A STUDY OF GOSSIP ALGORITHMS FOR INTERNET-SCALE CARDINALITY ESTIMATION OF DISTRIBUTED XML DATA

Vasil G. Slavov
Computer Science & Electrical Engineering
University of Missouri-Kansas City

June 22nd 2012

Acknowledgements
National Science Foundation (IIS-1115871)

Thesis Committee

- Praveen R. Rao, Ph.D., Committee Chair
- Yugyung Lee, Ph.D.
- Deep Medhi, Ph.D.

Overview

- Introduction
- Background
- Thesis objectives
- Design
  - VanillaXGossip
  - XGossip
- Implementation
- Performance evaluation
  - Amazon Elastic Compute Cloud (EC2)
- Conclusions

Introduction

- XML and XPath – W3C standards

```
<ClinicalDocument>
  <typeId extension="POCD_HD000040" root="2.16.840.1.113883.1.3"/>
  <id extension="CSE001" root="2.16.840.1.113883.19.4"/>
  <code code="8647-0" codeSystem="2.16.840.1.113883.6.1" codeSystemName="LOINC" displayName="Hospital Consultations"/>
  <confidentialityCode code="V" codeSystem="2.16.840.1.113883.5.25"/>
- <RecordTarget>
  - <PatientRole>
      <ID>711</ID>
      <Patient>F</Patient>
      <ProviderOrganization>id root=2.16.840.1.113883.19.5</ProviderOrganization>
    </PatientRole>
  </RecordTarget>
- <Author>
    <representedOrganization>UMKC School of Computing & Engineering (Research)</representedOrganization>
```

XPath

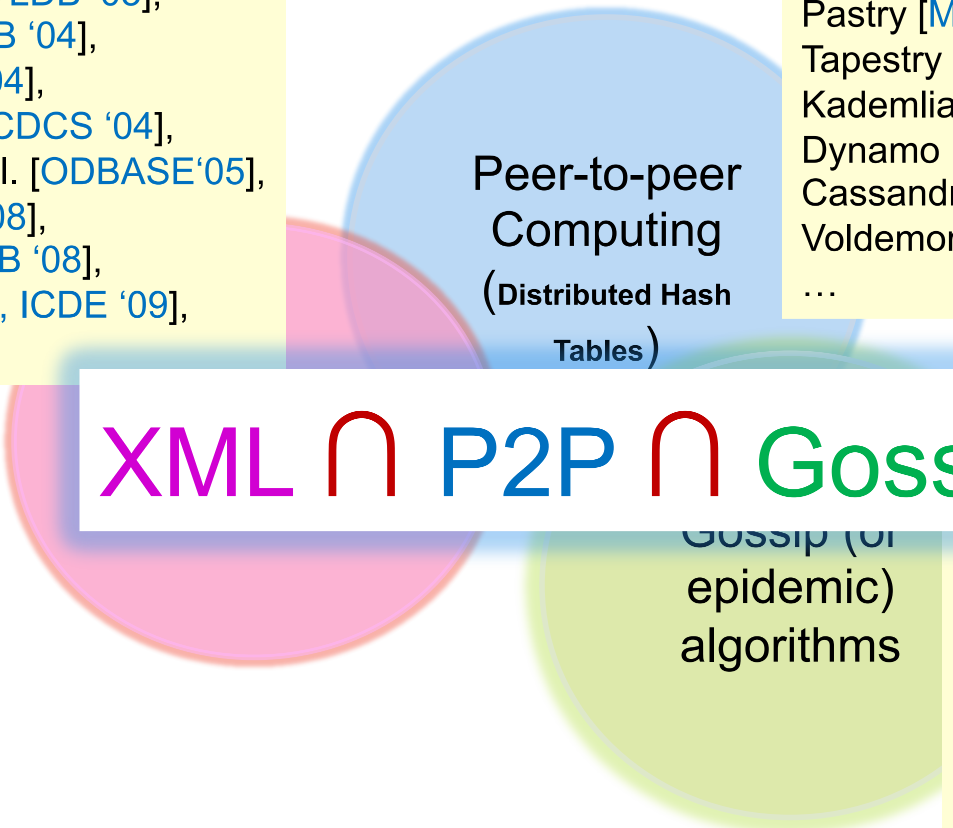
```
/ClinicalDocument[RecordTarget/PatientRole/Patient = "M"][RecordTarget/PatientRole/ID]
```

```
  - <section>
      <code code="" codeSystem="2.16.840.1.113883.6.1" codeSystemName="LOINC"/>
      <title>History of Present Illness</title>
    - <text>
        The patient was a 108-year-old nursing home resident , who was admitted with a two-day history of increased respiratory
        secretions and a 24-hour history of elevated fever . Despite Augmentin , the patient 's delirium worsened in the 24 hours prior to
        admission , and her temperature was up to 102 . She was refusing to take p.o.'s
      </text>
    </section>
  - <section>
      <code code="11348-0" codeSystem="2.16.840.1.113883.6.1" codeSystemName="LOINC"/>
      <title>Past Medical History</title>
```

The Story So Far…

Galanis et.al. [VLDB ‘03],
XPeer [P2P&DB ‘04],
XP2P [WIDM ‘04],
Garces et al. [ICDCS ‘04],
Skobeltsyn et.al. [ODBASE‘05],
KadoP [ICDE ‘08],
XTreeNet [VLDB ‘08],
*psi*X [TKDE ‘09, ICDE ‘09],
...

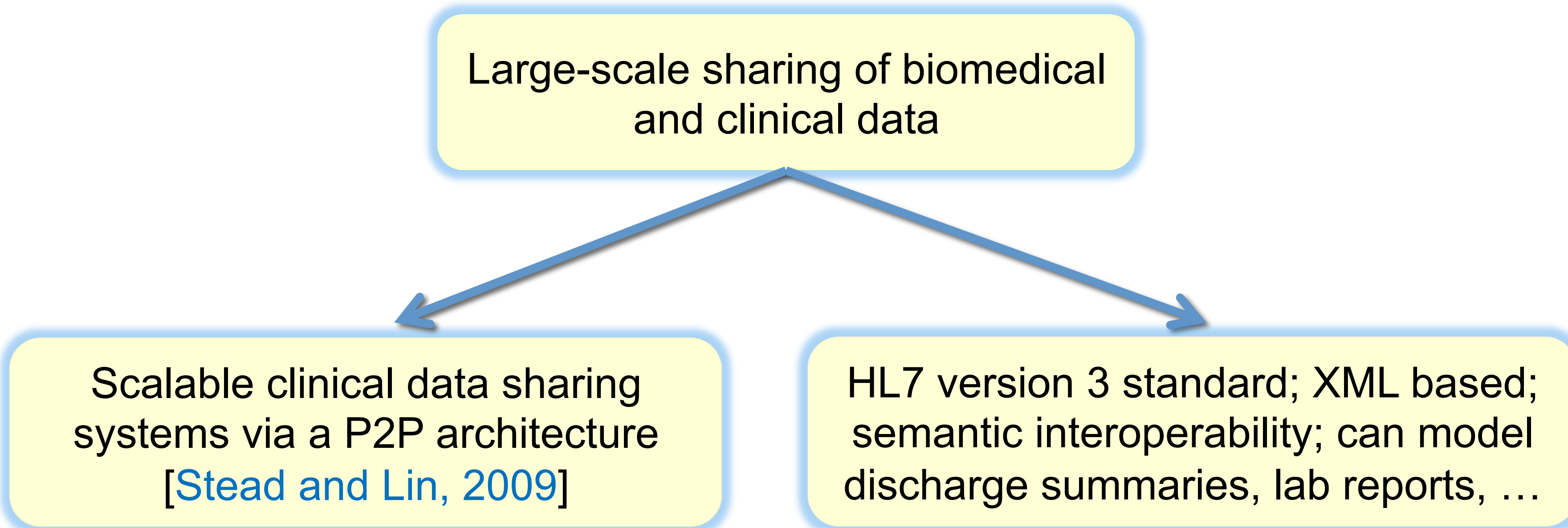
Chord [SIGCOMM ‘01],
CAN [SIGCOMM ‘01],
Pastry [Middleware ‘01],
Tapestry [JSAC ‘04],
Kademlia [IPTPS ‘02]
Dynamo [SOSP ‘07],
Cassandra [SIGMOD ‘08],
Voldemort [ICDE ‘11],
…



XML ∩ P2P ∩ Gossip

t. al. [PODC ‘87],
Karp et. al. [FOCS ’00],
Kempe et.al. [FOCS ‘03],
Berger et.al. [SODA ’05],
Ganesh et. al. [INFOCOMM ‘ 05],
Boyd et. al. [INFOCOMM ’05],
Jelasity et. al. [TOCS ‘05],
Kashyap et. al. [PODS ‘06],
Georgiou et. al. [PODC ‘08],
Mosk-Aoyama et. al. [TOIT ’08],
…

XML ∩ P2P = ?



