Ranking and Sorting in Unreliable Single Hop Radio Network

Marcin Kik `Marcin.Kik@pwr.wroc.pl`
INSTITUTE OF MATHEMATICS AND COMPUTER SCIENCE, Wrocław University of Technology, Wrocław, Poland (`www.im.pwr.wroc.pl`)

Wrocław University of Technology

Introduction

Model of computation:

- *radio network* (set of stations communicating by radio messages)
- *single hop* (all stations are within the range of each message)
- *synchronized* (time is divided into slots)
- *single channel* (in one time slot only one message can be broadcast)
- if station listens then *probability of successful reception* is p
  (In *reliable* network $p = 1$. In *unreliable* network $p < 1$.)
- during each time slot any station can be either:
  - *idle* (using no energy), or
  - *broadcasting* or *listening* (using one unit of energy)

Complexity measures:

- *energetic cost* – **maximum** over all stations of the energy used. (Stations are powered by batteries.)
- *time* – number of time slots used by the computation.

Remark: By "$\lg$" we mean "$\log_2$".

Ranking

(We assume that $n = 2^k$.) A sorted sequence of keys: $b_0, \ldots, b_{n-1}$ permuted by a fixed permutation π_k is transmitted periodically. (In time slot t the key b_x is broadcast, where $x = \pi_k^{-1}(t \bmod n)$.)

Station a containing $\mathsf{key}[a]$ has to compute the *rank* of $\mathsf{key}[a]$ in $b_0, \ldots, b_{n-1}$ (or its approximation). (*Rank* of $\mathsf{key}[a]$ is $|\{b_i \mid b_i \le \mathsf{key}[a]\}|$.) Station a contains variables $\mathsf{minR}[a]$ and $\mathsf{maxR}[a]$ that are the lower and upper bound on the rank, respectively. Initially $\mathsf{minR}[a] = 0$ and $\mathsf{maxR}[a] = n$. a can start its computation in arbitrary time slot. In time slot t, the station a does:

> Let $x = \pi_k^{-1}(t \bmod n)$. If $\mathsf{minR}[a] \le x < \mathsf{maxR}[a]$ then a listens.
> If a received the message (i.e. b_x), then a does:
> **if** $\mathsf{key}[a] < b_x$ **then** $\mathsf{maxR}[a] \leftarrow x$ **else** $\mathsf{minR}[a] \leftarrow x + 1$

Note that the rank of $\mathsf{key}[a]$ is always in the interval $[\mathsf{minR}[a], \mathsf{maxR}[a]]$.

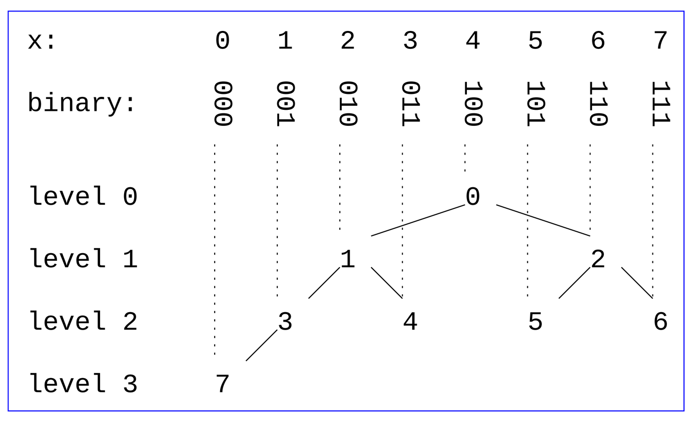
Lemma 1. *Let c be a positive integer. After $c \cdot n$ time slots $\mathsf{minR}[a] = \mathsf{maxR}[a]$ (i.e. the exact rank is computed) with probability at least $1 - 2 \cdot (1-p)^c$.*

