Optimal Tracking of Distributed Heavy Hitters and Quantiles

Qin Zhang

Hong Kong University of Science & Technology

Joint work with Ke Yi

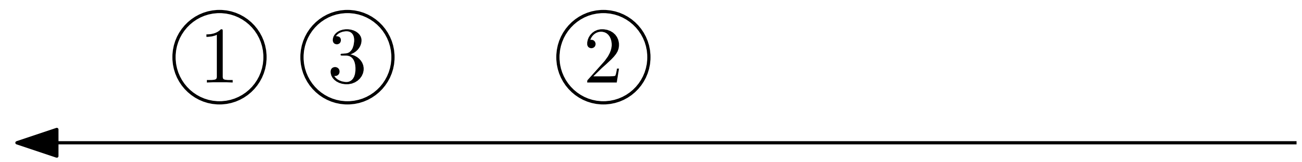
PODS 2009

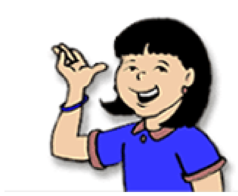
June 30, 2009

Models and Problems

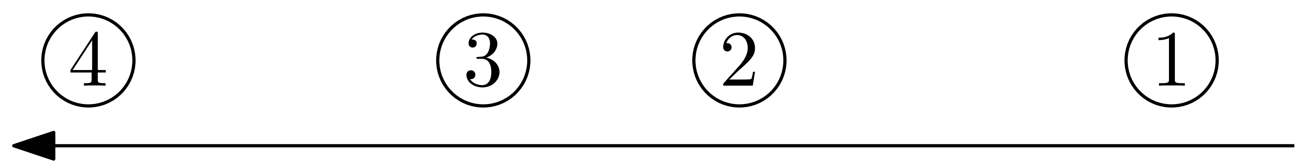
Distributed streaming model [Babcock, Olston SIGMOD 2003]

Alice observes
$A(t)$ by time t





Carole tries to track $f(A(t) \cup B(t))$ for all t (an ϵ-approximation is usually allowed)

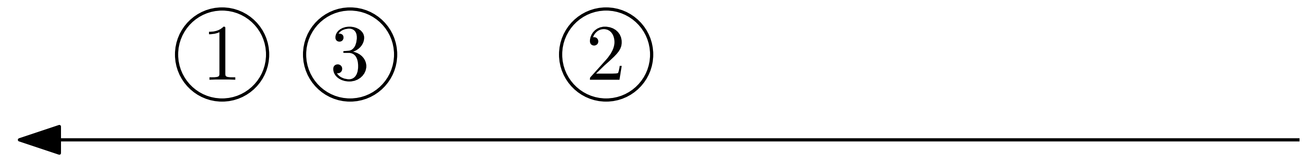


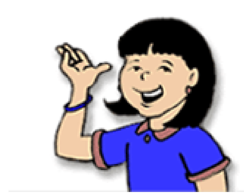
Bob observes
$B(t)$ by time t

$A(t), B(t)$: multisets

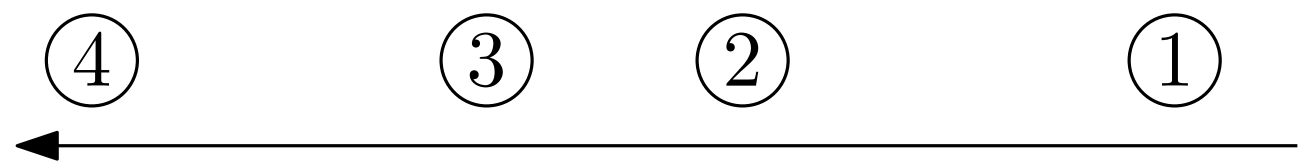
Distributed streaming model [Babcock, Olston SIGMOD 2003]

Alice observes
$A(t)$ by time t





Carole tries to track $f(A(t) \cup B(t))$ for all t (an ϵ-approximation is usually allowed)