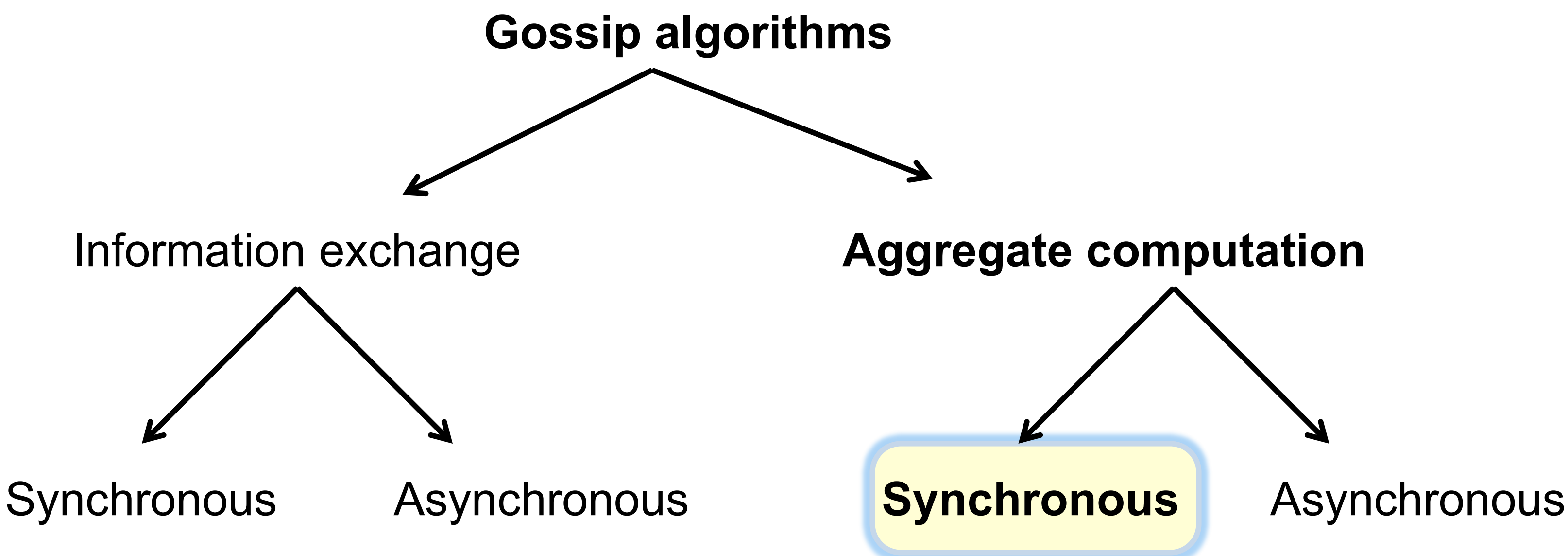
The Cancer Biomedical Informatics Grid (caBIG): a real world data sharing platform

The Problem: XPath Cardinality Estimation

- Compute the number of documents in the network that contain a match for the expression /Gene//goAcc
- Useful for query optimization
- Desired properties
  - Scalability
  - Decentralization
  - Fault-tolerance
  - Efficient usage of bandwidth
  - Provable guarantee on the quality of the estimate

Gossip Algorithms

- Communication, computation, and information spreading
- Attractive in large-scale, distributed systems
- Real world examples: Amazon S3, Dynamo, Cassandra



Can further classify based on the topology of the network

Push-Sum Protocol

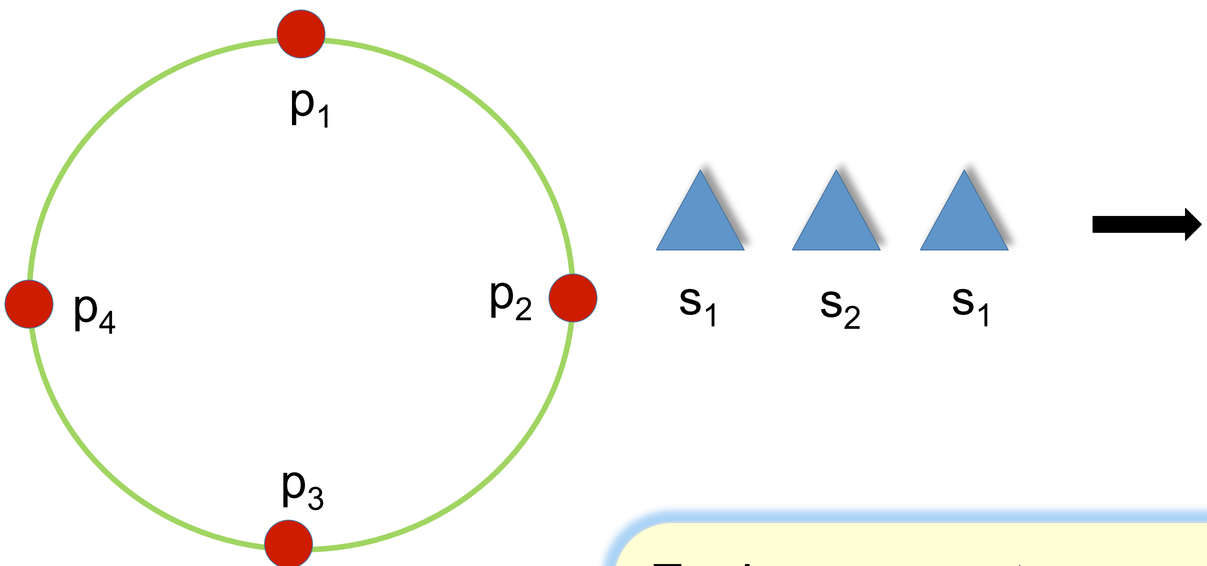
- By Kempe, Dobra, and Gehrke [FOCS ‘03]
- Each peer wishes to know the average
- Each peer maintains a (sum, weight) pair during gossip
  - In round @ t = 0: send (f_1,1) to itself
  - In any round @ t > 0: add up sums & weights, send (s1, s2)/2, (w2, w2)/2 to itself and to random peer
- Convergence (*n* peers)
  - Rounds: O(log(*n*) + log(1/ε) + log(1/δ))
  - Messages: *n* messages per round
- Proof is based on the property of “mass conservation”

Thesis Objectives

1. Implementing gossip in an Internet-scale environment
2. Conducting a comprehensive evaluation
3. Analyzing the experimental results

Design of VanillaXGossip

- Builds on Push-Sum
- XML documents are mapped to their signatures
  - *psi*X [Rao et.al. TKDE ‘09, ICDE ‘09]
  - XML doc. → data signature; XPath query → query signature



T

s_1	(2,1)
s_2	(1,1)
$\perp$	(0,1)

Each peer creates a sorted tuple list

Merging Phase

- Suppose a peer receives 3 tuple lists during a gossip round

T_1	T_2	T_3
$s_1,(f_1,w_1)$	$s_1,(f_3,w_3)$	$s_1,(f_4,w_4)$
$s_2,(f_2,w_2)$	$s_3,(f_5,w_5)$	$\perp, (f_c,w_c)$
$\perp, (f_a,w_a)$	$\perp, (f_b,w_b)$	