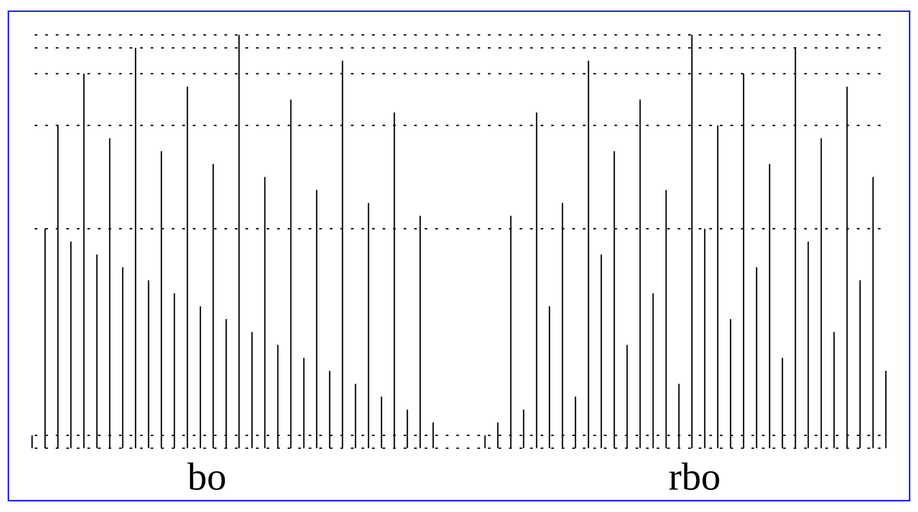
(Station a has c chances of receiving the direct neighbors of $\mathsf{key}[a]$.)

Lemma 2. *The expected value of $\Delta = \mathsf{maxR}[a] - \mathsf{minR}[a]$ after n time slots is not greater than $2/p - 2$.*

(Let r be the rank of $\mathsf{key}[a]$ in $b_0, \ldots, b_{n-1}$. Then $\mathsf{maxR}[a] - r + 1$ is bounded by a random variable X with geometric distribution: $Pr(X = m) = (1-p)^{m-1} \cdot p$ and $E[X] = 1/p$. Thus $E[\mathsf{maxR}[a] - r] = 1/p - 1$ (and, by symmetry, $E[r - \mathsf{minR}[a]] = 1/p - 1$.)

Bisection ordering permutation (bo_k): Illustrated by the figure to the right: Each x is connected by vertical dotted line with the node labelled $\mathsf{bo}_k(x)$. (Note that level of the tree and position within the level can be easily read from binary representation of x. Hence, bo_k is "easily computable" function.) If the network is reliable and $\pi_k = \mathsf{bo}_k$ and a starts in time slot $t = 0$ then the energy used by a is at most k. However, if a can start in arbitrary time slot, then the energy used by a may be as large as $n/2$.





Recursive bisection ordering permutation (rbo_k): First we permute all the elements by bo_k and then we permute each level of the binary tree of bo_k (except the first and the last one, which are singletons) by recursive bisection ordering. (rbo is also "easily computable".)

The permutation π_k can be imagined as a set of parallel vertical blades cutting of parts of the horizontal interval containing the rank of $\mathsf{key}[a]$ as it falls downwards. The figure to the left illustrates the difference between bo and rbo.

Lemma 3. *Let π_k be bo_k or rbo_k. Let the station a start in time slot 0. The expected energy used by a during the first $n = 2^k$ time slots is at most $1 + k \cdot (2/p - 1)$.*

(This follows from Lemma 2 applied to each level of the binary tree of bo_k.)

Ranking with rbo in a reliable network

Theorem 1. *If $\pi_k = \mathsf{rbo}_k$ and $p = 1$ and station a starts in arbitrary time slot, then the station a listens at most $4 \lg n$ times before it learns its rank.*

(The proof is rather technical: We bound by 2 or by 4 the energy used within levels of the bisection binary trees formed on the same or different levels of recursion within rbo.)