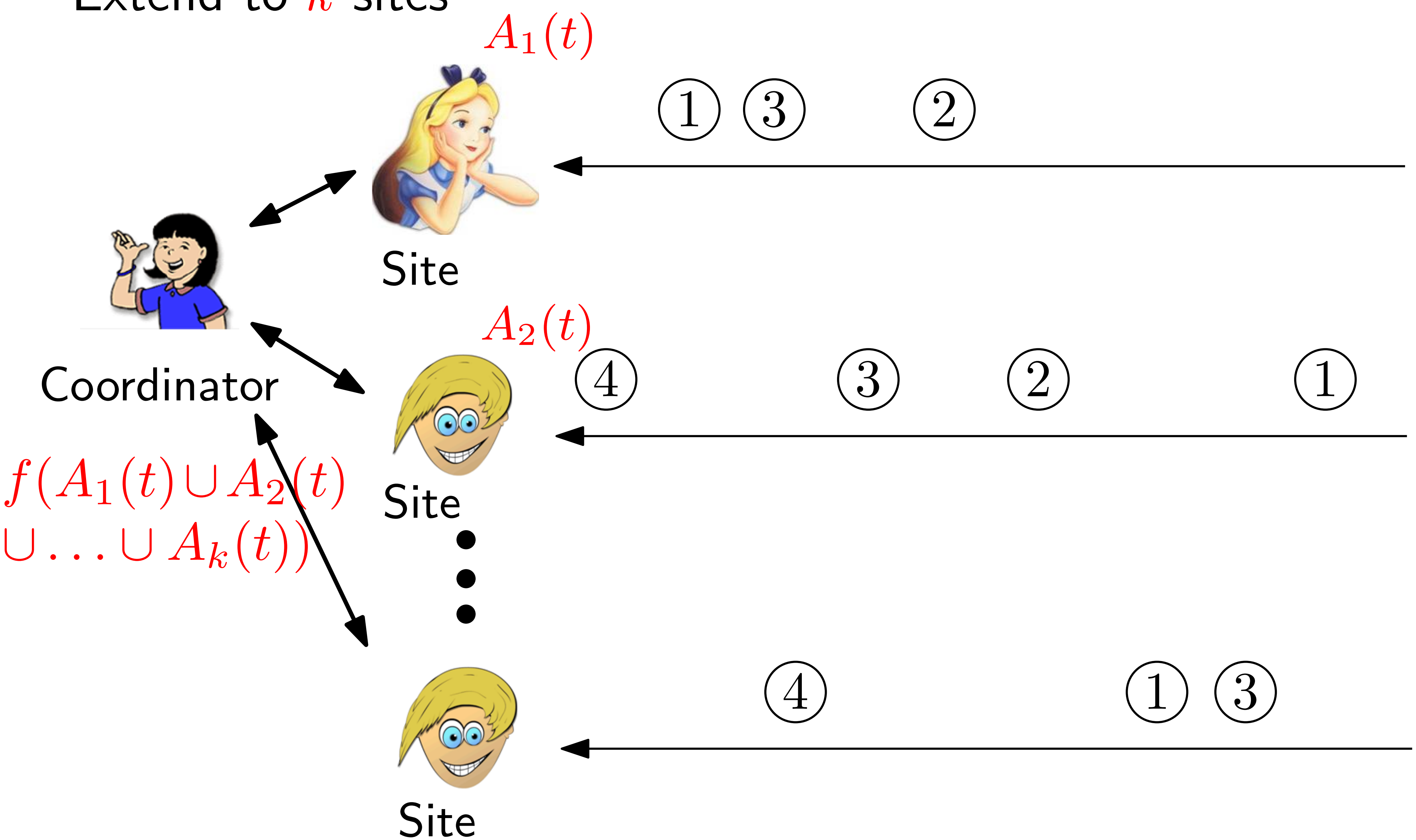
Bob observes
$B(t)$ by time t

Find an (approximate) tracking protocol
while minimizing communication

$A(t), B(t)$: multisets

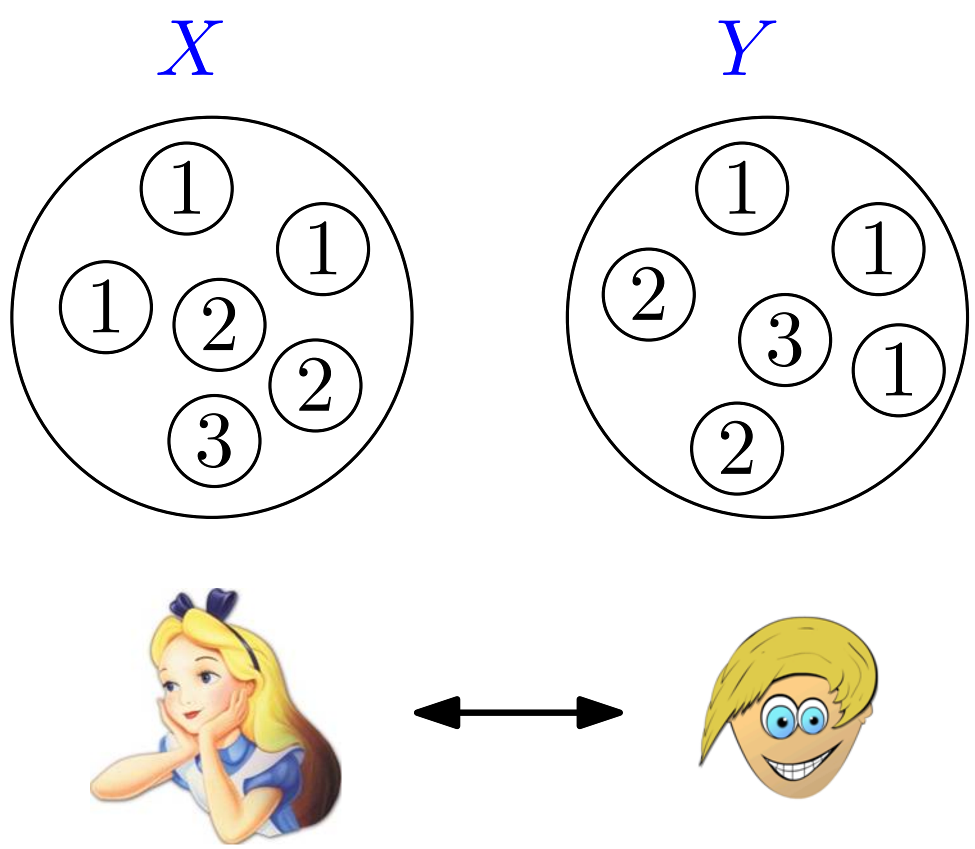
Distributed streaming model [Babcock, Olston SIGMOD 2003]

Extend to k sites



Find a protocol to minimize communication.

