IOWA STATE UNIVERSITY

Department of Electrical and Computer Engineering

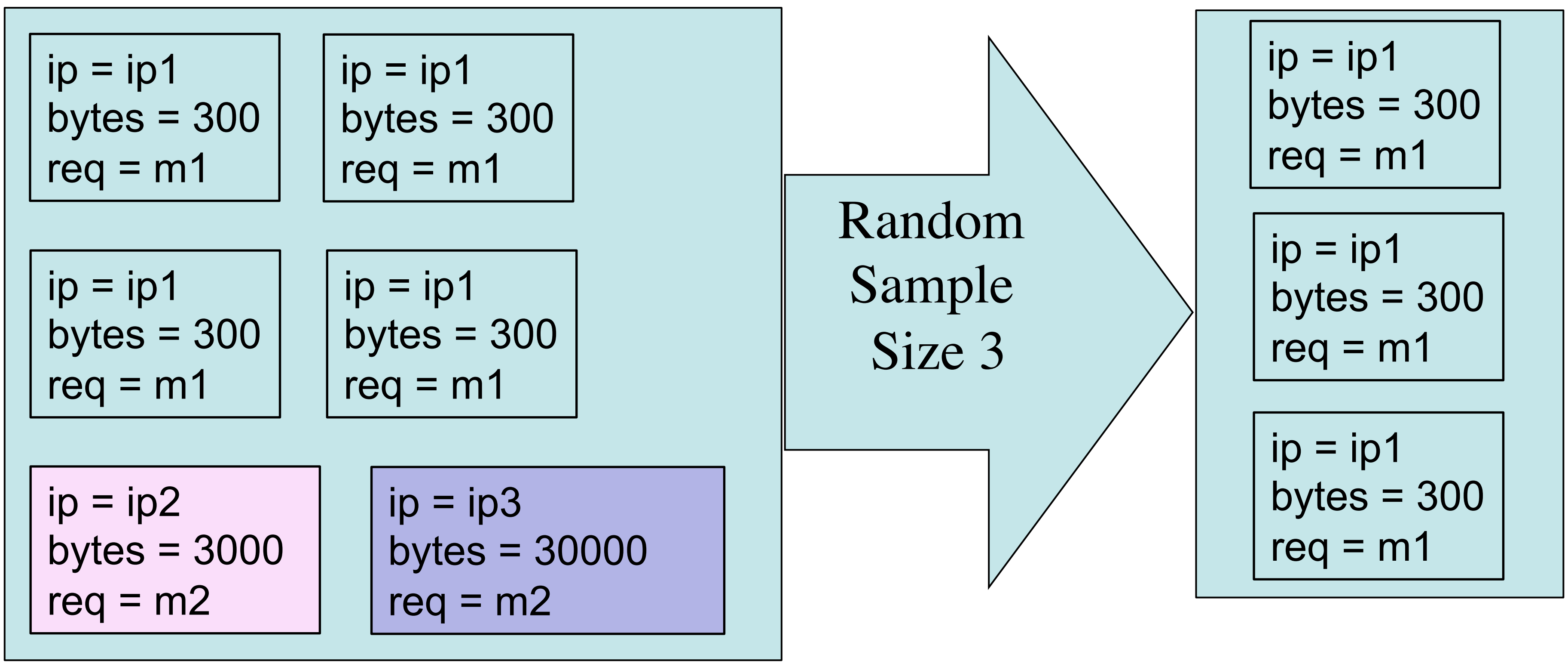
Distinct Random Sampling from a Distributed Stream

Yung-Yu Chung, Srikanta Tirthapura

Electrical and Computer Engineering

Iowa Sate University

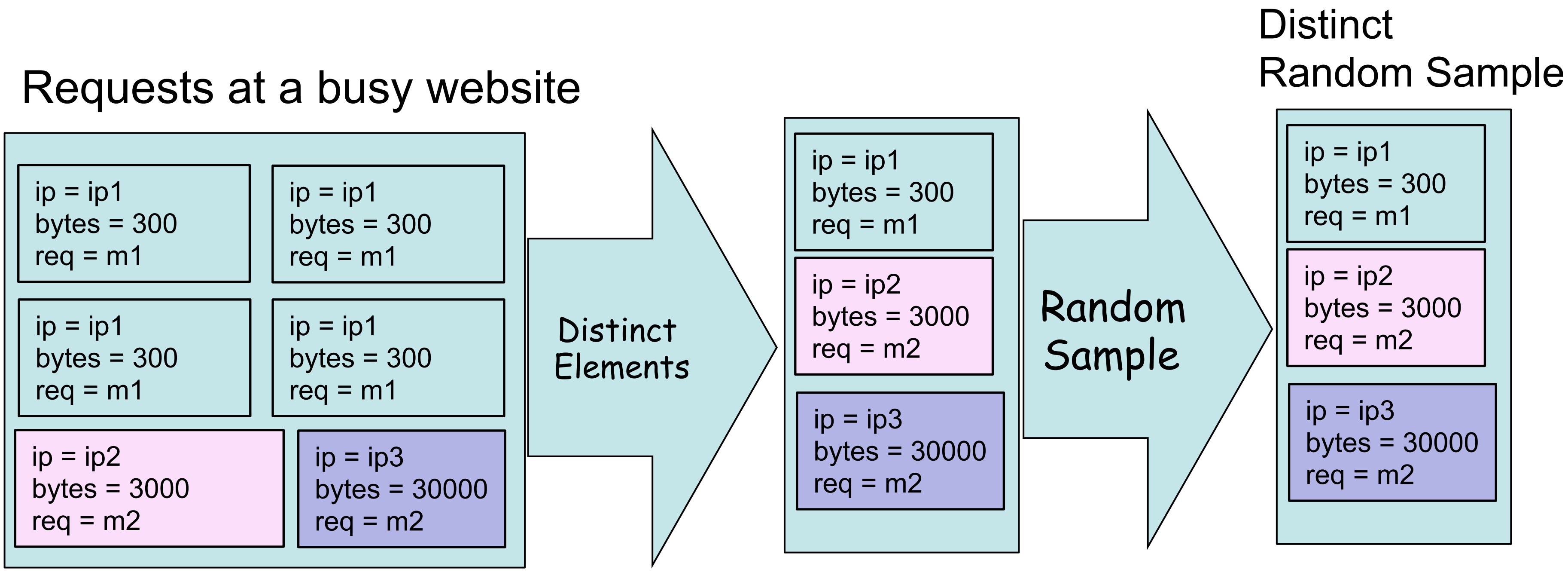
Random Sampling from Data



Requests at a busy website

Distinct Random Sampling

Sample from the Set of Distinct Elements within data



Conceptual View

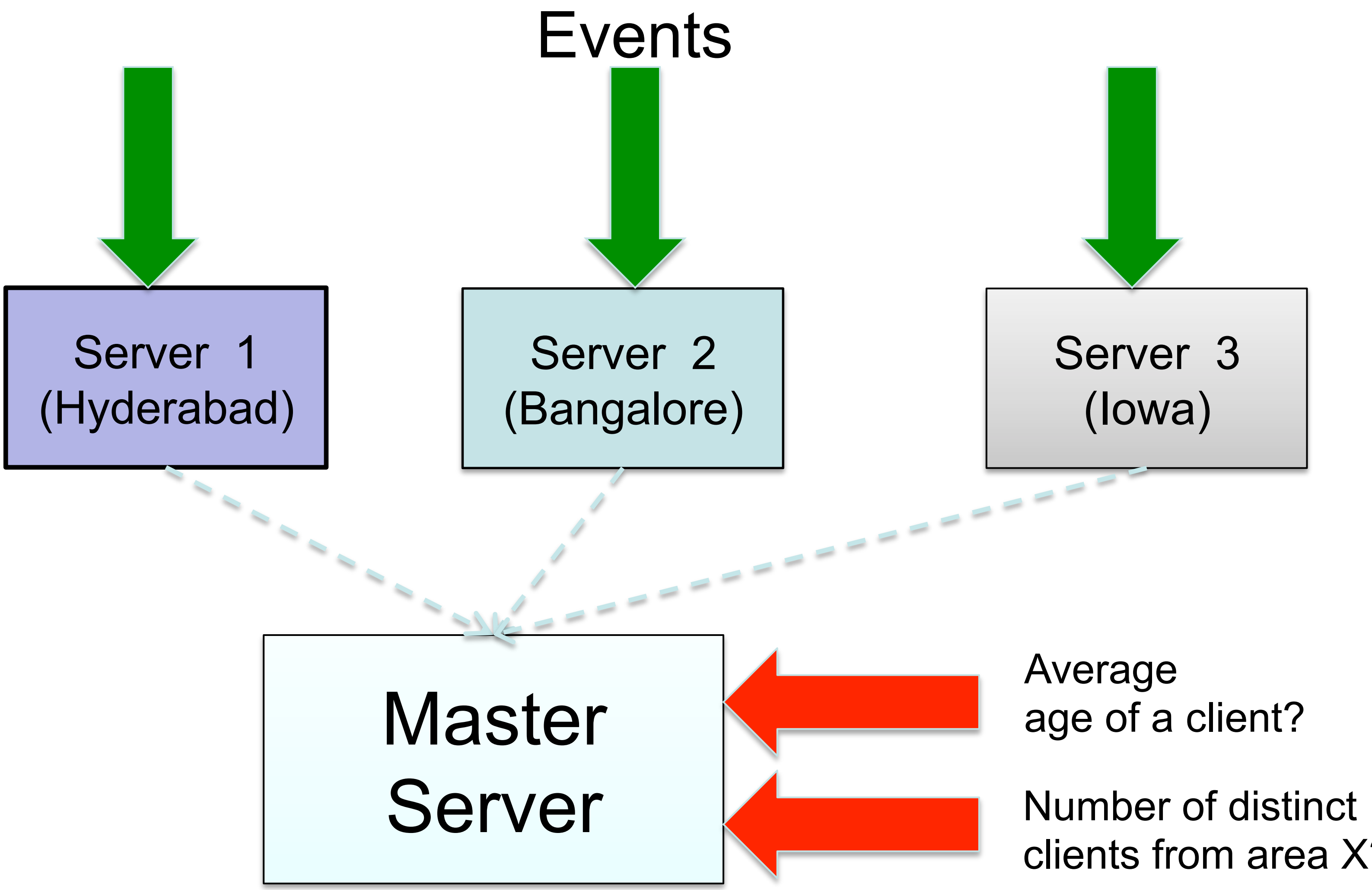
Problem at a High Level

Continuously maintain a distinct random sample from a data source whose elements are arriving as continuous stream of updates at multiple geographically distributed sites.

