INTERNET-SCALE CARDINALITY ESTIMATION OF XPATH QUERIES OVER DISTRIBUTED XML DATA

Praveen R. Rao and Vasil G. Slavov

Computer Science & Electrical Engineering

University of Missouri-Kansas City

Acknowledgements

National Science Foundation (IIS-1115871), University of Missouri Research Board, IBM Smarter Planet Faculty Innovation Award 2010

Roadmap

- Introduction
- Background
- XPath cardinality estimation
  - VanillaXGossip
  - XGossip
- Implementation and evaluation
  - Amazon Elastic Compute Cloud (EC2)
- Conclusions

By the end of this talk…

- ♦“Gossip is good” in large-scale distributed systems

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>
  - <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>
```

XPath

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

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],
...

Peer-to-peer Computing (Distributed Hash Tables)

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

Gossip (or epidemic) algorithms

…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 = ?

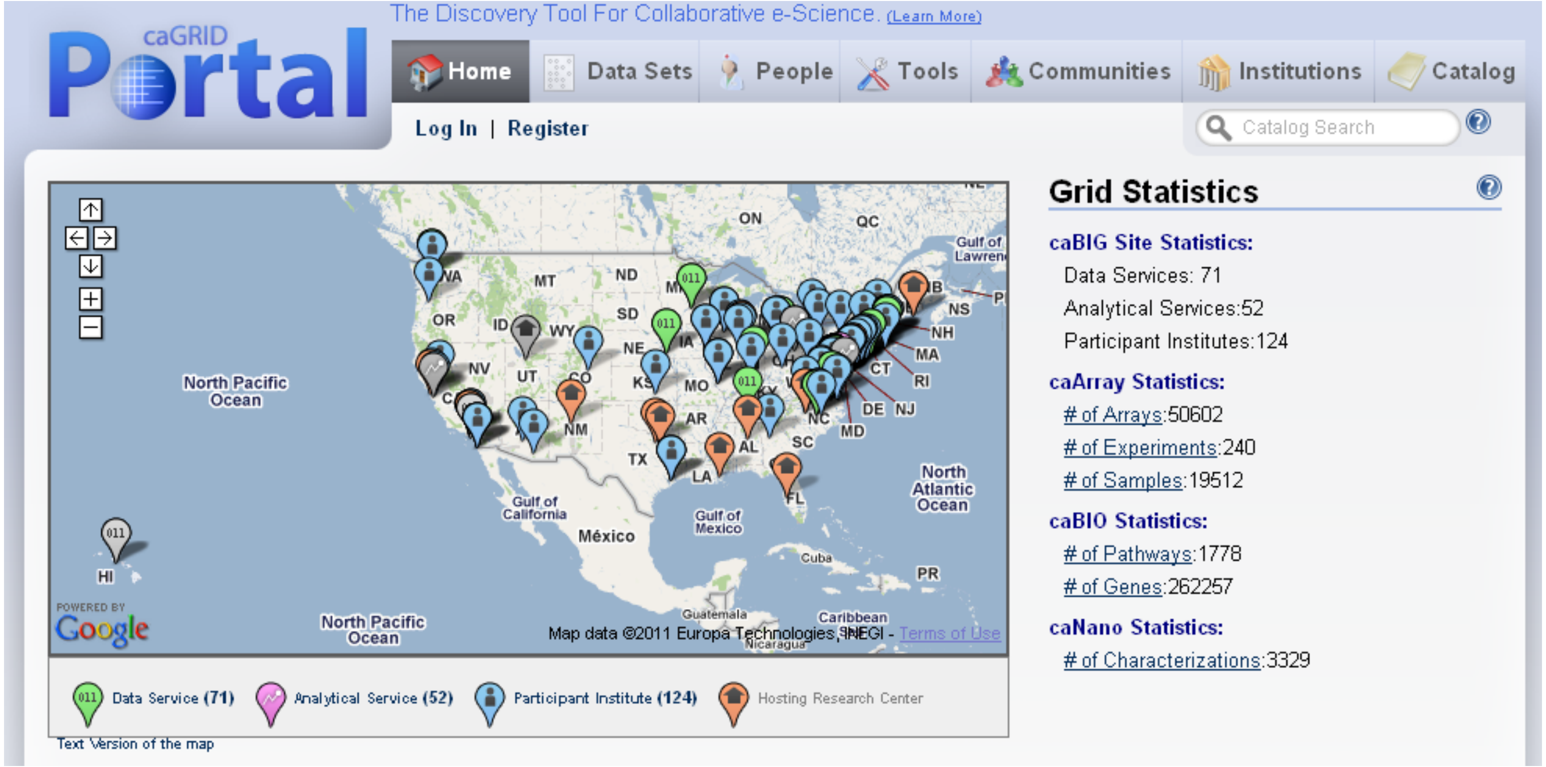
Large-scale sharing of biomedical and clinical data

Scalable clinical data sharing systems via a P2P architecture [Stead and Lin, 2009]

HL7 version 3 standard; XML based; semantic interoperability; can model discharge summaries, lab reports, …

A Real World Data Sharing Platform

- The Cancer Biomedical Informatics Grid (caBIG)
  - Source: http://cagrid-portal.nci.nih.gov/
  - About 124 participants across the US (SOA, XML databases)

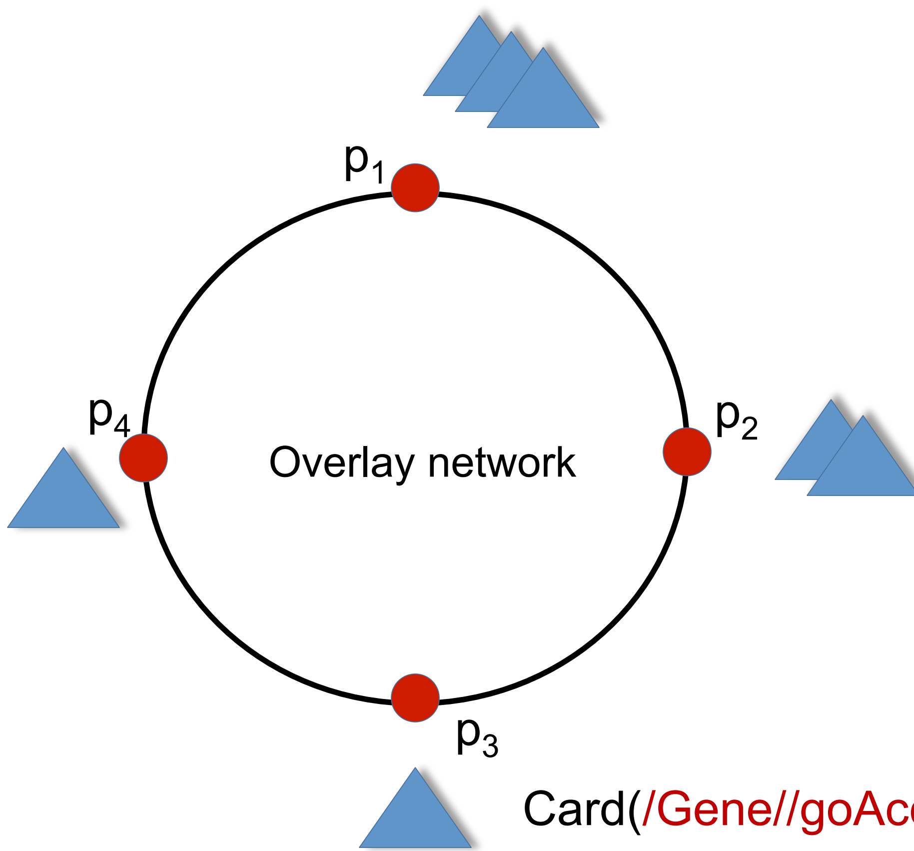


Example from caBIG

“Find all the expression data where there are at least 50 conditions for genes found in the vacuole”

```
FOR  $gene IN service
("http://cabio.osu.edu/GeneService.wsdl")/Gene,
$go IN service
("http://cabio.osu.edu/GeneOntologyService.wsdl")/GeneOntology,
$microarray IN service
("http://caarray.duke.edu/caArrayService.wsdl")/Microarray
LET $subject := $microarray/experiment/subject
WHERE
  $go/term='vacuole' AND $gene/goAcc=$go/acc AND
  $gene/gbAcc=$microarray/data/geneId AND
  count($microarray/data[geneId=$gene/$gbAcc]/condition)>50
RETURN
<subject>
  <subjectId>{ $subject/lsid }</subjectId>
  <species>{ $subject/species }</species>
  <microarrayData>
    { $microarray/data }
  </microarrayData>
</subject>
```

The Problem: XPath Cardinality Estimation



Card(/Gene//goAcc) = ?