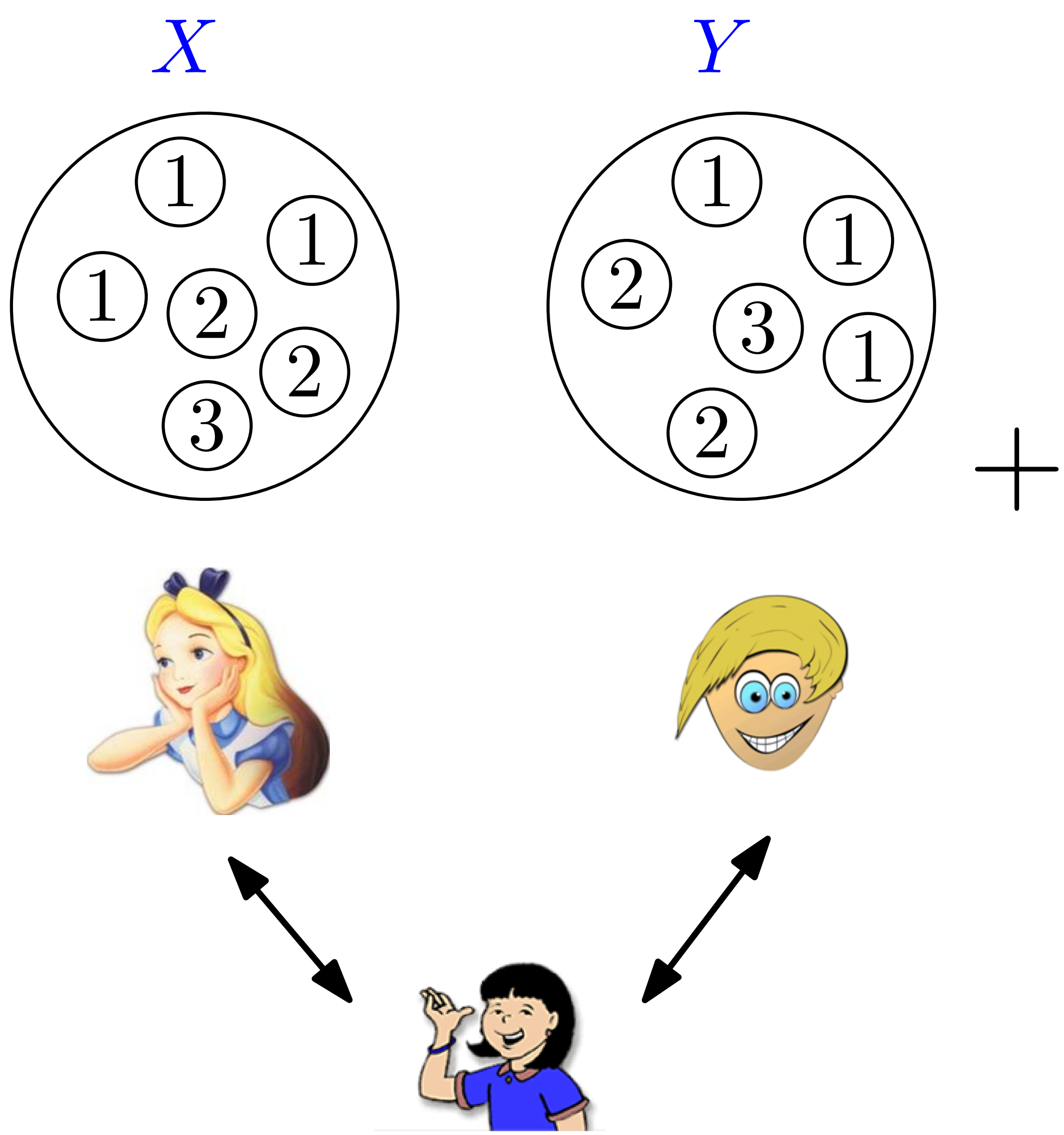
Combination of two models



(One-shot) communication model

$f(X \cup Y)$; minimize communication

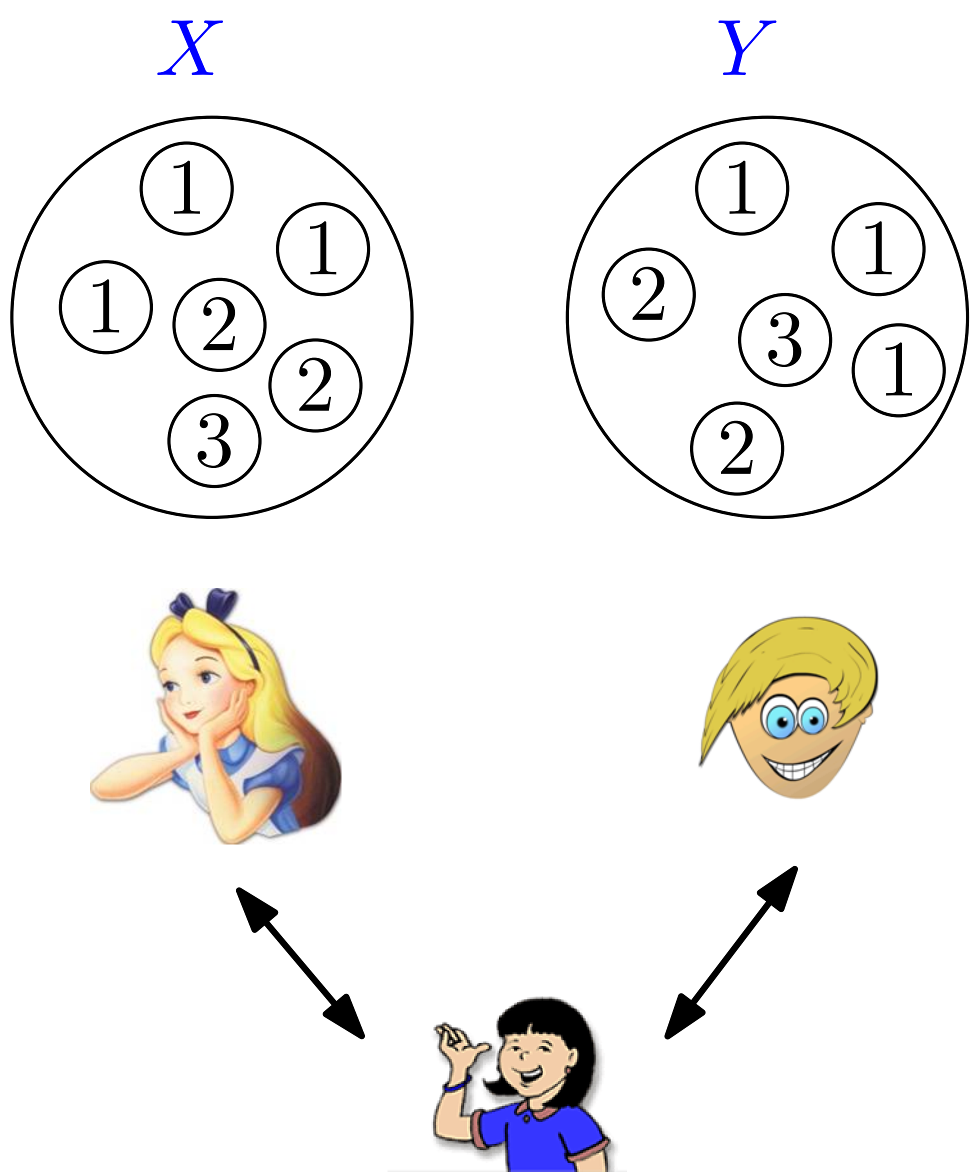
Combination of two models



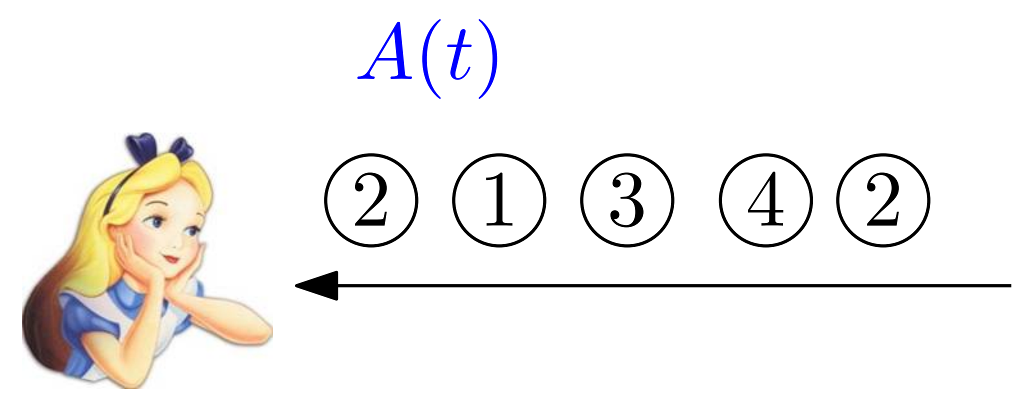
(One-shot) communication model

$f(X \cup Y)$; minimize communication

Combination of two models



$+$



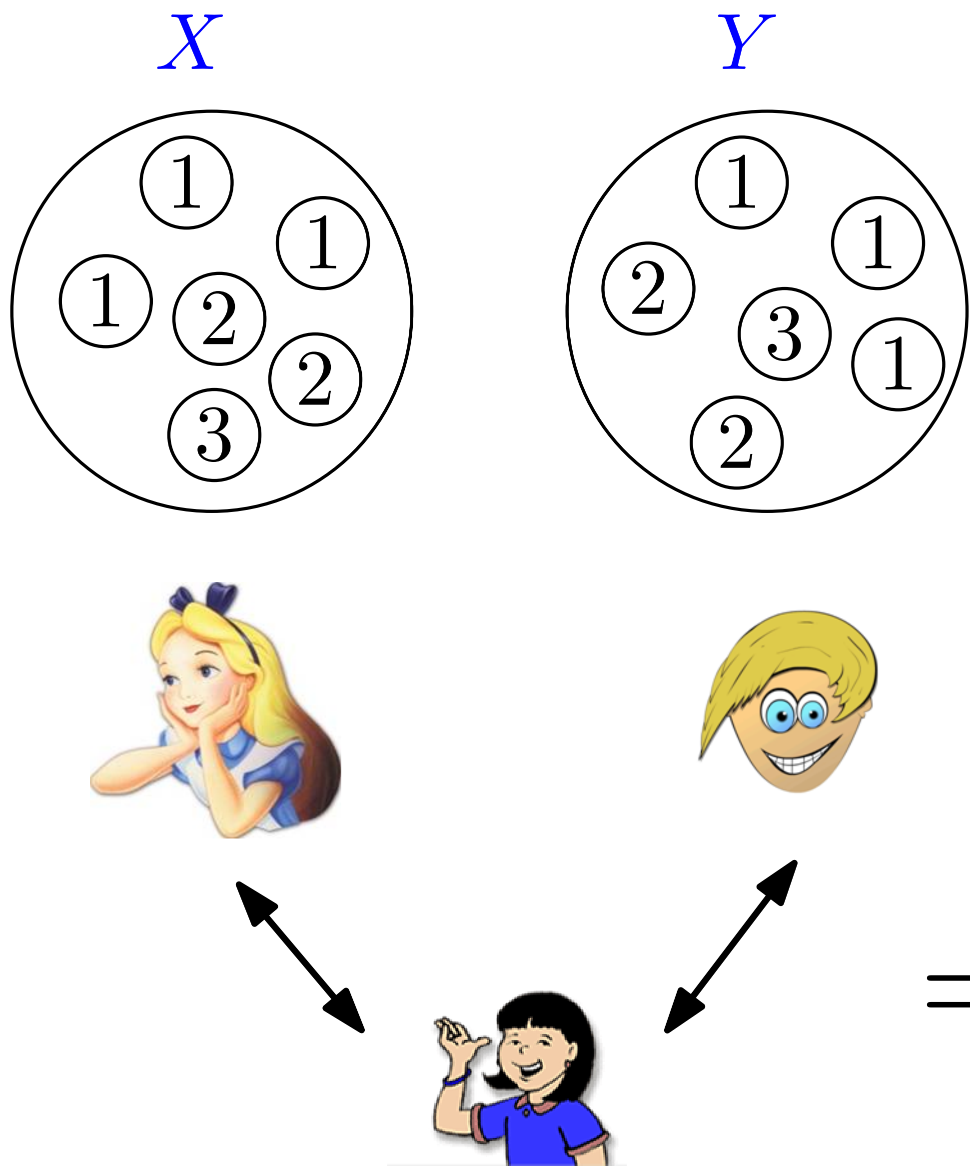
Streaming model

$f(A(t))$; minimize space

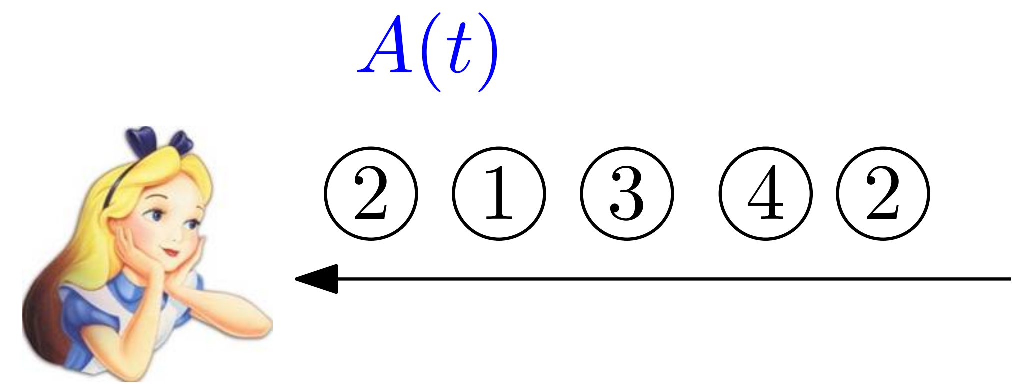
(One-shot) communication model

$f(X \cup Y)$; minimize communication

Combination of two models



$+$



Streaming model

$f(A(t))$; minimize space

$\Rightarrow$ Distributed streaming model

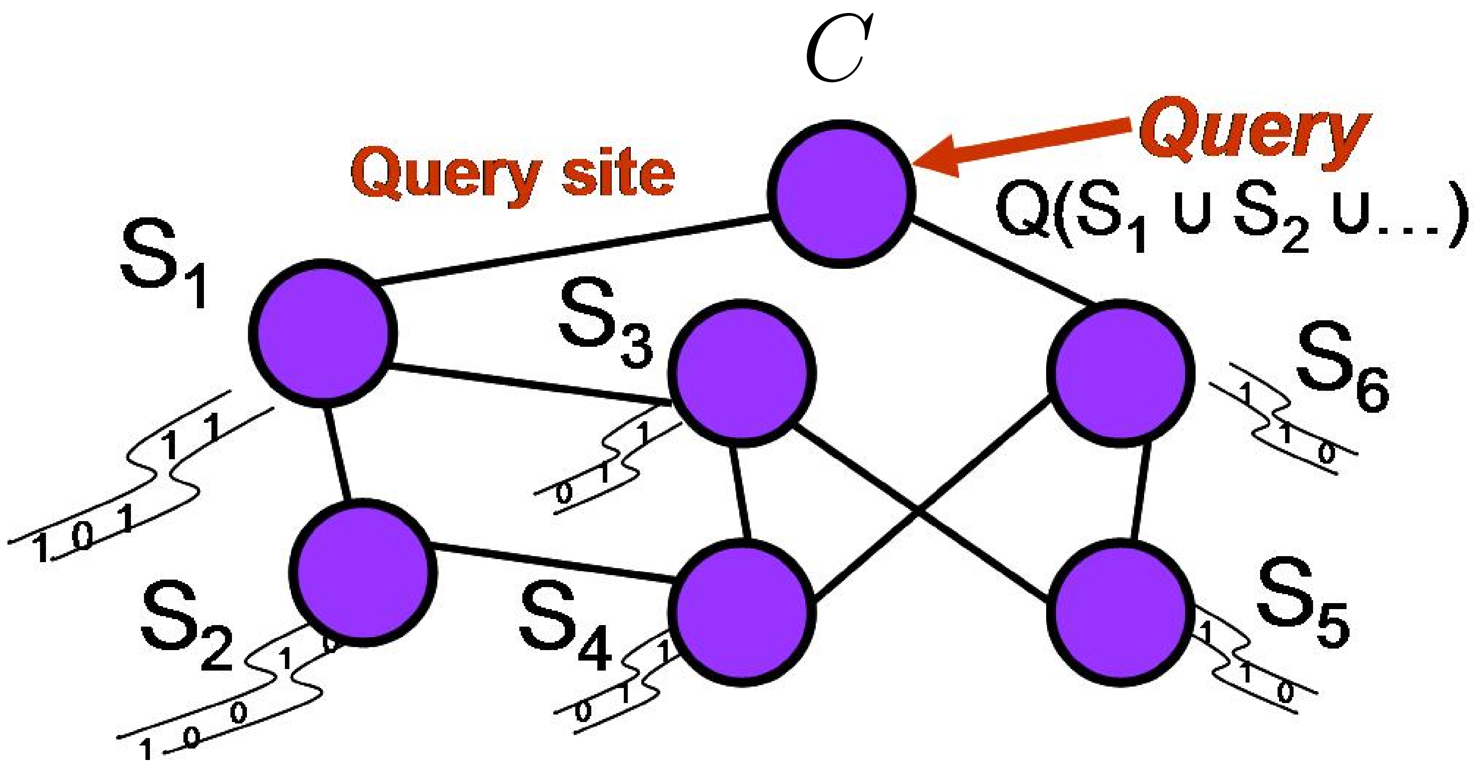
a.k.a. Continuous commun. model

Only interested in communication cost

(One-shot) communication model

$f(X \cup Y)$; minimize communication

Applied motivation: distributed monitoring



Large-scale distributed "holistic" querying/monitoring.

Picture from the tutorial "Streaming in a connected world: Querying and tracking distributed data streams" at VLDB$'06$ and SIGMOD$'07$

Three different fs

- Heavy hitters: For a multiset A, item i is a
  - heavy hitter if (count of i) $> \phi|A|$
  - non-heavy hitter if (count of i) $< (\phi - \epsilon)|A|$
- Application: Find out IP addresses that appear $\geq 1\%$ over a network

Three different fs

- **Heavy hitters**: For a multiset A, item i is a
  - heavy hitter if (count of i) $> \phi|A|$
  - non-heavy hitter if (count of i) $< (\phi - \epsilon)|A|$
  - Application: Find out IP addresses that appear $\geq 1\%$ over a network
- **Single Quantile**: Median, 1/4-quantile.
  - Application: The median size of packets over a network.

Three different fs

- Heavy hitters: For a multiset A, item i is a
  - heavy hitter if (count of i) $> \phi|A|$
  - non-heavy hitter if (count of i) $< (\phi - \epsilon)|A|$
  - Application: Find out IP addresses that appear $\geq 1\%$ over a network
- Single Quantile: Median, $1/4$-quantile.
  - Application: The median size of packets over a network.
- All Quantiles: A is a set of distinct elements from a totally ordered universe. Quantile of $x = |y < x, y \in A|/|A|$, usually allows a $\pm\epsilon$ error. Need a data structure to extract ϵ-approximate quantile for any x.

