Optimal Sampling from Distributed Streams

Qin Zhang

Joint work with
Graham Cormode (AT&T)
S. Muthukrishnan (Rutgers)
Ke Yi (HKUST)

Sept. 17, 2010
MSRA

Reservoir sampling [Waterman '??; Vitter '85]

- **Problem:** Maintain a (uniform) sample (w/o replacement) of size s from a stream of n items
  - Every subset of size s has equal probability to be the sample

Reservoir sampling [Waterman '??; Vitter '85]

- **Problem:** Maintain a (uniform) sample (w/o replacement) of size s from a stream of n items
  - Every subset of size s has equal probability to be the sample

- **Solution:** When the i-th item arrives
  - With probability s/i, use it to replace an item in the current sample chosen uniformly at ranfom
  - With probability $1 - s/i$, throw it away

Reservoir sampling [Waterman '??; Vitter '85]

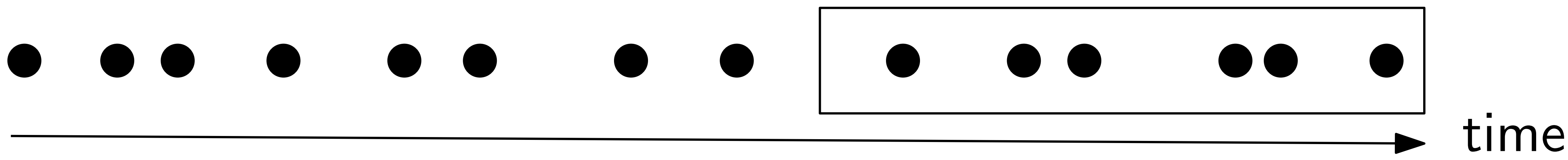
- **Problem:** Maintain a (uniform) sample (w/o replacement) of size s from a stream of n items
  - Every subset of size s has equal probability to be the sample
- **Solution:** When the i-th item arrives
  - With probability s/i, use it to replace an item in the current sample chosen uniformly at ranfom
  - With probability $1 - s/i$, throw it away
- **Correctness:** intuitive

Reservoir sampling [Waterman '??; Vitter '85]

- **Problem:** Maintain a (uniform) sample (w/o replacement) of size s from a stream of n items
  - Every subset of size s has equal probability to be the sample
- **Solution:** When the i-th item arrives
  - With probability s/i, use it to replace an item in the current sample chosen uniformly at ranfom
  - With probability $1 - s/i$, throw it away
- **Correctness:** intuitive
- **Cost:** Space: $O(s)$, time $O(1)$

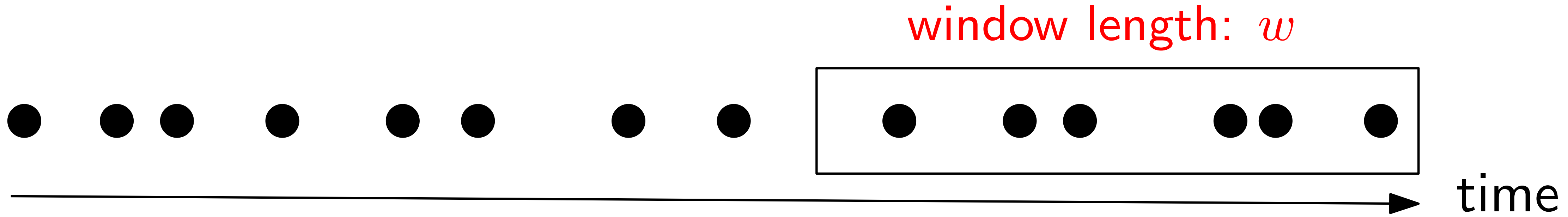
Sampling from a sliding window

[Babcock, Datar, Motwani, SODA'02; Gemulla, Lehner, SIGMOD'08; Braverman, Ostrovsky, Zaniolo, PODS'09]



Sampling from a sliding window

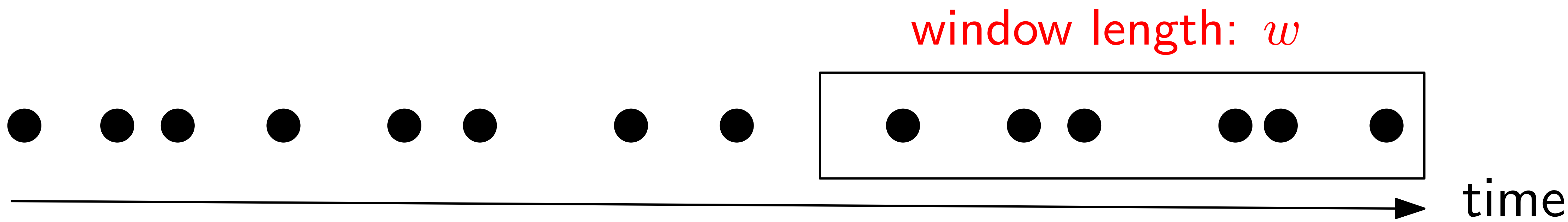
[Babcock, Datar, Motwani, SODA'02; Gemulla, Lehner, SIGMOD'08; Braverman, Ostrovsky, Zaniolo, PODS'09]



Time based window and sequence based window

Sampling from a sliding window

[Babcock, Datar, Motwani, SODA'02; Gemulla, Lehner, SIGMOD'08; Braverman, Ostrovsky, Zaniolo, PODS'09]

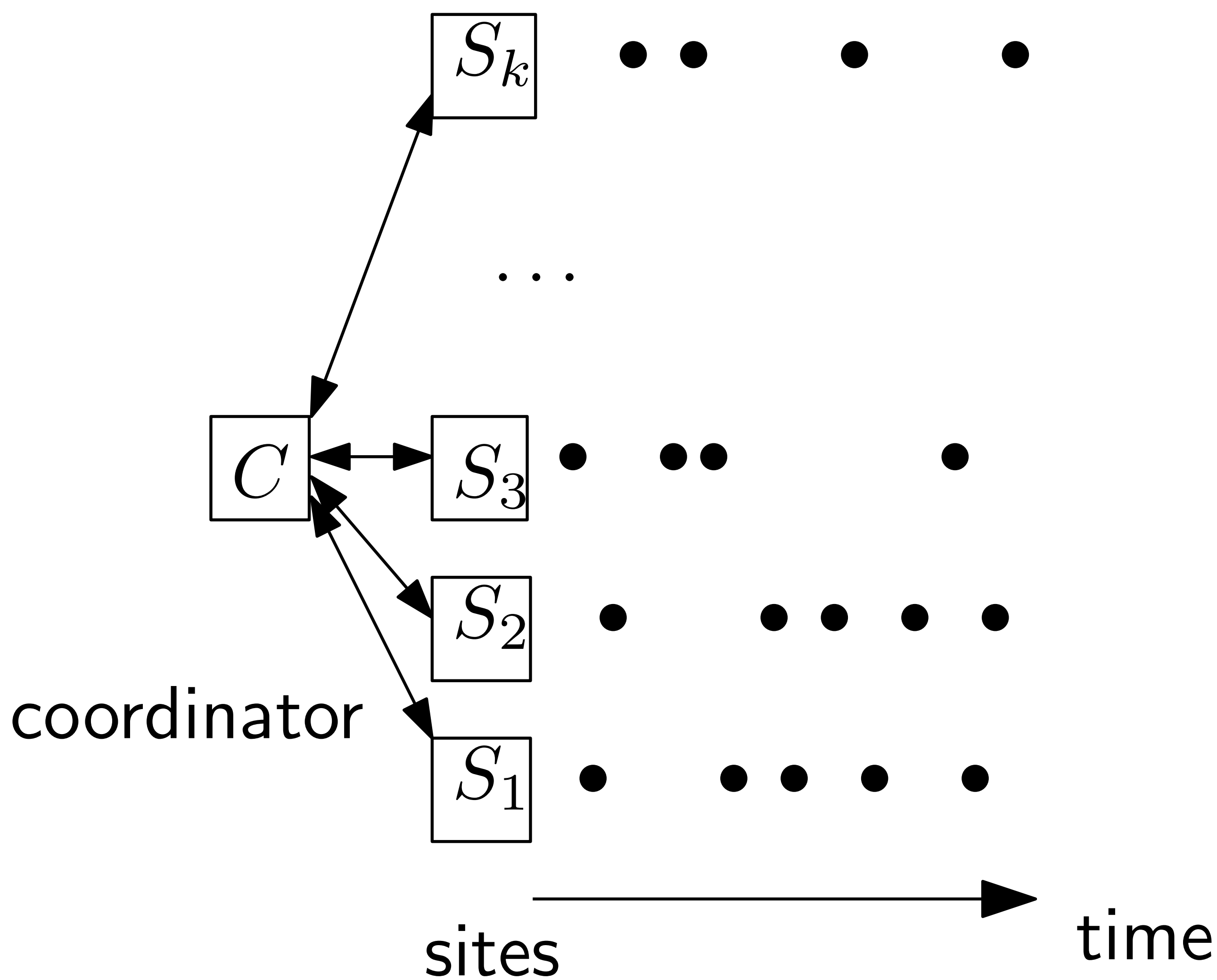


Time based window and sequence based window

- Space: $\Theta(s \log w)$
  - w: number of items in the sliding window
- Time: $\Theta(\log w)$

Sampling from distributed streams

- Maintain a (uniform) sample (w/o replacement) of size s from k streams of a total of n items