$\xrightarrow{\text{MergeLists(...)}}$

T_m
$s_1,(sum_{f_1},sum_{w_1})$
$s_2,(sum_{f_2},sum_{w_2})$
$s_3,(sum_{f_3},sum_{w_3})$
$\perp, (sum_{f_4},sum_{w_4})$

$f_a = f_b = f_c = 0$

$sum_{f_1} = (f_1+f_3+f_4)/2$

$sum_{w_1} = (w_1+w_3+w_4)/2$

$sum_{f_2} = (f_2+f_b+f_c)/2$

$sum_{w_2} = (w_2+w_b+w_c)/2$

$sum_{f_3} = (f_a+f_5+f_c)/2$

$sum_{w_3} = (w_a+w_5+w_c)/2$

$sum_{f_4} = (f_a+f_b+f_c)/2$

$sum_{w_4} = (w_a+w_b+w_c)/2$

$T_m \rightarrow$ randomly selected peer

VanillaXGossip

- Special multiset $\perp$
  - Placeholder for signatures not yet known to a peer during a gossip round
  - Preserves the property of "mass conservation"
- Convergence
  - Rounds: $O(\log(n) + \log(1/\varepsilon) + \log(1/\delta))$

Problem ☹
A peer will end up with all the distinct signatures
More memory, more bandwidth

XGossip

- Idea
  - Divide-and-conquer approach using Locality Sensitive Hashing (LSH)
  - A subset of peers are responsible for gossiping a subset of distinct signatures

Benefits ☺

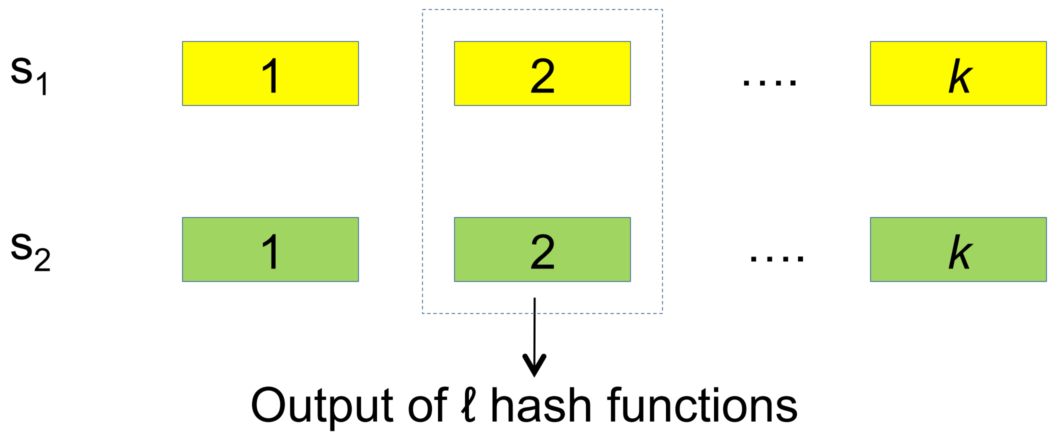
Less memory, less bandwidth, faster convergence

Locality Sensitive Hashing (LSH)

- Introduced by Indyk and Motwani [STOC '98]
- Applications
  - Web clustering, computer vision, computational biology, etc.
- Idea
  - Use many hash functions
  - Probability of collision is higher for inputs that are more similar
- LSH on sets using Jaccard index [WWW '02, WWW '05]
  - $P[h(s_1) = h(s_2)] = |s_1 \cap s_2| / |s_1 \cup s_2|$
  - $h()$ → min-hashing

LSH on Sets

- Suppose $\mathbf{p} = |s_1 \cap s_2| / |s_1 \cup s_2|$
- Pick k x ℓ random linear hash functions

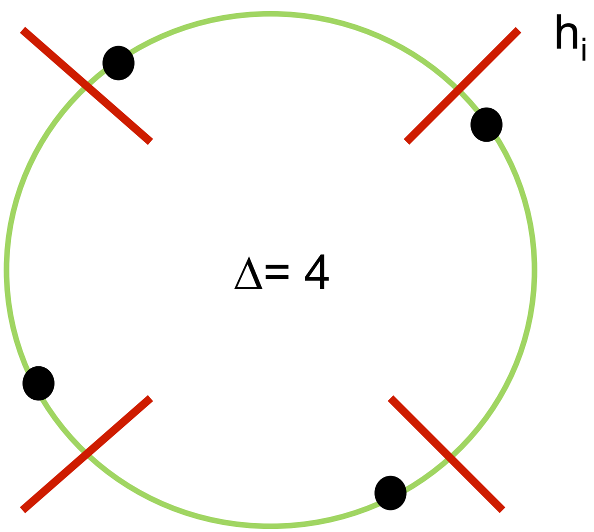


P[at least one p ... $= 1 - (1 - \mathbf{p}^{\ell})^{k}$

Can pick k and ℓ s.t.
High probability if similarity ≥ **p**
Low probability if similarity < **p**

XGossip (1/2)

- Define k teams for a signature **s**
  - LSH(**s**) → $\{h_1, \ldots, h_k\}$
  - Each team has id h_i, $1 \le i \le k$; Δ denotes team size
  - $\alpha = 1 - (1 - \mathbf{p}^{\ell})^k$



● denotes a peer

Cardinality estimation: more likely to find all the required signatures in the same team

XGossip (2/2)

- Initialization and execution phases
- Convergence
  - Rounds: $O(\log(\Delta) + \log(1/\varepsilon) + \log(1/\delta))$
- Bandwidth reduction
  - Compress signatures when sending a gossip message

VanillaXGossip: Cardinality Estimation

- Suppose a peer wants to compute card(/Gene//goAcc)

Tuple list T @ the peer

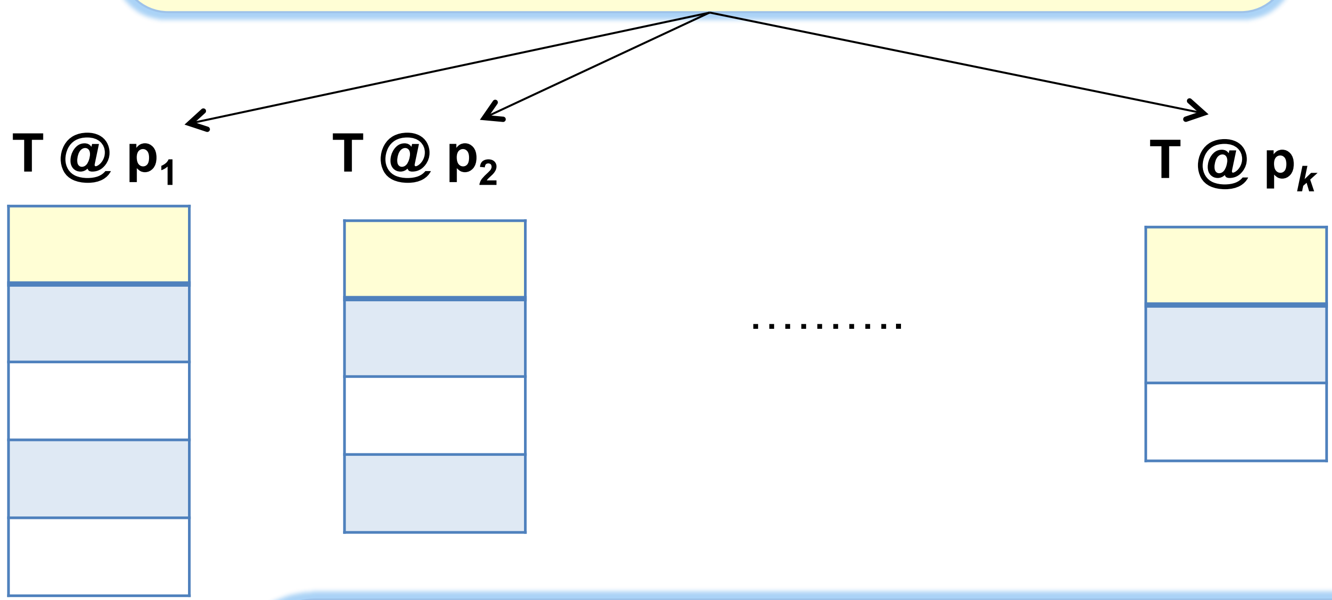
Signature	(sum,weight)
s_i	(f_i,w_i)