Previous and our results

Previous results in distributed streaming model

- Frequent moments ($F_p = \sum_i (\text{count of } i)^p$)
  - Total count F_1
    - Simple: each site sends an update every time its local count has increased by a $(1+\epsilon)$ factor
    - Complexity $O(k/\epsilon \cdot \log n)$
      n: total # items received at all sites, ϵ: relative error

Previous results in distributed streaming model

- Frequent moments ($F_p = \sum_i (\text{count of } i)^p$)
  - Total count F_1
    - Simple: each site sends an update every time its local count has increased by a $(1+\epsilon)$ factor
    - Complexity $O(k/\epsilon \cdot \log n)$
      n: total # items received at all sites, ϵ: relative error
  - Distinct count F_0, Second frequency moment F_2
    - F_0: $O(k/\epsilon^2 \cdot \log n \log(n/\delta))$ [Cormode, Muthukrishnan, Yi, SODA'08]
    - F_2: $O((k^2/\epsilon^2 + k^{3/2}/\epsilon^4) \log n \log(n/\delta))$ [same paper]

Previous results in distributed streaming model

- Frequent moments ($F_p = \sum_i (\text{count of } i)^p$)
  - Total count F_1
    - Simple: each site sends an update every time its local count has increased by a $(1+\epsilon)$ factor
    - Complexity $O(k/\epsilon \cdot \log n)$
      n: total # items received at all sites, ϵ: relative error
  - Distinct count F_0, Second frequency moment F_2
    - F_0: $O(k/\epsilon^2 \cdot \log n \log(n/\delta))$ [Cormode, Muthukrishnan, Yi, SODA'08]
    - F_2: $O((k^2/\epsilon^2 + k^{3/2}/\epsilon^4) \log n \log(n/\delta))$ [same paper]
- Entropy of the stream
  - Shannon entropy and related entropies [Arackaparambil, Brody, Chakrabarti, ICALP'09]

Previous results: Heavy Hitters

- Streaming model (space complexity)
  - $O(1/\epsilon)$, [Karp, Shenker, Papadimitriou TODS'03], [Demaine, Munro, Lopez-Ortiz, ESA'02], [Metwally, Agrawal, Abbadi TODS'06], [Misra, Gries 1982]

Previous results: Heavy Hitters

- Streaming model (space complexity)
  - $O(1/\epsilon)$, [Karp, Shenker, Papadimitriou TODS'03], [Demaine, Munro, Lopez-Ortiz, ESA'02], [Metwally, Agrawal, Abbadi TODS'06], [Misra, Gries 1982]

- One-shot communication model
  - $O(k/\epsilon)$ - easy

Previous results: Heavy Hitters

- Streaming model (space complexity)
  - $O(1/\epsilon)$, [Karp, Shenker, Papadimitriou TODS'03], [Demaine, Munro, Lopez-Ortiz, ESA'02], [Metwally, Agrawal, Abbadi TODS'06], [Misra, Gries 1982]
- One-shot communication model
  - $O(k/\epsilon)$ - easy
- Distributed streaming model
  - Heuristics [e.g. Babcock, Olston, SIGMOD'03]

Previous results: All Quantiles

- Streaming model (space complexity)
  - $O(1/\epsilon \cdot \log(\epsilon n))$ [Greenwald, Khanna, SIGMOD'01]
  - $O(1/\epsilon \cdot \log(1/\delta))$, with failure probability δ [Cormode, Muthukrishnan, J. Alg '05]

Previous results: All Quantiles

- Streaming model (space complexity)
  - $O(1/\epsilon \cdot \log(\epsilon n))$ [Greenwald, Khanna, SIGMOD'01]
  - $O(1/\epsilon \cdot \log(1/\delta))$, with failure probability δ [Cormode, Muthukrishnan, J. Alg '05]

- One-shot communication model
  - $O(k/\epsilon)$ - easy

Previous results: All Quantiles

- Streaming model (space complexity)
  - $O(1/\epsilon \cdot \log(\epsilon n))$ [Greenwald, Khanna, SIGMOD'01]
  - $O(1/\epsilon \cdot \log(1/\delta))$, with failure probability δ [Cormode, Muthukrishnan, J. Alg '05]

- One-shot communication model
  - $O(k/\epsilon)$ - easy

- Distributed streaming model
  - $O(k/\epsilon^2 \cdot \log n)$ [Cormode, Garofalakis, Muthukrishnan, Rastogi, SIGMOD'05]

Our results

- Tracking heavy hitters
  - Complexity: $\Theta(k/\epsilon \cdot \log n)$

Our results

- Tracking heavy hitters
  - Complexity: $\Theta(k/\epsilon \cdot \log n)$
- Tracking one quantile (in particular: Median)
  - Complexity: $\Theta(k/\epsilon \cdot \log n)$

Our results

- Tracking heavy hitters
  - Complexity: $\Theta(k/\epsilon \cdot \log n)$
- Tracking one quantile (in particular: Median)
  - Complexity: $\Theta(k/\epsilon \cdot \log n)$
- Tracking all quantiles
  - Upper bound: $O(k/\epsilon \cdot \log^2(1/\epsilon) \log n)$

Technical details

Tracking Heavy Hitters

- Divide into multiple rounds: Every time m increase by a factor of $1 + \epsilon$. m: total $\#$ elements inserted

Tracking Heavy Hitters

- Divide into multiple rounds: Every time m increase by a factor of $1+\epsilon$. m: total # elements inserted
- In each round
  - Compute total count m: $O(k)$ messages
  - Broadcast m: $O(k)$ messages
  - Each site:
    - For each i, sends out a message every time count(i) increases by $\epsilon m/(2k)$
    - Sends out a message every time total local count increases by $\epsilon m/(2k)$
  - Coordinator:
    - Terminate the round when $O(k)$ messages have been received

Tracking Heavy Hitters

- Divide into multiple rounds: Every time m increase by a factor of $1+\epsilon$. m: total # elements inserted
- In each round
  - Compute total count m: $O(k)$ messages
  - Broadcast m: $O(k)$ messages
  - Each site:
    - For each i, sends out a message every time count(i) increases by $\epsilon m/(2k)$
    - Sends out a message every time total local count increases by $\epsilon m/(2k)$
  - Coordinator:
    - Terminate the round when $O(k)$ messages have been received

Track the count of each element with error at most ϵm

Tracking Heavy Hitters

Total messages in one round: $O(k)$

\# rounds: $O(\log_{1+\epsilon} n) = O(\log n/\epsilon)$

- Divide into multiple rounds: Every time m increase by a factor of $1+\epsilon$. m: total # elements inserted
- In each round
  - Compute total count m: $O(k)$ messages
  - Broadcast m: $O(k)$ messages
  - Each site:
    - For each i, sends out a message every time count(i) increases by $\epsilon m/(2k)$
    - Sends out a message every time total local count increases by $\epsilon m/(2k)$
  - Coordinator:
    - Terminate the round when $O(k)$ messages have been received

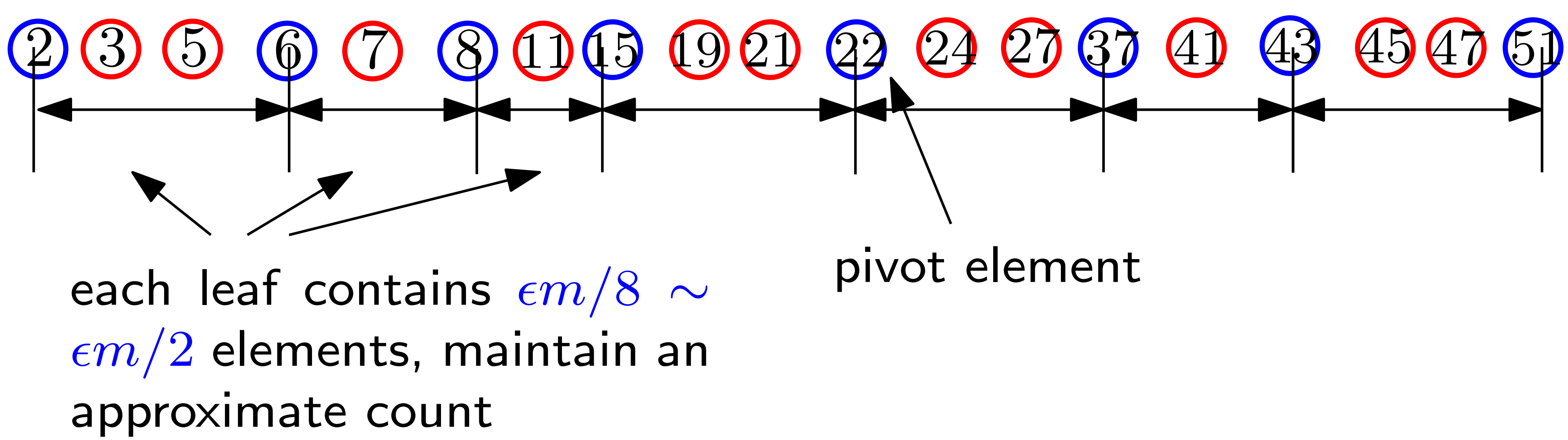
Track the count of each element with error at most ϵm

