Merging and Merge-sort in a Single Hop Radio Network

Marcin Kik

Wrocław University of Technology

Poland

SOFSEM 2006

Model of computation

Radio network:

- n stations communicating by radio messages

Model of computation

Radio network:

- n stations communicating by radio messages
- single-hop

Model of computation

Radio network:

- n stations communicating by radio messages
- single-hop
- synchronized (time is slotted)

Model of computation

Radio network:

- n stations communicating by radio messages
- single-hop
- synchronized (time is slotted)
- in a one slot at most one message (in single channel)

Model of computation

Radio network:

- n stations communicating by radio messages
- single-hop
- synchronized (time is slotted)
- in a one slot at most one message (in single channel)
- single message contains $O(\lg n)$ bits

Model of computation

Radio network:

- n stations communicating by radio messages
- single-hop
- synchronized (time is slotted)
- in a one slot at most one message (in single channel)
- single message contains $O(\lg n)$ bits
- broadcasting/listening to a single message requires unit of energetic cost

Model of computation

Radio network:

- n stations communicating by radio messages
- single-hop
- synchronized (time is slotted)
- in a one slot at most one message (in single channel)
- single message contains $O(\lg n)$ bits
- broadcasting/listening to a single message requires unit of energetic cost
- memory of single station limited to constant number of words of $O(\lg n)$ bits

Model of computation

Radio network:

- n stations communicating by radio messages
- single-hop
- synchronized (time is slotted)
- in a one slot at most one message (in single channel)
- single message contains $O(\lg n)$ bits
- broadcasting/listening to a single message requires unit of energetic cost
- memory of single station limited to constant number of words of $O(\lg n)$ bits

Energetic cost of the algorithm is the maximal energy dissipated by a single station.

Statement of the sorting problem

- We have n enumerated stations $a_1 \ldots a_n$.

Statement of the sorting problem

- We have n enumerated stations $a_1 \ldots a_n$.
- Each a_i stores a single key in its local variable $key[a_i]$.

Statement of the sorting problem

- We have n enumerated stations $a_1 \ldots a_n$.
- Each a_i stores a single key in its local variable $key[a_i]$.
- We want to rearrange the keys between the stations so that finally the sequence $key[a_1], \ldots, key[a_n]$ is sorted.

Statement of the sorting problem

- We have n enumerated stations $a_1 \ldots a_n$.
- Each a_i stores a single key in its local variable $key[a_i]$.
- We want to rearrange the keys between the stations so that finally the sequence $key[a_1], \ldots, key[a_n]$ is sorted.

We assume that single key can be sent in a single message.

Sorting on single hop:

- From comparator networks: (each comparator simulated in two slots)
  - AKS energy: $\Theta(\lg n)$, time: $\Theta(n \lg n)$
  - Batcher energy: $\approx \lg^2 n$, time: $\Theta(n \lg^2 n)$

Sorting on single hop:

- From comparator networks: (each comparator simulated in two slots)
  - AKS energy: $\Theta(\lg n)$, time: $\Theta(n \lg n)$
  - Batcher energy: $\approx \lg^2 n$, time: $\Theta(n \lg^2 n)$
- Singh, Prasanna – energy: $\Theta(\lg n)$, time: $\Theta(n \lg n)$
  (*Quick-sort with energetically balanced selection*)

Sorting on single hop:

- From comparator networks: (each comparator simulated in two slots)
  - AKS energy: $\Theta(\lg n)$, time: $\Theta(n \lg n)$
  - Batcher energy: $\approx \lg^2 n$, time: $\Theta(n \lg^2 n)$
- Singh, Prasanna – energy: $\Theta(\lg n)$, time: $\Theta(n \lg n)$
  (*Quick-sort with energetically balanced selection*)

In this paper:

- practical Merge-sort
  energy: $\frac{1}{2} \lg^2 n + \frac{7}{2} \lg n$,
  time: $2n \cdot \lg n$.
  (*Merging two sequences of length* m
  energy: $\lceil \lg(m+1) \rceil + 3$*, time* $4m$*.*)

Sorting on single hop:

- From comparator networks: (each comparator simulated in two slots)
  - AKS energy: $\Theta(\lg n)$, time: $\Theta(n \lg n)$
  - Batcher energy: $\approx \lg^2 n$, time: $\Theta(n \lg^2 n)$
- Singh, Prasanna – energy: $\Theta(\lg n)$, time: $\Theta(n \lg n)$
  (*Quick-sort with energetically balanced selection*)

In this paper:

- practical Merge-sort
  energy: $\frac{1}{2} \lg^2 n + \frac{7}{2} \lg n$,
  time: $2n \cdot \lg n$.
  (*Merging two sequences of length* m
  energy: $\lceil \lg(m+1) \rceil + 3$*, time* $4m$*.*)
- Merge-sort – energy: $O(\lg n \lg^* n)$, time: $O(n \lg n \lg^* n)$
  (*Merging – energy:* $O(\lg^* m)$*, time* $O(m \lg^* m)$).

Sorting on single hop:

- From comparator networks: (each comparator simulated in two slots)
  - AKS energy: $\Theta(\lg n)$, time: $\Theta(n \lg n)$
  - Batcher energy: $\approx \lg^2 n$, time: $\Theta(n \lg^2 n)$
- Singh, Prasanna – energy: $\Theta(\lg n)$, time: $\Theta(n \lg n)$
  (*Quick-sort with energetically balanced selection*)

In this paper:

- practical Merge-sort
  energy: $\frac{1}{2}\lg^2 n + \frac{7}{2}\lg n$ (improved: $\frac{1}{2}\lg^2 n + \frac{3}{2}\lg n + 2$) ,
  time: $2n \cdot \lg n$ (improved: $n \lg n + n$) .
  (*Merging two sequences of length* m
  energy: $\lceil \lg(m+1) \rceil + 3$, *time* $4m$.)
- Merge-sort – energy: $O(\lg n \lg^* n)$, time: $O(n \lg n \lg^* n)$
  (*Merging – energy:* $O(\lg^* m)$, *time* $O(m \lg^* m)$).