Sum the frequency estimates of signatures that are supersets of the query signature; multiply by n

XGossip: Cardinality Estimation

- Suppose a peer wants to compute card(/Gene//goAcc)

Apply LSH on the query signature → *k* teams

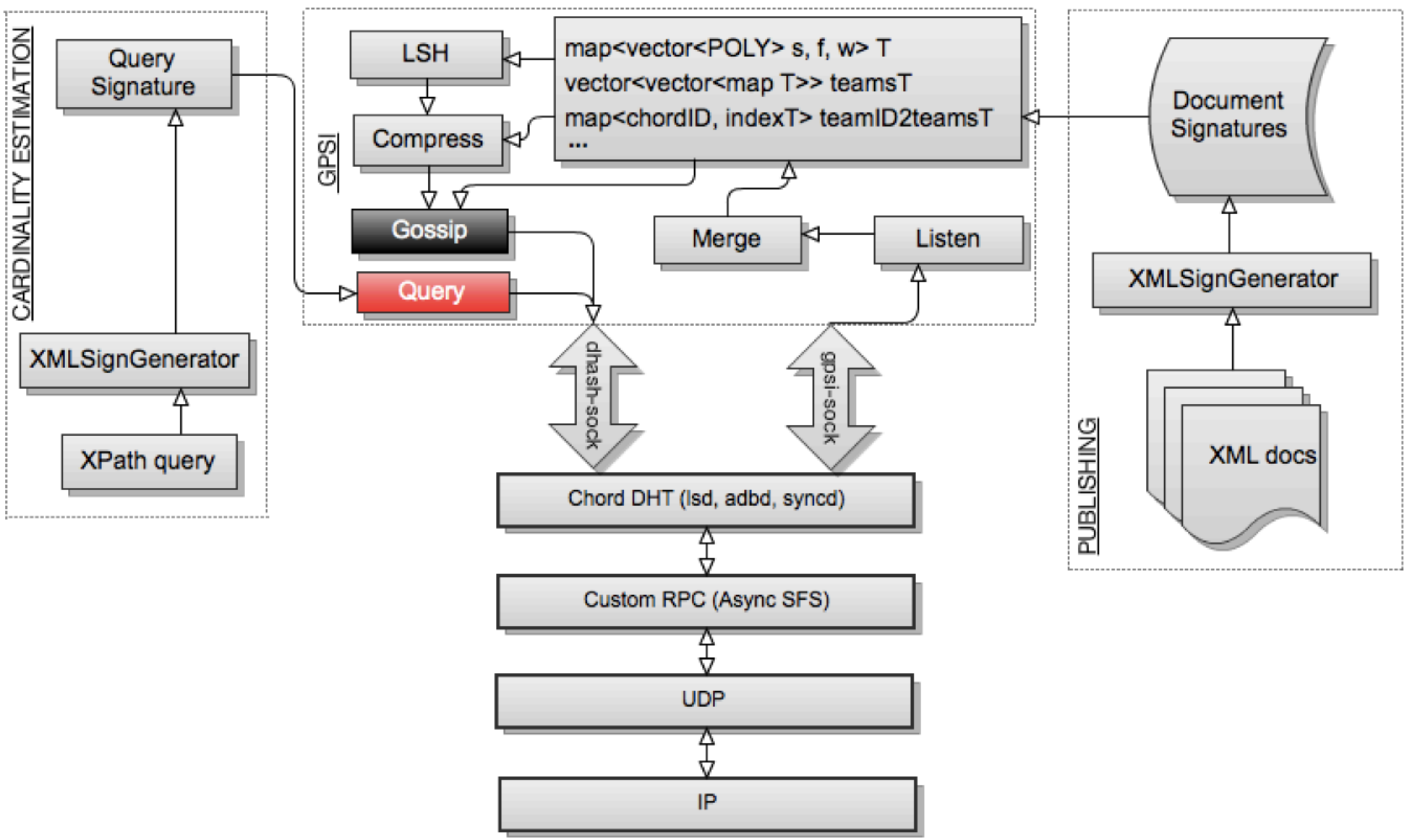


Merge the frequency estimates of signatures that are supersets of the query signature; multiply by Δ

Implementation

- Built on top of the Chord DHT [SIGCOMM '01]
  - Use Chord for routing (key-value pairs)
  - 4 processes per peer: Chord (lsd, syncd, adbd), Gossip (gpsi)
  - Communicate over UNIX sockets
  - Read signatures from files, store in main memory
- Implemented in C++
  - Data structures from STL
    - typedef std::map<std::vector<POLY>, std::vector<double>, CompareSig> mapType;
    - typedef std::map<chordID, std::vector<int> > teamid2totalT;
  - SFSlite asynchronous library
- Challenges

System Architecture



Performance Evaluation

- Datasets
  - Generated by the IBM Synthetic XML generator using well-known DTDs (Treebank, DBLP, etc.)
  - Uniformly distributed among all peers

Dataset	# of DTDs	Avg # of docs per DTD	Total # of docs	Avg. document signature size
D1	11	190,809	2,098,900	114 bytes
D2	13	192,223	2,498,900	127 bytes

Performance Evaluation

- Query sets
  - XPath queries generated by YFilter [TODS ‘03]
  - A total of 753 queries
  - 2 query approaches
    - LSH(XPath sig)
    - LSH(proxy sig)
  - Query subsets based on p_{min} value

Query Set	Value of p_{min}	# of queries
Q_0	[0, 0.5)	101
Q_1	[0.5, 1]	652
Q_2	[0.6, 1]	356
Q_3	[0.7, 1]	300
Q_4	[0.8, 1]	277
Q_5	[0.9, 1]	26

Network Setup and Distribution of Documents

- Amazon EC2
  - 20 medium instances (2 cores, 1.7GB memory)
  - US East availability zone
- Peers use local clock
- 120s long rounds
- Variables
  - team size (Δ): 8, 16
  - LSH parameter k: 4, 8
  - LSH parameter l: 10 (fixed): $\alpha = 0$, when $p < 0.5$
  - compression
  - # of peers: 500, 1000, 2000

Total # of peers in the network (*n*)	# of peers per EC2 instance	# of peers picked per DTD (*z*)
500	25	250
1000	50	500
2000	100	1000