Compute the number of documents in the network that contain a match for the expression /Gene//goAcc

XPath Cardinality Estimation

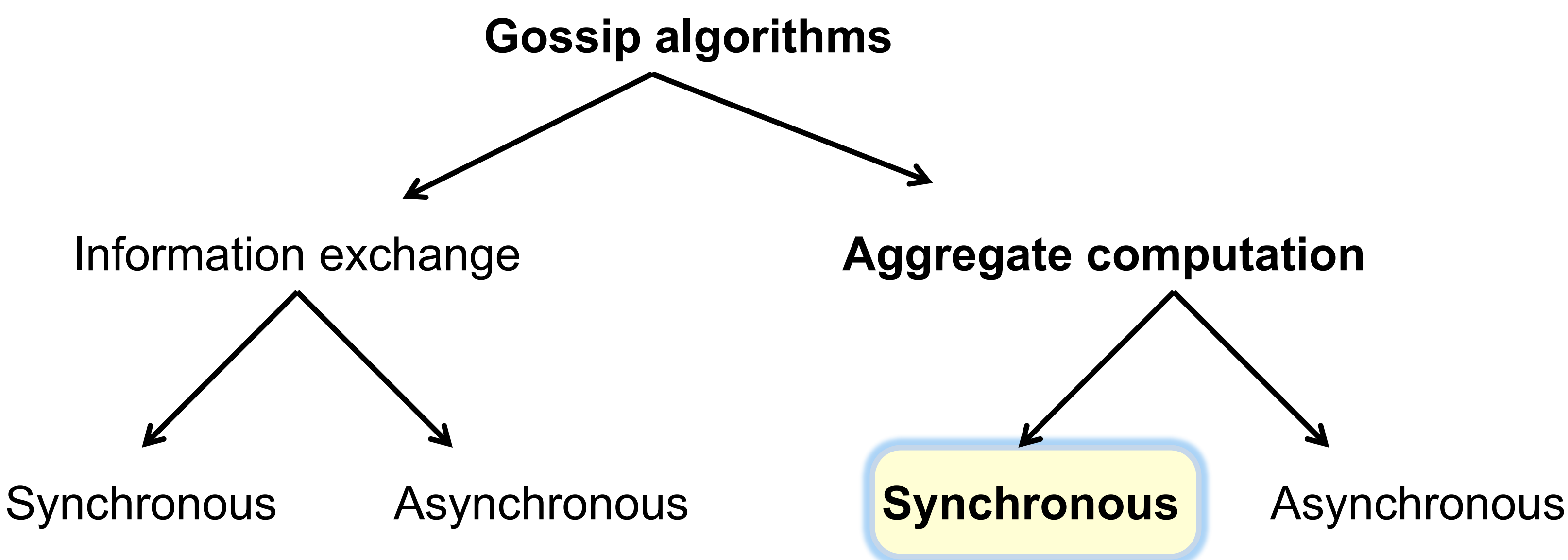
- Useful for
  - XQuery optimization
    - E.g., to select a particular join ordering
  - Designing IR-style ranking schemes
  - Designing clinical studies
    - E.g., to determine if sufficient number of samples available to conduct a study

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



Can further classify based on the topology of the network

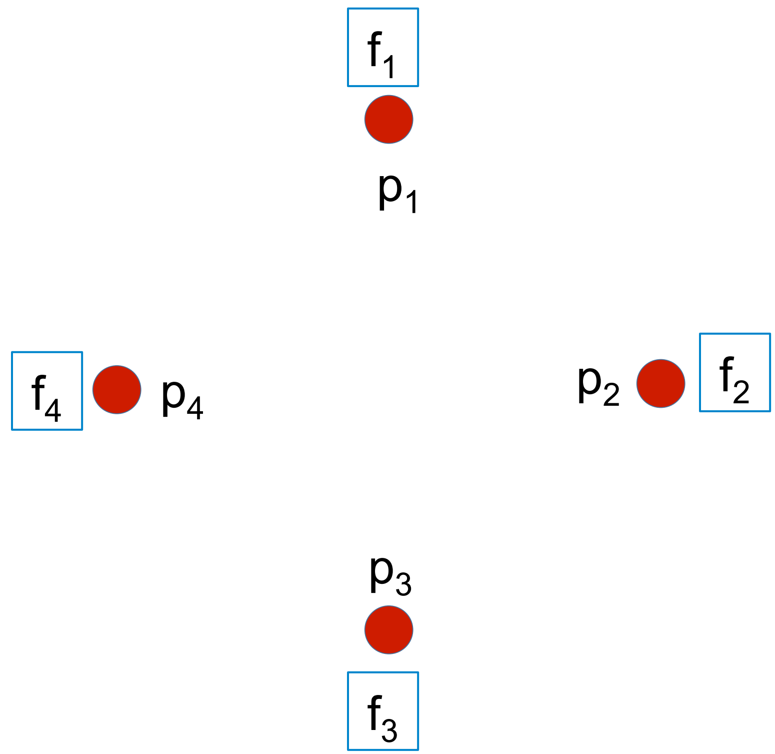
Real World Examples

- Gossip algorithms are used in practice
  - Amazon’s Dynamo (key-value store)
  - Amazon’s S3 data centers
  - Facebook’s Cassandra (key-value store)

Push-Sum Protocol (1/4)

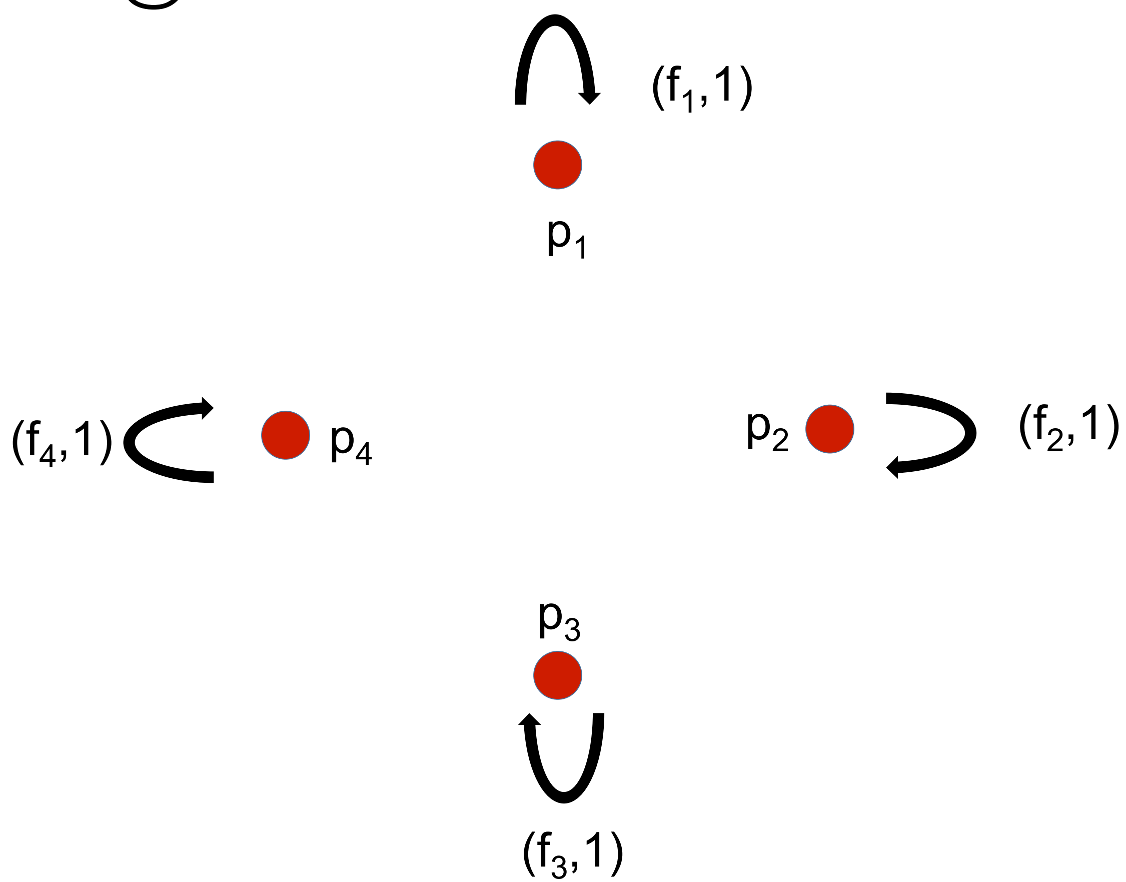