Merging?

What is the optimal energetic cost of merging?

Merging?

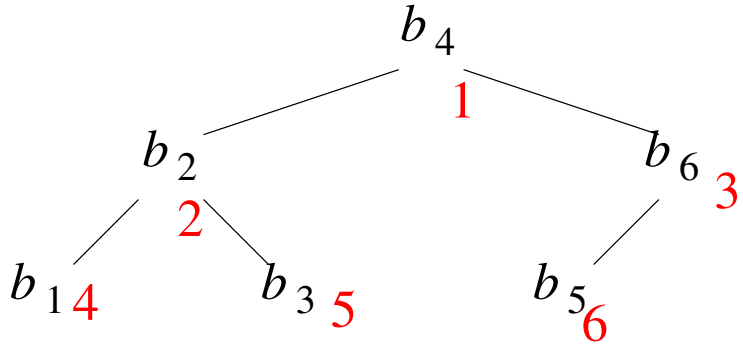
What is the optimal energetic cost of merging?

If we simulate the simple sequential algorithm, then the total energy for broadcasting and listening is $O(n)$, but it is not balanced.

Ranking

Ranking of $a_1 \ldots a_m$ in $b_1 \ldots b_m$:

- $a_1 \ldots a_m$ – sorted
- $b_1 \ldots b_m$ – sorted
  (balanced binary tree T_m)
- all values distinct
- in-order indexes
- "preorder" indexes – slots
- ranks
- timers – slots



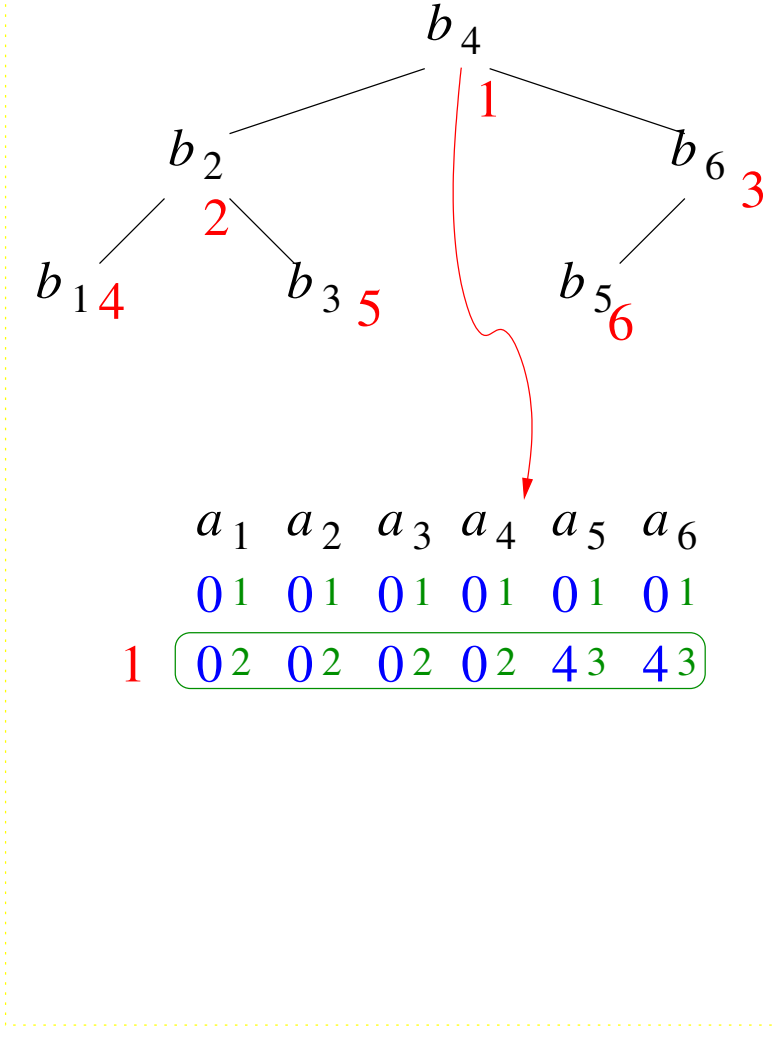
a_1 a_2 a_3 a_4 a_5 a_6

0 1 0 1 0 1 0 1 0 1 0 1

Ranking

Ranking of $a_1 \ldots a_m$ in $b_1 \ldots b_m$:

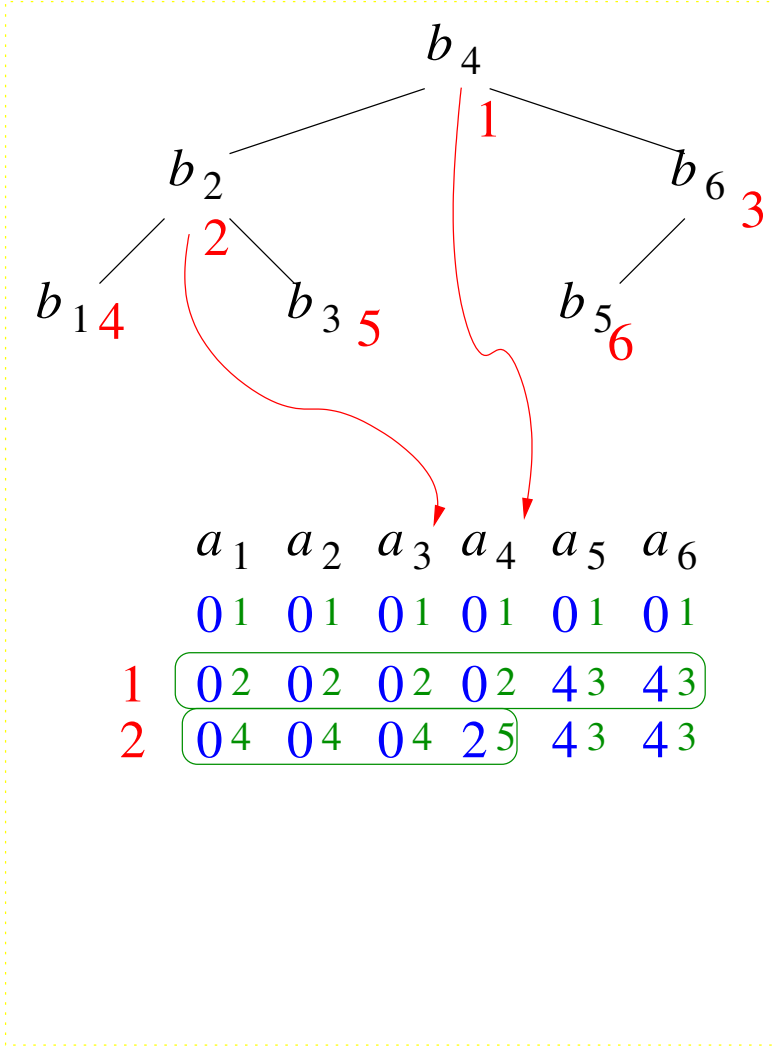
- $a_1 \ldots a_m$ – sorted
- $b_1 \ldots b_m$ – sorted
  (balanced binary tree T_m)
- all values distinct
- in-order indexes
- "preorder" indexes – slots
- ranks
- timers – slots