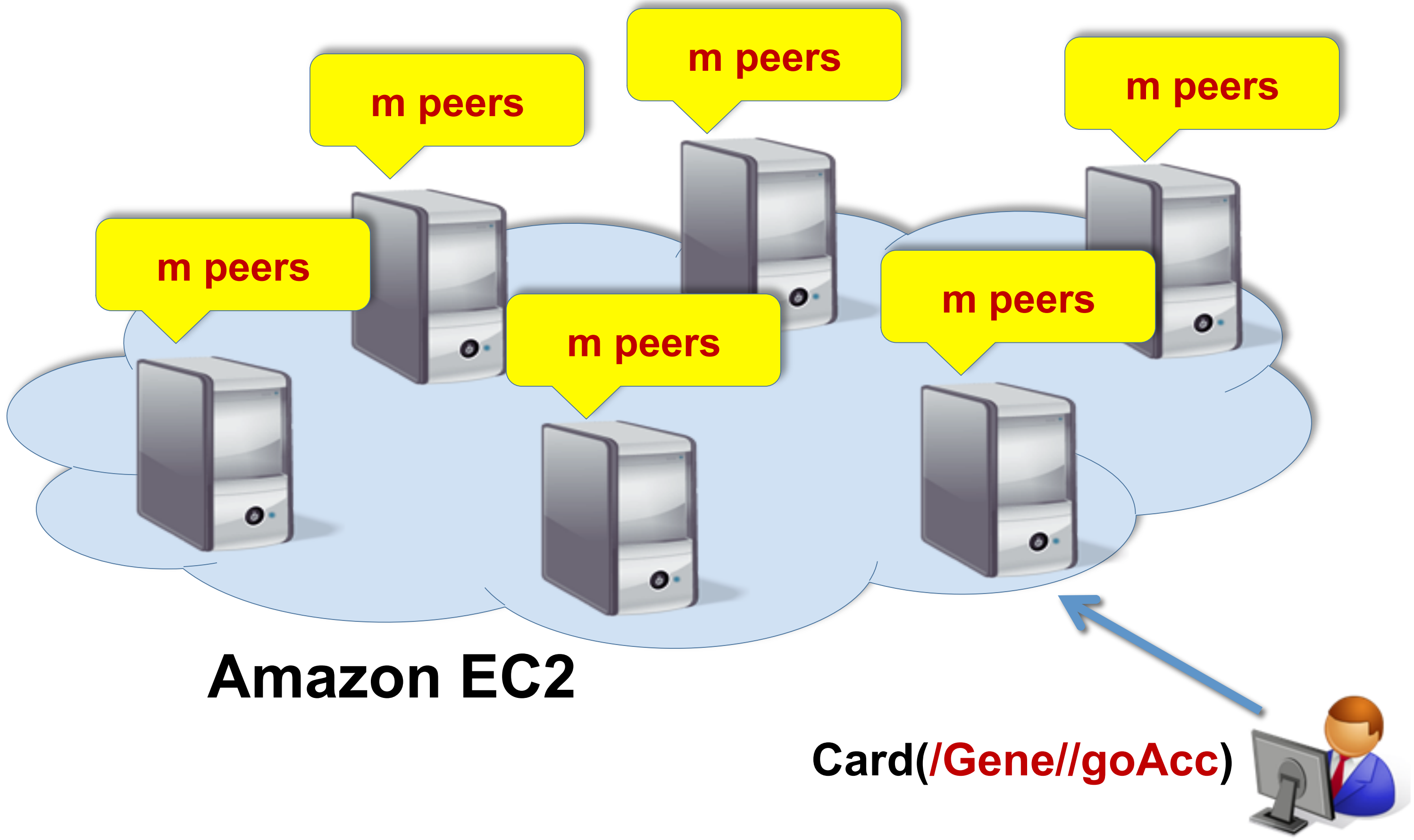
Evaluation Metrics

- Convergence speed of the frequency of signatures
  - Mean absolute relative error (MARE) of the frequency estimate of the document signatures

$$\frac{1}{M} \times \sum_{i=1}^{M} \frac{|ef_i - f_i|}{f_i}$$

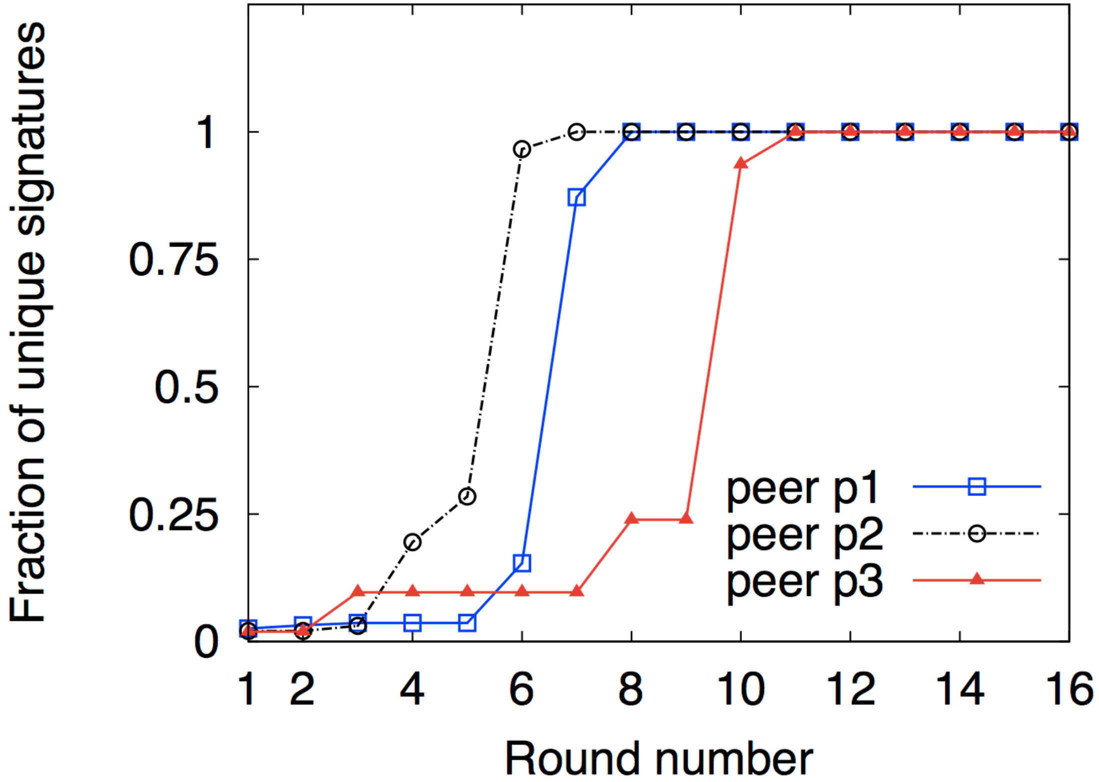
  - Where
    - M: tuple list size
    - f_i: true signature frequency
    - ef_i: estimated signature frequency
      - VanillaXGossip: freq / weight * n
      - XGossip: freq / weight * Δ
- Accuracy of cardinality estimation
  - MARE of the cardinality estimate of the queries
- Bandwidth consumption during gossip
  - Amount of data transmitted per round by all peers

XGossip in the Cloud



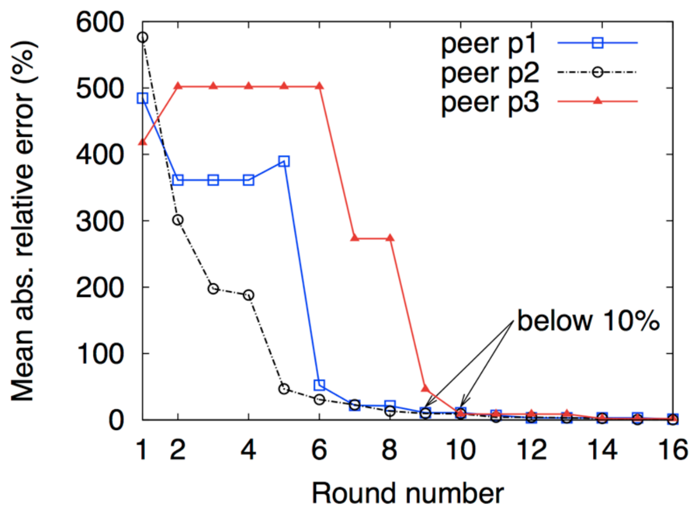
Diffusion Speed of Signatures

VanillaXGossip



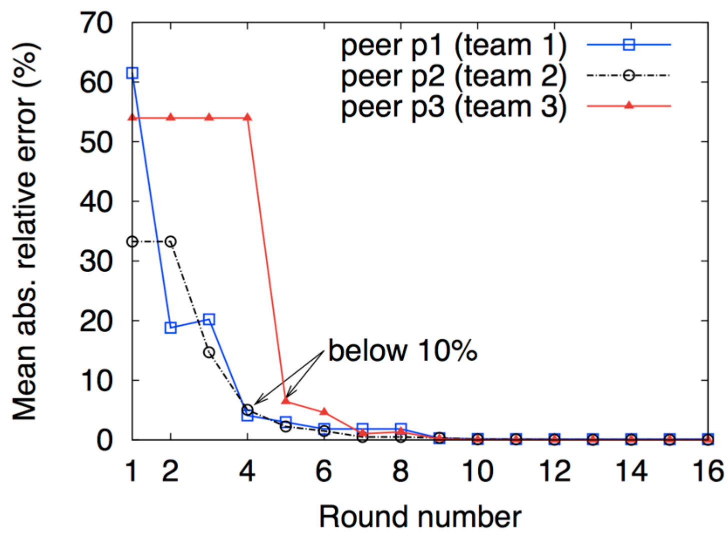
Convergence of Frequencies of Signatures