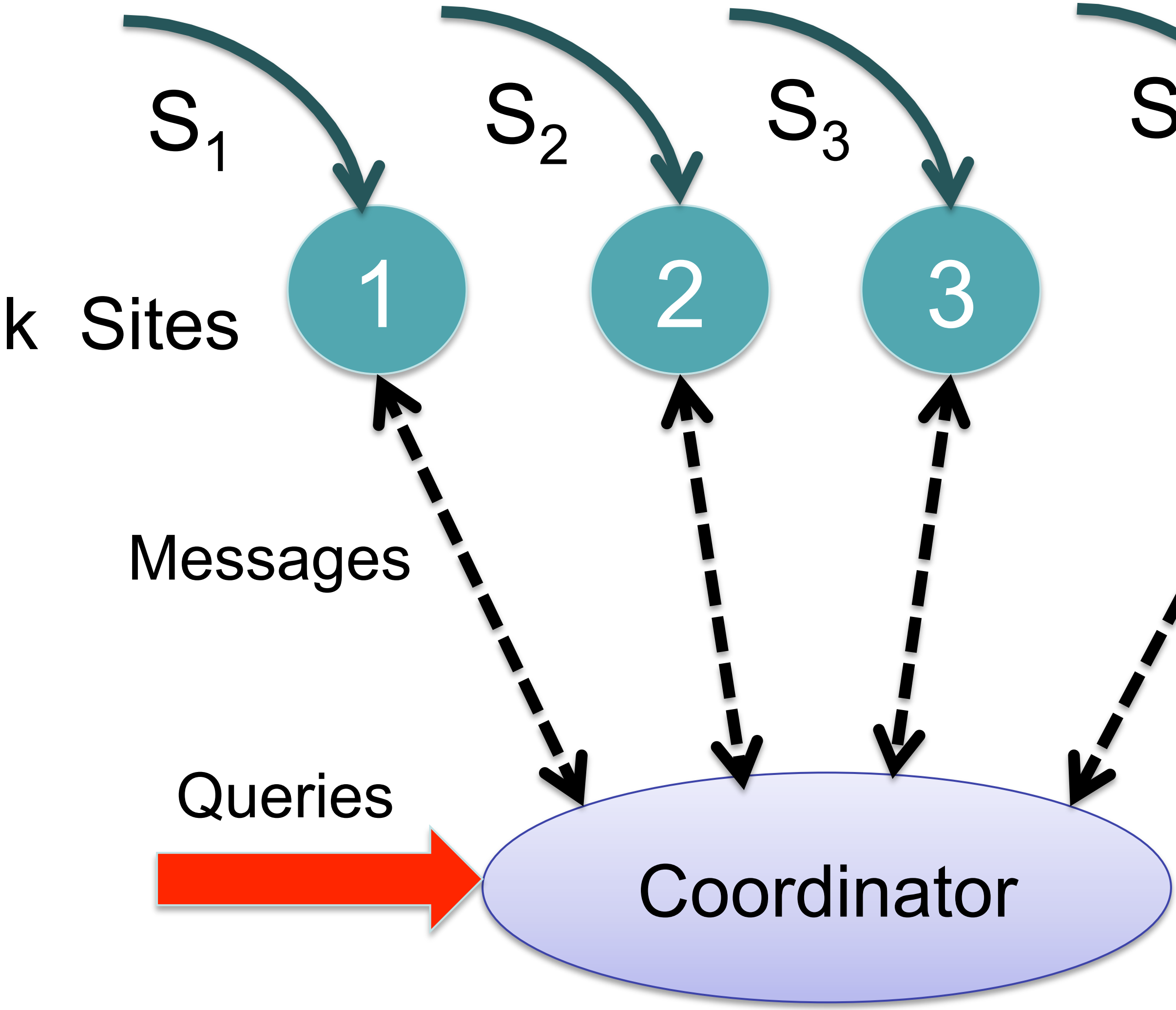
Importance of Distinct Random Sampling

- Network Anomaly Detection (Venkataraman et al. NDSS 2005)
- Database Query Optimization (Ganguly et al. VLDB 2005, Poddar 2011, Gibbons VLDB 2001)
- Sampling Operators a core part of Streaming Systems
  - Sampling Algorithms in a Stream Operator (Johnson et al. SIGMOD 2005)
  - IBM Infosphere Streams
- BlinkDB "Big Data" database is based on random sampling for approximate answers (Agarwal et al. Eurosys 2013)

Distributed Streams



Distinct Sampling Problem Definition (1)



Let $S = S_1 \cup S_2 \cup \ldots \cup S_k$
Let distinct(S) be set of distinct elements in S

Task: continuously maintain a random sample of size s chosen without replacement from distinct(S)

Distinct Sampling Problem Definition (2)

- Cost: Number of messages transmitted between sites and coordinator

- Synchronous Model
  - Execution proceeds in rounds
  - In each round, each site observes one or more items, and can send a message to coordinator, receive a response

- Two Versions:
  - Infinite Window: Sample drawn from all items seen so far
  - Sliding Window: Sample drawn from items seen in recent rounds

Our Results (Upper Bound)

An algorithm that continuously maintains a distinct sample from a distributed stream S, with the following performance guarantees

- Expected total messages for processing all of S is $2ks \ln (de/s)$
- O(s) memory consumption per site
- O(s) memory at the coordinator, and
- O(1) processing time per element

k = number of sites
d = size of distinct(S)
s = sample size desired

Our Results (Lower Bound)

For any algorithm A and parameter d, there exists an input distributed stream, I_A with d distinct elements such that the expected number of messages sent by the algorithm upon receiving I_A is at least $(ks/2) \ln (de/s)$

k = number of sites
d = size of distinct(S)
s = sample size desired

Results Summary

	Our Algorithm	Lower Bound
Number of Messages	2ks ln (de/s)	(ks/2) ln (de/s)
Memory at Coordinator (in words)	$O(s)$	$\Omega(s)$

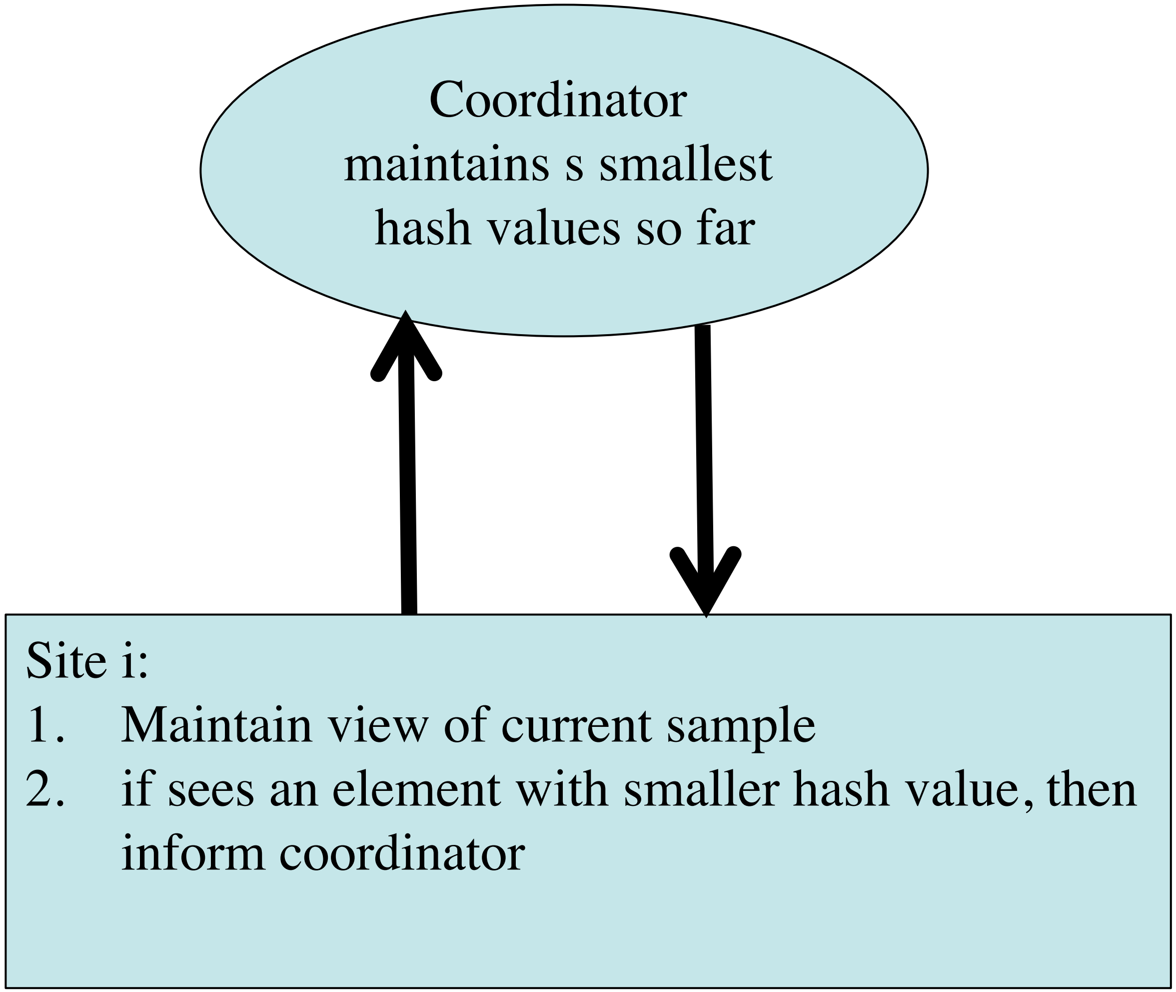
Prior and Related Work

- Distinct Sampling on a Stream and Applications (Gibbons and Tirthapura SPAA 2001, Gibbons VLDB 2001)
- Continuous Distributed Streaming Model (Cormode et al. SODA 2008, and many other works)
- Continuous Random Sampling on Dist. Streams
  - Cormode, Muthu, Yi, Zhang, JACM 2011
  - Tirthapura and Woodruff, DISC 2011
- Stream Sampling has a rich history starting from the reservoir sampling algorithm