Ranking

Ranking of $a_1 \ldots a_m$ in $b_1 \ldots b_m$:

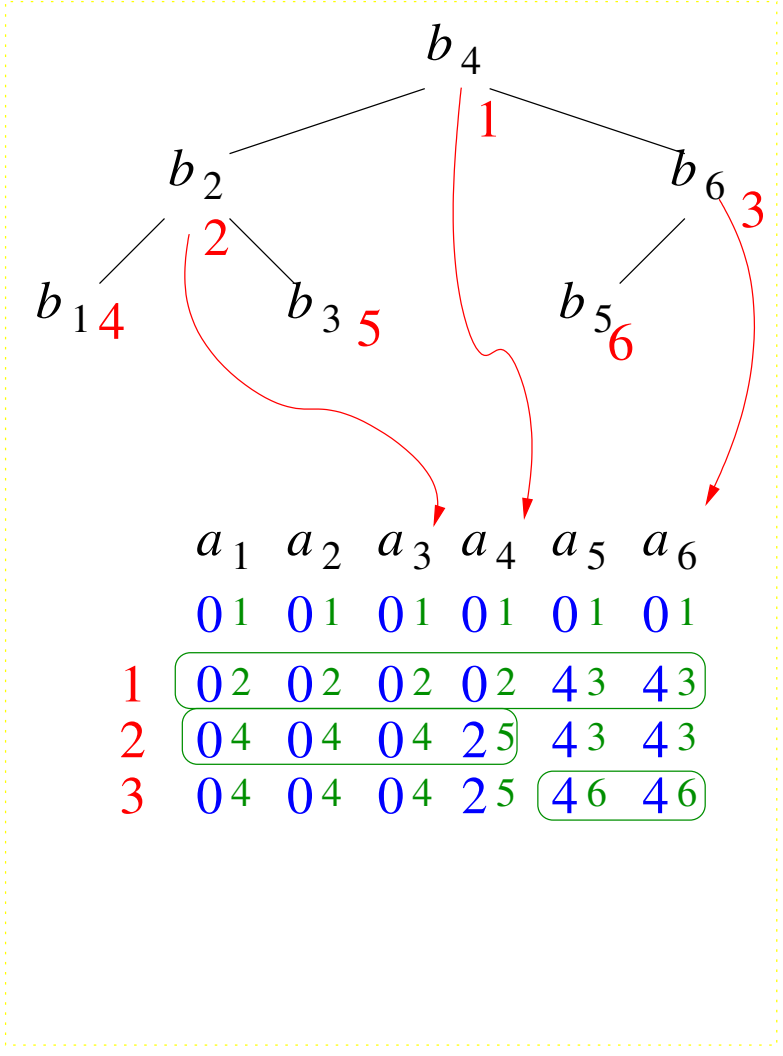
- $a_1 \ldots a_m$ – sorted
- $b_1 \ldots b_m$ – sorted
  (balanced binary tree T_m)
- all values distinct
- in-order indexes
- "preorder" indexes – slots
- ranks
- timers – slots



Ranking

Ranking of $a_1 \ldots a_m$ in $b_1 \ldots b_m$:

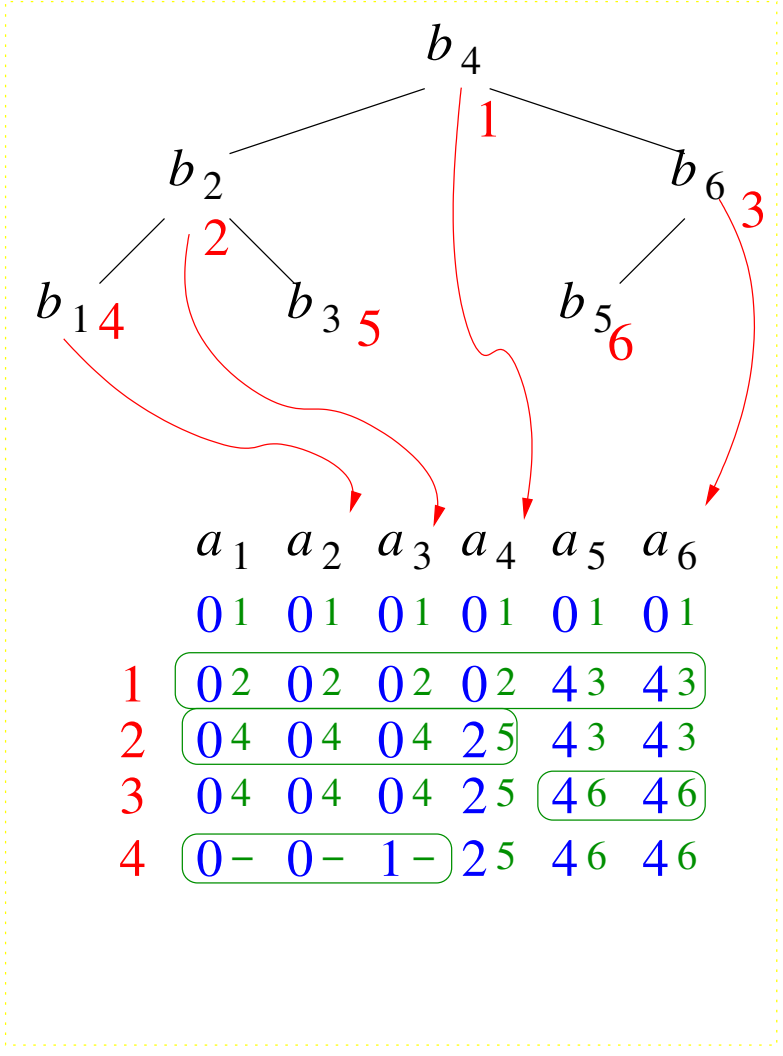
- $a_1 \ldots a_m$ – sorted
- $b_1 \ldots b_m$ – sorted
  (balanced binary tree T_m)
- all values distinct
- in-order indexes
- "preorder" indexes – slots
- ranks
- timers – slots