VanillaXGossip



Dataset D_1, n = 1000

XGossip

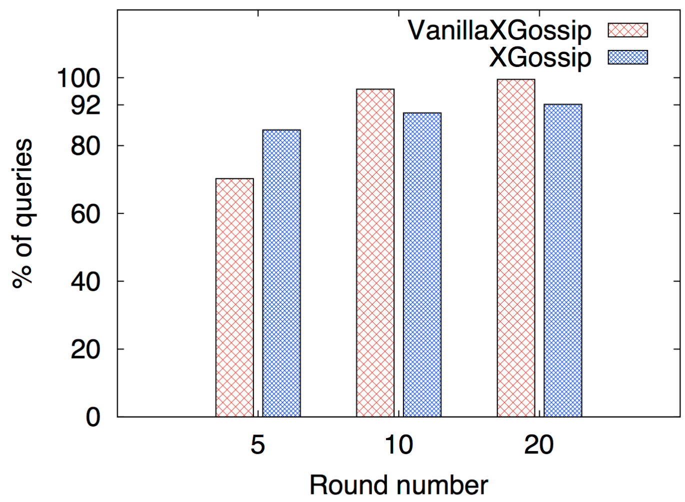


Dataset D_1, n = 1000
Δ = 8, k = 8, l = 10

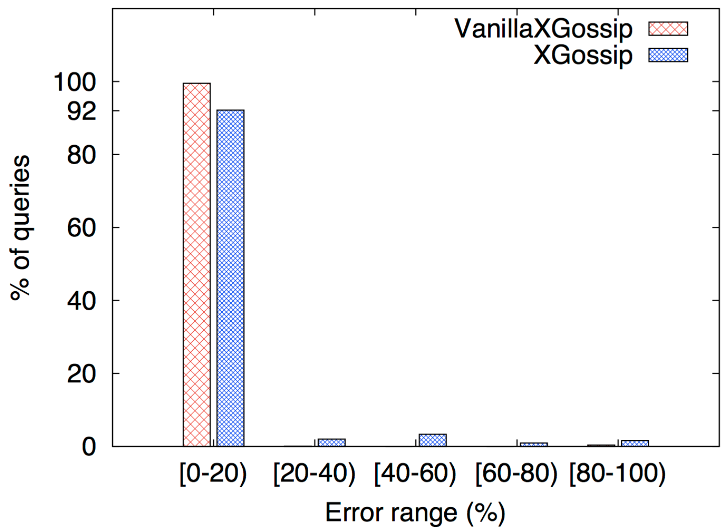
Accuracy of Cardinality Estimation (1/6)

VanillaXGossip vs. XGossip

At different rounds



After 20 rounds

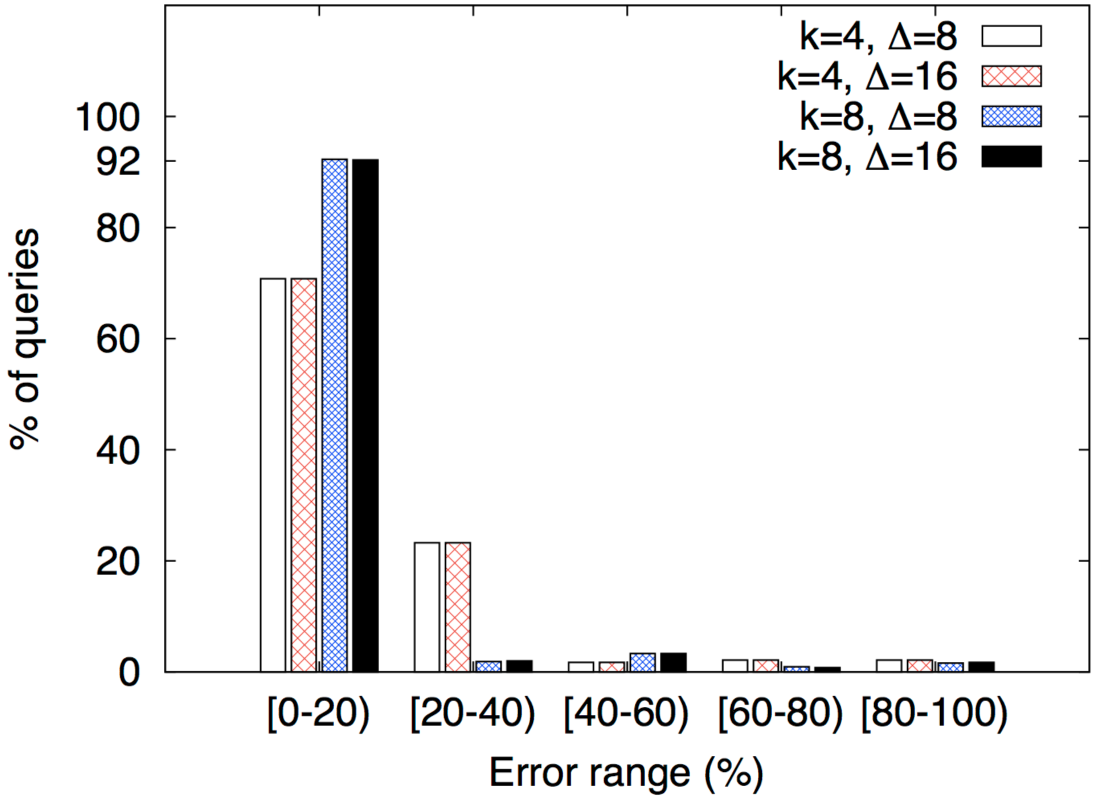


Relative error below 20%

n = 1000, Δ = 8, k = 8, l = 10

Accuracy of Cardinality Estimation (2/6)

XGossip: LSH and team size

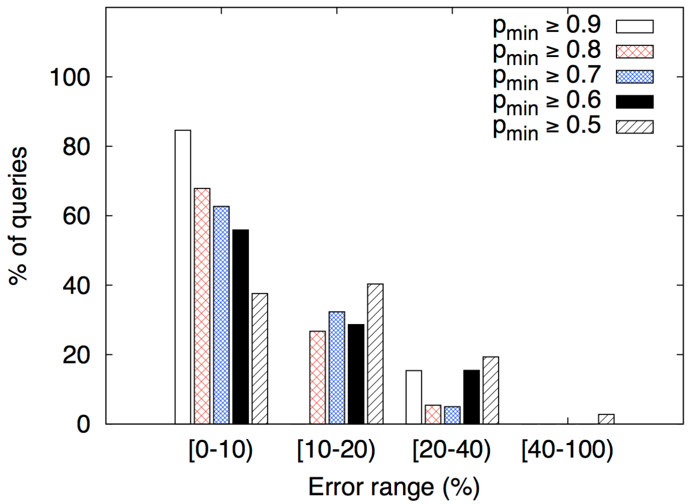


After 30 rounds, n = 1000, l = 10

Accuracy of Cardinality Estimation (3/6)