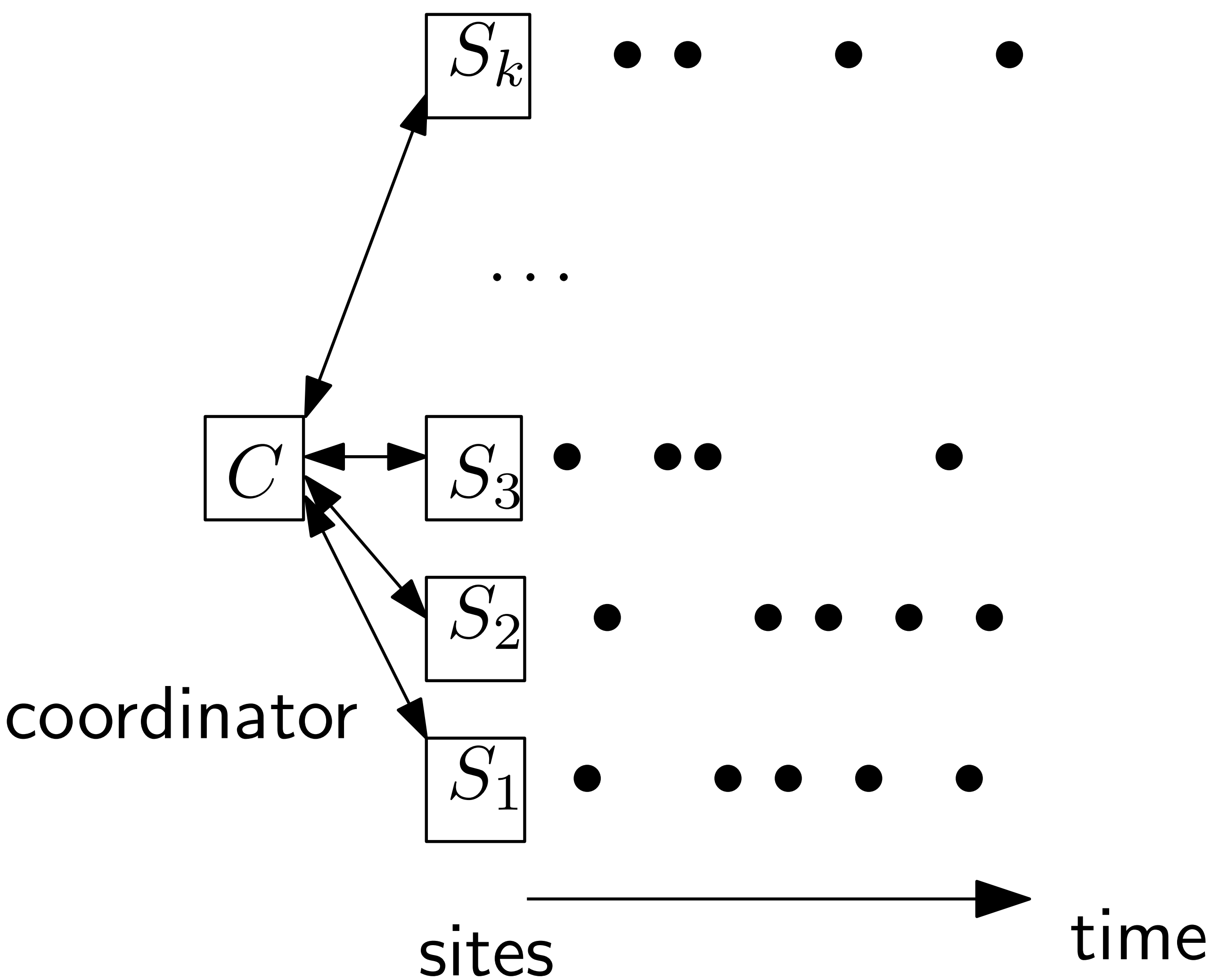


Primary goal:
communication

Secondary goal:
space/time at coordinator/site

Sampling from distributed streams

- Maintain a (uniform) sample (w/o replacement) of size s from k streams of a total of n items



Primary goal:
communication

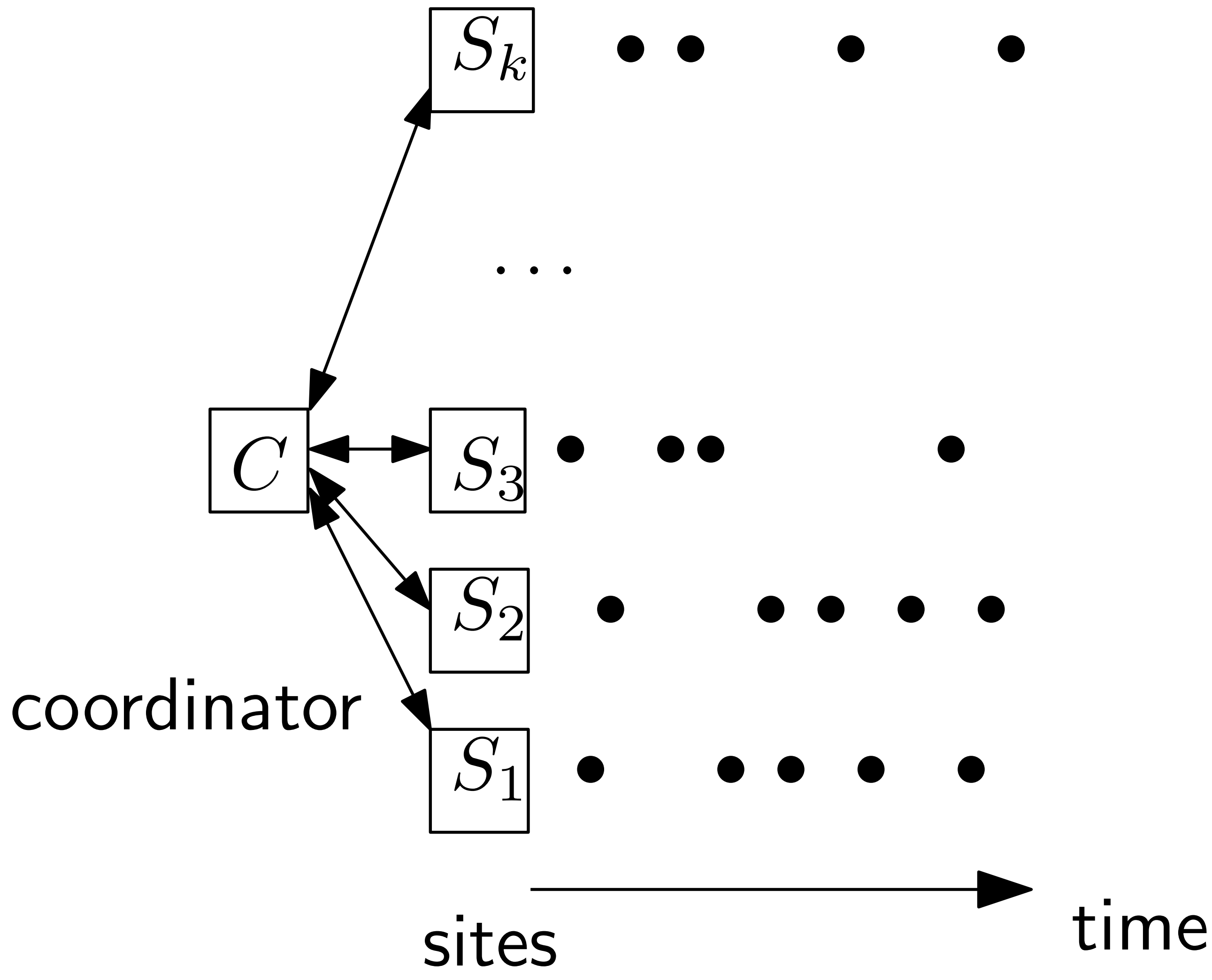
Secondary goal:
space/time at coordinator/site

Applications:

Internet routers
Sensor networks
Distributed computing

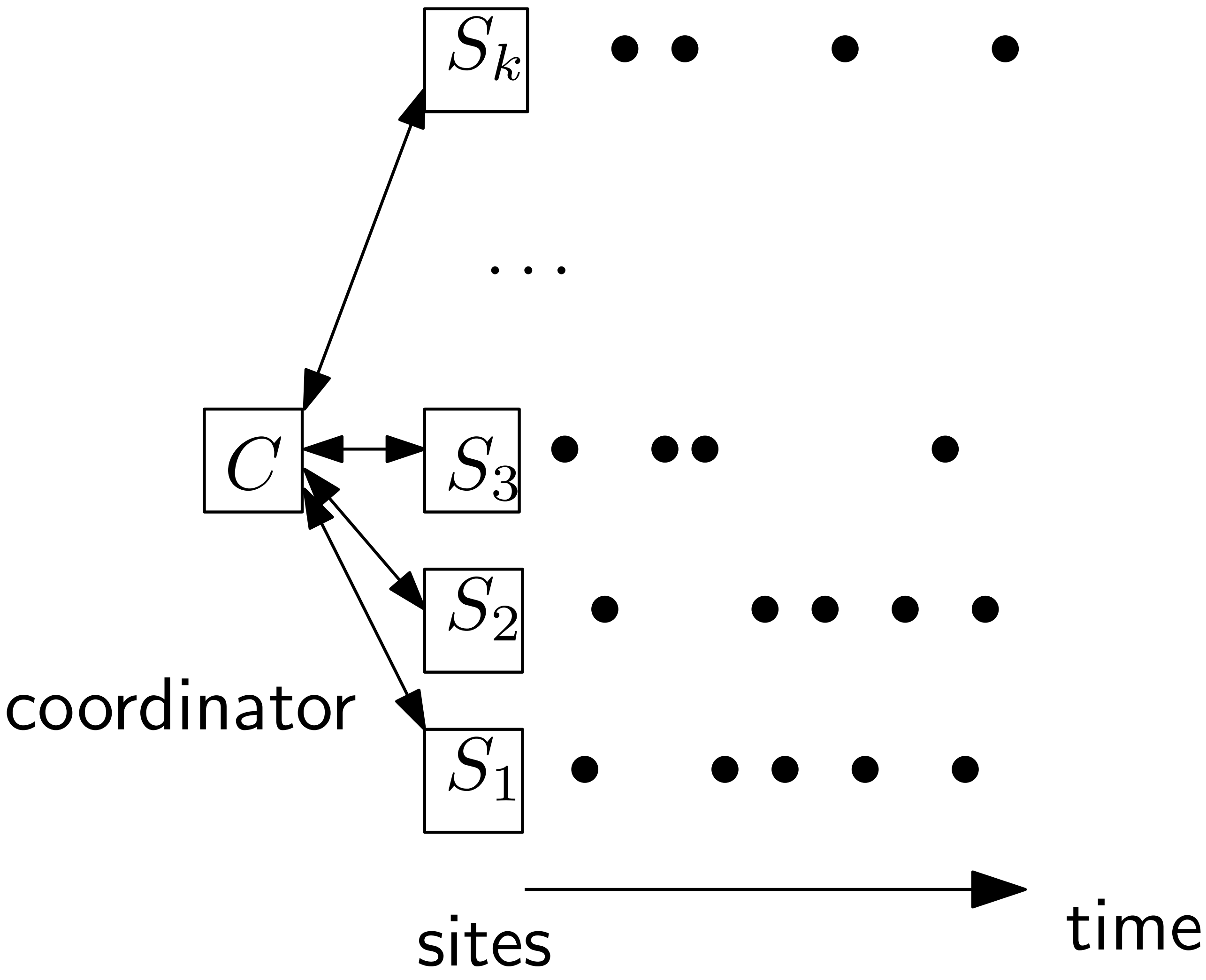
Why existing solutions don't work

- When $k = 1$, reservoir sampling has communication $\Theta(s \log n)$



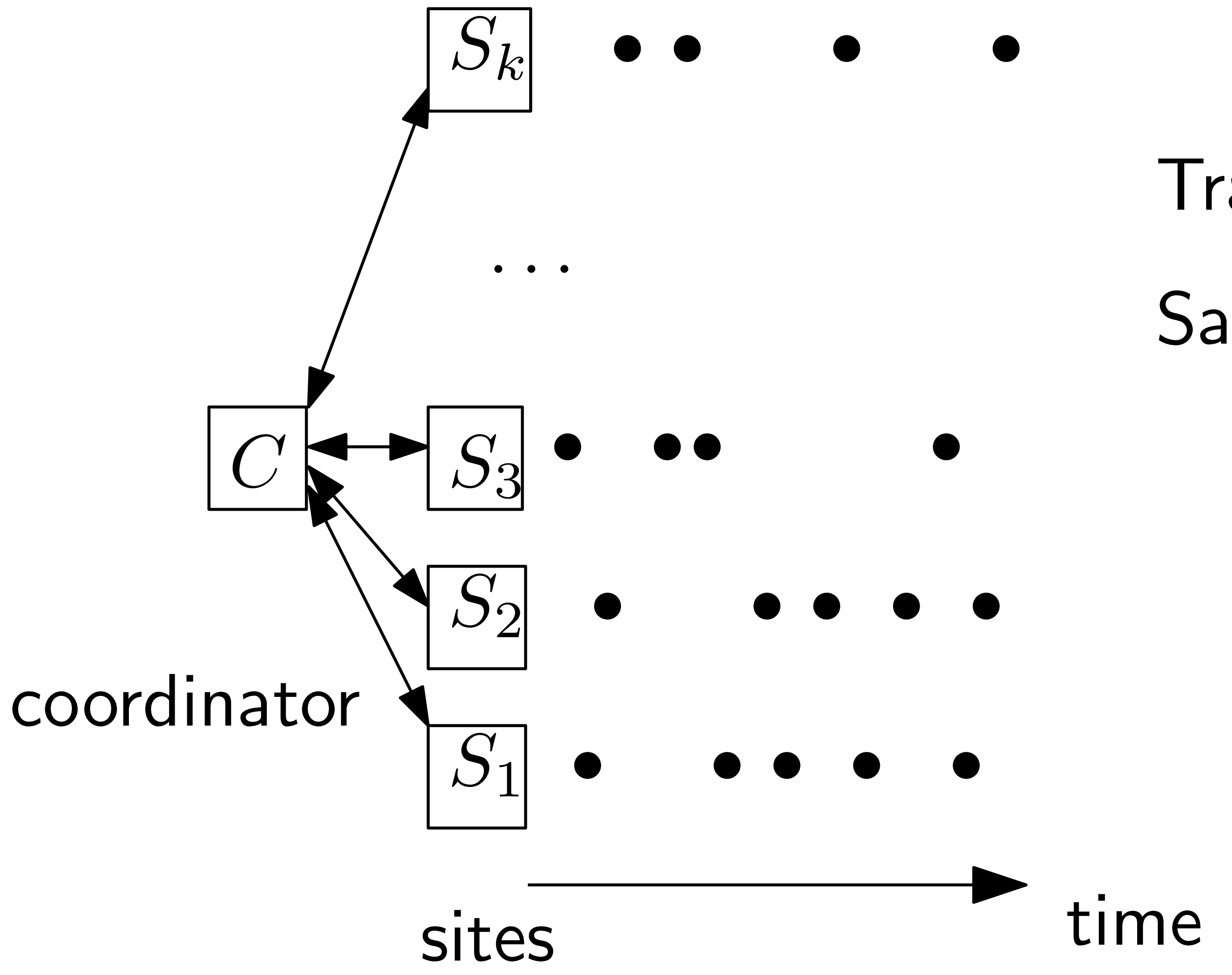
Why existing solutions don't work

- When $k = 1$, reservoir sampling has communication $\Theta(s \log n)$
- When $k \geq 2$, it has cost $O(n)$ because it's costly to track i



Why existing solutions don't work

- When $k = 1$, reservoir sampling has communication $\Theta(s \log n)$
- When $k \geq 2$, it has cost $O(n)$ because it's costly to track i

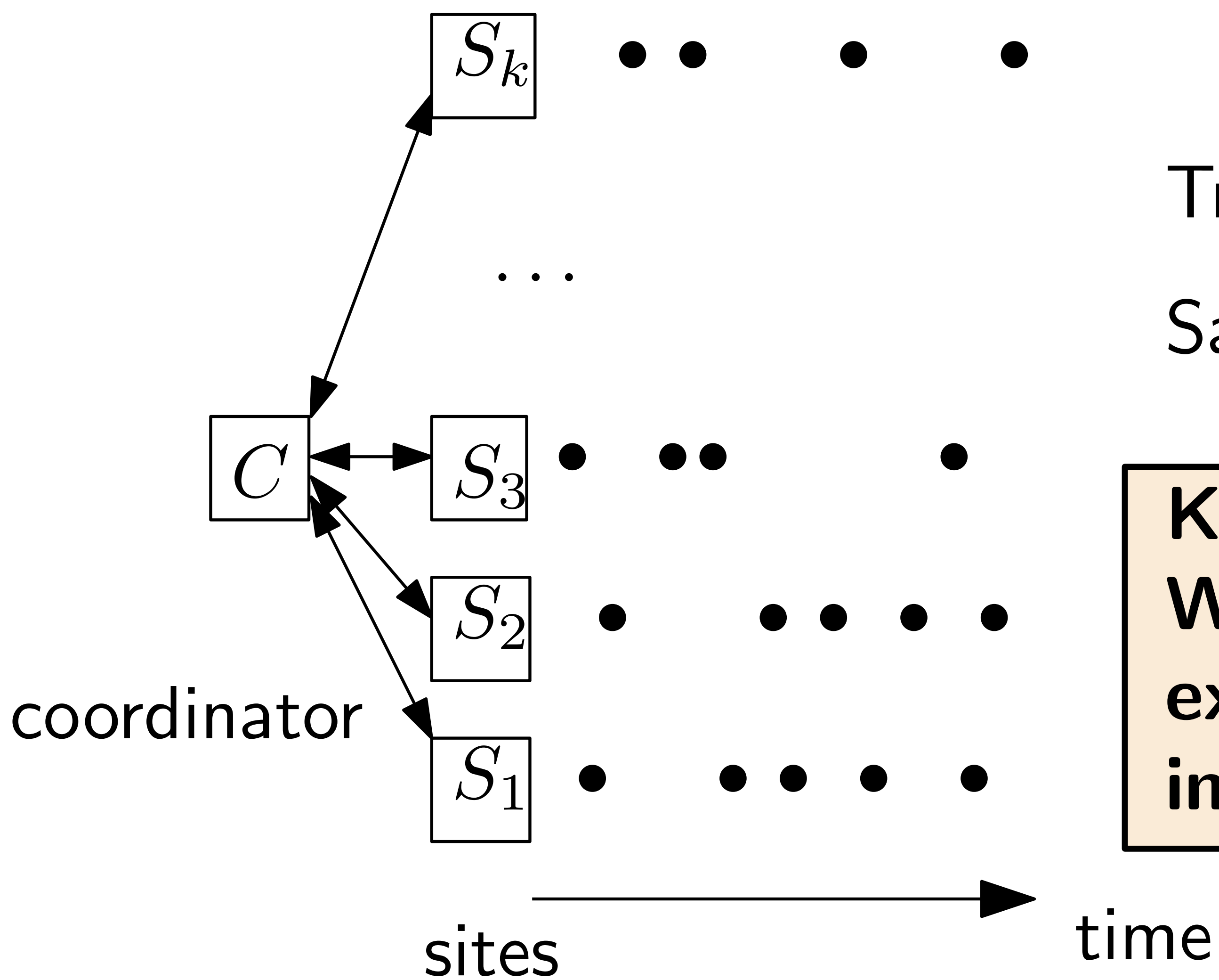


Tracking i approximately?