- By Kempe, Dobra, and Gehrke [FOCS ‘03]
- Each peer wishes to know the average

$$avg = \frac{(f_1+f_2+f_3+f_4)}{4}$$



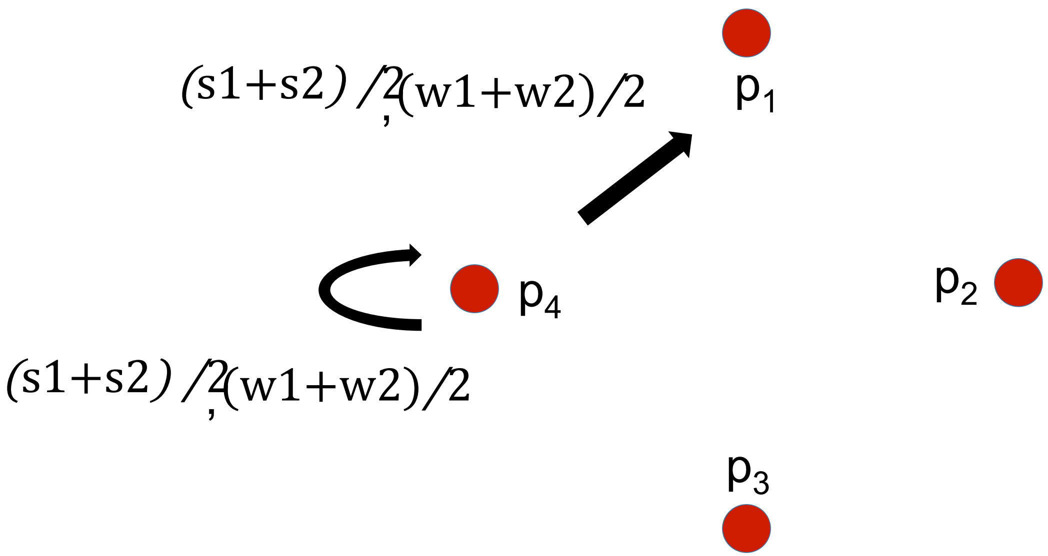
Push-Sum Protocol (2/4)

- Each peer maintains a (sum, weight) pair during gossip
- In the round @ t = 0



Push-Sum Protocol (3/4)

- In any round @ t > 0

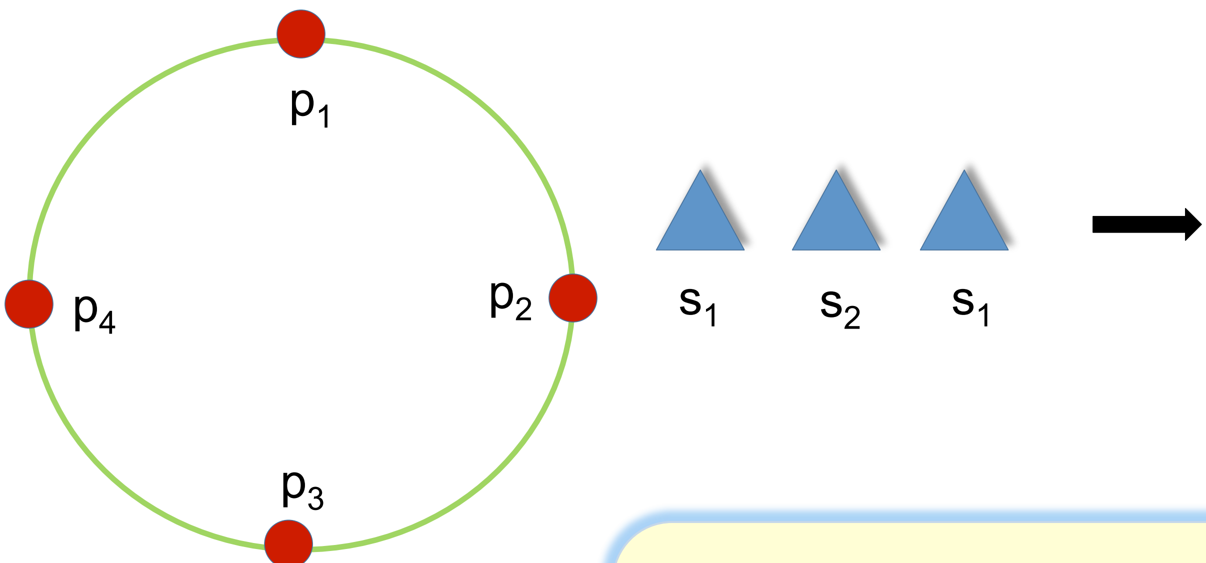


Proof is based on the property of "mass conservation"

Messages: *n* messages per round

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 ☺

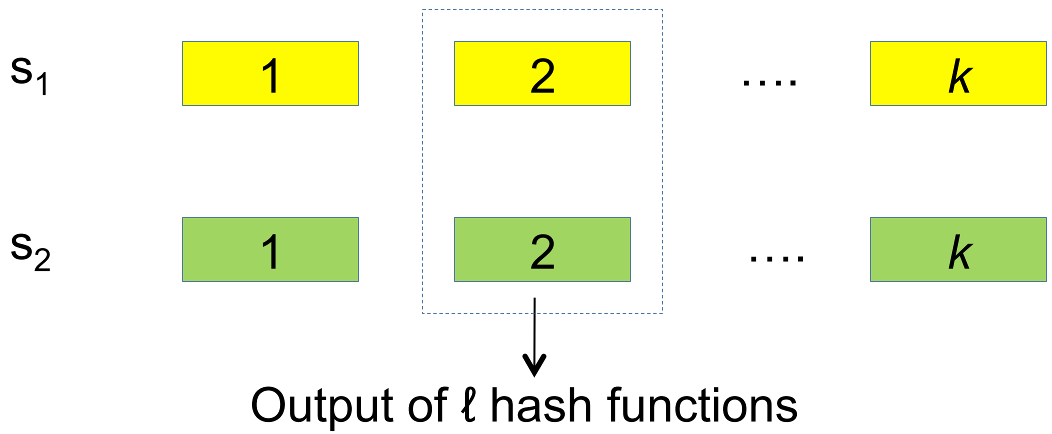
Less memory, less bandwidth, faster convergence

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 **p** = $|s_1 \cap s_2| / |s_1 \cup s_2|$
- Pick k x ℓ random linear hash functions

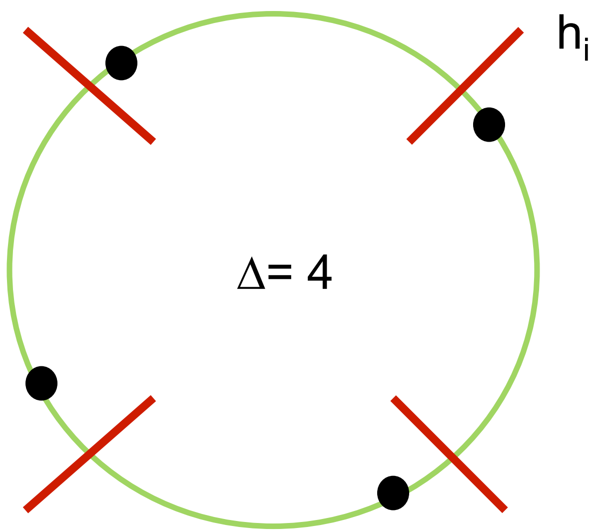


P[at least one p … = $1 - (1 - \mathbf{p}^{\ell})^{k}$