XGossip

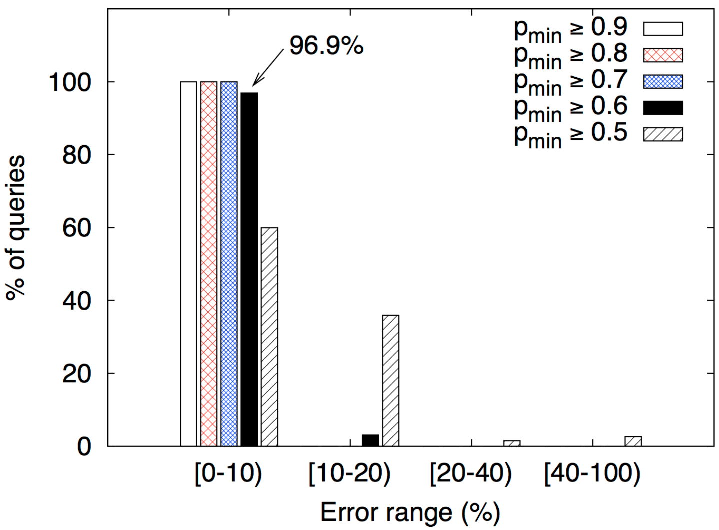
Δ = 8, k = 4



n = 1000, after 30 rounds

Δ = 16, k = 4 is almost identical

Δ = 8, k = 8

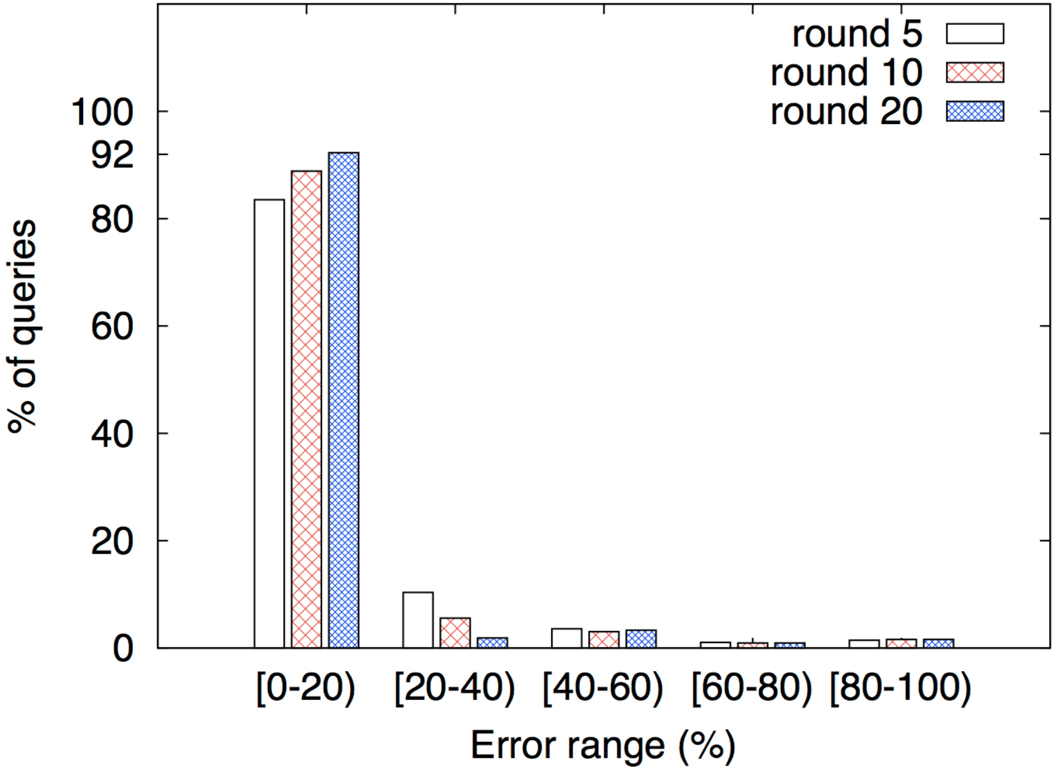


n = 1000, after 30 rounds

Δ = 16, k = 8 is almost identical

Accuracy of Cardinality Estimation (4/6)

XGossip: increasing # of rounds

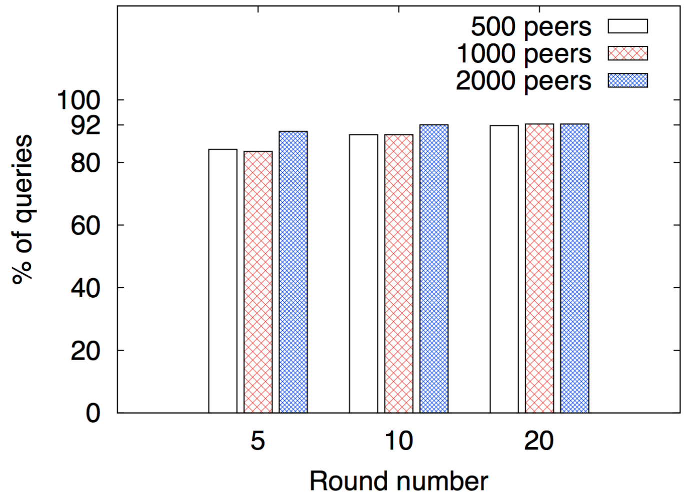


n = 1000, Δ = 8, k = 8, l = 10

Accuracy of Cardinality Estimation (5/6)

XGossip

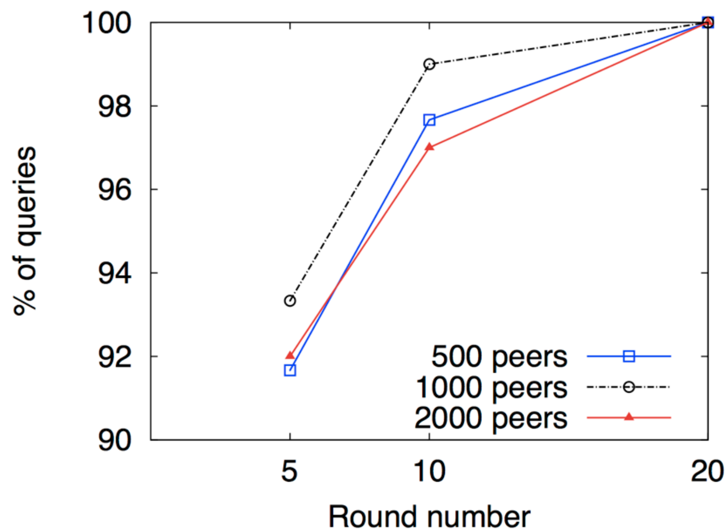
Varying # of peers



Relative error below 20%

Δ = 8, k = 8, l = 10

Varying # of peers (query set Q_3: [0.7, 1])



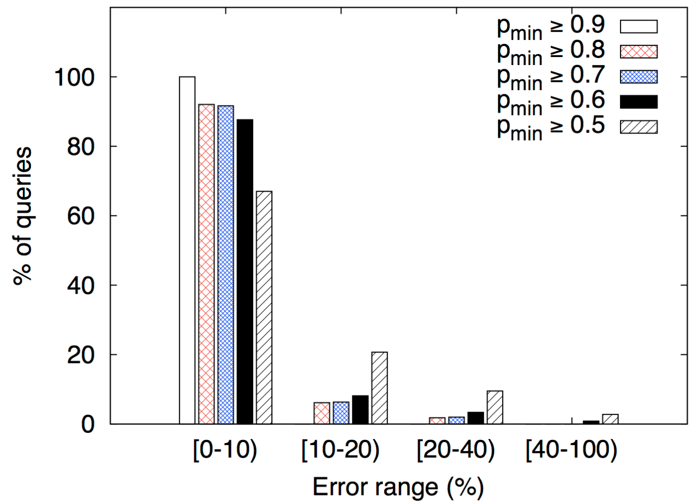
Relative error below 10%

n = 1000, Δ = 8, k = 8, l = 10

Accuracy of Cardinality Estimation (6/6)