Tracking Heavy Hitters: lower bound

- Construct an input where the set of heavy hitters undergoes $\Omega(\log n/\epsilon)$ updates
- In each update, one item changes from non-heavy to heavy
- Argue that $\Omega(k)$ sites need to be contacted in order to correctly identify this item at the right time

Tracking the Median

- Divide tracking period into rounds: m (roughly) doubles in each round
- In each round
  - Maintain a dynamic set of intervals each has $\frac{\epsilon}{8}m$ to $\frac{\epsilon}{2}m$ items: # messages $O(k/\epsilon)$ in each round



Tracking the Median

- Divide tracking period into rounds: m (roughly) doubles in each round
- In each round
  - Maintain a dynamic set of intervals each has $\frac{\epsilon}{8}m$ to $\frac{\epsilon}{2}m$ items: # messages $O(k/\epsilon)$ in each round
  - After $\Theta(\epsilon m)$ items, recalculate the median from the old median by using the dynamic set of intervals.

    $\#$ messages $O(k)$ per recalculation, $O(1/\epsilon)$ re-calculations in each round

Tracking the Median

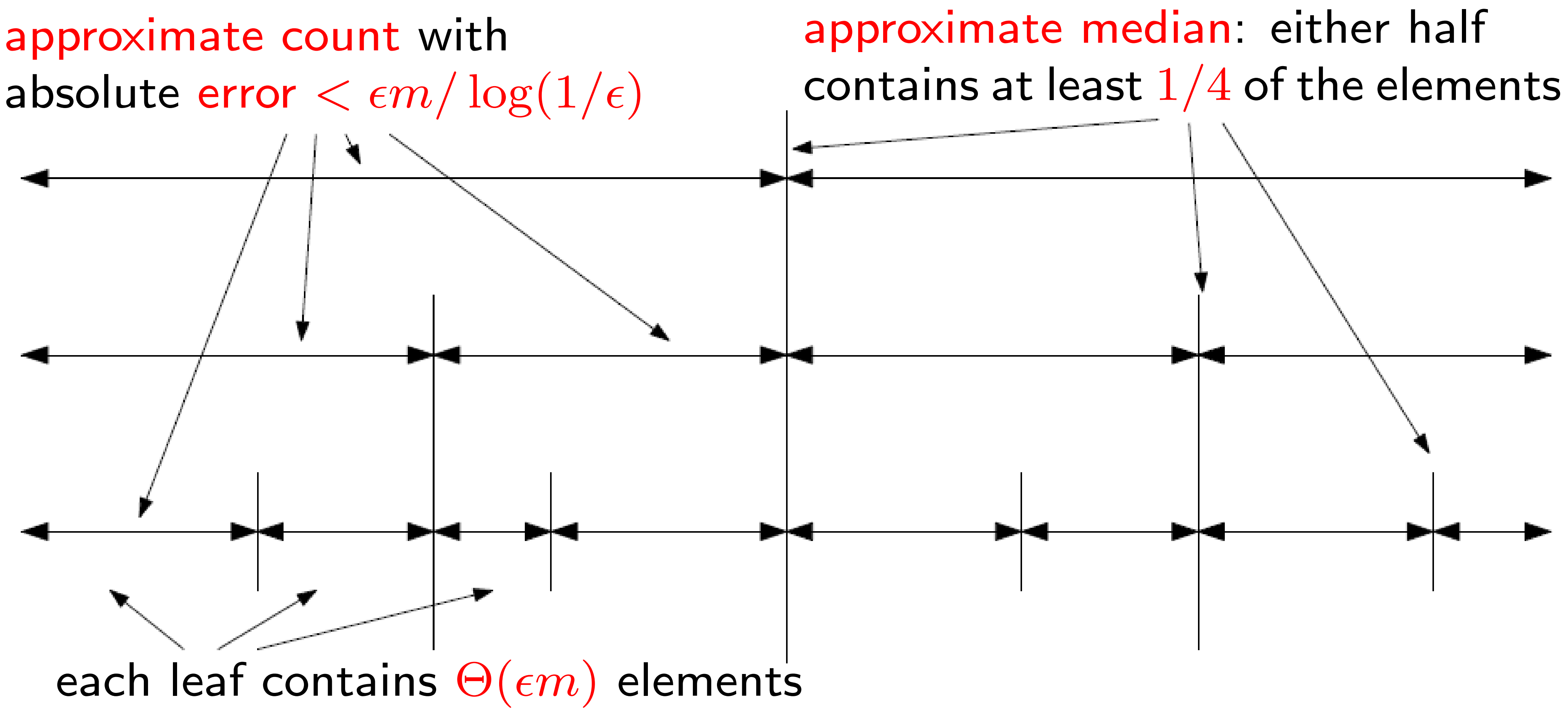
Total messages in one round: $O(k/\epsilon)$
rounds: $O(\log n)$

- Divide tracking period into rounds: m (roughly) doubles in each round
- In each round
  - Maintain a dynamic set of intervals each has $\frac{\epsilon}{8}m$ to $\frac{\epsilon}{2}m$ items: # messages $O(k/\epsilon)$ in each round
  - After $\Theta(\epsilon m)$ items, recalculate the median from the old median by using the dynamic set of intervals.