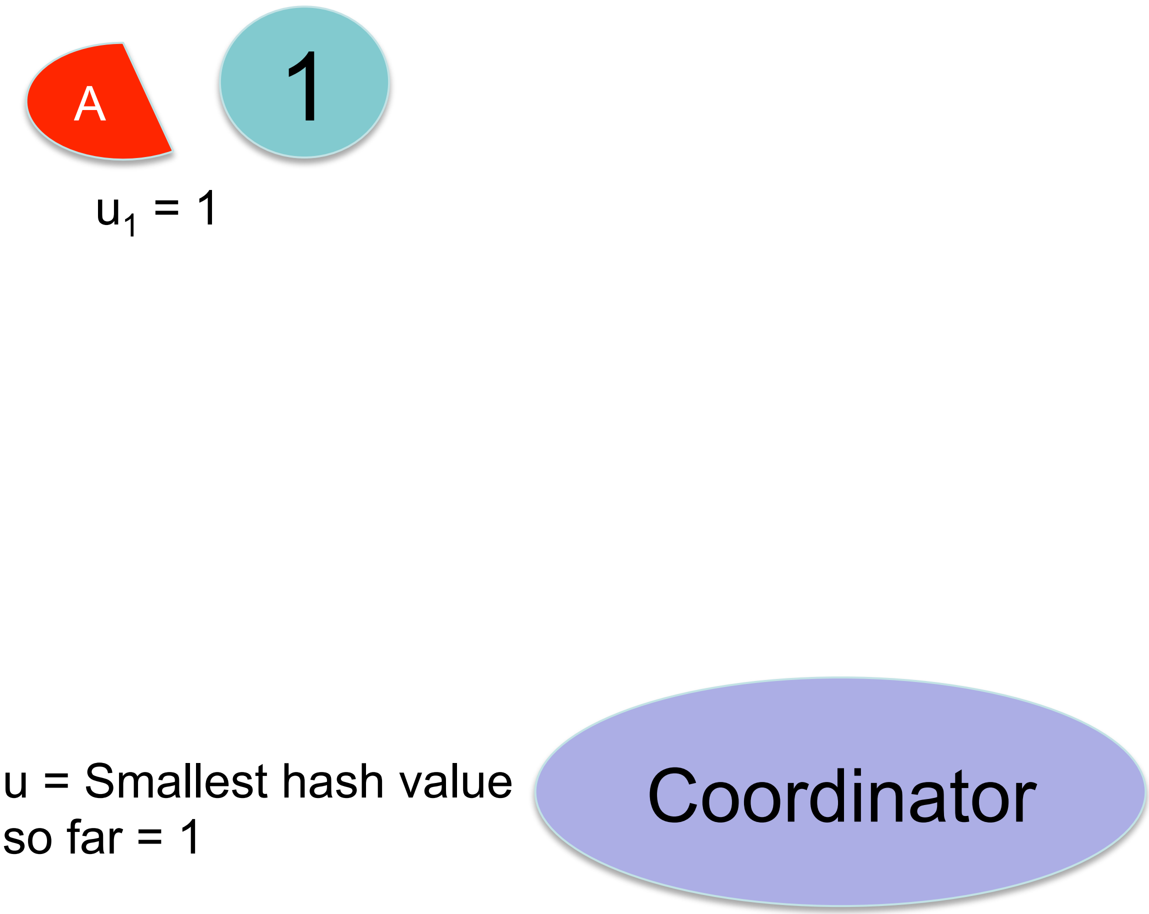
Sampling Algorithm Basics

- U be the universe of all elements in S
- Algorithm first chooses h: U → [0,1], a hash function that assigns a real number in [0,1] to each element in U
  - On same input v, h always yields same output h(v)
  - On distinct inputs, outputs of h are mutually independent random variables
- Random Sample of size s from S is the set of elements R that have the s smallest hash values in {h(x) | x in S}

Distributed Maintenance of Sample

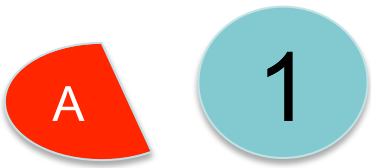


Algorithm: Element Arrives at Site 1 (maintain a sample of size 1)