Sampling won't be uniform

Why existing solutions don't work

- When $k = 1$, reservoir sampling has communication $\Theta(s \log n)$
- When $k \geq 2$, it has cost $O(n)$ because it's costly to track i



Tracking i approximately?

Sampling won't be uniform

Key observation:
We don't have to know the exact size of the population in order to sample!

Previous results on distributed streaming

- A lot of heuristics in the database/networking literature
  - But random sampling has not been studied, even heuristically

Previous results on distributed streaming

- A lot of heuristics in the database/networking literature
  - But random sampling has not been studied, even heuristically
- Threshold monitoring, frequency moments [Cormode, Muthukrishnan, Yi, SODA'08]
- Entropy [Arackaparambil, Brody, Chakrabarti, ICALP'08]
- Heavy hitters and quantiles [Yi, Zhang, PODS'09]
- Basic counting, heavy hitters, quantiles in sliding windows [Chan, Lam, Lee, Ting, STACS'10]

Previous results on distributed streaming

- A lot of heuristics in the database/networking literature
  - But random sampling has not been studied, even heuristically
- Threshold monitoring, frequency moments [Cormode, Muthukrishnan, Yi, SODA'08]
- Entropy [Arackaparambil, Brody, Chakrabarti, ICALP'08]
- Heavy hitters and quantiles [Yi, Zhang, PODS'09]
- Basic counting, heavy hitters, quantiles in sliding windows [Chan, Lam, Lee, Ting, STACS'10]

- All of them are deterministic algorithms, or use randomized sketches as black boxes. And the trackings are "approximate".

Our results on random sampling

window	upper bounds	lower bounds
infinite	$O(k \log_{k/s} n + s \log n)$	$\Omega(k \log_{k/s} n + s \log n)$
sequence-based	$O(ks \log(w/s))$	$\Omega(ks \log(w/ks))$
time-based	$O((k+s) \log w)$	$\Omega(k + s \log w)$

(per window)

Our results on random sampling

window	upper bounds	lower bounds
infinite	$O(k \log_{k/s} n + s \log n)$	$\Omega(k \log_{k/s} n + s \log n)$
sequence-based	$O(ks \log(w/s))$	$\Omega(ks \log(w/ks))$
time-based	$O((k+s) \log w)$	$\Omega(k + s \log w)$

(per window)

- Applications
  - Heavy hitters and quantiles can be tracked in $\tilde{O}(k + 1/\epsilon^2)$
    Beats deterministic bound $\tilde{\Theta}(k/\epsilon)$ for $k \gg 1/\epsilon$
  - Also for sliding windows

Our results on random sampling

window	upper bounds	lower bounds
infinite	$O(k \log_{k/s} n + s \log n)$	$\Omega(k \log_{k/s} n + s \log n)$
sequence-based	$O(ks \log(w/s))$	$\Omega(ks \log(w/ks))$
time-based	$O((k+s)\log w)$	$\Omega(k + s \log w)$

(per window)

- Applications
  - Heavy hitters and quantiles can be tracked in $\tilde{O}(k + 1/\epsilon^2)$
    Beats deterministic bound $\tilde{\Theta}(k/\epsilon)$ for $k \gg 1/\epsilon$
  - Also for sliding windows
  - ϵ-approximations in bounded VC dimensions: $\tilde{O}(k + 1/\epsilon^2)$
  - ϵ-nets: $\tilde{O}(k + 1/\epsilon)$
  - $\cdots$

ISWoR

- The protocol
  - **Site:** always maintains an upper bound u (initialized to be 1) and lower bound l (initialized to be 0), and only sends those items with rank in the range $[l, u]$.

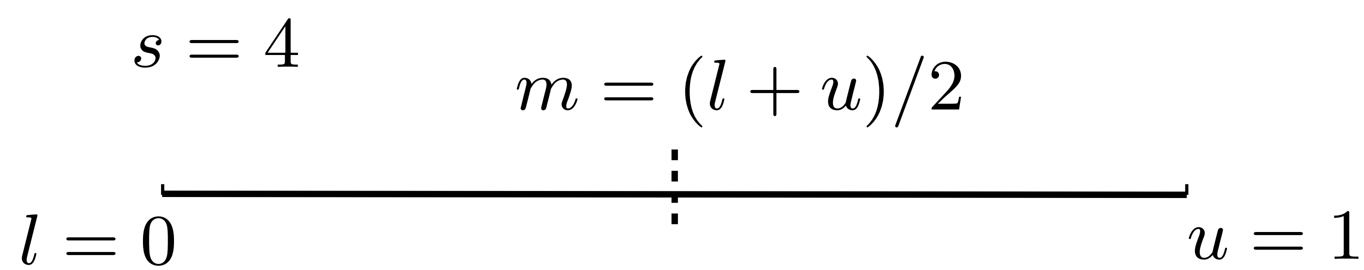
Rank: for each item coming, generate a random number in $[0, 1]$ as its rank.

ISWoR

- The protocol
  - **Site:** always maintains an upper bound u (initialized to be 1) and lower bound l (initialized to be 0), and only sends those items with rank in the range $[l, u]$.
  - **Coordinator:** let $m = (l + u)/2$, waits until
    - # items receiced in the range $[l, m]$ becomes $\geq s$, updates each site with $u = m$.
    - # items receiced in the range $[m, u]$ becomes $\geq s$, updates each site with $l = m$.

  Report:
  subsamples s items from all items in $[l, u]$.

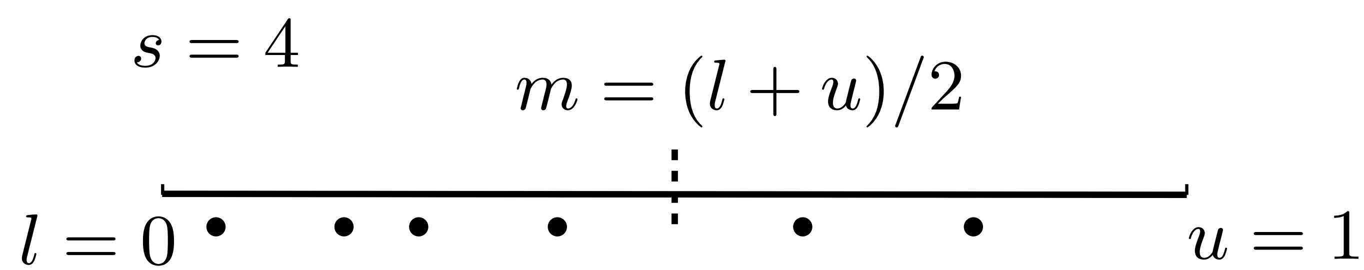
ISWoR



- The protocol
  - **Site:** always maintains an upper bound u (initialized to be 1) and lower bound l (initialized to be 0), and only sends those items with rank in the range $[l, u]$.
  - **Coordinator:** let $m = (l+u)/2$, waits until
    - # items receiced in the range $[l, m]$ becomes $\geq s$, updates each site with $u = m$.
    - # items receiced in the range $[m, u]$ becomes $\geq s$, updates each site with $l = m$.

  Report:
  subsamples s items from all items in $[l, u]$.

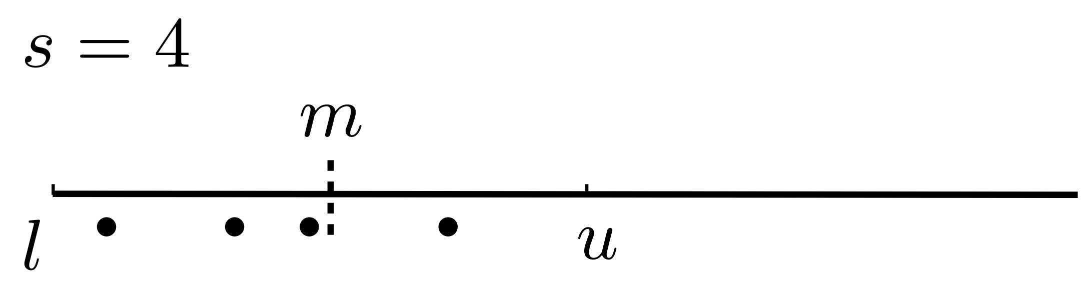
ISWoR



- The protocol
  - **Site:** always maintains an upper bound u (initialized to be 1) and lower bound l (initialized to be 0), and only sends those items with rank in the range $[l, u]$.
  - **Coordinator:** let $m = (l + u)/2$, waits until
    - # items receiced in the range $[l, m]$ becomes $\geq s$, updates each site with $u = m$.
    - # items receiced in the range $[m, u]$ becomes $\geq s$, updates each site with $l = m$.

    Report:
    subsamples s items from all items in $[l, u]$.

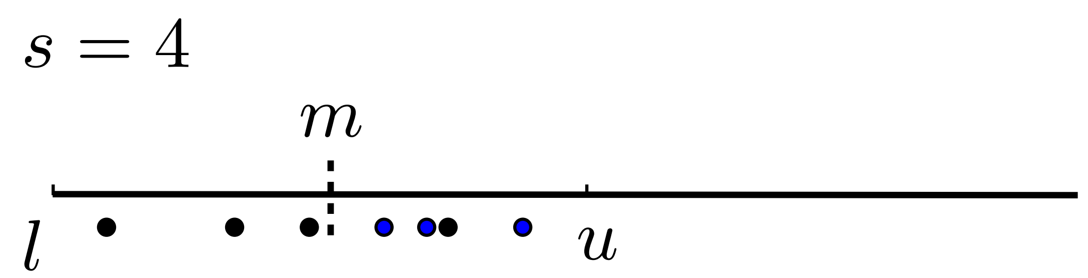
ISWoR



- The protocol
  - **Site:** always maintains an upper bound u (initialized to be 1) and lower bound l (initialized to be 0), and only sends those items with rank in the range $[l, u]$.
  - **Coordinator:** let $m = (l + u)/2$, waits until
    - $\#$ items receiced in the range $[l, m]$ becomes $\geq s$, updates each site with $u = m$.
    - $\#$ items receiced in the range $[m, u]$ becomes $\geq s$, updates each site with $l = m$.