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, $1 \leq i \leq k$; Δ denotes team size



● 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

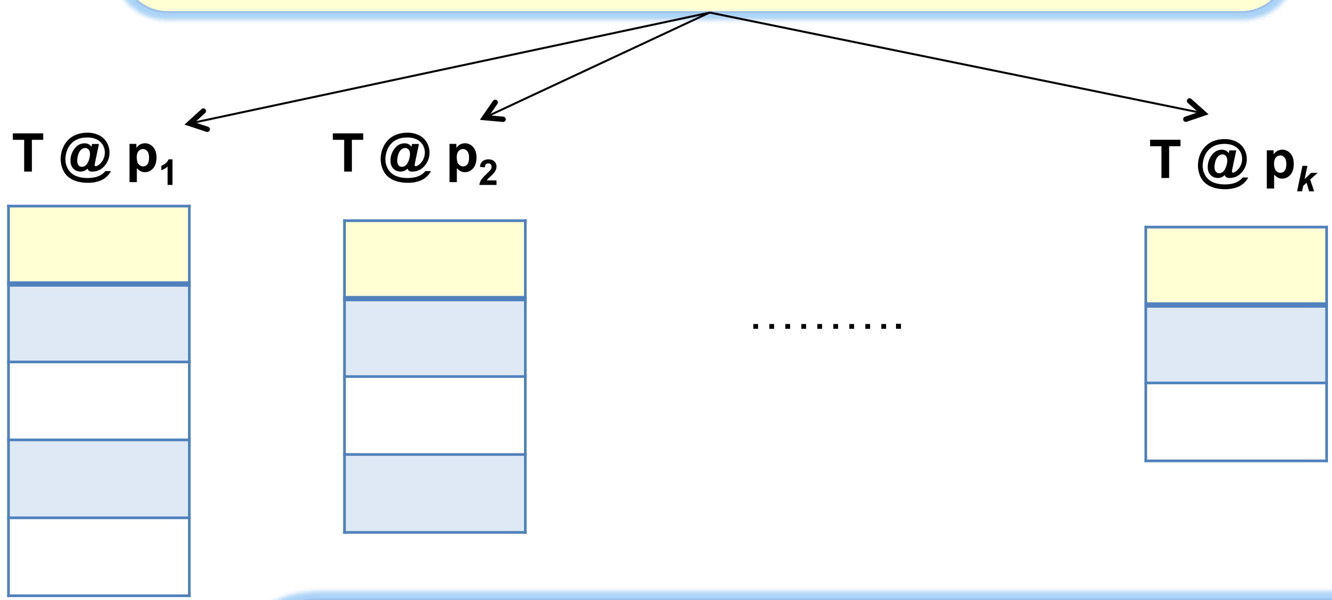
Signature	(sum,weight)
s_i	(f_i,w_i)

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

Asymptotic Analysis

Metric	VanillaXGossip	XGossip
Accuracy	$r\epsilon$	$r\epsilon$
Confidence	$(1-\delta)$	$(1-\delta)$
Convergence (# of rounds)	$O(log(n) + log(\frac{1}{\epsilon}) + log(\frac{1}{\delta}))$	$O(log(\Delta) + log(\frac{1}{\epsilon}) + log(\frac{\alpha}{\alpha+\delta-1}))$
Bandwidth	$O(nD)$	$O(log(n)kD\Delta)$
Messages	$O(n\ log(n))$	$O(\frac{log(n)}{n}kD\Delta log(\Delta))$

r → # of signatures that are supersets of a query signature

α depends on the minimum similarity between the query signature and the *r* distinct document signatures that it divides, and *k* and *ℓ*