Algorithm: Element Arrives at Site 1

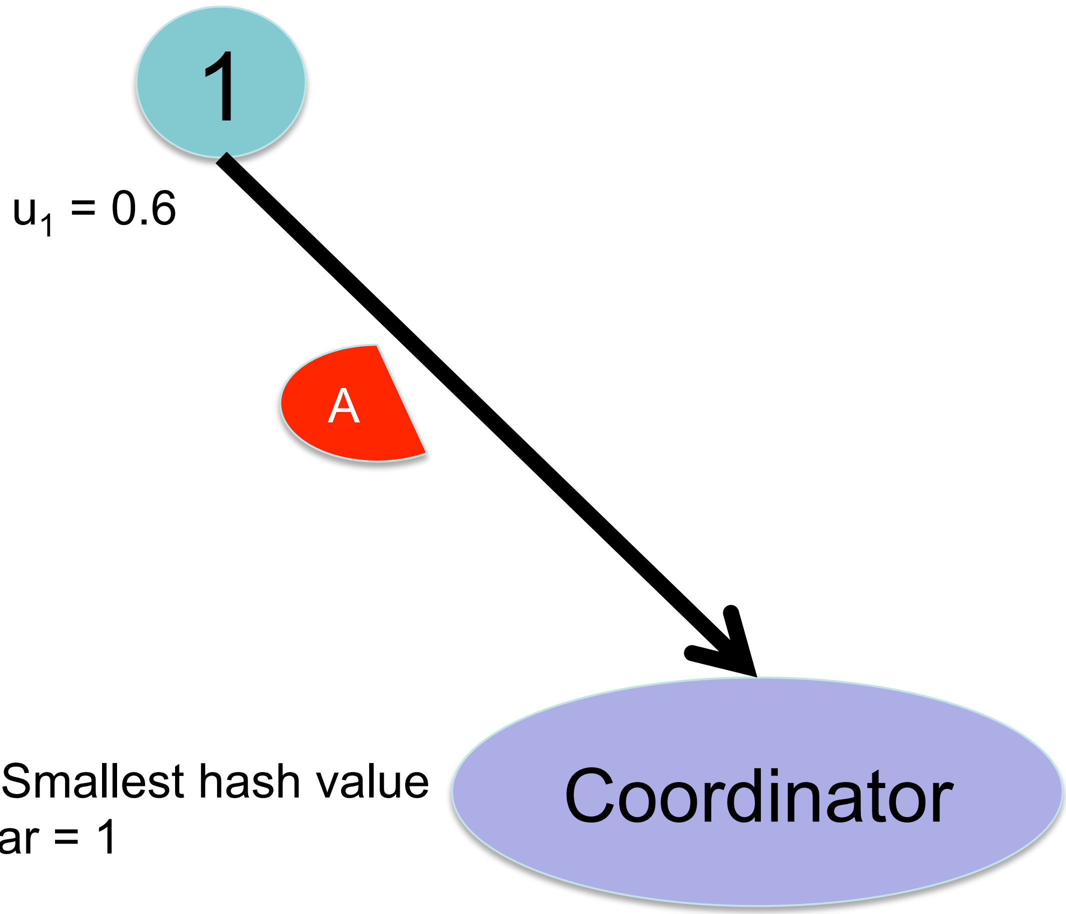
Weight = h(A) = 0.6



$u_1 = 1$

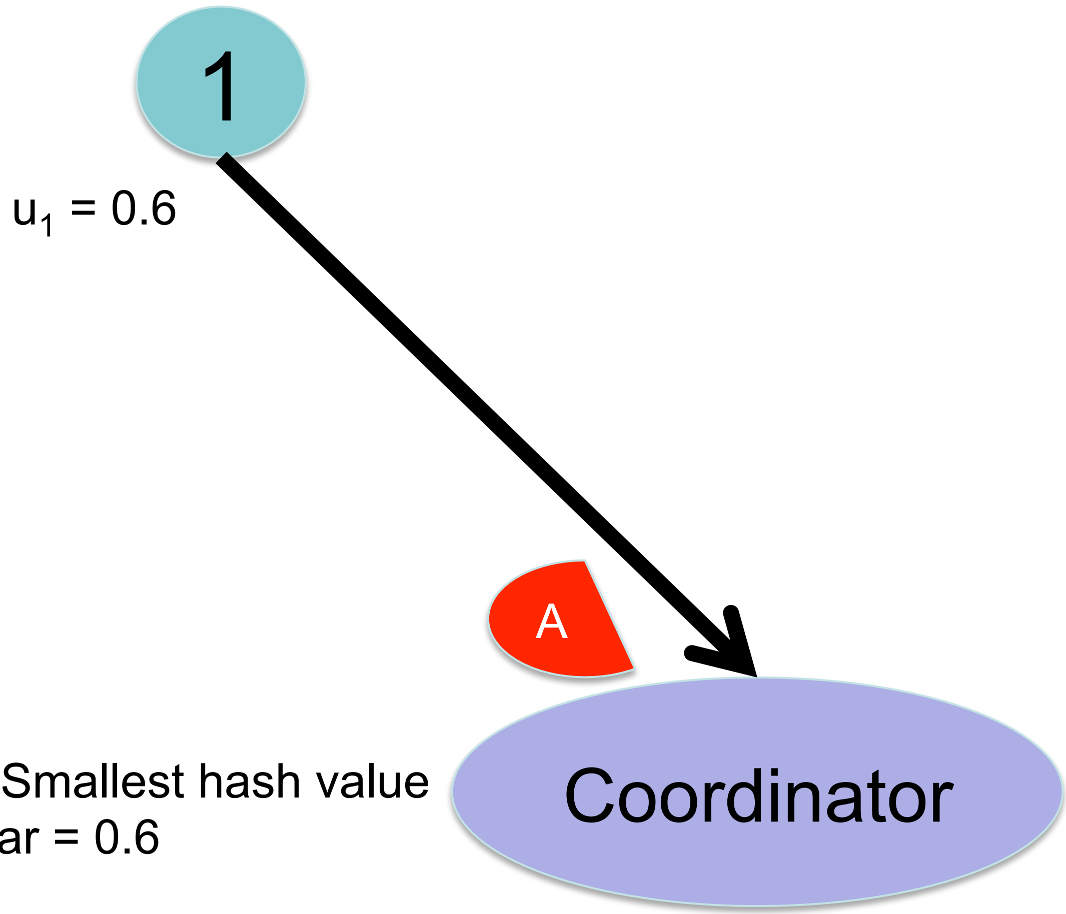
u = Smallest hash value so far = 1

Algorithm: Element Arrives at Site 1



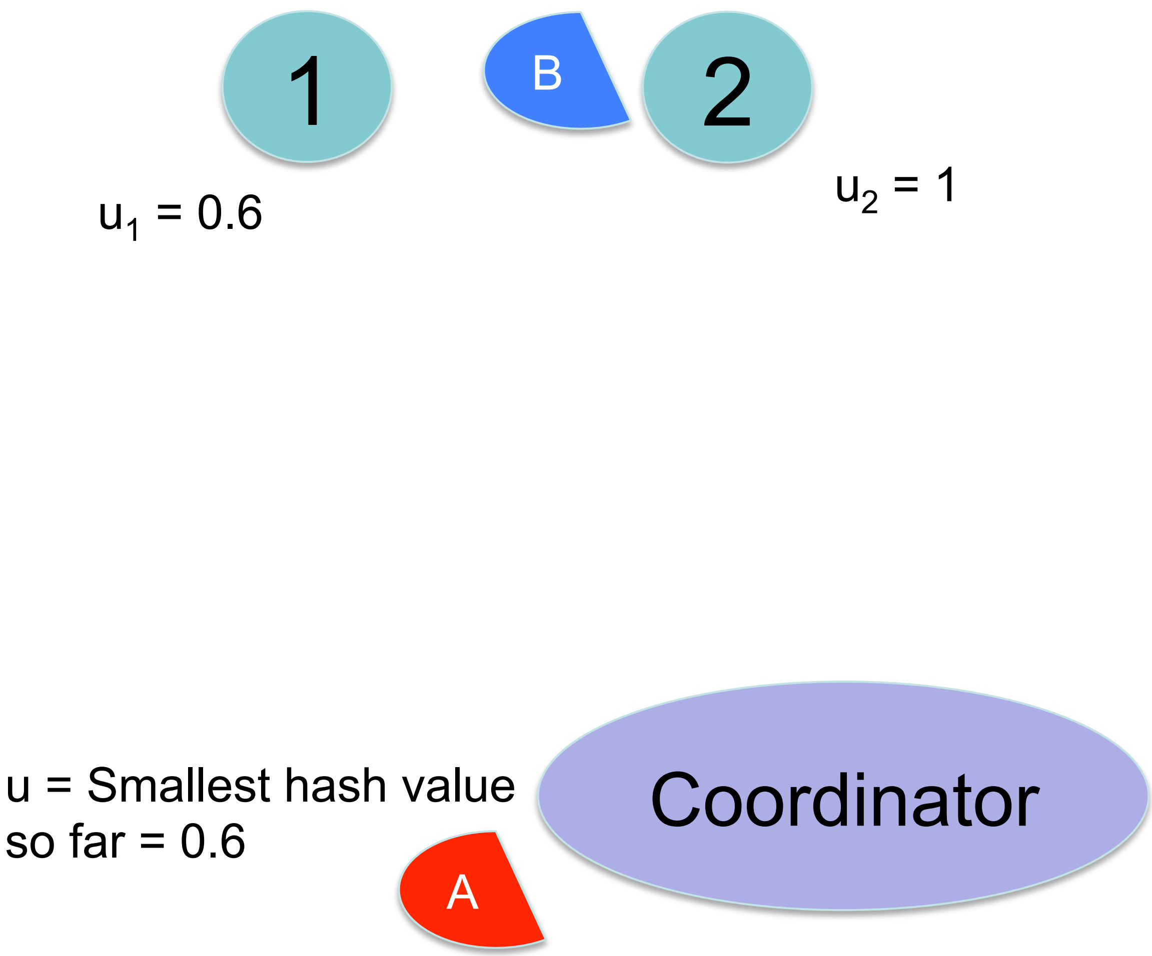
u = Smallest hash value so far = 1

Algorithm: Element Arrives at Site 1

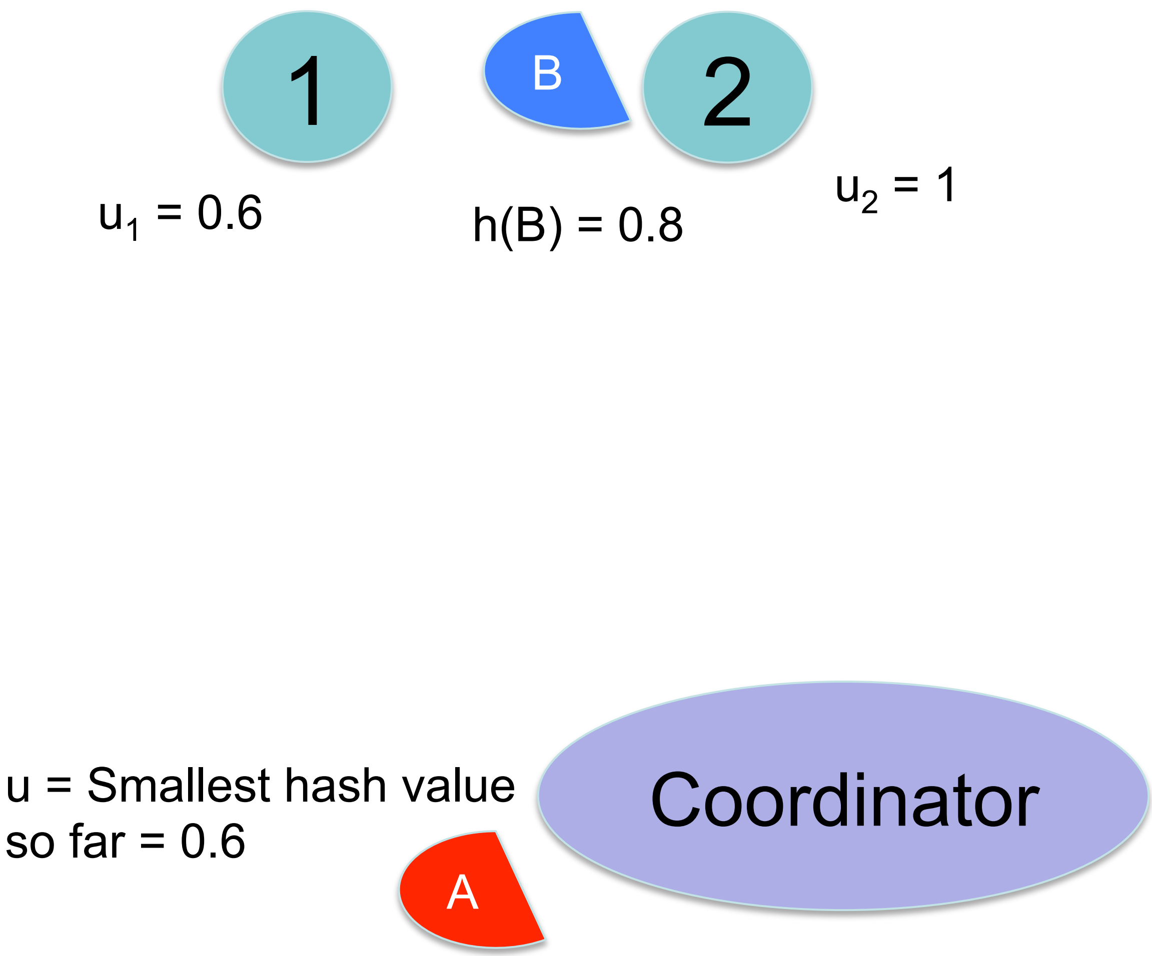


u = Smallest hash value so far = 0.6

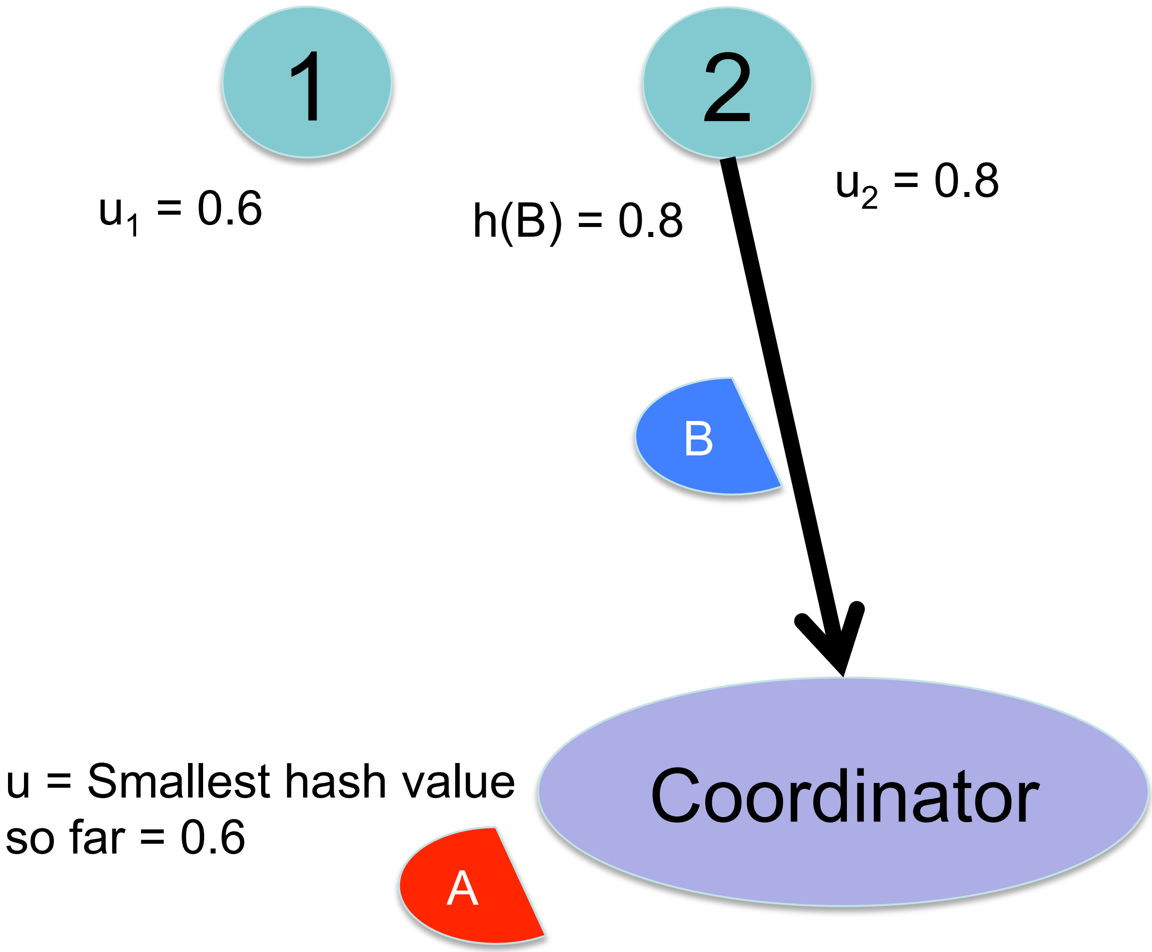
Algorithm: Element Arrives at Site 2



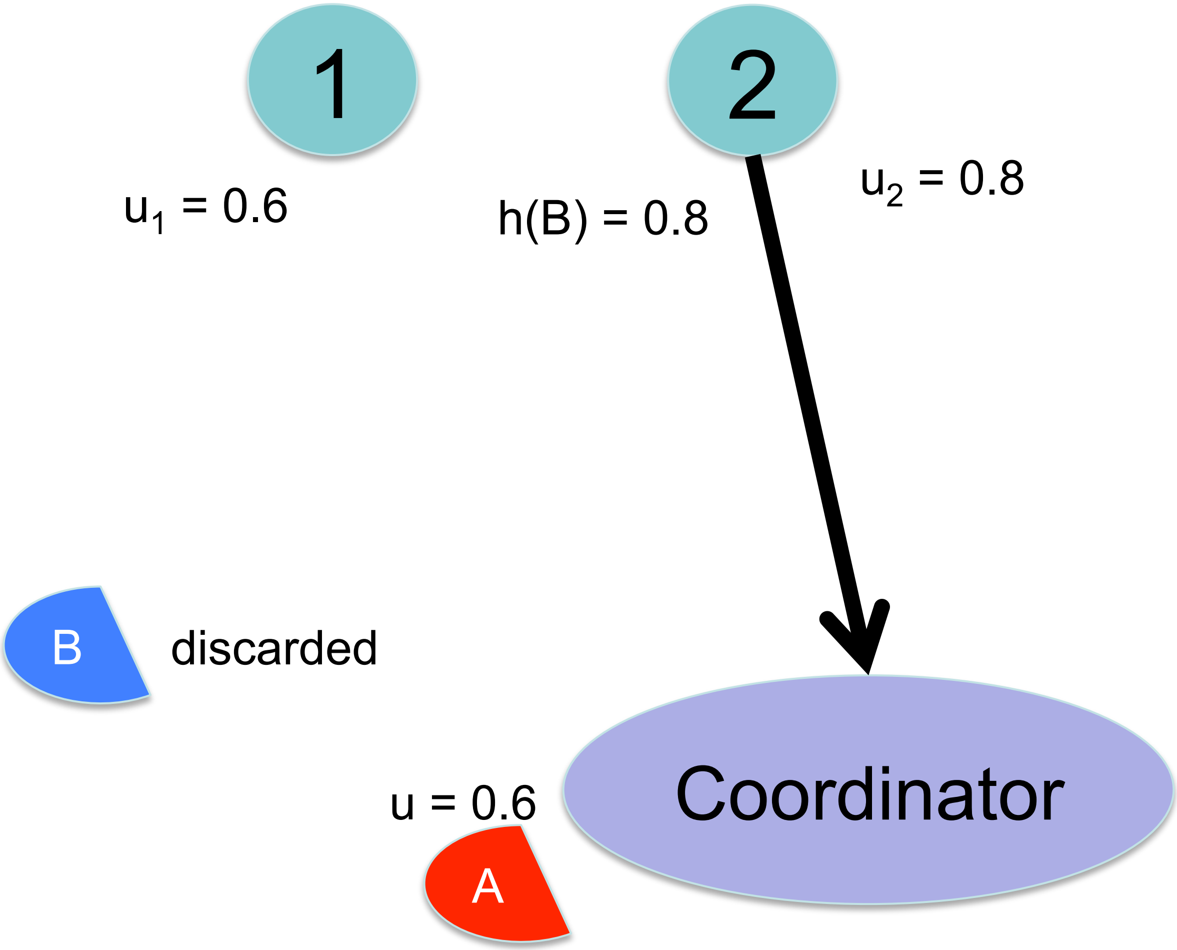
Algorithm: Element Arrives at Site 2



Algorithm: Element Arrives at Site 2

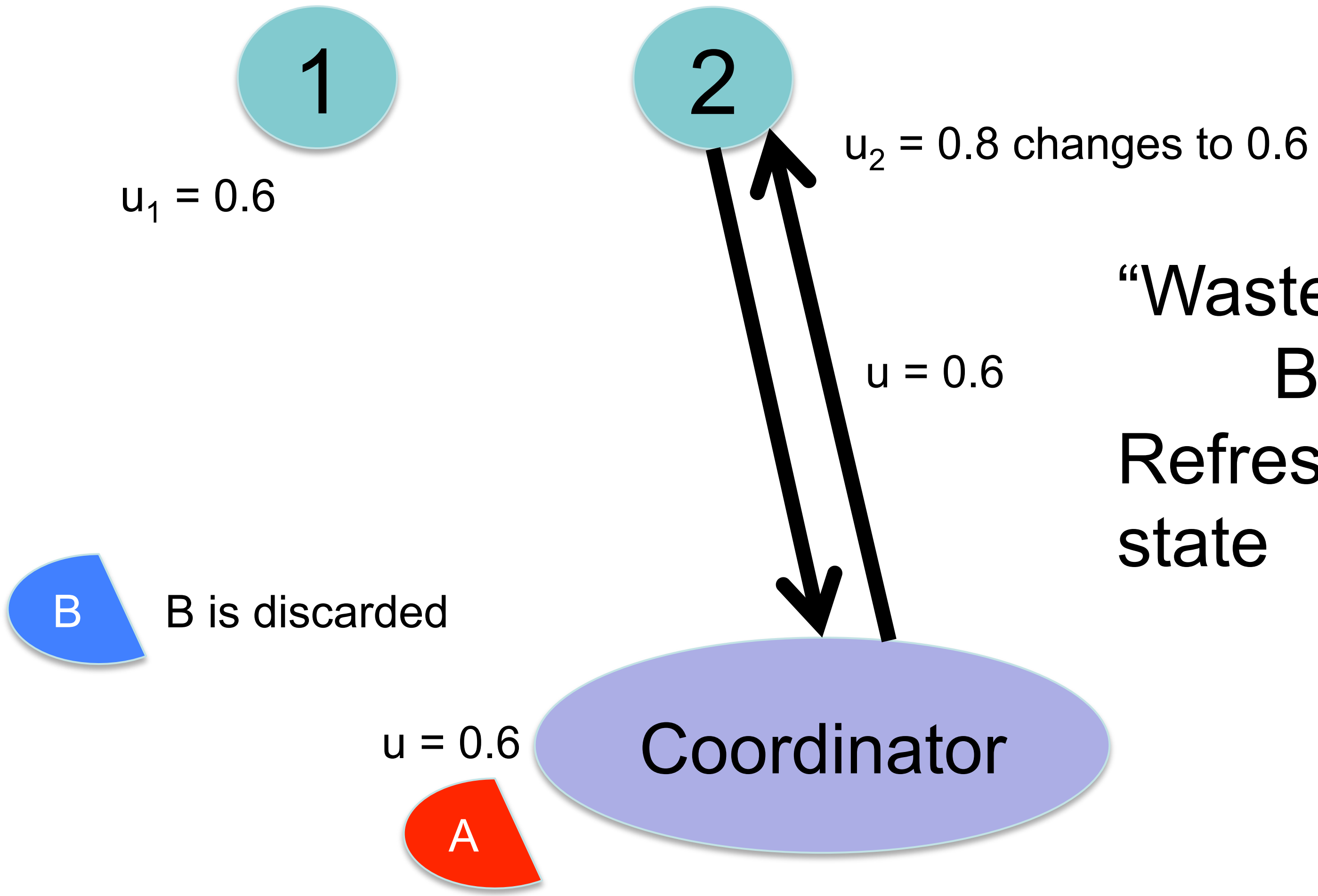


Algorithm: Element Arrives at Site 2



"Wasteful" Message

Algorithm: Element Arrives at Site 2

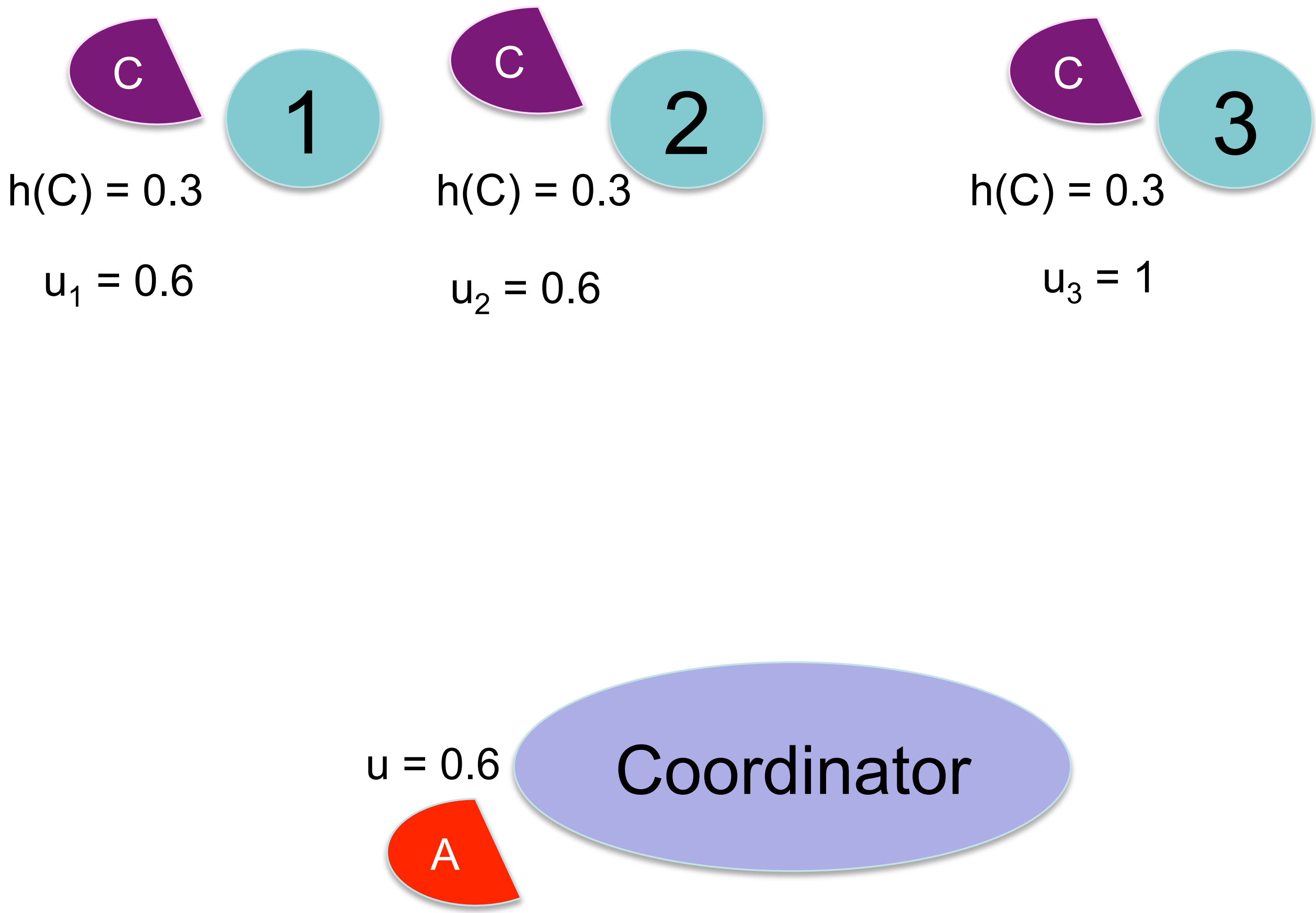


“Wasteful” Message

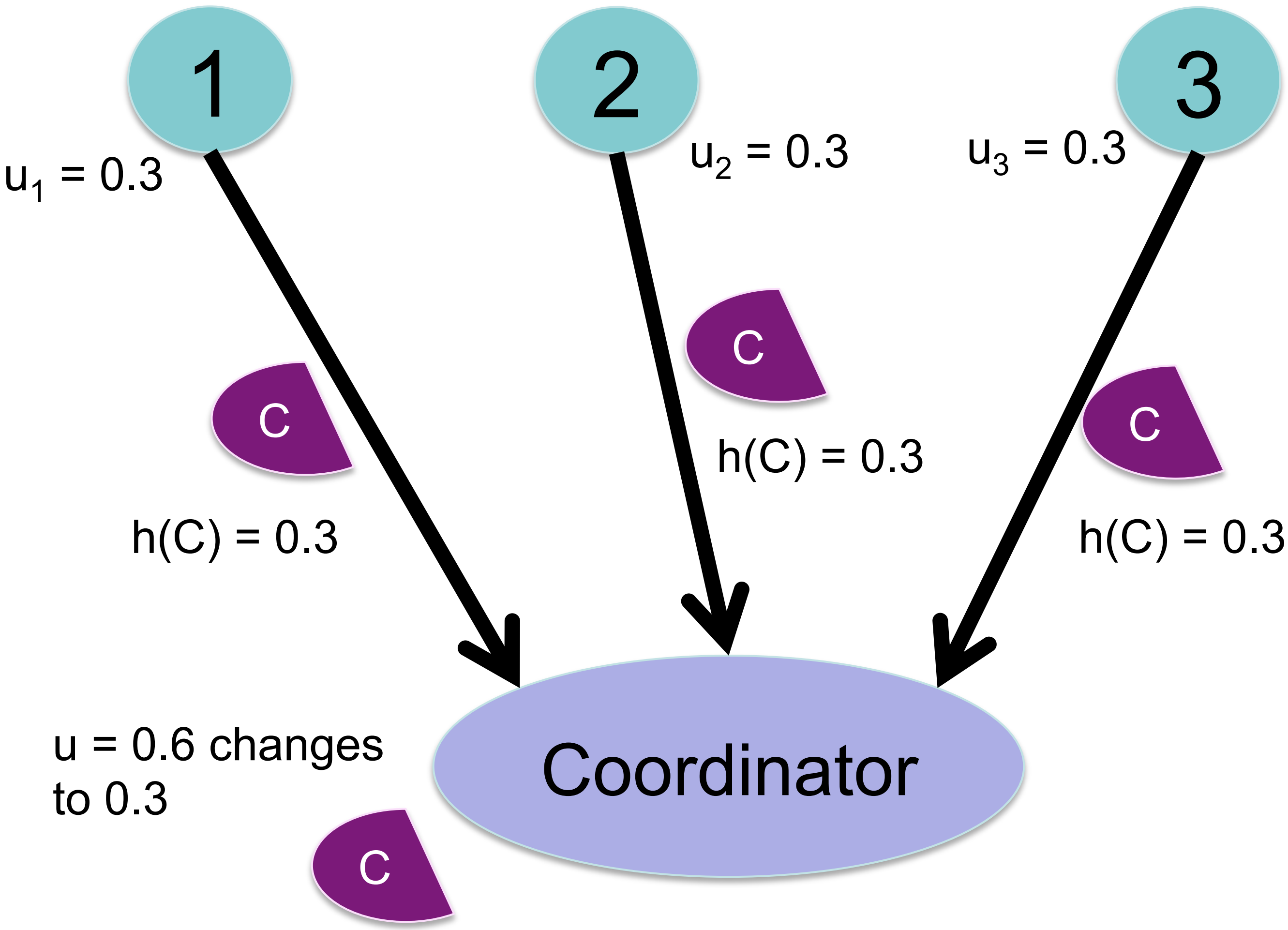
BUT

Refresh Node 2’s state

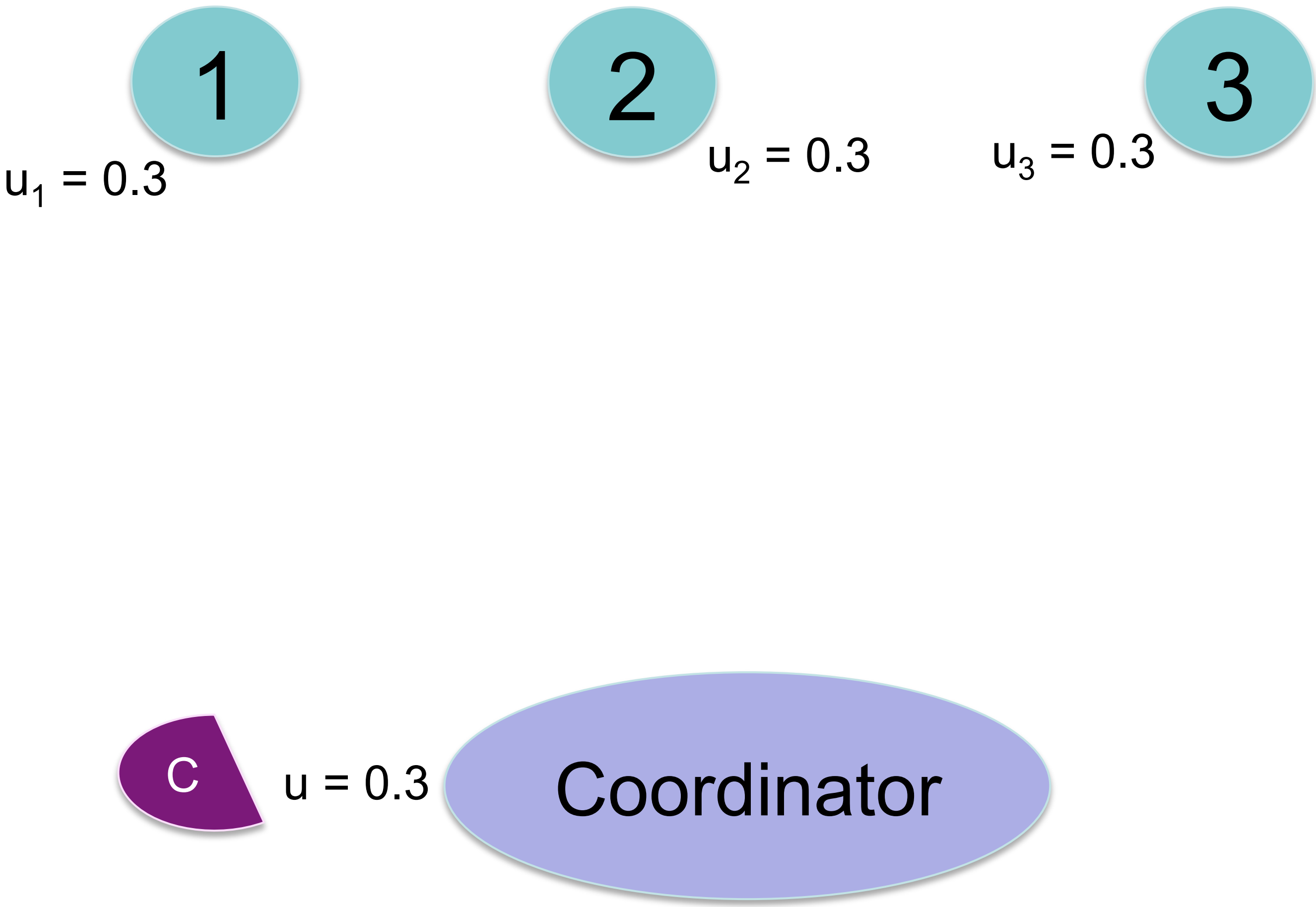
Algorithm: Element Arrives at Sites 1, 2, 3



