensors in inaccessible places
- battery operated, exchanging batteries very problematic
- sensor field management via a broadcasting channel
- event driven activities
- frequently: the sensors have to sleep almost all the time

Design goals

- the sensor field should work as long as possible
- most energy are used for keeping antenna active
- **switch off the antenna whenever possible**

Ad hoc network for security/ disaster recovery
application scenario

Technical issues

- mobility of receivers
- information critical - if a receiver is in the broadcasting range, **it must not skip any message designated to it**
- energy should be saved – problem of exhausting batteries
- emergency situations are non typical - no a priori knowledge and no a priori set up

targets

- the receiver can emerge at any moment
- no message lost, but the antennas deactivated whenever almost all the time
- overhead due to message headers should be relatively small

Sharing frequencies
application scenario

Broadcasting services and local communication

creating local subnetworks:

- take advantage of signal propagation decay: a weak but close sender can be better heard as remote strong station
- local network may reuse the broadcast frequency
- local subnetwork sending locally a weak signal on the broadcast frequency – whenever not receiving data from the broadcaster

Targets

- good time synchronization despite unpredictable a priori broadcast destinations

Naïve solution

Simple inefficient solution

- broadcast cycle consisting of a fixed number of fixed size slots
- each slot starts with destination address/key
- the messages sent in the order corresponding to the keys

slot_1: key_1 | data_1
slot_2: key_2 | data_2
...
slot_k: key_k | data_k

Naïve solution
disadvantaged

slot_1: key_1 | data_1
slot_2: key_2 | data_2
...
slot_k: key_k | data_k

Problems

- a receiver has to listen all the time, until the key becomes higher
- the worst case: listening the whole cycle
- the messages to one receiver comes in a contiguous interval
  being late or burst error $\Rightarrow$ the whole transmission lost

RBO
Broadcasting with Reversed Binary Ordering

Bit reversal

Bit reversal

If t is a k-bit number, then $\mathrm{rev}_k(t)$ is the k-bit number with the same representation, but written in a reverse order.

e.g.

$\mathrm{rev}_5(00001) = 10000$
$\mathrm{rev}_5(00010) = 01000$
$\mathrm{rev}_5(11000) = 00011$

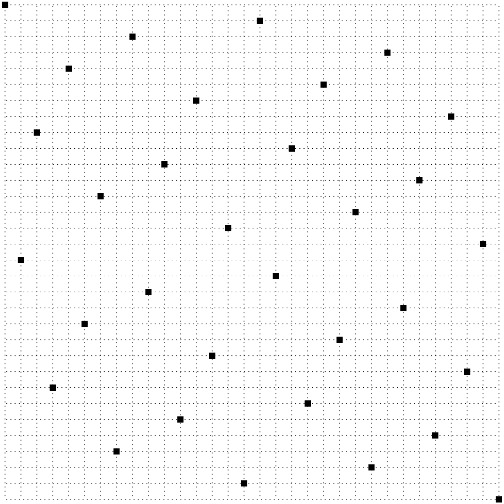
Bit Reversal Scheduling

Scheduling

1. keys after sorting: $a_1, \ldots, a_k$
2. **the message with the key a_i is sent at the slot**

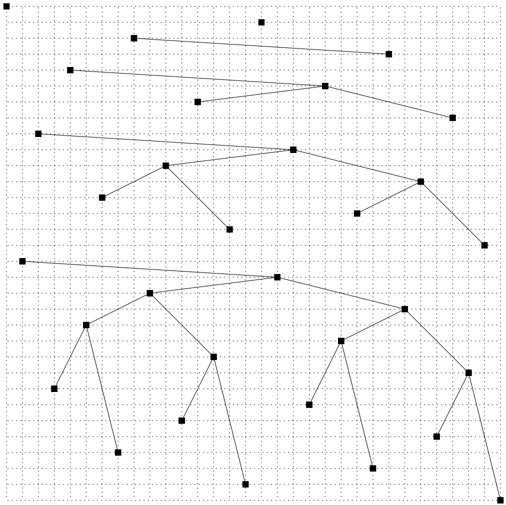
$$\mathrm{rev}_k(i)$$

RBO Scheduling



y axis: time, *x* axis: position *i* in the sorted sequence

RBO Scheduling



y axis: time, *x* axis: position *i* in the sorted sequence

RBO -main properties

Data model

- a message to be sent in one slot consists of
  - a key
  - data
- a receiver is interested in messages with keys from some **interval**
- the intervals of different receivers may overlap