Ranking

Ranking of $a_1 \ldots a_m$ in $b_1 \ldots b_m$:

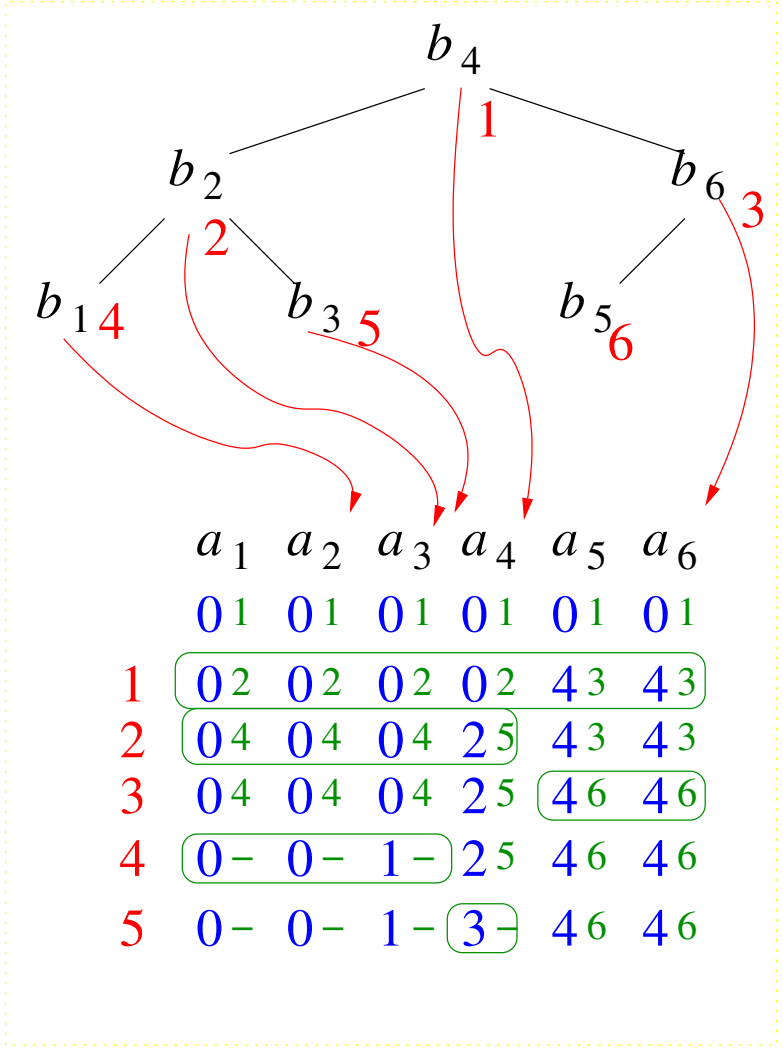
- $a_1 \ldots a_m$ – sorted
- $b_1 \ldots b_m$ – sorted
  (balanced binary tree T_m)
- all values distinct
- in-order indexes
- "preorder" indexes – slots
- ranks
- timers – slots



Ranking

Ranking of $a_1 \ldots a_m$ in $b_1 \ldots b_m$:

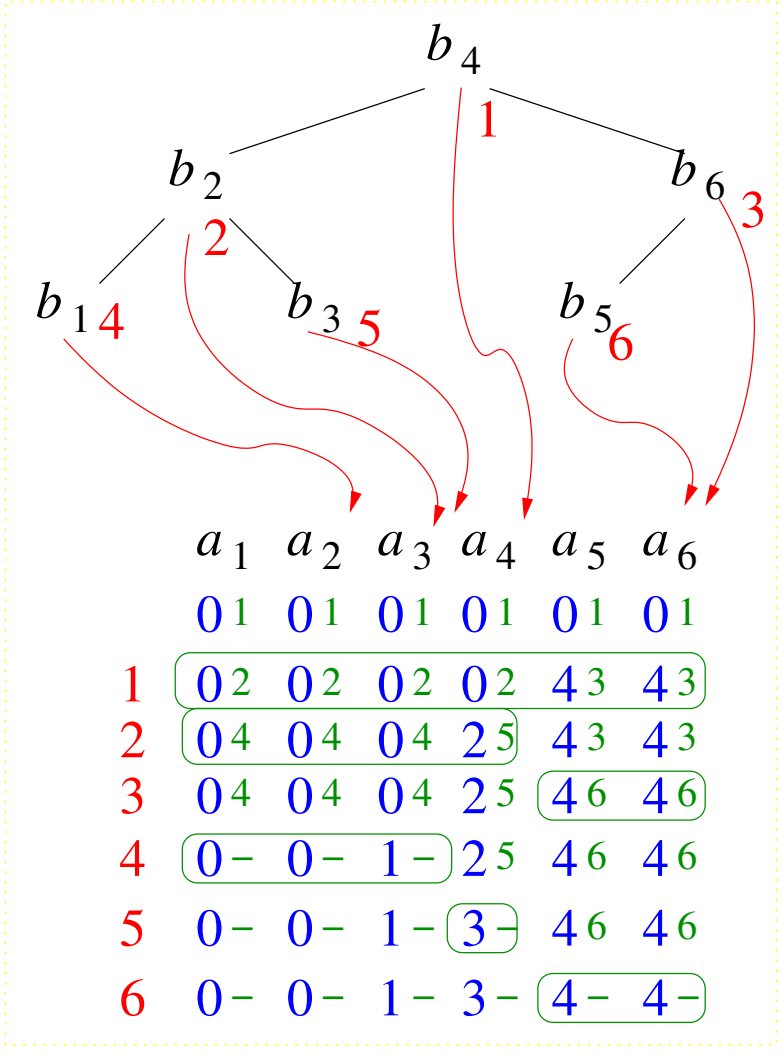
- $a_1 \ldots a_m$ – sorted
- $b_1 \ldots b_m$ – sorted
  (balanced binary tree T_m)
- all values distinct
- in-order indexes
- "preorder" indexes – slots
- ranks
- timers – slots