Algorithm: Element Arrives at Sites 1, 2, 3



Algorithm: Element Arrives at Sites 1, 2, 3



Distributed Algorithm Notes

1. State of coordinator is always current
2. State of site maybe out of sync, but is “safe”
3. A message from site either updates coordinator or results in an update to the state of the site
4. Each site maintains a view of current sample, to prevent sending the same element repeatedly to the coordinator

Algorithm at Site i

Init: Receive h from coordinator, set $u_i \leftarrow 1$, $P_i \leftarrow$ phi

Repeat forever

    If receive element e in stream S_i

        If $(h(e) < u_i)$ and (e not in P_i)

            Insert e into P_i

            Send e to coordinator

    If receive value u from coordinator

        $u_i \leftarrow u$

        Discard all elements e from pi such that $h(e) \geq u_i$

Algorithm at Coordinator

Init: P ← empty, u ← 1. Send hash function h to all sites

Repeat Forever

    If receive e from site i

        If $h(e) < u$:

            If e not in P, add it

            If $|P| > s$

                Discard element with largest hash value from P

                $u \leftarrow \max\{h(e) \mid e \text{ in } P\}$

        Send u to site i

    If receive query for random sample, then Return P

Analysis of Algorithm

1. Analyze communication from site i to coordinator
2. Multiply by two (coordinator feedback)
3. Sum over all sites

We sketch an analysis parameterized by the number of distinct elements d

Analysis of Algorithm (Upper Bound)

Lemma: The expected number of messages transmitted by site 1 <= $s \log (d_1 e/s)$ where d_1 is number of distinct elements observed at site 1

Proof. Consider distinct element arrivals $j=1,2,3,d_1$ at site 1
Let Y = total number of messages transmitted by site

For $j = 1$ to d_1, let $Y_j = 1$ if j^{th} arrival causes a message , 0 otherwise

$$Y = Y_1 + Y_2 + \ldots Y_{d1}$$

$$E[Y] = E[Y_1] + E[Y_2] + \ldots E[Y_{d1}]$$

$E[Y_j]$ = Pr[arrival of j^{th} distinct element causes a message to be sent]