RBO -main properties

Data model

- a message to be sent in one slot consists of
  - a key
  - data
- a receiver is interested in messages with keys from some **interval**
- the intervals of different receivers may overlap

- intervals enable to send longer data with fixed size slots
- intervals may be related to location of a receiver or its function
- a receiver may be interested in more than one interval

Receiver algorithm

Challenge

- the keys for the messages sent during a cycle are unknown for the receiver
- based on the messages received so far, the receiver has to determine which slots to skip

Idea

- a receiver interested in interval J holds
  - a lower bound lb
  - an upper bound ub

  for indexes i such that a_i holds the key from J
- after slot is received and the key is not from J, then lb or ub may be updated

Receiver algorithm

- in time-slot t, if lb $\leq \mathrm{rev}_k(t) \leq$ ub, then the receiver activates the antenna and gets a message with key $\kappa = \kappa_{\mathrm{rev}_k(t)}$.
- Updating lb and ub: according to the value of κ:
  - Case 1 if $\kappa < \kappa'$, then lb $:= \mathrm{rev}_k(t) + 1$,
  - Case 2 if $\kappa'' < \kappa$, then ub $:= \mathrm{rev}_k(t) - 1$,
  - Case 3 if $\kappa' \leq \kappa \leq \kappa''$, then proper message received
- If after the update lb $>$ ub, then there is no message with a key from J in the current broadcast cycle and the receiver switches off its antenna.

Correctness

No message from the range is skipped

follows immediately by the construction

Computational complexity

- simple and short program computing bit reversal
- trial implementation on TinyOS works fine

efficient Java implementation available from Marcin Kik

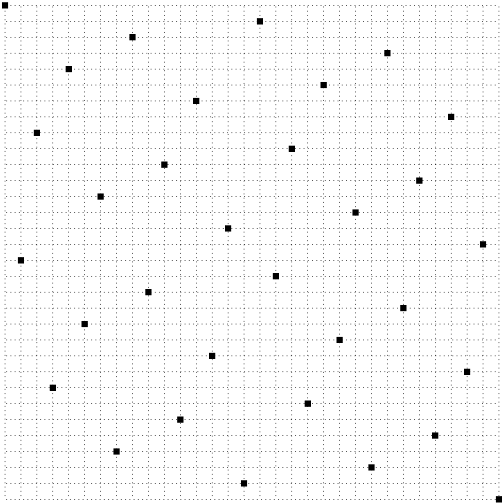
Efficiency

what is the number of updates of lb and ub small?

Remarkable properties of RBO

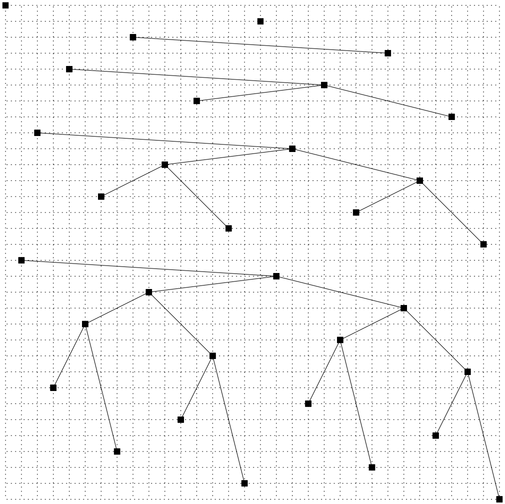
1. the ordering of positions sent is that the new points come always almost in the middle between the closest positions already sent
2. self similarity: if the receiver starts not at the beginning of the broadcast cycle, then almost the same structure can be observed

RBO - Binary search structure



y axis: time, *x* axis: position *i* in the sorted sequence

RBO Scheduling



y axis: time, *x* axis: position *i* in the sorted sequence

Analysis

Problem

Energy efficiency

- **What is the number of time slots that the receiver listens to the channel, but the key is outside the target interval?**
  (it is used merely for updating lb *and* ub*)*
- the number of these time slots is called **extra energy**

Lower bound for extra energy

Lower bound

- There are examples showing that for a cycle consisting of 2^k slots extra energy is $2k - 1$.
- intuitively: k time slots might be necessary to find the starting position of keys from J in the sorted sequence of keys.
- Analogously, $\log_2(2^k) = k$ steps might be needed to find the upper part.