  Report: subsamples s items from all items in $[l, u]$.

ISWoR

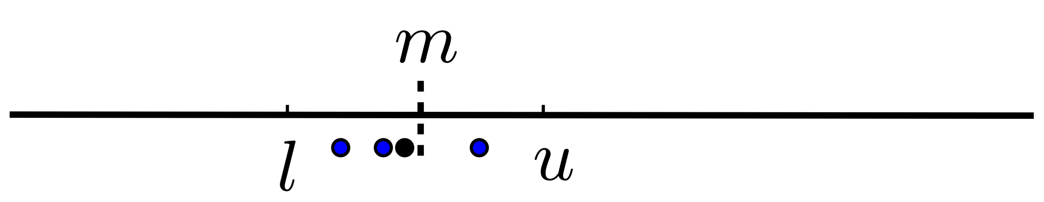


- The protocol
  - **Site:** always maintains an upper bound u (initialized to be 1) and lower bound l (initialized to be 0), and only sends those items with rank in the range $[l, u]$.
  - **Coordinator:** let $m = (l + u)/2$, waits until
    - # items receiced in the range $[l, m]$ becomes $\geq s$, updates each site with $u = m$.
    - # items receiced in the range $[m, u]$ becomes $\geq s$, updates each site with $l = m$.

    Report:
    subsamples s items from all items in $[l, u]$.

ISWoR

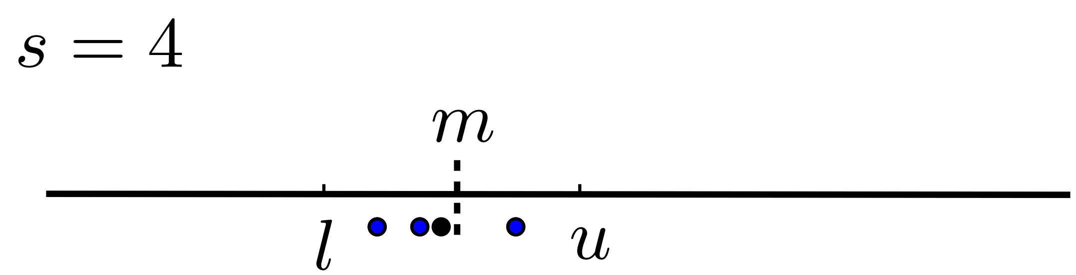
$s = 4$



- The protocol
  - **Site:** always maintains an upper bound u (initialized to be 1) and lower bound l (initialized to be 0), and only sends those items with rank in the range $[l, u]$.
  - **Coordinator:** let $m = (l + u)/2$, waits until
    - $\#$ items receiced in the range $[l, m]$ becomes $\geq s$, updates each site with $u = m$.
    - $\#$ items receiced in the range $[m, u]$ becomes $\geq s$, updates each site with $l = m$.

  Report:
  subsamples s items from all items in $[l, u]$.

ISWoR

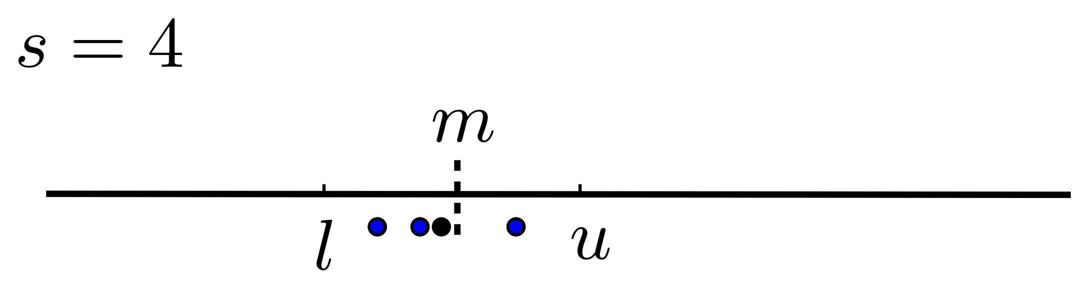


- The protocol
  - **Site:** always maintains an upper bound u (initialized to be 1) and lower bound l (initialized to be 0), and only sends those items with rank in the range $[l, u]$.
  - **Coordinator:** let $m = (l + u)/2$, waits until
    - \# items receiced in the range $[l, m]$ becomes $\geq s$, updates each site with $u = m$.
    - \# items receiced in the range $[m, u]$ becomes $\geq s$, updates each site with $l = m$.

  Report:
  subsamples s items from all items in $[l, u]$.

Like Binary Search :)

ISWoR



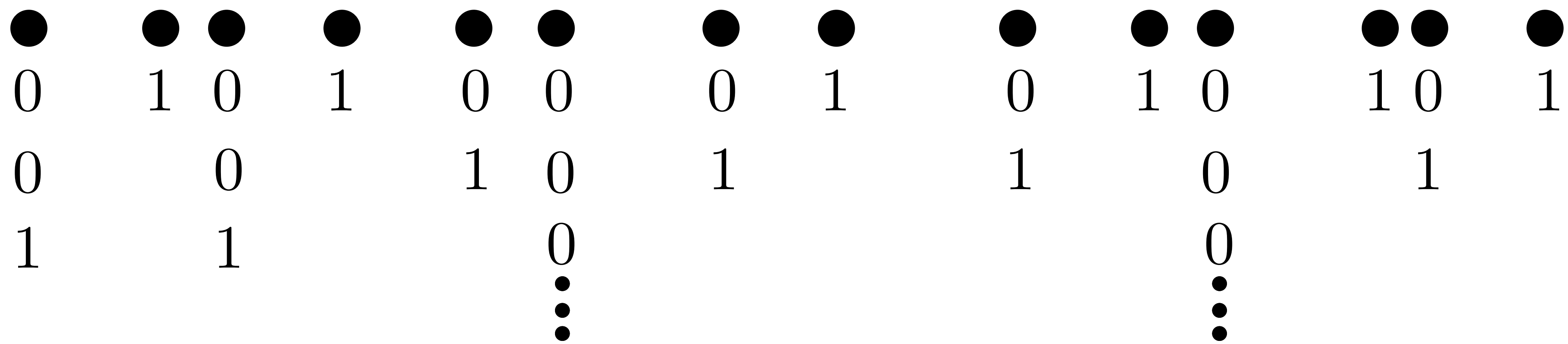
- The protocol
  - **Site:** always maintains an upper bound u (initialized to be 1) and lower bound l (initialized to be 0), and only sends those items with rank in the range $[l, u]$.
  - **Coordinator:** let $m = (l + u)/2$, waits until
    - # items receiced in the range $[l, m]$ becomes $\geq s$, updates each site with $u = m$.
    - # items receiced in the range $[m, u]$ becomes $\geq s$, updates each site with $l = m$.

  Report: subsamples s items from all items in $[l, u]$.

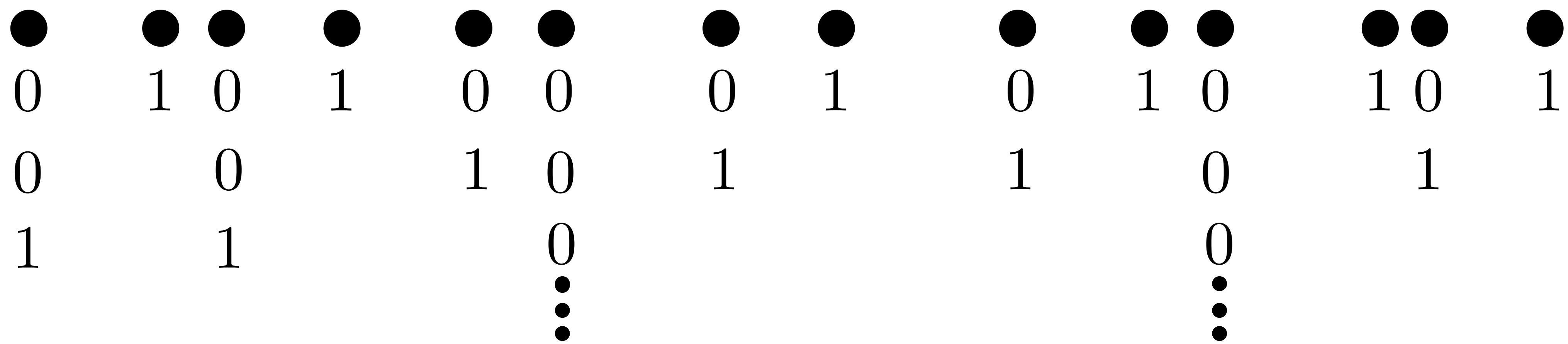
Communication cost: $O((k + s) \log n)$

The basic idea: Binary Bernoulli sampling

The basic idea: Binary Bernoulli sampling

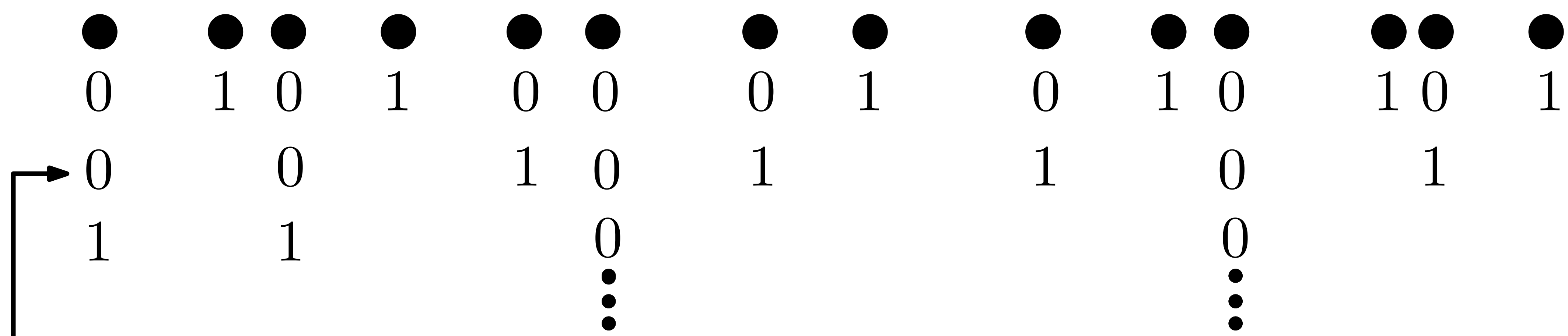


The basic idea: Binary Bernoulli sampling



Conditioned upon a row having $\geq s$ active items, we can draw a sample from the active items