Remarks:

- In the simulations, the energy used by a was never greater than $2 \lg n$. (The estimation $4 \lg n$ does not seem to be very precise.)
- The experimental results suggest that rbo is also energetically efficient in *unreliable* networks when a starts in arbitrary time slot.

Sorting

We assume that $\lg n$ is positive integer. The network consists of n stations: $s_0, \ldots, s_{n-1}$. Each s_i contains single key: $\mathsf{key}[s_i]$. Each s_i also contains the following variables:

- $\mathsf{idx}_0[s_i], \ldots, \mathsf{idx}_{\lg n}[s_i]$
- $\mathsf{minR}_0[s_i], \ldots, \mathsf{minR}_{\lg n - 1}[s_i]$
- $\mathsf{maxR}_0[s_i], \ldots, \mathsf{maxR}_{\lg n - 1}[s_i]$

procedure init
Each s_i does (in parallel):
begin
  $\mathsf{idx}_0[s_i] \leftarrow 0$;
  for $k \leftarrow 1$ **to** $\lg n$ **do**
    $\mathsf{idx}_k[s_i] \leftarrow NIL$;
  for $k \leftarrow 0$ **to** $\lg n - 1$ **do**
    $\mathsf{minR}_k[s_i] \leftarrow 0$;
    $\mathsf{maxR}_k[s_i] \leftarrow 2^k$;
end

These variables are initialized by the procedure init. On *level* k we partition the sequence $s_0, \ldots, s_{n-1}$ into *blocks* of length 2^k. $\mathsf{idx}_k[s_i]$ should become the index of $\mathsf{key}[s_i]$ in the sorted sequence of the keys from its block on level k. The ultimate *goal of sorting* is to compute in each $\mathsf{idx}_{\lg n}[s_i]$ the index of $\mathsf{key}[s_i]$ in the sorted sequence of all the keys.

procedure $\mathsf{rank}(k, l, d, \pi_k)$
for $0 \le i < 2^k$:
- let a_i denote $s_{l \cdot 2^{k+1} + d \cdot 2^k + i}$, and
- let b_i denote $s_{l \cdot 2^{k+1} + (1-d) \cdot 2^k + i}$.

for *time slot* $t \leftarrow 0$ **to** $2^k - 1$ **do**
  the (at most one) b_j with $\pi_k(\mathsf{idx}_k[b_j]) = t$
  broadcasts $key = \mathsf{key}[b_j]$;
  let $x = \pi_k^{-1}(t)$;
  each a_i with $\mathsf{minR}[a_i] \le x < \mathsf{maxR}_k[a_i]$ does:
  begin
    a_i listens;
    if a_i *received* key **then**
      (* comparison for *stable* ranking *)
      if *($d = 0$ and $\mathsf{key}[a_i] \le key$) or*
      ($d = 1$ and $\mathsf{key}[a_i] < key$) **then**
        $\mathsf{maxR}_k[a_i] \leftarrow x$;
      else
        $\mathsf{minR}_k[a_i] \leftarrow x + 1$;
      (* cascading computation of indexes *)
      $k' \leftarrow k$;
      while $k' < \lg n$ *and* $\mathsf{idx}_{k'}[a_i] \ne NIL$ *and*
      $\mathsf{minR}_{k'}[a_i] = \mathsf{maxR}_{k'}[a_i]$ **do**
        $\mathsf{idx}_{k'+1}[a_i] \leftarrow \mathsf{idx}_{k'}[a_i] + \mathsf{minR}_{k'}[a_i]$;
        $k' \leftarrow k' + 1$;
  end

$\mathsf{rank}(k, l, d, \pi_k)$, for $d \in \{0, 1\}$, is used for merging the lth pair of sorted blocks on level k into one block on level $k+1$. For $d = 0$ (respectively $d = 1$), each station from the left (respectively right) block attempts to find the rank of its key in the other block (using one round of the ranking algorithm). The sum of the index in its own block and the rank in the other block is the index in the merged block. To ensure that the sorting is *stable* and that the computed indexes are unique, whenever we compare two equal keys, the one from the right block is considered to be the greater.
(*Cascading* computation of indexes could be useful if some ranking on lower level follows some some ranking on higher level. In $\mathsf{sorting}_{q'}$ (defined below) it is enough to compute only the index on the next level.)