Ranking

Ranking of $a_1 \ldots a_m$ in $b_1 \ldots b_m$:

- $a_1 \ldots a_m$ – sorted
- $b_1 \ldots b_m$ – sorted
  (balanced binary tree T_m)
- all values distinct
- in-order indexes
- "preorder" indexes – slots
- ranks
- timers – slots



Ranking

procedure Rank($\langle a_1, \ldots, a_m \rangle$, $\langle b_1, \ldots, b_m \rangle$)
begin
  Each a_i does: $timer[a_i] \leftarrow 1$; $rank[a_i] \leftarrow 0$;
  for *time slot* $d \leftarrow 1$ **to** m **do**
    Let x be such that d is preorder index of x.
    b_x broadcasts its key k.
    Each a_j with $timer[a_j] = d$ listens and does:
    if $key[a_j] < k$ **then**
      $timer[a_j] \leftarrow$ preorder index of left child of x
    else
      $timer[a_j] \leftarrow$ preorder index of right child of x
      $rank[a_j] \leftarrow x$;
end

Energetic cost and time

Energy:

- Each b_i broadcasts once.
- Each a_i listens $\lceil \lg(m+1) \rceil$ (height of T_m)times

Time: m slots.

Merging

Merging $a_1 \ldots a_m$ with $b_1 \ldots b_m$:

Merging

Merging $a_1 \ldots a_m$ with $b_1 \ldots b_m$:

- Ranking $a_1 \ldots a_m$ in $b_1 \ldots b_m$

Merging

Merging $a_1 \ldots a_m$ with $b_1 \ldots b_m$:

- Ranking $a_1 \ldots a_m$ in $b_1 \ldots b_m$
- Ranking $b_1 \ldots b_m$ in $a_1 \ldots a_m$

Merging

Merging $a_1 \ldots a_m$ with $b_1 \ldots b_m$:

- Ranking $a_1 \ldots a_m$ in $b_1 \ldots b_m$
- Ranking $b_1 \ldots b_m$ in $a_1 \ldots a_m$
- Each a_i and b_i computes final *index* of its *key* as: $i + ($*its rank in the other sequence*$)$.

Merging

Merging $a_1 \ldots a_m$ with $b_1 \ldots b_m$:

- Ranking $a_1 \ldots a_m$ in $b_1 \ldots b_m$
- Ranking $b_1 \ldots b_m$ in $a_1 \ldots a_m$
- Each a_i and b_i computes final *index* of its *key* as: $i + ($*its rank in the other sequence*$)$.
- `Permutation routing`: In step t, the element with *index*$= t$ broadcasts its *key* to the tth element of the merged sequence

Simple Merge

procedure Merge($\langle a_1, \ldots, a_m \rangle$, $\langle b_1, \ldots, b_m \rangle$)
begin
  Rank($\langle a_1, \ldots, a_m \rangle$,$\langle b_1, \ldots, b_m \rangle$);
  Rank($\langle b_1, \ldots, b_m \rangle$,$\langle a_1, \ldots, a_m \rangle$);
  Each a_i does: $idx[a_i] \leftarrow i + rank[a_i]$;
  Each b_i does: $idx[b_i] \leftarrow i + rank[b_i]$;
  For $1 \leq i \leq m$ let $c_i = a_i$ and $c_{m+i} = b_i$.
  for *time slot* $t \leftarrow 1$ **to** $2m$ **do**
    c_i with $idx[c_i] = t$ broadcasts its key k.
    c_t listens and does: $new[c_t] \leftarrow k$;
  Each c_i does: $key[c_i] \leftarrow new[c_i]$;
end

Merge: Energetic cost

Energy:

- First Rank:
  - each a_i: $\lceil \lg(m+1) \rceil$
  - each b_i: 1
- Second Rank:
  - each b_i: $\lceil \lg(m+1) \rceil$
  - each a_i: 1
- last for loop:
  - each station: 2 (once listens, once broadcasts)
- Totally: energ. cost: $\lceil \lg(m+1) \rceil + 3$

Merge: time

- First Rank: m
- Second Rank: m
- Last for loop: $2m$
- Total time: $4m$

Merge-sort

Obvious application of Merge: (Let $n = 2^k$)

- Energy

$$\begin{aligned} E(n) &= \sum_{l=0}^{k-1} (\lceil \lg(2^l + 1) \rceil + 3) \\ &= \frac{1}{2} \lg^2 n + \frac{7}{2} \lg n \end{aligned}$$

- Time

$$\begin{aligned} T(n) &= \sum_{l=0}^{k-1} n/(2 \cdot 2^l) \cdot (4 \cdot 2^l) \\ &= 2n \lg n \end{aligned}$$

Remark:

In all mergings except the last one:

Instead of the last “for” loop (i.e. `permutation routing`): Each station c_i internally temporarily modifies its global number to the number of the destination of its key and acts as its destination.

Remark:

In all mergings except the last one:

Instead of the last "for" loop (i.e. `permutation routing`): Each station c_i internally temporarily modifies its global number to the number of the destination of its key and acts as its destination.