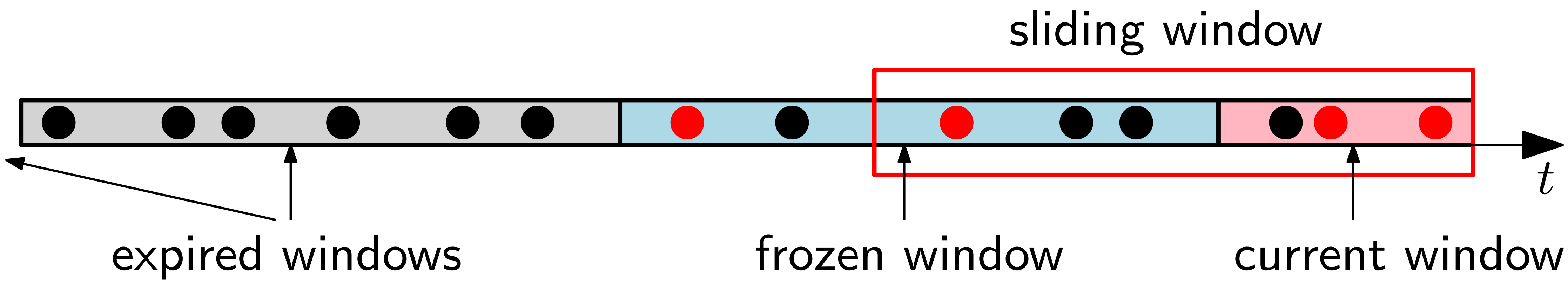
The basic idea: Binary Bernoulli sampling



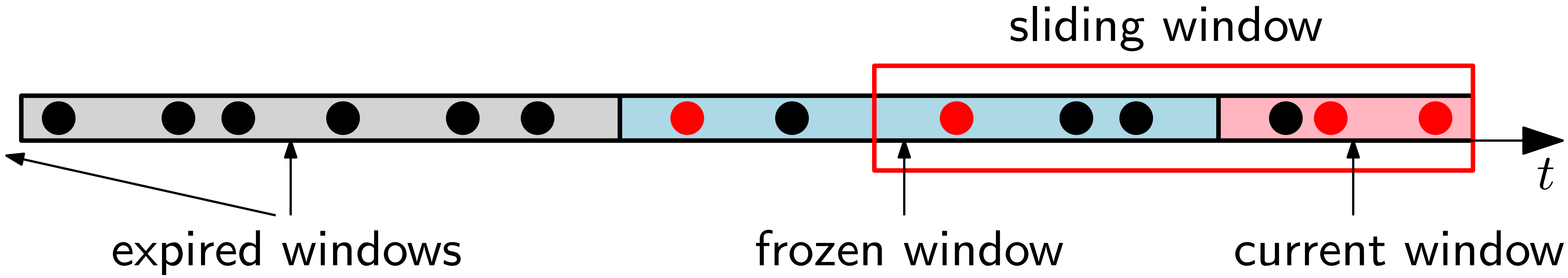
Conditioned upon a row having $\geq s$ active items, we can draw a sample from the active items

The coordinator could maintain a Bernoulli sample of size between s and $O(s)$

Sampling from a sliding window: Idea



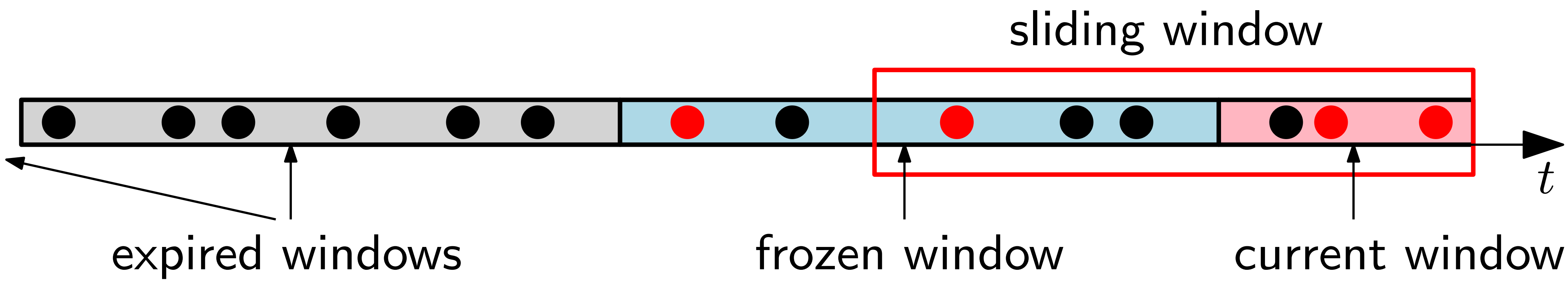
Sampling from a sliding window: Idea



- Sample for sliding window =
  (1) a subsample of the (unexpired) sample of frozen window +
  (2) a subsample of the sample of current window by ISWoR

need new ideas

Sampling from a sliding window: Idea

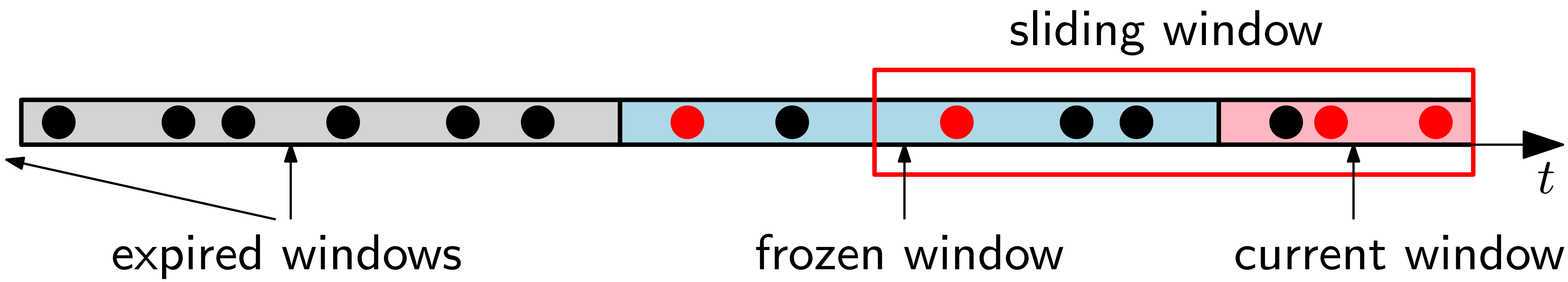


- Sample for sliding window =
  (1) a subsample of the (unexpired) sample of frozen window +
  (2) a subsample of the sample of current window by ISWoR

  need new ideas

  - (1), (2) may be sampled by different rates.
    But as long as both have sizes $\geq \min\{s, \#$ live items$\}$, fine.

Sampling from a sliding window: Idea



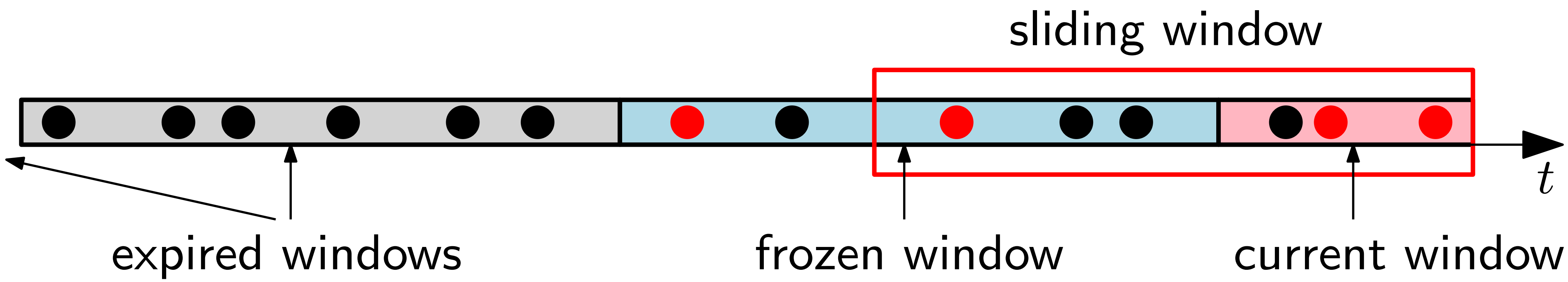
- Sample for sliding window $=$
  (1) a subsample of the (unexpired) sample of frozen window $+$
  (2) a subsample of the sample of current window by ISWoR

  need new ideas

  - (1), (2) may be sampled by different rates.
    But as long as both have sizes $\geq \min\{s, \#$ live items$\}$, fine.