D → # of distinct signatures in the network

A peer becomes a successor of O(*kD* log(*n*)/*n*) teams (using the property of consistent hashing)

# of distinct signatures per team → O(*kD*Δ log(*n*)/*n*)

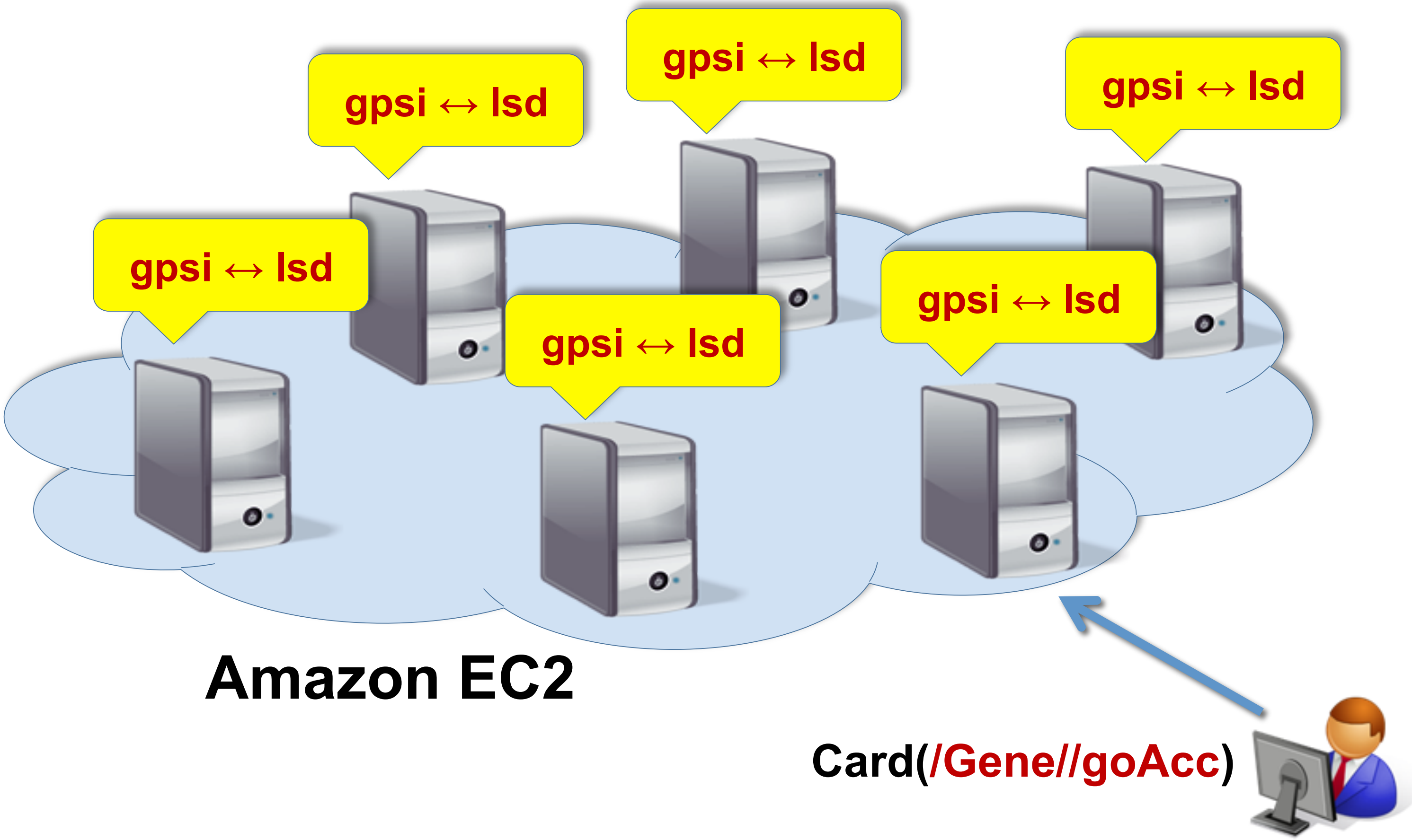
VanillaXGossip: will always find all the superset signatures

XGossip: may miss a superset signature if its similarity with the query signature is below the threshold used by LSH

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

System Architecture



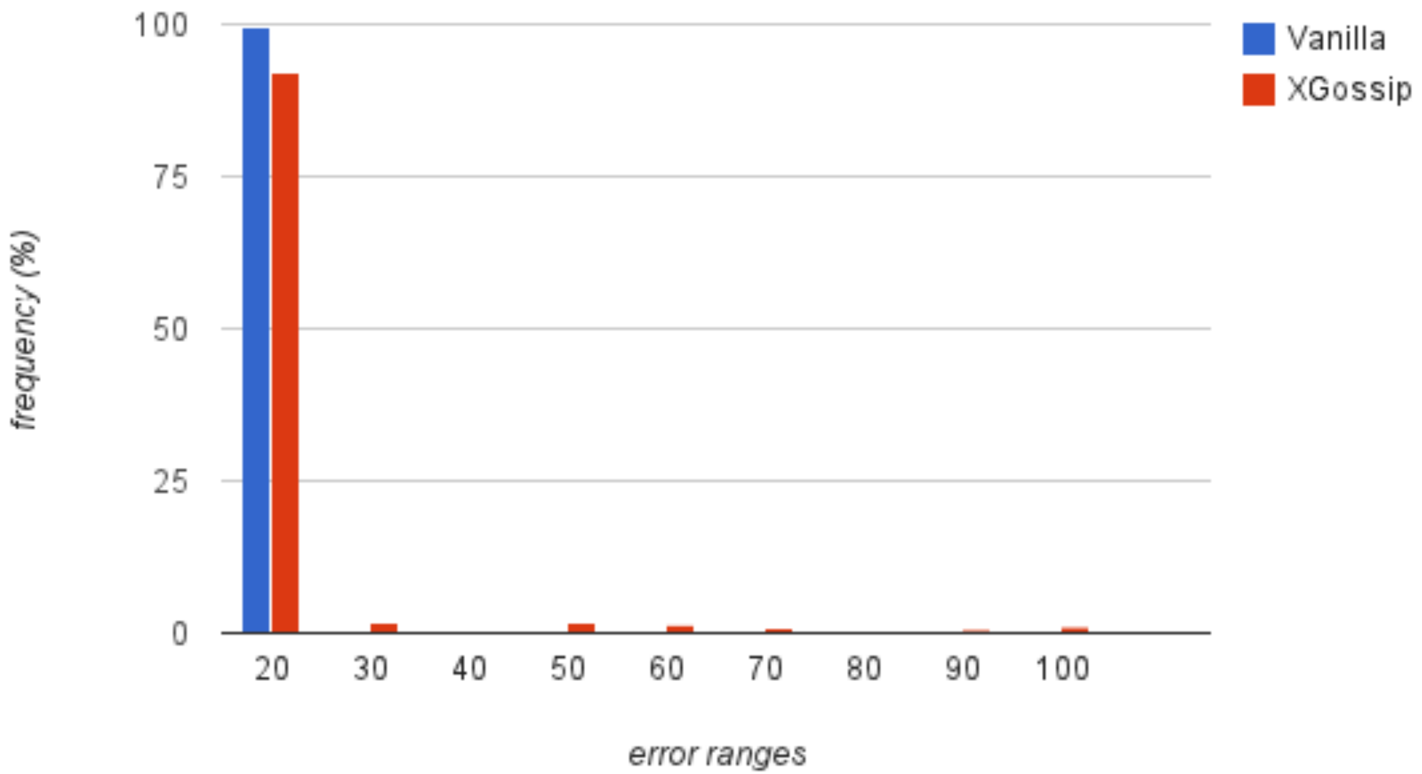
Performance Evaluation

- Amazon EC2
  - 20 medium instances (2 cores, 1.7GB memory), 50 peers/instance, 1000 peers
- Datasets
  - Generated by the IBM Synthetic XML generator using well-known DTDs (Treebank, DBLP, etc.)