$$= 1 \quad \text{for } j=1 \text{ to } s,$$

$$= s/j \quad \text{for } j > s$$

Summation over all j leads to the lemma

Analysis Notes

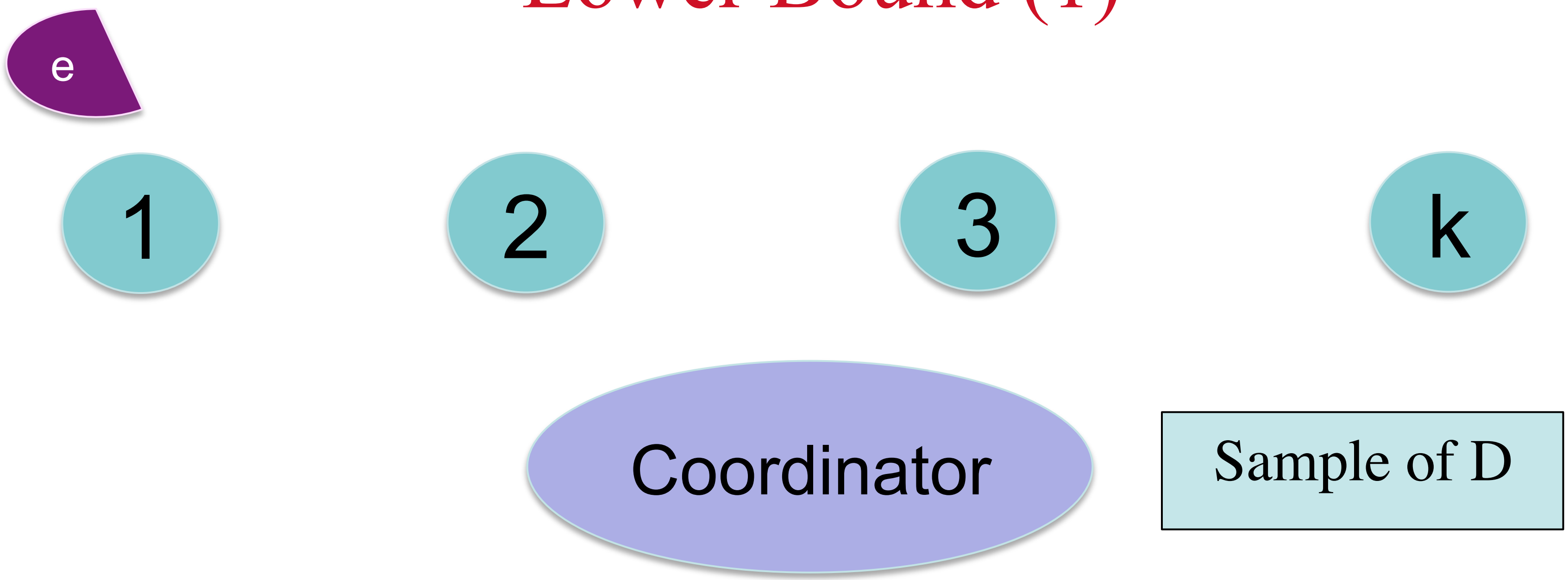
- Analysis does not assume any specific input distribution, hence worst-case
- Can achieve better bounds when more is known about input distribution
- Can a different algorithm do better in general?

Lower Bound

- For any algorithm A and parameter d, there exists an input distributed stream, I_A with d distinct elements such that the expected number of messages sent by the algorithm upon receiving I_A is at least $(ks/2) \ln (de/s)$

- Probability space is one from which the random sample is chosen

Lower Bound (1)



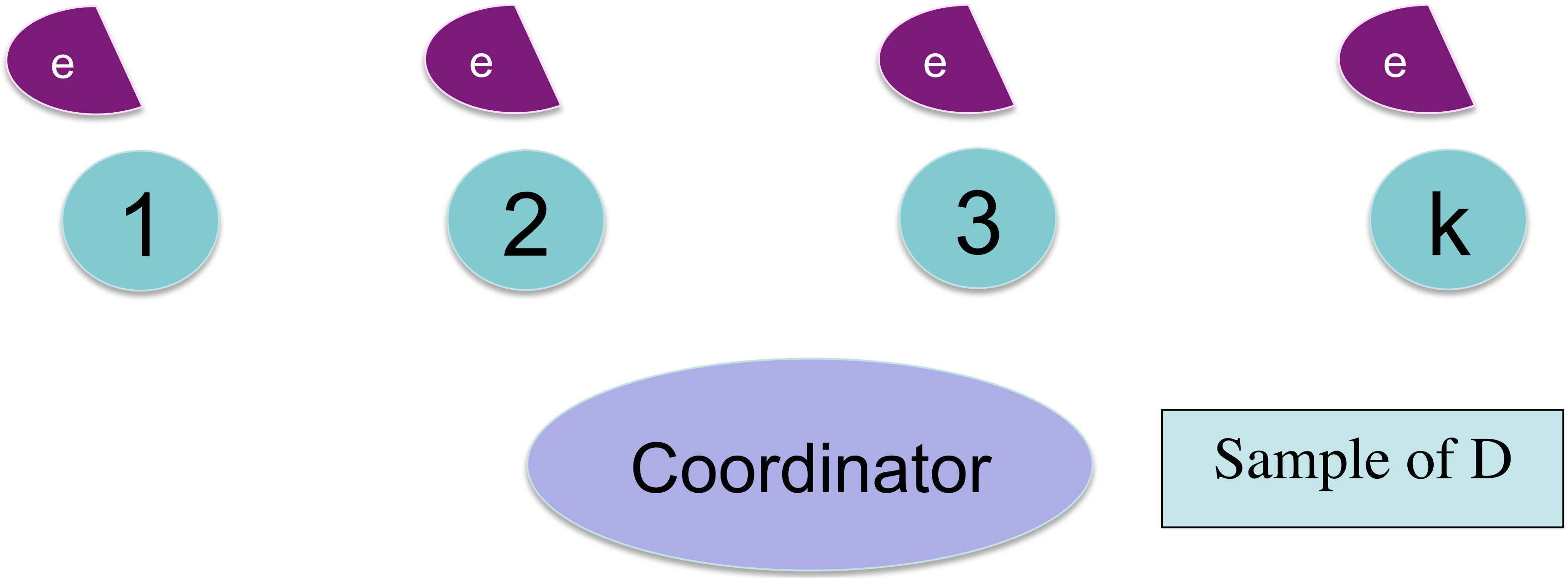
Suppose we have seen set of distinct elements D so far

Supply an element e to site 1 that does not belong to D

e will belong to sample with probability s/(|D|+1)

Site 1 will send a message to Coordinator with probability at least s/2(|D|+1)

Lower Bound (2)



Suppose we have seen set of distinct elements D

Supply same element e (outside of D) to all sites 1,2,..,k

e will belong to sample with probability s/(|D|+1)

Site 1 will send a message to Coordinator with probability s/2(|D|+1)
So will every other site.

Lower Bound (3)

Round 1: All sites see element e_1

Round 2: All sites see element e_2

- - - - -

Round j: All sites see element e_j

Expected number of messages sent in a round $\geq sk/2(|D|+1)$

Continue this process in rounds 1,2,3,…,d

Expected number of messages sent $\geq sk/2\ \{1/2 + 1/3 + .. + 1/(d+1)\}$

Comparison with Simple Random Sampling

k= number of sites, s = sample size n= number of elements d = number of distinct elements	Number of Messages Over Entire Execution
Distributed Simple Random Sampling	max{k,s} log(n/s) (T, Woodruff & Cormode et al.)
Distributed Distinct Random Sampling	k s log (d/s)

Sampling With Replacement

- Requirement: Each element in sample chosen uniformly from entire population
- Solution: Repeat s copies of single element sampling algorithm, in parallel
- Improvement: Combine messages of different copies of algorithm, reducing duplication

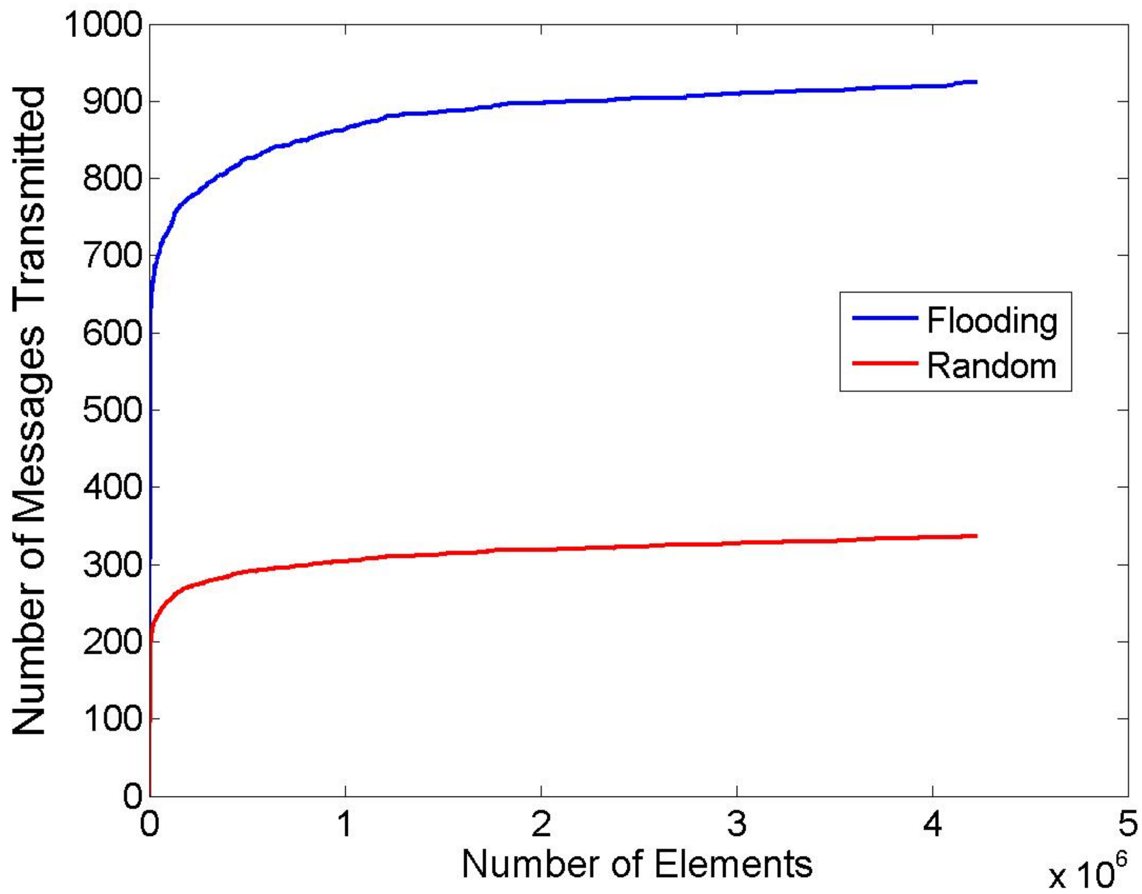
Sliding Window

- Random sample chosen from set of all elements observed in the w most recent time steps
- Idea: choose the elements with the smallest hash values from among the w most recent time steps
- Problem: maintaining minimum weight element within a sliding window is hard (communication wise)
- Idea: Use the fact that these are not arbitrary weights, but randomly chosen weights