    # messages $O(k)$ per recalculation, $O(1/\epsilon)$ re-calculations in each round

Tracking all Quantiles

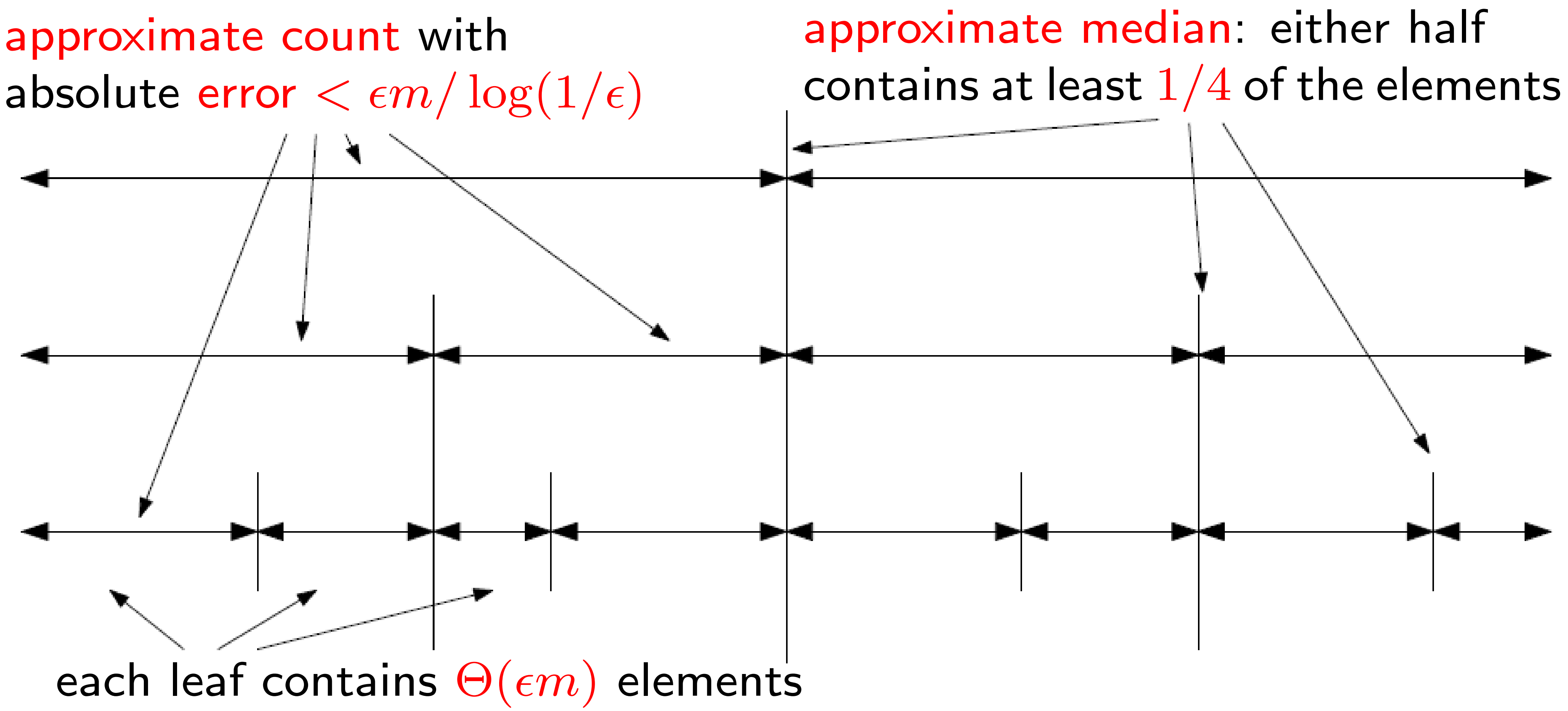
A balanced tree



structure can be used to extract the rank of any x with absolute error $< \epsilon m \Rightarrow$ quantile with error $< \epsilon$

Tracking all Quantiles

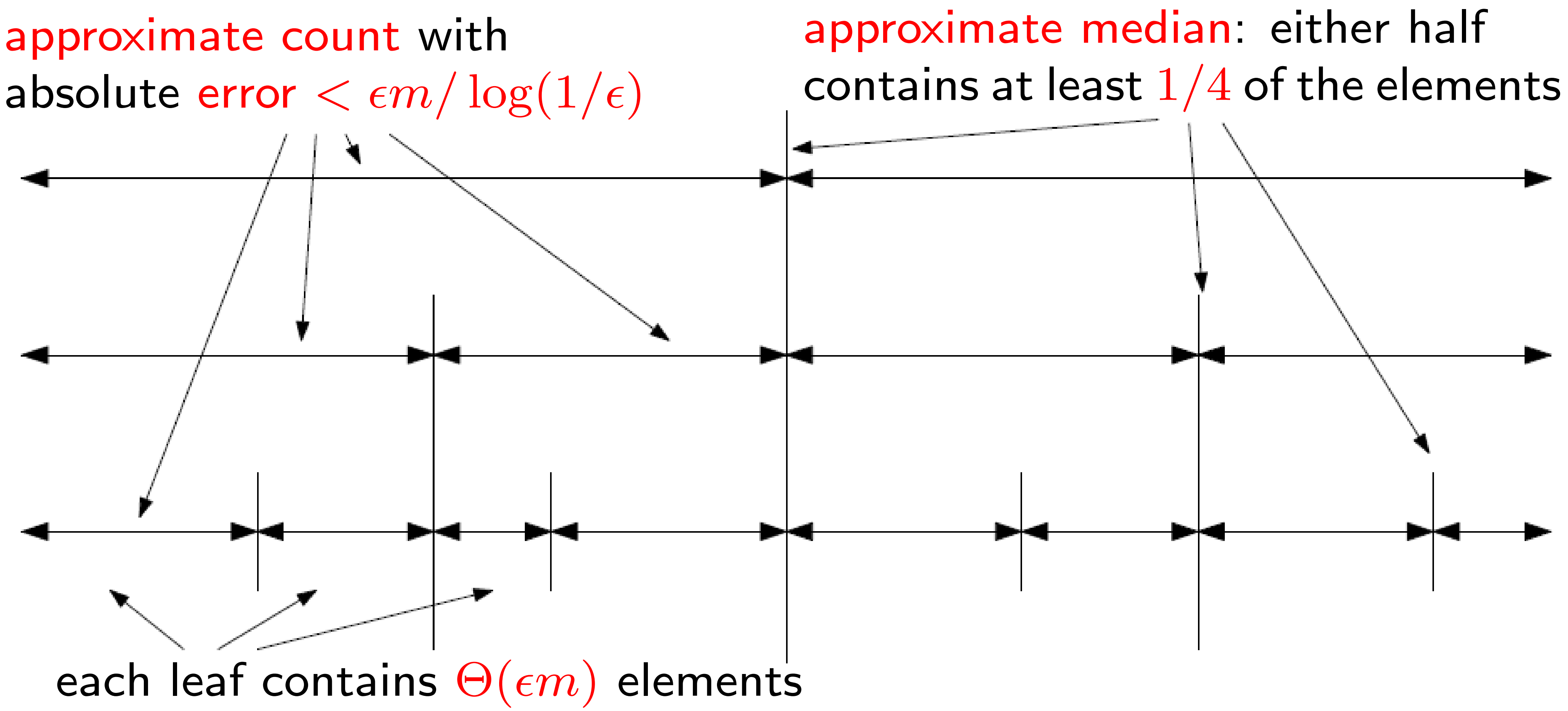
A balanced tree



- Divide tracking period into rounds: m (roughly) doubles in each round

Tracking all Quantiles

A balanced tree



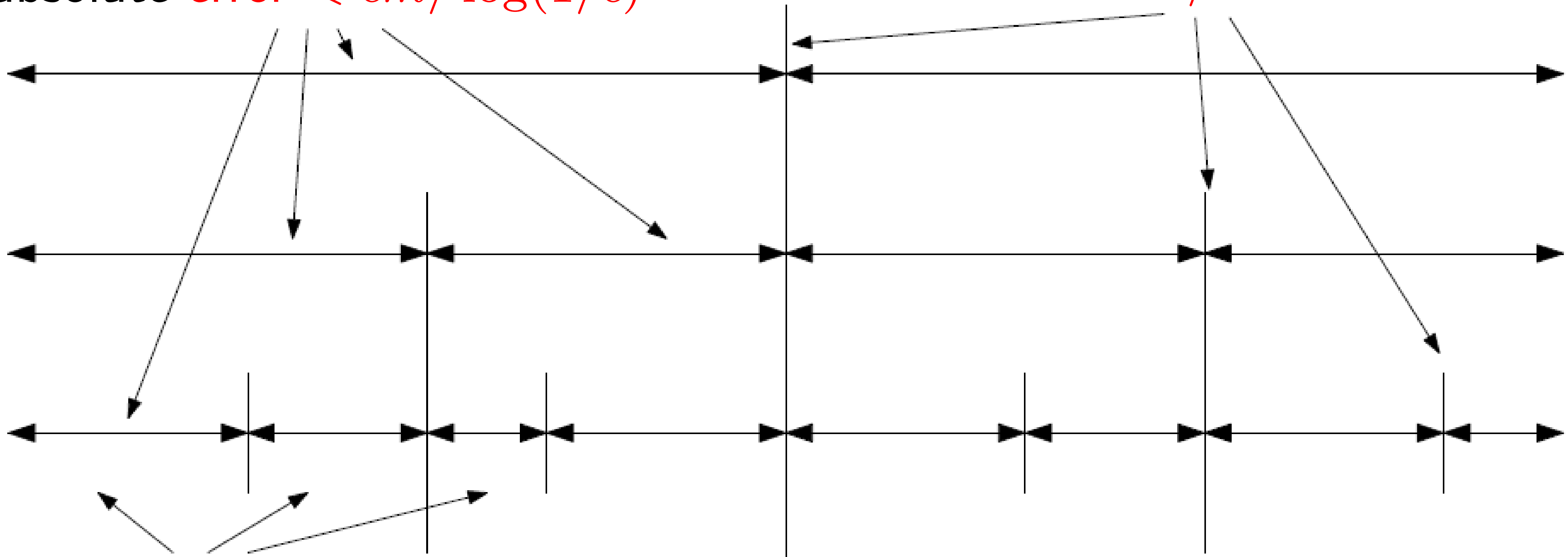
- Divide tracking period into rounds: m (roughly) doubles in each round
- At the beginning of each round, initialize the structure with $O(k/\epsilon)$ communication, then broadcast.