  - Uniformly distributed among all peers (25 hosts/DTD)
- Queries
  - XPath queries generated by YFilter [TODS '03]
  - LSH(XPath): VanillaXGossip, XGossip
  - LSH(proxy sig): XGossip
- Variables: team size (8, 16); LSH: K (4, 8), L (10); compression

Number of DTDs	Avg # of docs per DTD	Total # of docs	Avg. document signature size	Total # of queries
11	9,540	104,945	114 bytes	1977
13	9,611	124,945	127 bytes	2355

Query Error (1/3)

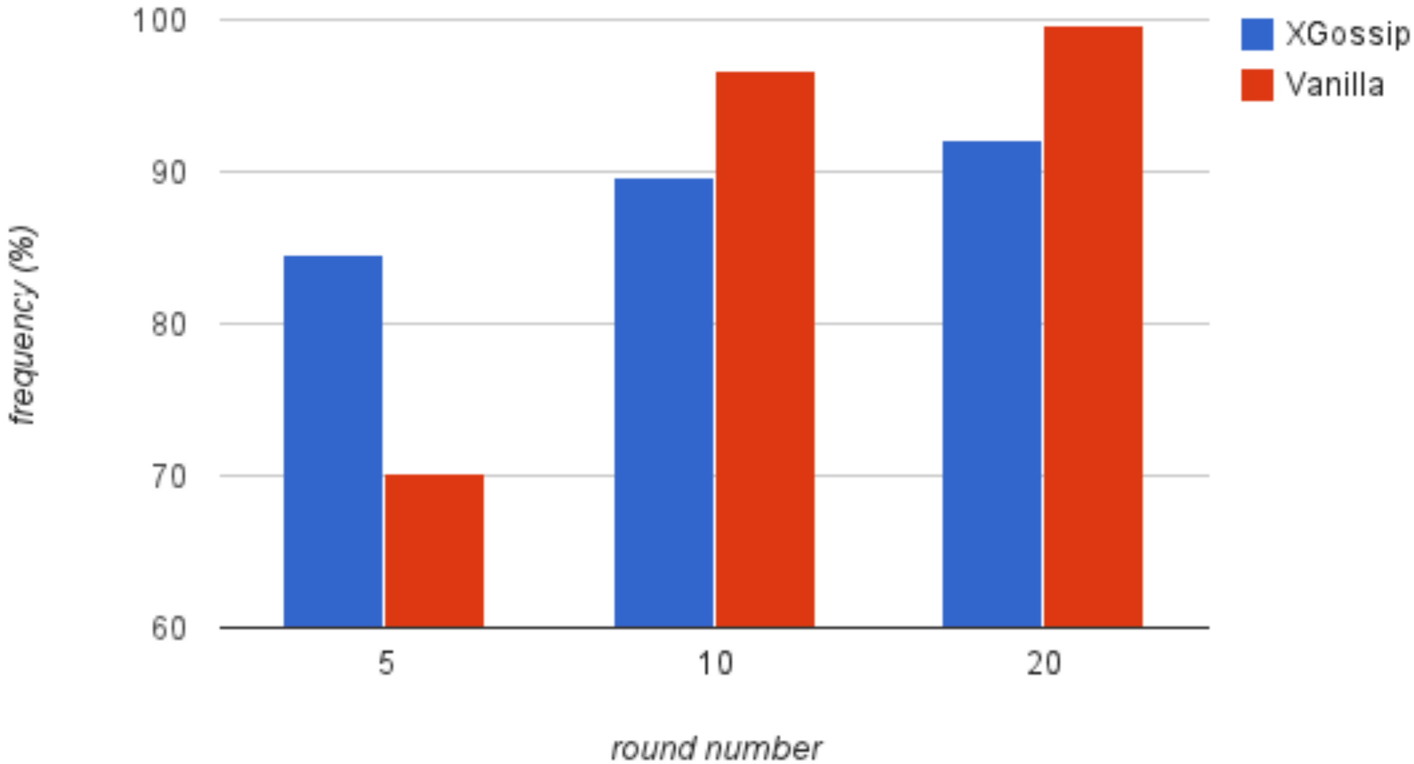
VanillaXGossip vs XGossip