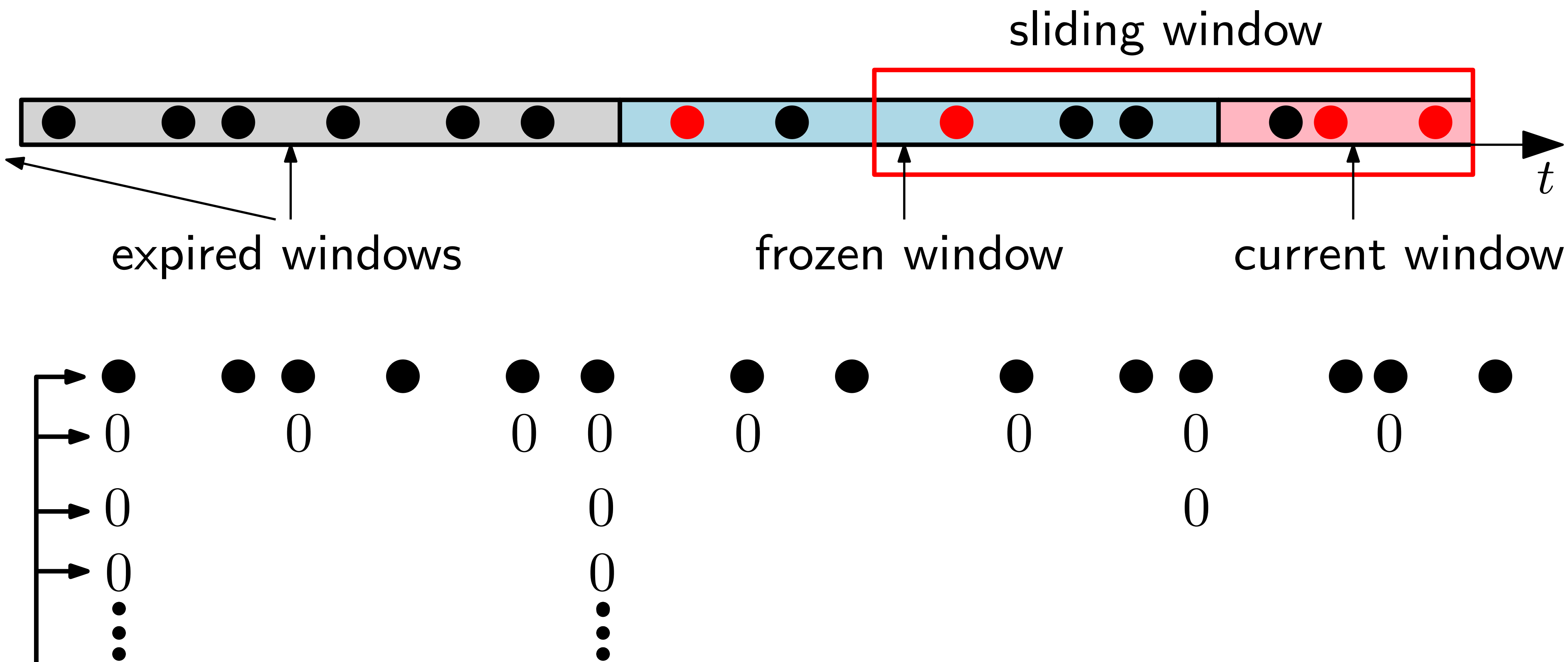
- The key issue: how to guarantee "both have sizes $\geq s$"?
  as items in the frozen window are expiring ...

Sampling from a sliding window: Idea



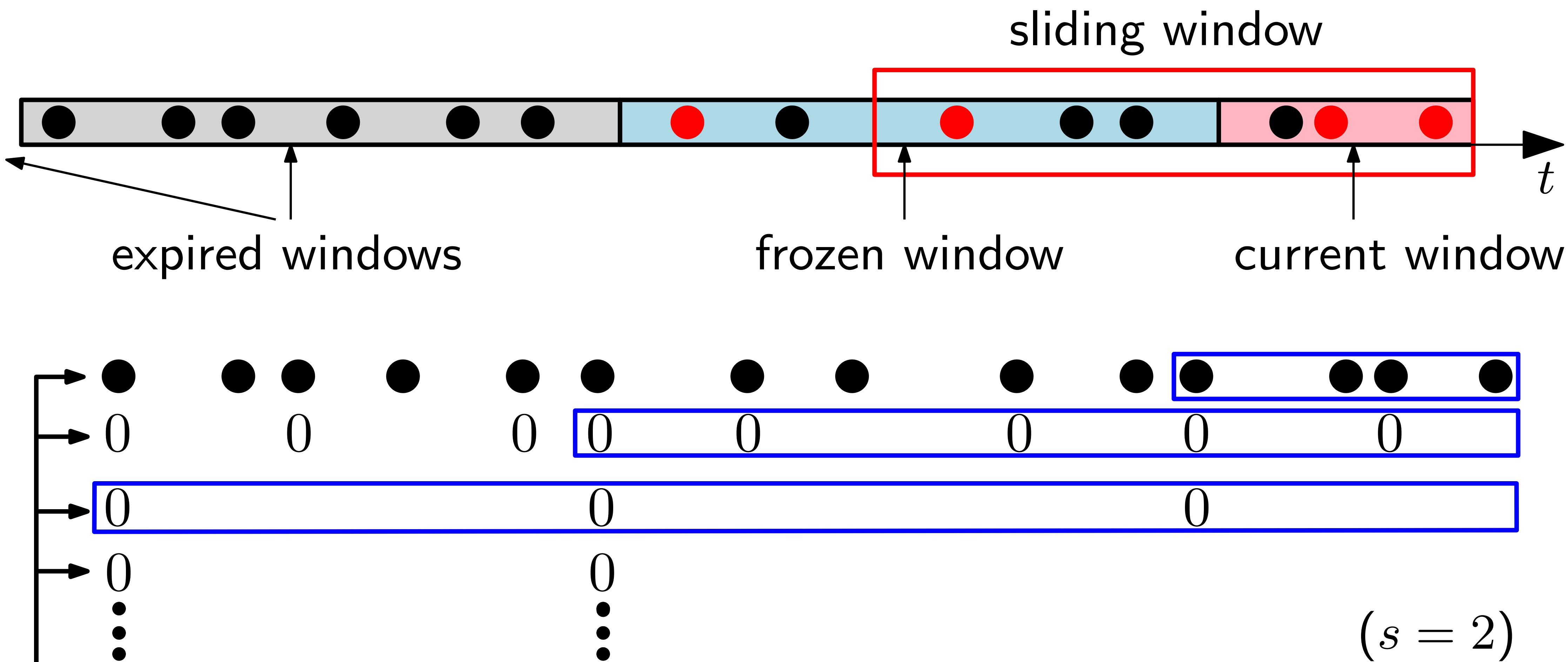
- Sample for sliding window $=$
  (1) a subsample of the (unexpired) sample of frozen window $+$
  (2) a subsample of the sample of current window by ISWoR
  need new ideas
  - (1), (2) may be sampled by different rates.
    But as long as both have sizes $\geq \min\{s, \#$ live items$\}$, fine.
- The key issue: how to guarantee "both have sizes $\geq s$"?
  as items in the frozen window are expiring ...
  - Solution: In the frozen window, find a good sample rate such that the sample size $\geq s$.