- Energy

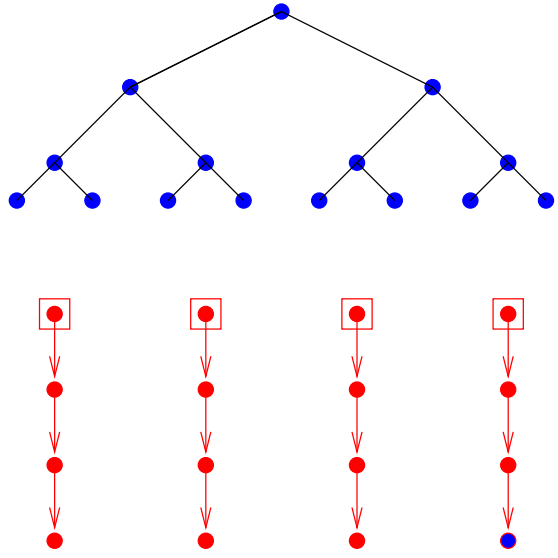
$$E(n) = \frac{1}{2}\lg^2 n + \frac{3}{2}\lg n + 2$$

- Time

$$T(n) = n\lg n + n$$

Balancing of listening energy

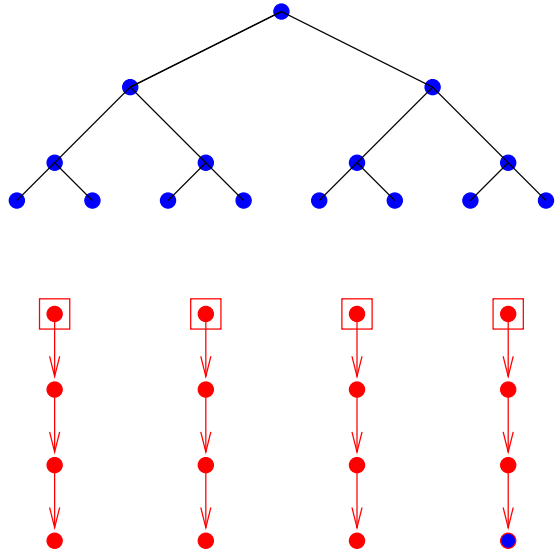
Each *block* of $\lceil \lg(m+1) \rceil$ *listeners* computes the rank of its *leader*. (The message to the next *listener*: *(leader's key, timer, rank)*)



Balancing of listening energy

Each *block* of $\lceil \lg(m+1) \rceil$ *listeners* computes the rank of its *leader*. (The message to the next *listener*: *(leader's key, timer, rank)*)
Energetic cost:

- broadcasting: 1
- listening: 2

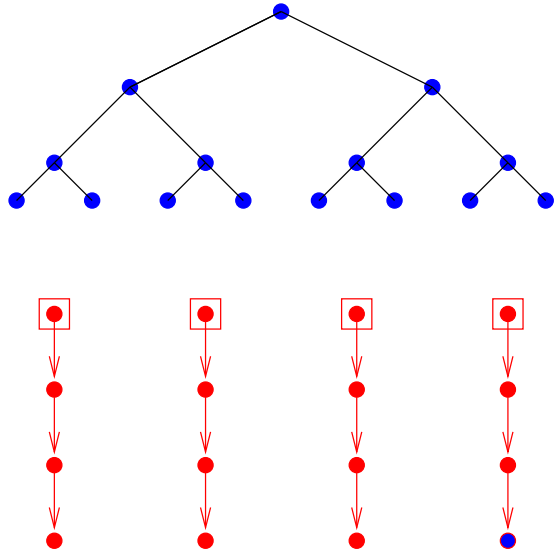


Balancing of listening energy

Each *block* of $\lceil \lg(m+1) \rceil$ *listeners* computes the rank of its *leader*. (The message to the next *listener*: *(leader's key, timer, rank)*)
Energetic cost:

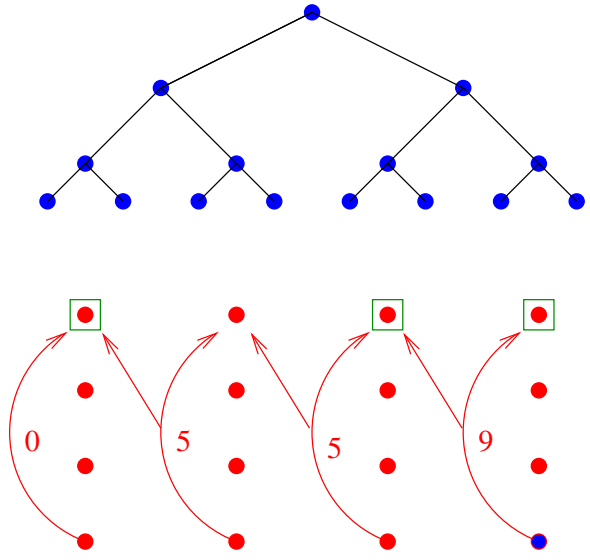
- broadcasting: 1
- listening: 2

Time: $2n$.



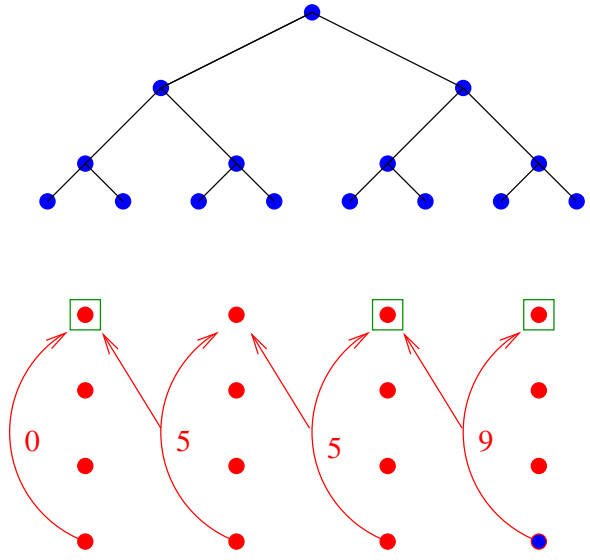
Selecting winners

Each *leader* is informed about its *rank* by the last *listener* in its block and overhears the *rank* of the next *leader*.
For each *rank*, the last leader with this *rank* becomes a *winner*.