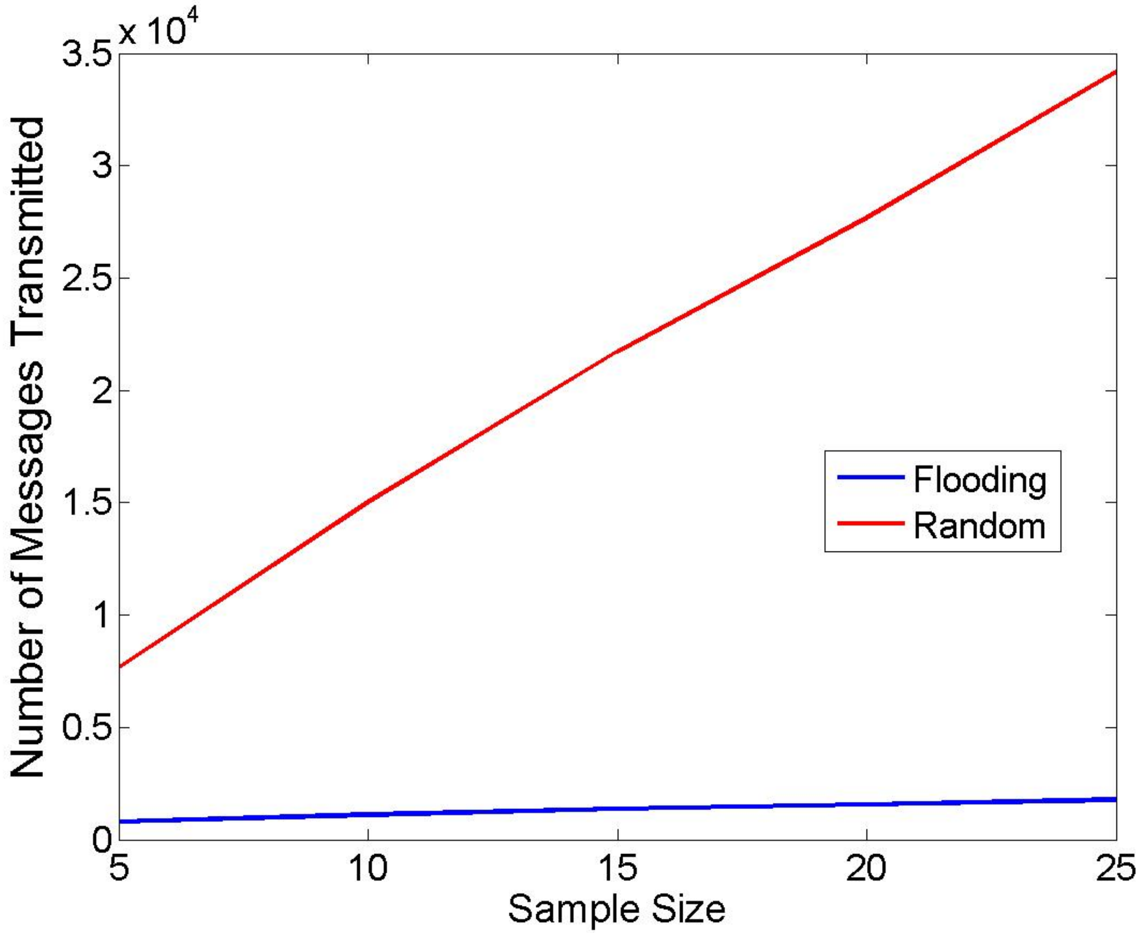
Experiments: Datasets

	# Elements	# Distinct
OC48 Network Trace	42 mil	4.3 mil
Enron Email	1.5 mil	0.37 mil

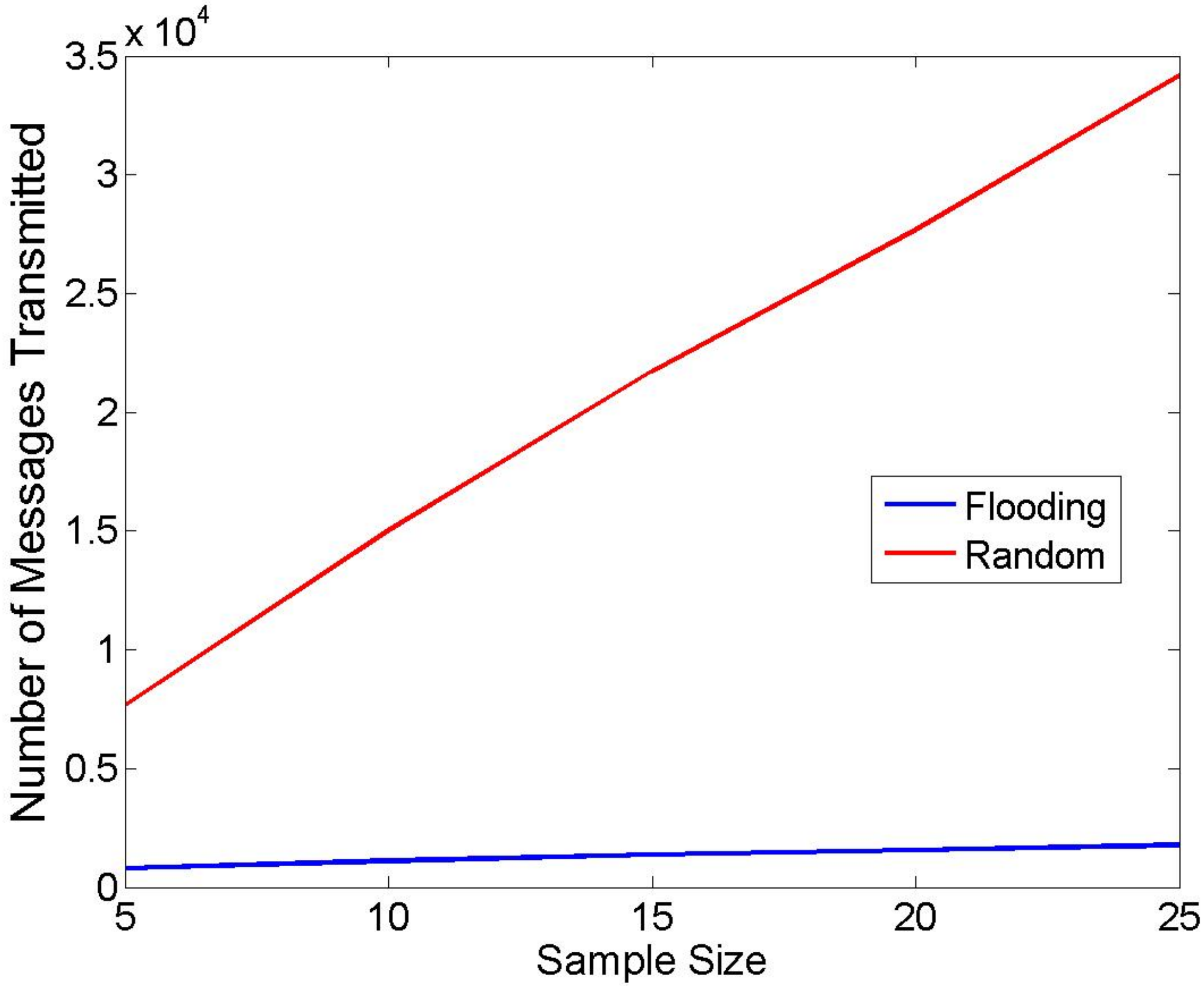
Number of Messages vs Stream Size (OC48)
k (number of sites) = 10, s (sample size) = 5



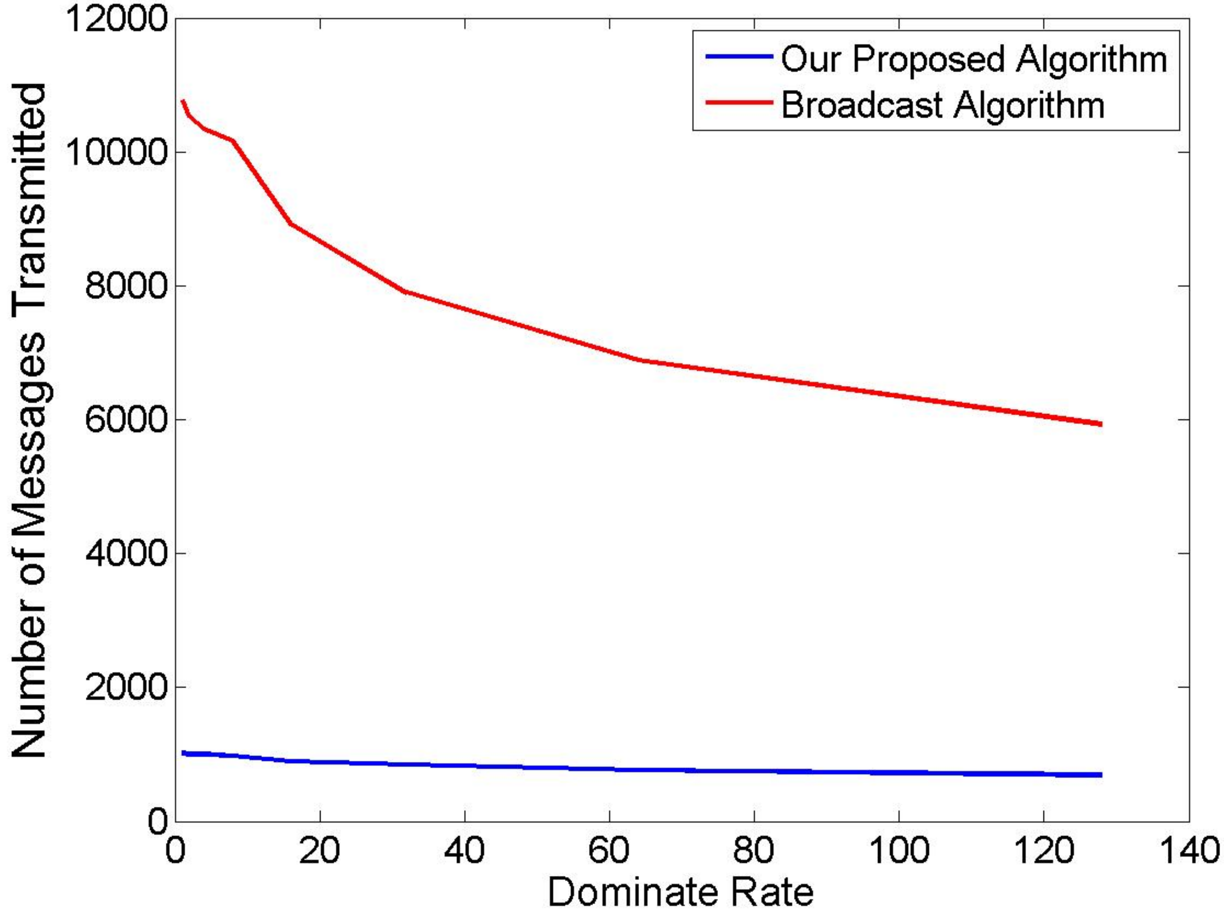
Number of messages vs sample size (OC48)
Number of sites = 50



Number of messages vs Number of sites (OC48)
Sample size = 20



Messages vs skew in data (OC48)
sites = 20, sample size = 20



Conclusion

- Message Optimal Algorithm for Continuous Distributed Distinct Sampling
- Easy to implement, good practical performance
- Message complexity of distinct sampling inherently greater than simple random sampling
- Sampling Without and With Replacement, Sliding Windows
- Works in Asynchronous Model

Future Work

- Better Lower Bounds for Sliding Windows
- Other properties in a continuous distributed streaming model, including properties on graphs