$\mathsf{levelRanking}(k, \pi_k)$ is used for merging all the pairs of blocks on level k. $\mathsf{levelRanking}(k, \pi_k)$ attempts to increase the number of computed indexes on level $k+1$. The time of each $\mathsf{levelRanking}$ is n and sorting algorithms can be defined as sequences of $\mathsf{levelRankings}$.

procedure $\mathsf{levelRanking}(k, \pi_k)$
for $l \leftarrow 0$ **to** $n/(2^{k+1}) - 1$ **do**
  $\mathsf{rank}(k, l, 1, \pi_k)$;
  $\mathsf{rank}(k, l, 0, \pi_k)$;

For $n > 0$ and $0 < q, q' < 1$, let $c(q, q', n) = \lceil \log_{1/q} \left(\frac{2n \lg n}{q'} \right) \rceil = \lceil (1 + \lg n + \lg\lg n + \lg(1/q'))/\lg(1/q) \rceil$.

procedure $\mathsf{sorting}_{q'}$
init;
Let $q = 1 - p$, where p is probability of successful reception;
for $k \leftarrow 0$ **to** $\lg n - 1$ **do**
  repeat $c(q, q', n)$ times $\mathsf{levelRanking}(k, \mathsf{rbo}_k)$;

$\mathsf{sorting}_{q'}$ is the simplest application of $\mathsf{levelRankings}$ that guaranties successful sorting with probability $1 - q'$. The low energetic cost follows from $\pi_k = \mathsf{rbo}_k$. (We could use bo_k as well.)

Theorem 2. *For $0 < q' < 1$, the procedure $\mathsf{sorting}_{q'}$ sorts any input sequence with probability greater or equal $1 - q'$.*

(This follows from Lemma 1 and from the definition of $c(q, q', n)$.)

Theorem 3. *For any input, the expected energy used for listening by any single station in $\mathsf{sorting}_{q'}$ is at most $\lg n \cdot (1 + (c-1)(2/p - 2) + (2/p - 1)(\lg n - 1)/2) + c(n-1)q'$, where $c = c(q, q', n)$. The energy used for broadcasting by any single station is $c \lg n$. Time of $\mathsf{sorting}_{q'}$ is $cn \lg n$.*

(If the computation is successful, then all indexes on level k are computed before the first $\mathsf{levelRanking}(k, \mathsf{rbo}_k)$. Thus we can use Lemma 3 to bound the energy used by the first $\mathsf{levelRanking}(k, \mathsf{rbo}_k)$. By Lemma 2 we have the bounds for the remaining $\mathsf{levelRankings}$ on this level. If the computation is unsuccessful (this happens with probability $\le q'$) then we use the worst case bound: $c(n-1)$.)

There is a tradeoff between energy and reliability. As one example, we have the following corollary:

Corollary. *Algorithm $\mathsf{sorting}_{1/n}$ sorts any input with probability at least $1 - \frac{1}{n}$ in time $O(n \lg^2 n)$ and, for each station s, the expected energy used by s is $O(\lg^2 n)$.*

Bibliographic coordinates of the conference paper:

Marcin Kik: *Ranking and Sorting in Unreliable Single Hop Radio Network.* In David Coudert, David Simplot-Ryl, and Ivan Stojmenovic (Eds.): Ad-hoc, Mobile and Wireless Networks 7th International Conference, ADHOC-NOW 2008, LNCS 5198, pp. 333-344, 2008. Springer-Verlag Berlin Heidelberg 2008.

This work has been supported by the ICT Programme of the European Union under contract number FP7-215270 (FRONTS)