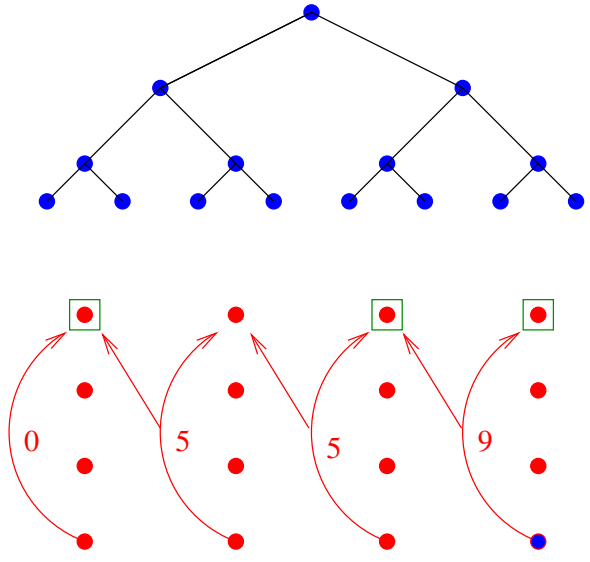
Selecting winners

Each *leader* is informed about its *rank* by the last *listener* in its block and overhears the *rank* of the next *leader*.
For each *rank*, the last leader with this *rank* becomes a *winner*.
Energetic cost: 2. (each *leader* listens to its own and its successor's rank)



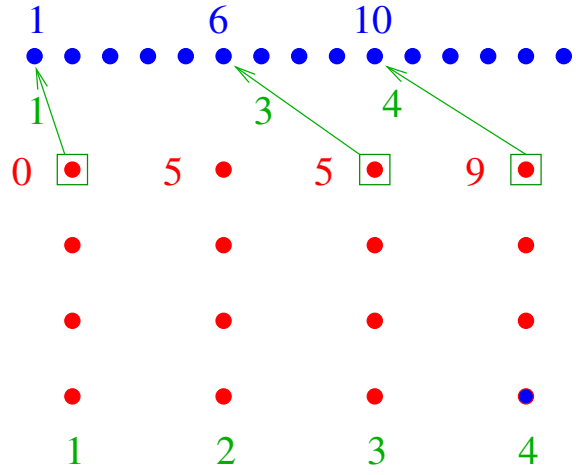
Selecting winners

Each *leader* is informed about its *rank* by the last *listener* in its block and overhears the *rank* of the next *leader*.
For each *rank*, the last leader with this *rank* becomes a *winner*.
Energetic cost: 2. (each *leader* listens to its own and its successor's rank)
Time: $\lceil m/\lceil \lg(m+1) \rceil \rceil$



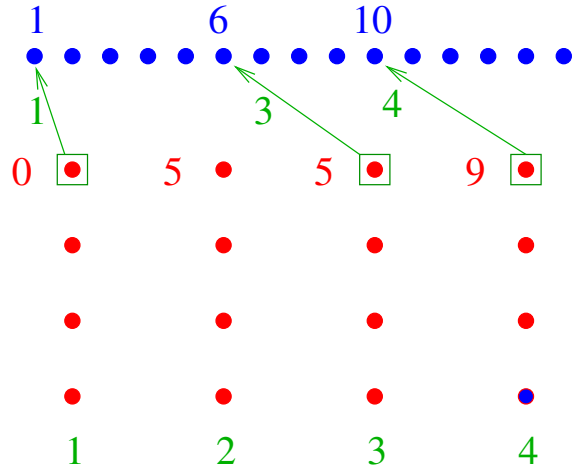
Setting group borders

Each *winner* informs its *successor* b_i in the *other sequence* about its *block number* in the i-th time slot.



Setting group borders

Each *winner* informs its *successor* b_i in the *other sequence* about its *block number* in the i-th time slot.

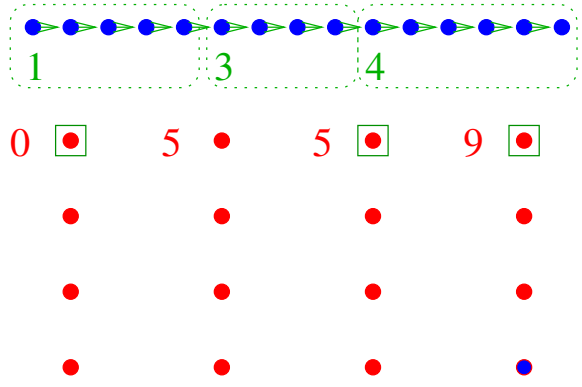


Energetic cost: 1.

Time: m.

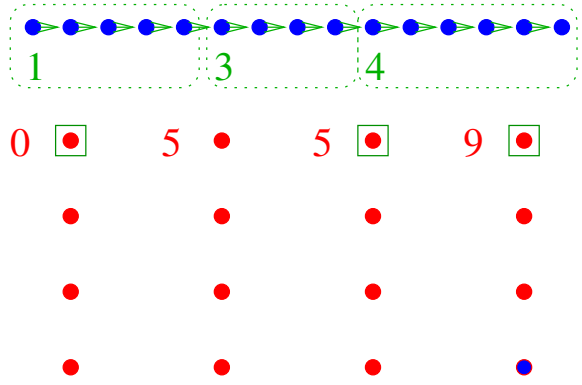
Grouping

Each informed *station* sends its *block number* to its successor.
Successor ignores this if it was earlier informed by some *winner*.