- Frequency of queries with relative error within the specified ranges

Query Error (2/3)

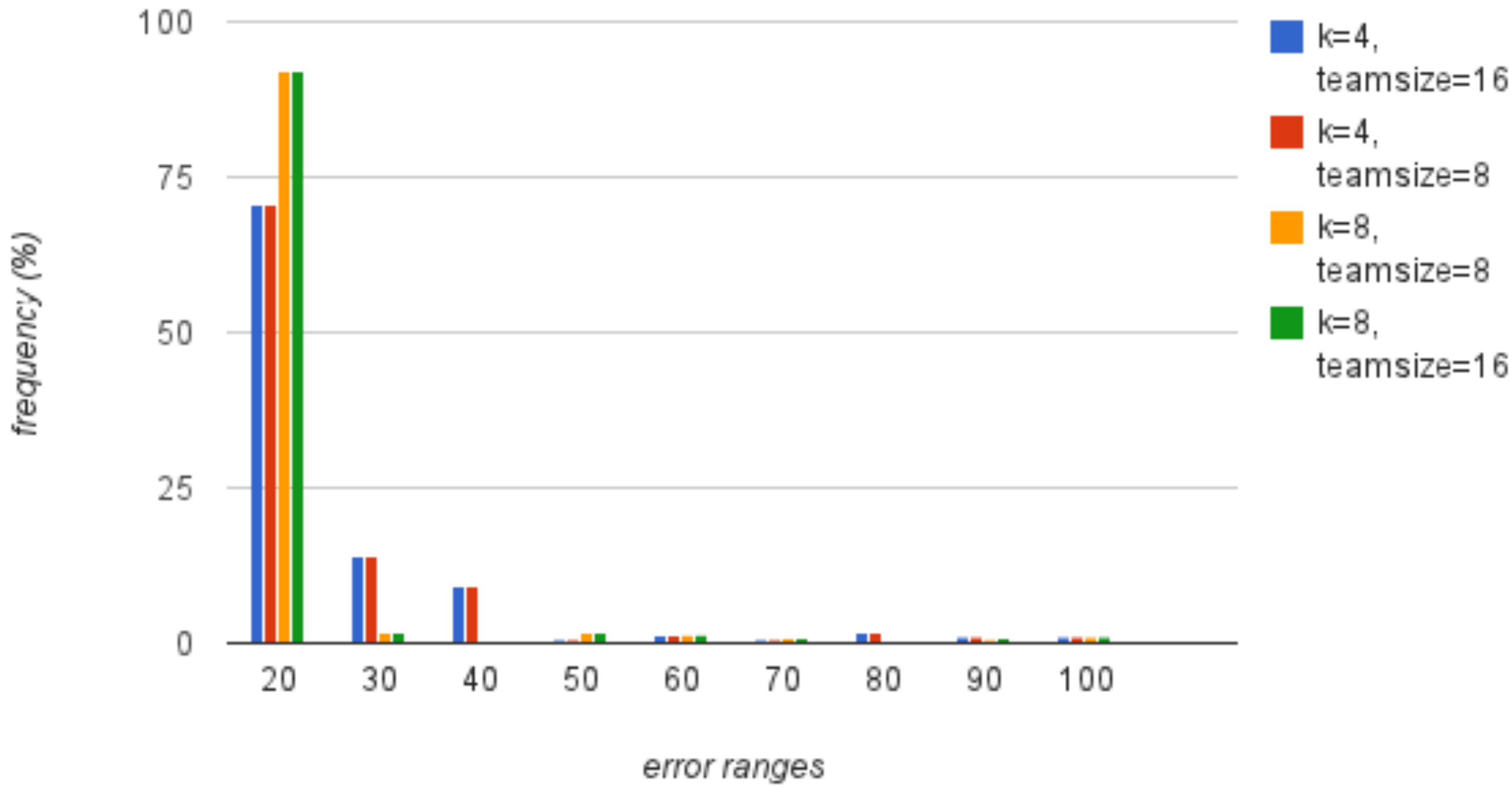
VanillaXGossip vs XGossip: rounds



- Frequency of queries with relative error below 20%

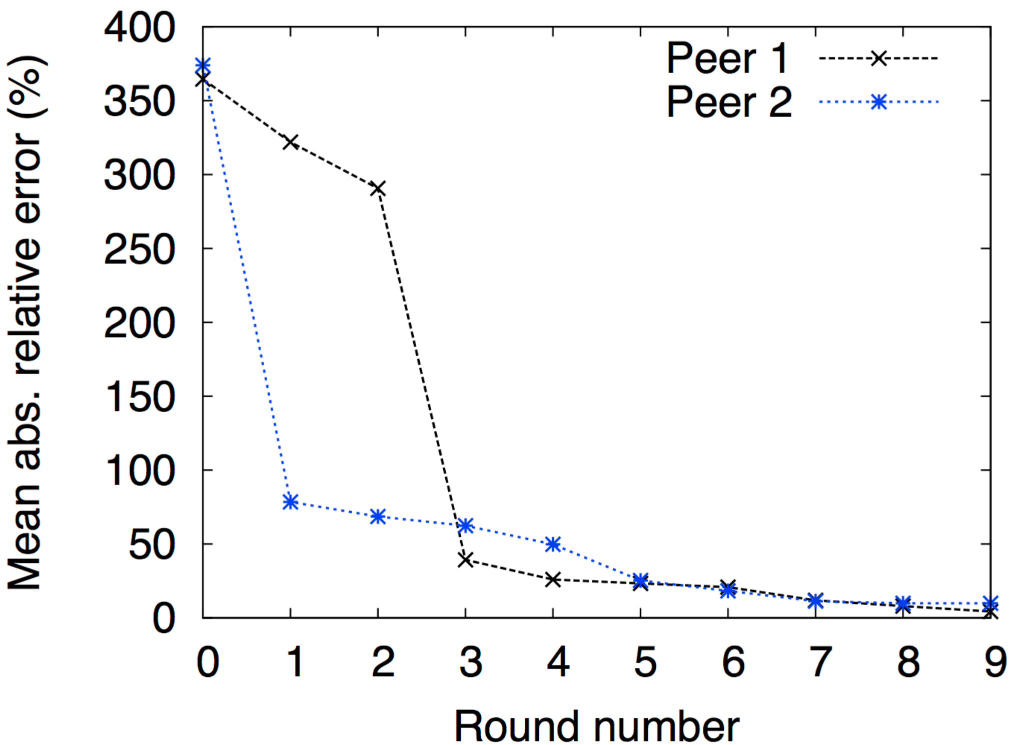
Query Error (3/3)

XGossip: LSH and team size



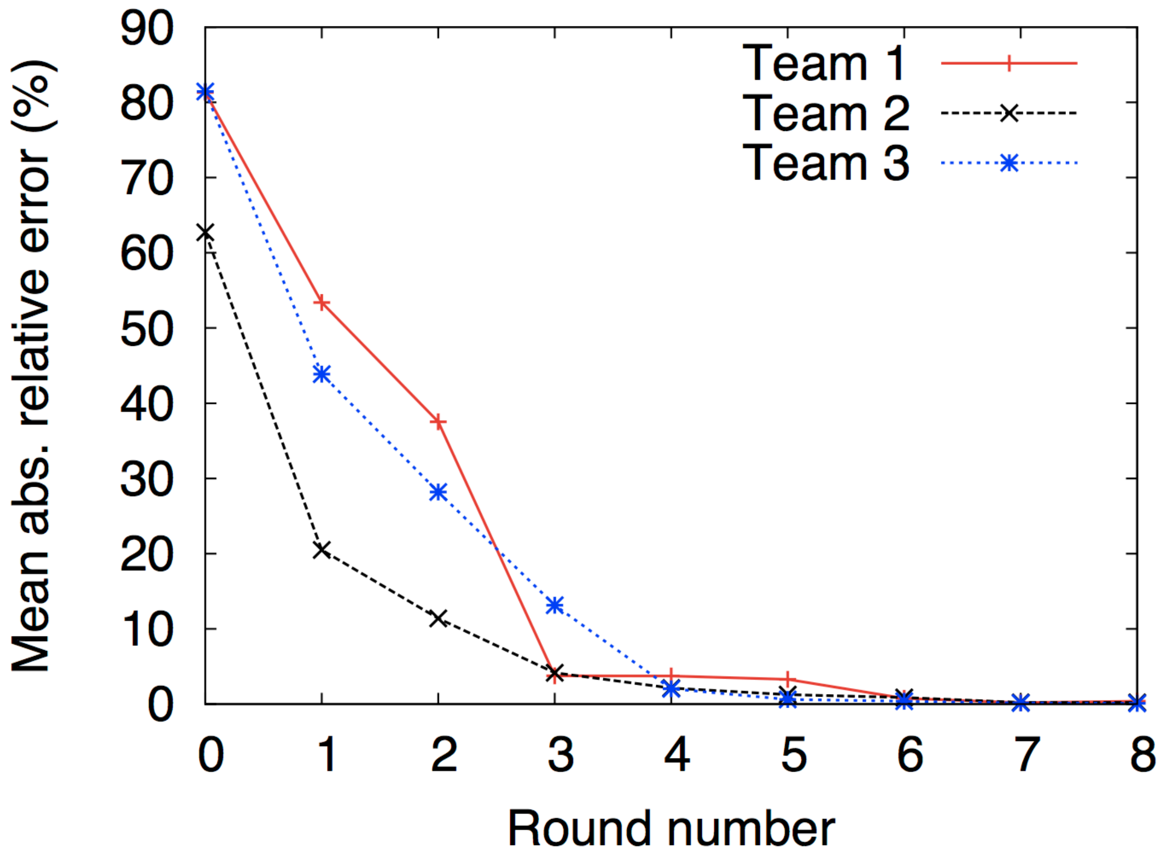
Signature Convergence (1/2)

VanillaXGossip



Signature Convergence (2/2)

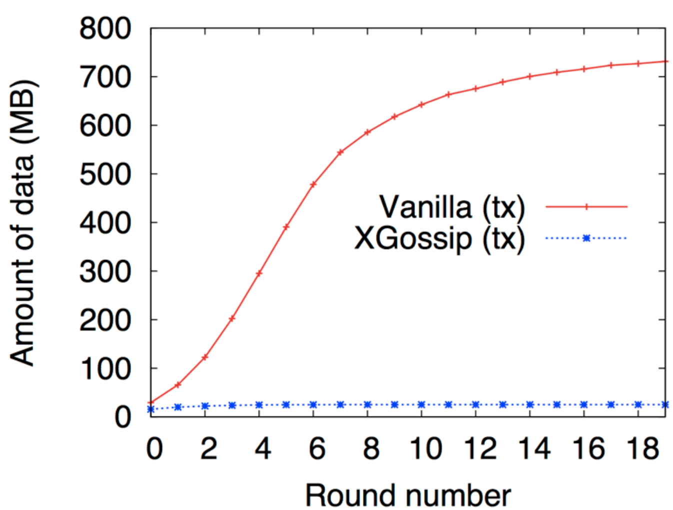
XGossip



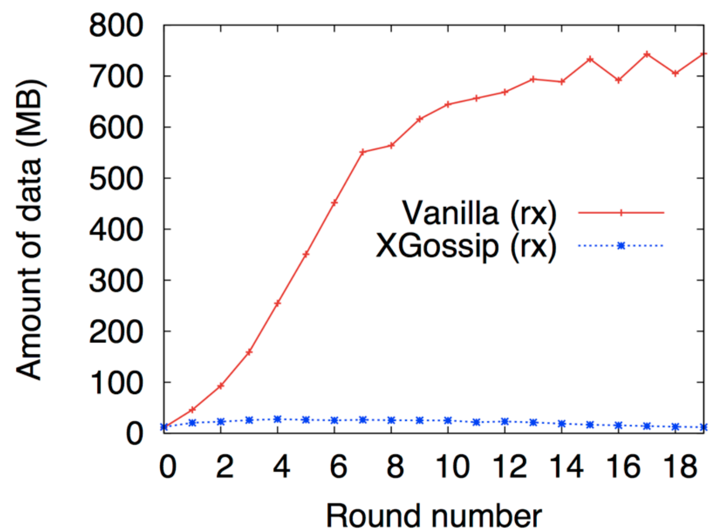
Bandwidth Consumption (1/2)

VanillaXGossip vs. XGossip