Tracking all Quantiles

A balanced tree

approximate count with absolute error $< \epsilon m / \log(1/\epsilon)$

approximate median: either half contains at least $1/4$ of the elements

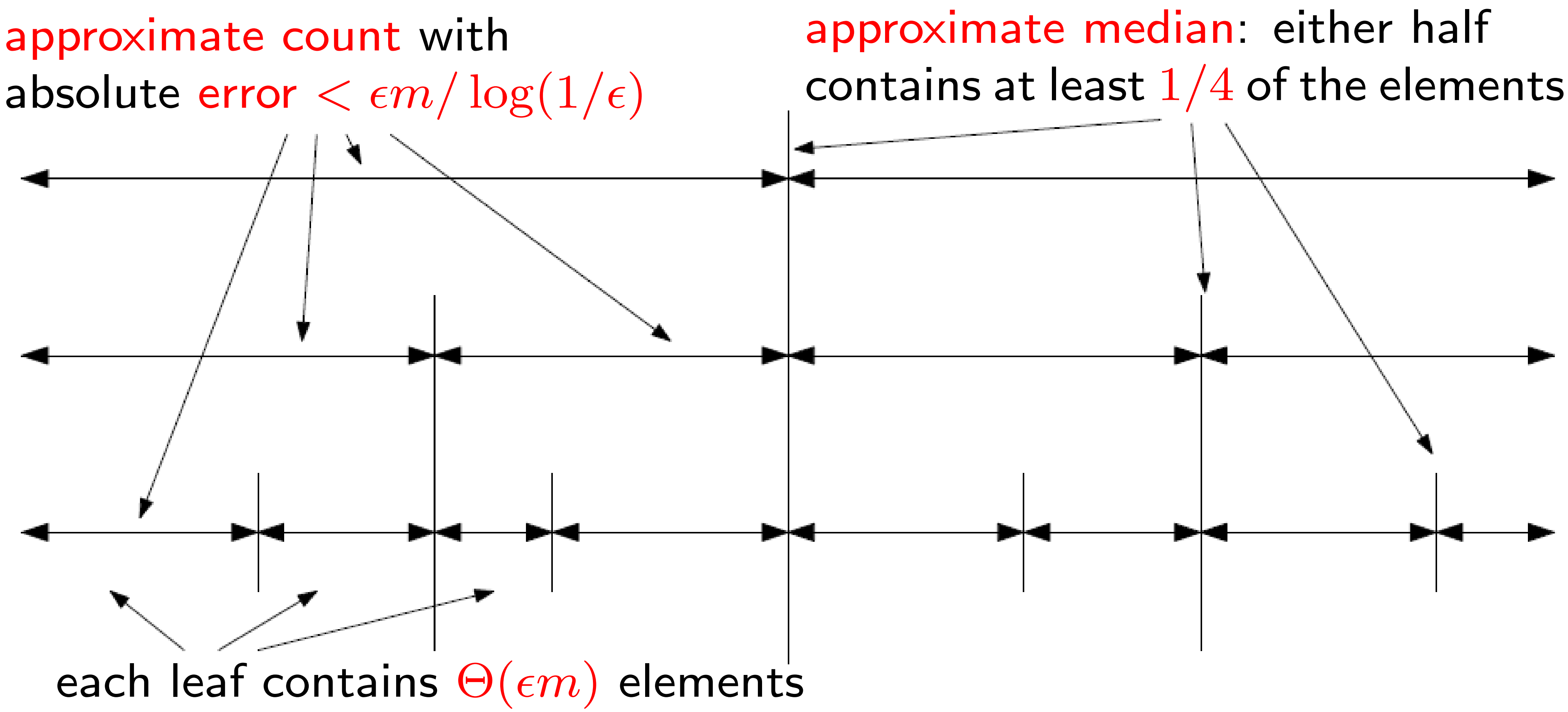


each leaf contains $\Theta(\epsilon m)$ elements

- Divide tracking period into rounds: m (roughly) doubles in each round
- At the beginning of each round, initialize the structure with $O(k/\epsilon)$ communication, then broadcast.
- Each site tracks each interval, sends a message for every $\Theta(\epsilon m/(k \log(1/\epsilon)))$ new elements arrive. Total: $O(k/\epsilon \cdot \log^2(1/\epsilon))$

Tracking all Quantiles

A balanced tree



- Divide tracking period into rounds: m (roughly) doubles in each round
- At the beginning of each round, initialize the structure with $O(k/\epsilon)$ communication, then broadcast.
- Each site tracks each interval, sends a message for every $\Theta(\epsilon m/(k \log(1/\epsilon)))$ new elements arrive. Total: $O(k/\epsilon \cdot \log^2(1/\epsilon))$
- Maintain the balance of the tree. Total: $O(k/\epsilon \cdot \log(1/\epsilon))$

Remarks and open problems

- Focused on communication only, but the algorithms can be implemented with small space
  - $O(1/\epsilon)$ each site for heavy hitter
  - $O(1/\epsilon \cdot \log(\epsilon n))$ each site for quantiles

Remarks and open problems

- Focused on communication only, but the algorithms can be implemented with small space
  - $O(1/\epsilon)$ each site for heavy hitter
  - $O(1/\epsilon \cdot \log(\epsilon n))$ each site for quantiles
- Randomized algorithms?
  - There is an upperbound $O\left((k + 1/\epsilon^2) \cdot poly\log(n, k, 1/\epsilon)\right)$

Remarks and open problems

- Focused on communication only, but the algorithms can be implemented with small space
  - $O(1/\epsilon)$ each site for heavy hitter
  - $O(1/\epsilon \cdot \log(\epsilon n))$ each site for quantiles
- Randomized algorithms?
  - There is an upperbound $O\left((k + 1/\epsilon^2) \cdot poly \log(n, k, 1/\epsilon)\right)$
- How about deletions?
  - One site, arbitrary function in Z^d, competitive analysis. (Yi and Zhang SODA 2009)

Remarks and open problems

- Focused on communication only, but the algorithms can be implemented with small space
  - $O(1/\epsilon)$ each site for heavy hitter
  - $O(1/\epsilon \cdot \log(\epsilon n))$ each site for quantiles
- Randomized algorithms?
  - There is an upperbound $O\left((k + 1/\epsilon^2) \cdot poly \log(n, k, 1/\epsilon)\right)$
- How about deletions?
  - One site, arbitrary function in Z^d, competitive analysis. (Yi and Zhang SODA 2009)
- Other tracking problems ...

The End

$\mathcal{THANK\ \ YOU}$

Q and A