Upper bound

Problems

- this is NOT exactly the binary search tree,
- one can show some cases when there is no “halving effect”
- what if the receiver starts not at the beginning of the cycle?
  – the *binary tree structure* gets complicated

Upper bound

Problems

- this is NOT exactly the binary search tree,
- one can show some cases when there is no “halving effect”
- what if the receiver starts not at the beginning of the cycle?
  – the *binary tree structure* gets complicated

- It is relatively easy to show that the extra energy is something like $4k$ for cycle of length 2^k.
- Trial executions all have shown a much better behavior. So what is the real energy cost (extra energy)?

Main Result

Main Theorem

The extra energy of RBO protocol is at most $2k + 3$ for any input configuration.

Remarks about the proof

1. look into proceedings:
   - the proof is extremely long
   - one of the difficulties is the extremely large number of different notions necessary to carry out the proof
   - the proof has been remarkably simplified compared to the first version

   ... but still it requires a big effort to follow it.

Remarks about the proof

1. look into proceedings:
   - the proof is extremely long
   - one of the difficulties is the extremely large number of different notions necessary to carry out the proof
   - the proof has been remarkably simplified compared to the first version

   ... but still it requires a big effort to follow it.
2. What is the minimal size/ combinatorial complexity of the proof for (nice) behavior of this simple protocol?

Left energy versus right energy

Hope

One may expect that there is some symmetry and it suffices to estimate the number of updates of lb or ub to get the result.

Left energy versus right energy

Hope

One may expect that there is some symmetry and it suffices to estimate the number of updates of lb or ub to get the result.

The hope is wrong

Estimating the number of cases of updating ub (*right energy*) is much more complicated than the case of updating lb (*left energy*).

There are many very subtle effects.

However it works!

You do not have to recheck the proof to take advantage of good properties of the RBO scheduling

Extensions

Fault model - random errors

Random errors

- it may happen that due to propagation problems the receiver may not read some slots
- this occurs in particular when the energy level of the signal is low

Fault model - random errors

Random errors

- it may happen that due to propagation problems the receiver may not read some slots
- this occurs in particular when the energy level of the signal is low

Behavior of RBO for random errors

- the receivers might be forced to spend more extra energy
- corresponds roughly to a binary search tree where we do not know the outcome on some nodes

Fault model - burst errors

burst errors

- due to some external sources during a short period of time all time slots are scrambled
- this occurs in particular due to shortcuts (trains, trams, ...), vehicles passing by and reflecting the waves, ... or some local problems of the receiver

Fault model - burst errors

burst errors

- due to some external sources during a short period of time all time slots are scrambled
- this occurs in particular due to shortcuts (trains, trams, ...), vehicles passing by and reflecting the waves, . . . or some local problems of the receiver

Behavior of RBO for burst errors

- bit reversal converts local error concentrated on an interval to single errors dispersed evenly over the whole interval

Streaming multiple video channels

one more application

How to organize assignment of time slots for video channels?

the scheme should be

- simple
- flexible (any division of time available to different video channels)
- relatively tolerant to burst errors

Streaming multiple video channels

one more application

How to organize assignment of time slots for video channels?

the scheme should be

- simple
- flexible (any division of time available to different video channels)
- relatively tolerant to burst errors

RBO solution

- assign each channel an interval, e.g. $[i, i+1, \dots, i+\Delta]$
- send slot i at time $\mathrm{rev}(i)$
- thereby the time channels are evenly dispersed
  - good because of burst errors
  - good for weak receiver devices - giving them time between consecutive frames

Thanks for your attention!

Contact data

1. Miroslaw.Kutylowski@pwr.wroc.pl
2. http://kutylowski.im.pwr.wroc.pl
3. +48 71 3202109, +48 71 3202105
   fax: +48 71 3202105