Grouping

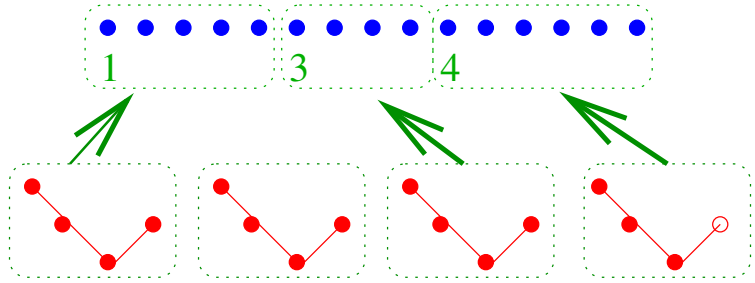
Each informed *station* sends its *block number* to its successor.
Successor ignores this if it was earlier informed by some *winner*.



Energetic cost: 2.
Time: $m - 1$.

Ranking with energy $O(\lg \lg m)$

Each station listens only to the tree with its group number



- Energetic cost: listening: $O(\lg \lg m)$, broadcasting: 1.
- Time: $O(m)$

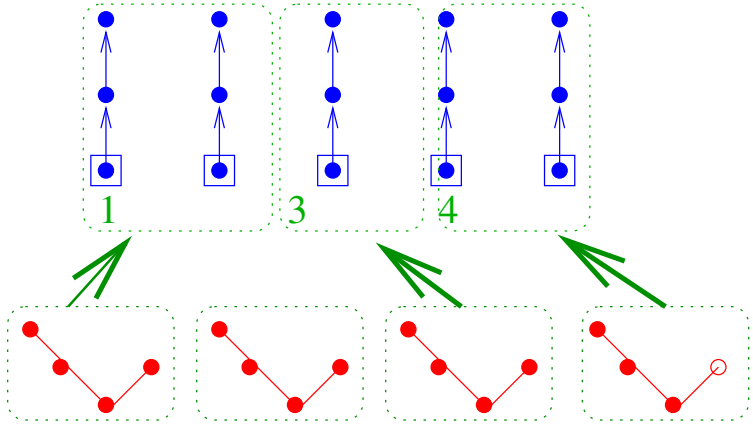
Iterating of regrouping

Instead of all stations computing their own ranks, blocks of $\approx \lg \lg m$ stations compute ranks of their *leaders*.

Now the density of the *leaders* is much higher.

The message to the next slave is:

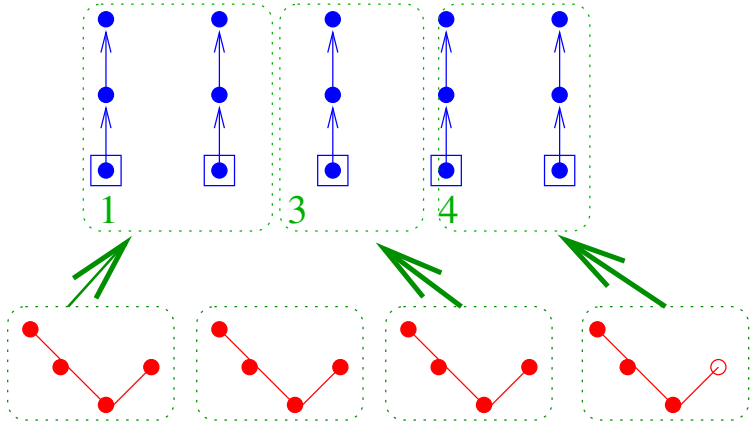
(leader's key, **group***, rank, timer)*



Then select the *winners* that split the other sequence into groups.

Iterating of regrouping

Instead of all stations computing their own ranks, blocks of $\approx \lg \lg m$ stations compute ranks of their *leaders*.
Now the density of the *leaders* is much higher.
The message to the next slave is:
(leader's key, **group***, rank, timer)*



Then select the *winners* that split the other sequence into groups.
Note that each iteration *works in opposite direction* to the previous one.

Time and energy

For the whole iteration (*computing ranks of leaders, selecting winners, splitting the other sequence into groups*):

- the energetic cost is $O(1)$
- time is $O(m)$

$h(m,i)$ and $l(m)$

$$h(m,i) = \begin{cases} m, & \text{if } i = 0 \\ \lceil \lg(m+1) \rceil, & \text{if } i \geq 1 \end{cases}$$

$h(m, i+1)$ is the *height* of *balanced binary tree* with $h(m,i)$ nodes.
Thus $h(m,i)$ is the *block* size in ith iteration.

$h(m,i)$ and $l(m)$

$$h(m,i) = \begin{cases} m, & \text{if } i = 0 \\ \lceil \lg(m+1) \rceil, & \text{if } i \geq 1 \end{cases}$$

$h(m, i+1)$ is the *height* of *balanced binary tree* with $h(m,i)$ nodes.
Thus $h(m,i)$ is the *block* size in ith iteration.

$$l(m) = \min\{i : h(m,i) \leq 2\}.$$

$$\underbrace{h(m,0), h(m,1), \ldots 2}_{l(m)}, 2, 2, 2, 2 \ldots$$

Note that: $l(m)$ is $O(\lg^* m)$.

$O(\lg^* m)$

After $2\lceil l(m)/2 \rceil + 2 = O(\lg^* m)$ iterations the group numbers in each sequence refer to blocks of size two in the other sequence.
Hence, each station can compute its rank by listening to at most two messages.

$O(\lg^* m)$

After $2\lceil l(m)/2 \rceil + 2 = O(\lg^* m)$ iterations the group numbers in each sequence refer to blocks of size two in the other sequence.
Hence, each station can compute its rank by listening to at most two messages.

Theorem: Merging of two sequences of length m can be done in time $\Theta(m \lg^* m)$ with energetic cost (of both listening and broadcasting) $\Theta(\lg^* m)$

Concluding remarks

- Exact asymptotic bounds for energetic cost?
  - of merging (constant?)
  - of sorting (we ignore cost of internal computations and stations may have larger memory) ($\Omega(\lg n)$?)

Concluding remarks

- Exact asymptotic bounds for energetic cost?
  - of merging (constant?)
  - of sorting (we ignore cost of internal computations and stations may have larger memory) ($\Omega(\lg n)$?)
- The case: each station stores k keys instead of one (Merging algorithm in DELIS-TR-0239: energ. cost: $\approx 8k + 4\lg m$, time: $\approx 6m \cdot k + m$)

Thank you!