Transmitted data



Received data

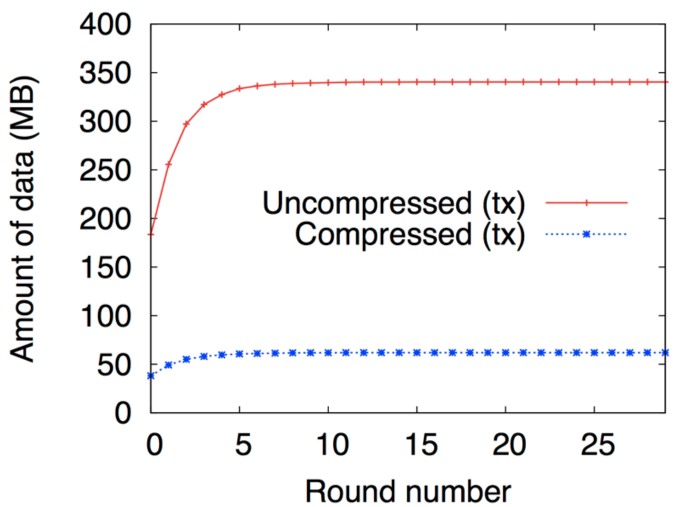


- VanillaXGossip: 10,309 MB
- XGossip: 484 MB

Bandwidth Consumption (2/2)

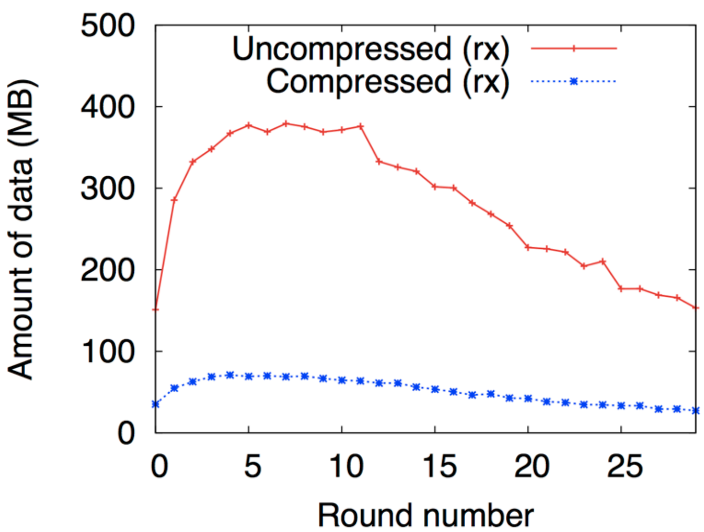
XGossip compression

Transmitted data



- Uncompressed: 9874 MB
- Compressed: 1805 MB

Received data



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.

- Acknowledgements
  - National Science Foundation (IIS-1115871), 2011-2014
  - University of Missouri Research Board, 2010-2012
  - IBM Smarter Planet Faculty Innovation Award, 2010