XGossip

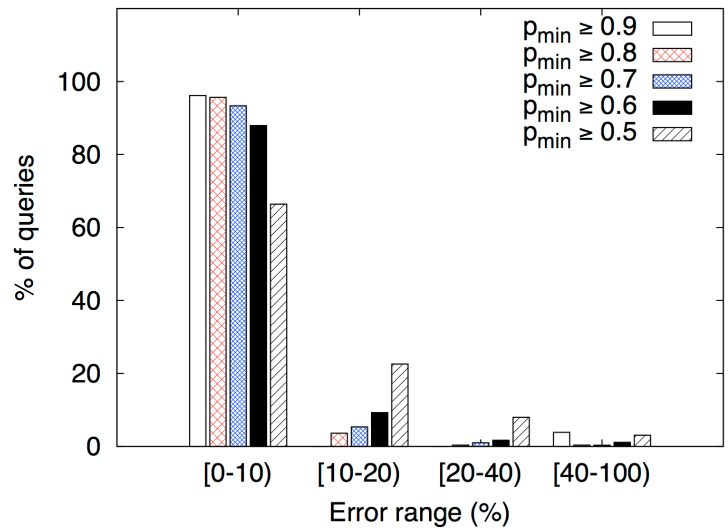
500 peers



After 5 rounds

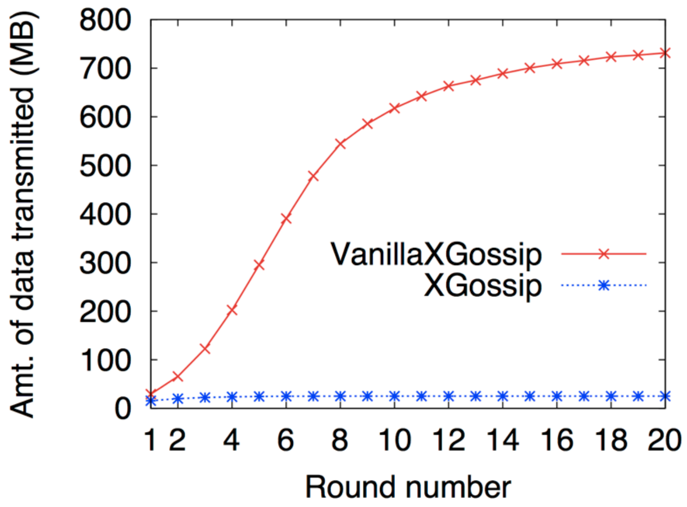
$\Delta = 8$, $k = 8$, $l = 10$

1000 peers



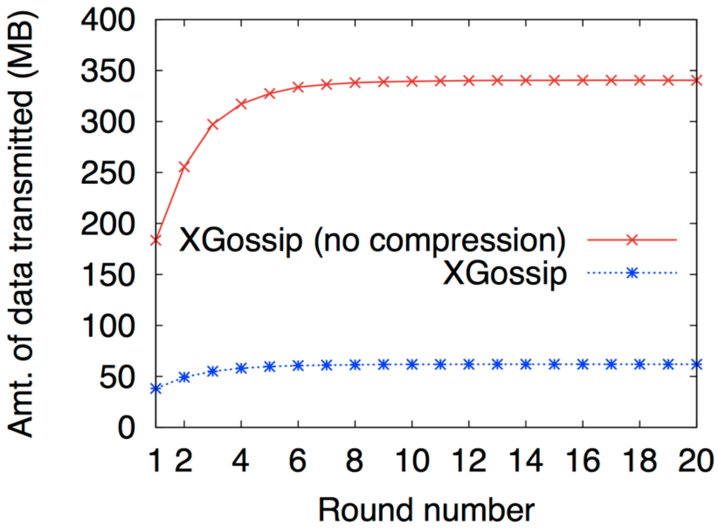
Bandwidth Consumption (1/2)

VanillaXGossip vs. XGossip (Dataset D_1)



VanillaXGossip: 10,309 MB
XGossip: 484 MB

XGossip compression (Dataset D_2)

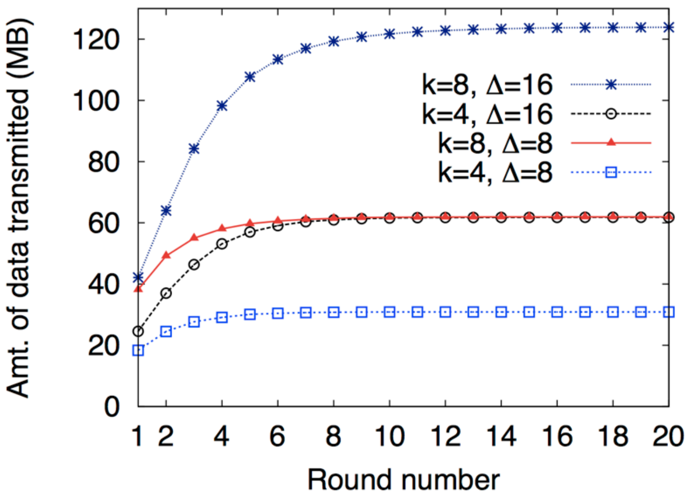


XGossip (no compression): 9,874.2 MB
XGossip: 1,805.9 MB

Bandwidth Consumption (2/2)

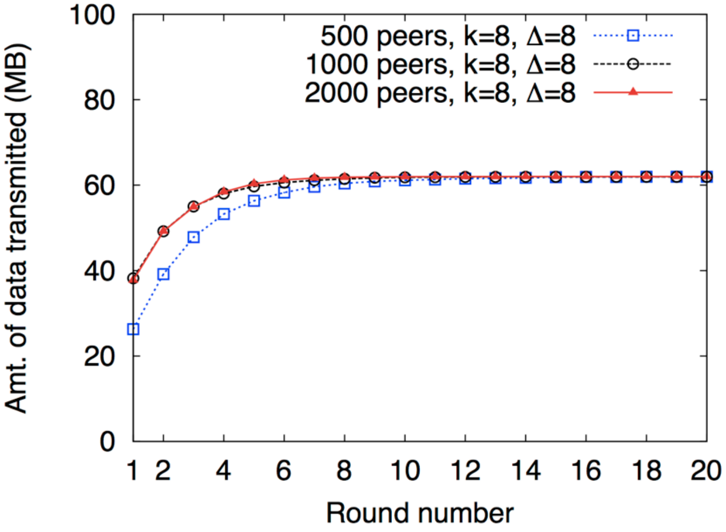
XGossip

Different values of k and Δ



n = 1000

Varying the # of peers



l = 10

Performance Analysis: Results Summary

- LSH: tune k and l
- Compression works
- XGossip scales, VanillaXGossip does not
- XGossip converges faster than VanillaXGossip
- XGossip transmits less data than VanillaXGossip

# of peers	Avg. # of teams/peer	Avg. # of sigs/peer	Avg. # of sigs/team	Avg. msg size/peer (bytes)	Total # of msgs
500	88.40	4,024.25	45.52	1,160.18	440,240
1000	44.82	2,040.81	45.52	1,265.64	440,240
2000	23.09	1051.20	45.52	1,244.91	441,750

Conclusion

- Thesis objectives
  1. Implementing gossip in an Internet-scale environment
  2. Conducting a comprehensive evaluation
  3. Analyzing the experimental results
- The results we obtained were consistent with the theoretical analysis of VanillaXGossip and XGossip.

Questions?

- References
  - Vasil G. Slavov, Praveen R. Rao - **Towards Internet-Scale Cardinality Estimation of XPath Queries over Distributed XML Data.** *Proceedings of 6th International Workshop on Networking Meets Databases* (NetDB 2011), Athens, Greece.
  - Praveen Rao and Vasil Slavov - **Towards Internet-Scale Cardinality Estimation of XPath Queries over Distributed XML Data**. University of Missouri-Kansas City, Kansas City, MO 64110, Tech. Rep. TR-DB-2011-01, Jun. 2011, http://r.faculty.umkc.edu/raopr/TR-DB-2011-01.pdf.
- Acknowledgements
  - National Science Foundation (IIS-1115871), 2011-2014