Dealing with the frozen window



Keep all the levels? Need $O(w)$ communication

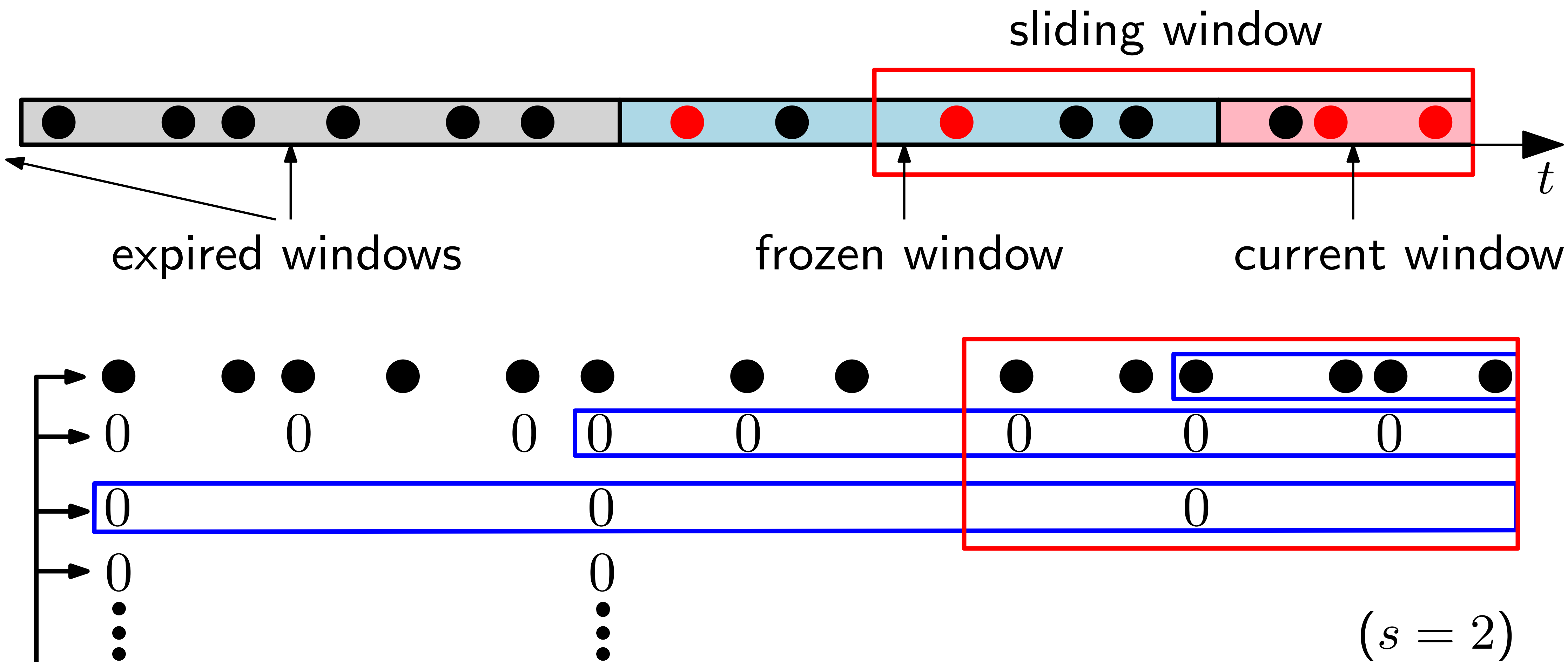
Dealing with the frozen window



Keep all the levels? Need $O(w)$ communication

Keep most recent sampled items in a level until s of them are also sampled at the next level. Total size: $O(s \log w)$

Dealing with the frozen window

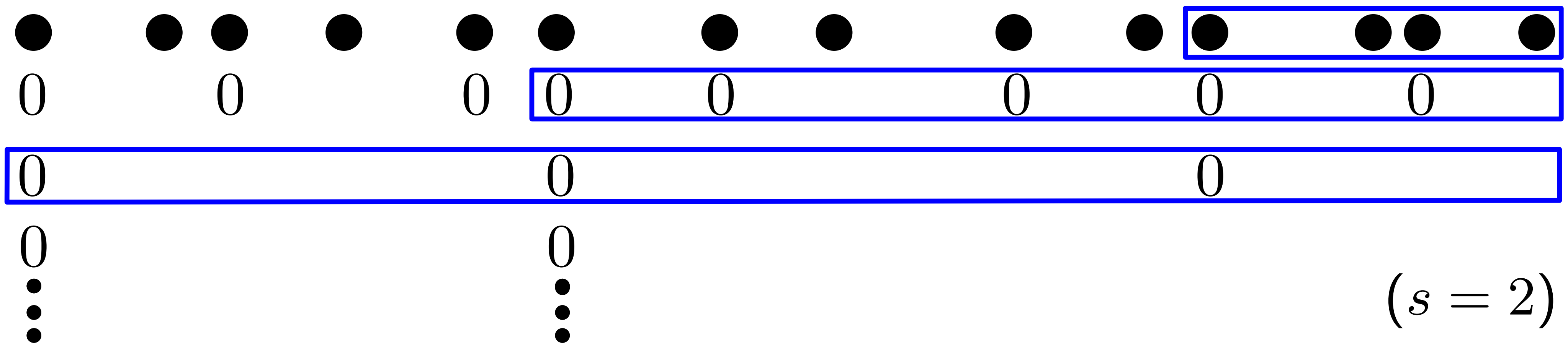


Keep all the levels? Need $O(w)$ communication

Keep most recent sampled items in a level until s of them are also sampled at the next level. Total size: $O(s \log w)$

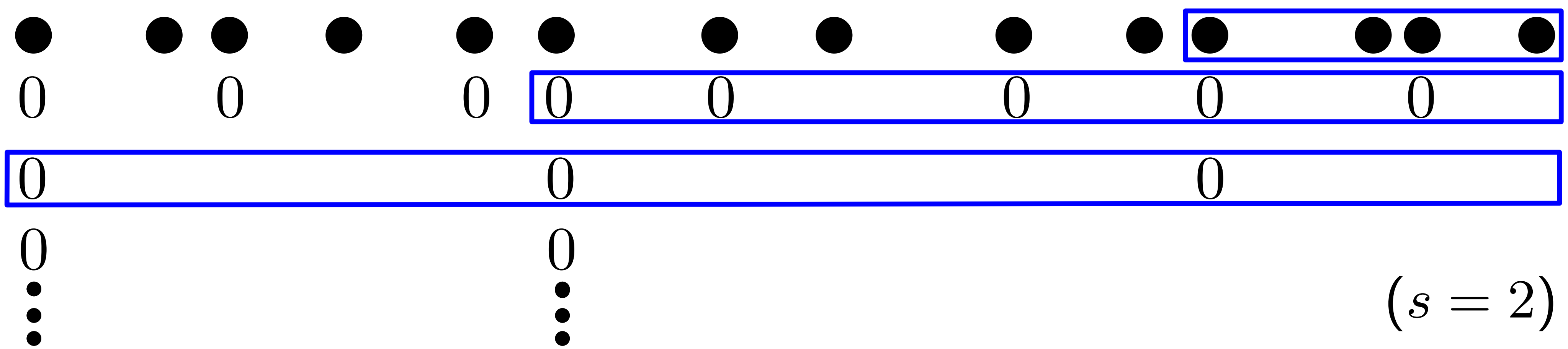
Guaranteed: There is a blue window with $\geq s$ sampled items that covers the unexpired portion of the frozen window

Dealing with the frozen window: The algorithm



- Each site builds its own level-sampling structure for the current window until it freezes
  - Needs $O(s \log w)$ space and $O(1)$ time per item

Dealing with the frozen window: The algorithm



- Each site builds its own level-sampling structure for the current window until it freezes
  - Needs $O(s \log w)$ space and $O(1)$ time per item

- When the current window freezes
  - For each level, do a k-way merge to build the level of the global structure at the coordinator. Total communication $O((k+s)\log w)$

Other results

- Similar results hold for sampling with replacement (WR)
  - There is a simple reduction from sampling WR to sampling WoR, but need to know n.

Other results

- Similar results hold for sampling with replacement (WR)
  - There is a simple reduction from sampling WR to sampling WoR, but need to know n.
  - Need some new ideas

Other results

- Similar results hold for sampling with replacement (WR)
  - There is a simple reduction from sampling WR to sampling WoR, but need to know n.
  - Need some new ideas
- Processing time per item is another complicated issue for WR. Finally we can get $O(1)$ (but complicated).
- Experiments show that our algorithms work well.

Future directions

- Direct applications
  - Heavy hitters and quantiles can be tracked in $\tilde{O}(k + 1/\epsilon^2)$
    Beats deterministic bound $\tilde{\Theta}(k/\epsilon)$ for $k \gg 1/\epsilon$
  - Also for sliding windows
  - ϵ-approximations in bounded VC dimensions: $\tilde{O}(k + 1/\epsilon^2)$
  - ϵ-nets: $\tilde{O}(k + 1/\epsilon)$
  - $\cdots$

Future directions

- Direct applications
  - Heavy hitters and quantiles can be tracked in $\tilde{O}(k + 1/\epsilon^2)$
    Beats deterministic bound $\tilde{\Theta}(k/\epsilon)$ for $k \gg 1/\epsilon$
  - Also for sliding windows
  - ϵ-approximations in bounded VC dimensions: $\tilde{O}(k + 1/\epsilon^2)$
  - ϵ-nets: $\tilde{O}(k + 1/\epsilon)$
  - $\cdots$
- Is random sampling the best way to solve these problems?

Future directions

- Direct applications
  - Heavy hitters and quantiles can be tracked in $\tilde{O}(k + 1/\epsilon^2)$
    Beats deterministic bound $\tilde{\Theta}(k/\epsilon)$ for $k \gg 1/\epsilon$
  - Also for sliding windows
  - ϵ-approximations in bounded VC dimensions: $\tilde{O}(k + 1/\epsilon^2)$
  - ϵ-nets: $\tilde{O}(k + 1/\epsilon)$
  - $\cdots$
- Is random sampling the best way to solve these problems?
  - New result: Heavy hitters and quantiles can be tracked in $\tilde{O}(k + \sqrt{k}/\epsilon)$, using a different sampling method

Future directions

- Direct applications
  - Heavy hitters and quantiles can be tracked in $\tilde{O}(k + 1/\epsilon^2)$
    Beats deterministic bound $\tilde{\Theta}(k/\epsilon)$ for $k \gg 1/\epsilon$
  - Also for sliding windows
  - ϵ-approximations in bounded VC dimensions: $\tilde{O}(k + 1/\epsilon^2)$
  - ϵ-nets: $\tilde{O}(k + 1/\epsilon)$
  - $\cdots$
- Is random sampling the best way to solve these problems?
  - New result: Heavy hitters and quantiles can be tracked in $\tilde{O}(k + \sqrt{k}/\epsilon)$, using a different sampling method
- Other problems: range-counting, extent measures, etc.

Multiparty private message CC

- Before, multiparty communication complexities are mainly used for other applications.
  - Number on the forehead
  - Public message
  - One-way communication

  • • •

Multiparty private message CC

- Before, multiparty communication complexities are mainly used for other applications.
  - Number on the forehead
  - Public message
  - One-way communication

  • • •
- But surprisingly, the most general, natural setting – "private message model" – has not been studied!
  - Possible reason: before "distributed streaming model", no direct application.

Multiparty private message CC

- Before, multiparty communication complexities are mainly used for other applications.
  - Number on the forehead
  - Public message
  - One-way communication

    • • •
- But surprisingly, the most general, natural setting – "private message model" – has not been studied!
  - Possible reason: before "distributed streaming model", no direct application.
- Now, it is the time!

The End

THANK YOU

Q and A