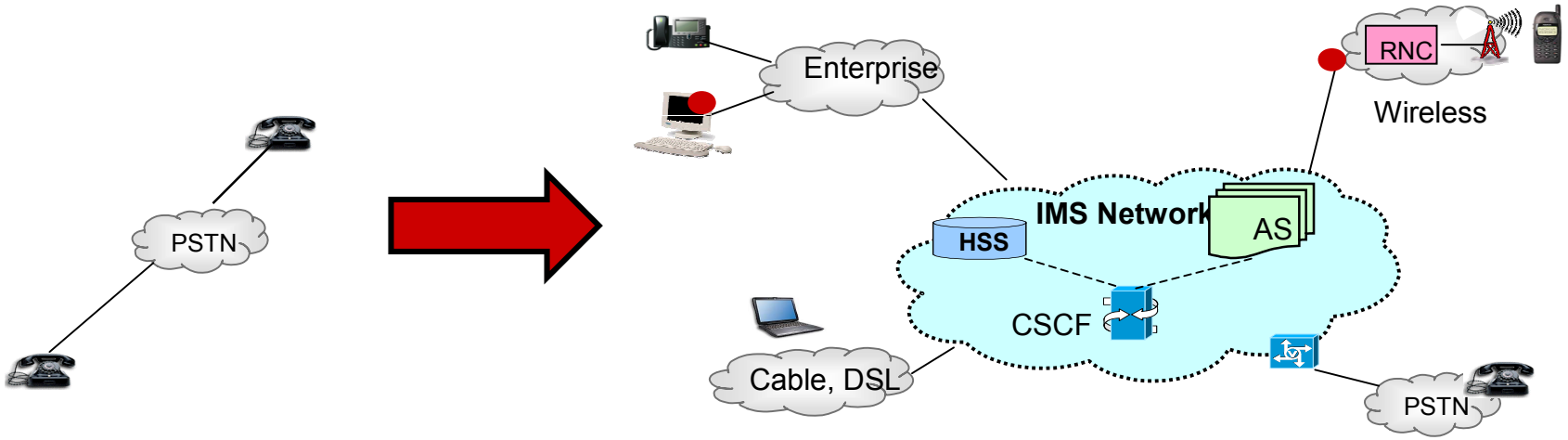
Tracking Inverse Distributions of Massive Data Streams

Graham Cormode
cormode@bell-labs.com

Lucent Technologies
Bell Labs Innovations

Network Monitoring

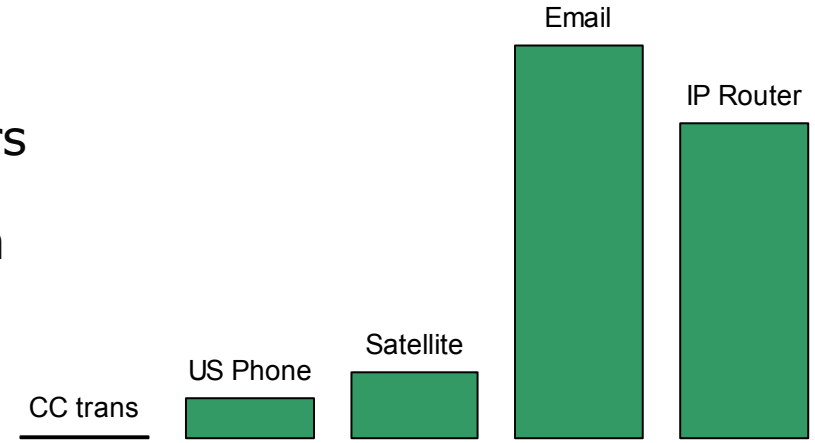


Today's converged networks bring many new challenges for monitoring

- Massive scale of data and connections
- No centralized control, inability to police what is connected
- Attacks, malicious usage, malware, misconfigurations…
- No per-connection records or infrastructure

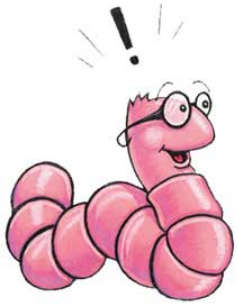
Scale of Data

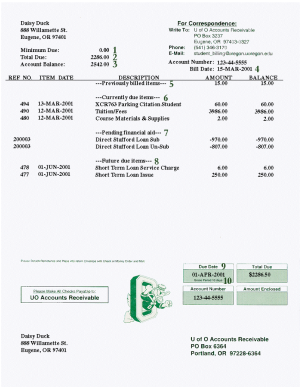
- IP Network Traffic: up to 1 Billion packets per hour per router. Each ISP has many (hundreds) of routers
- Scientific data: NASA's observation satellites each generate billions of readings per day.
- Compare to "human scale" data: "only" 1 billion worldwide credit card transactions per month.
- "Only" 3 Billion Telephone Calls in US each day
- "Only" 30 Billion emails daily, 1 Billion SMS, IMs.



Doing anything at all with such massive data is a challenge

Analysis Challenges





- Real-time security, attack detection and defense (DoS, worms)
- Service Quality Management
- Abuse tracking (bandwidth hogs, malicious calling, zombies)
- Usage tracking/billing, SLA enforcement

Focus

- In this talk, focus on inherent algorithmic challenges in analyzing high speed data in real time or near real time.
- Must solve fundamental problems with many applications.
- We cannot store all the data, in fact can only retain a tiny fraction, and must process quickly (at line speed)
- Exact answers to many questions are impossible without storing everything.
- We must use approximation and randomization with strong guarantees
- Techniques used are algorithm design, careful use of randomization and sampling.

Computation Model

Formally, we observe a *stream* of data, each update arrives once, and we have to compute some function of interest.

Analyze the resources needed, in terms of time per update, space, time for computing the function, communication and other resources.

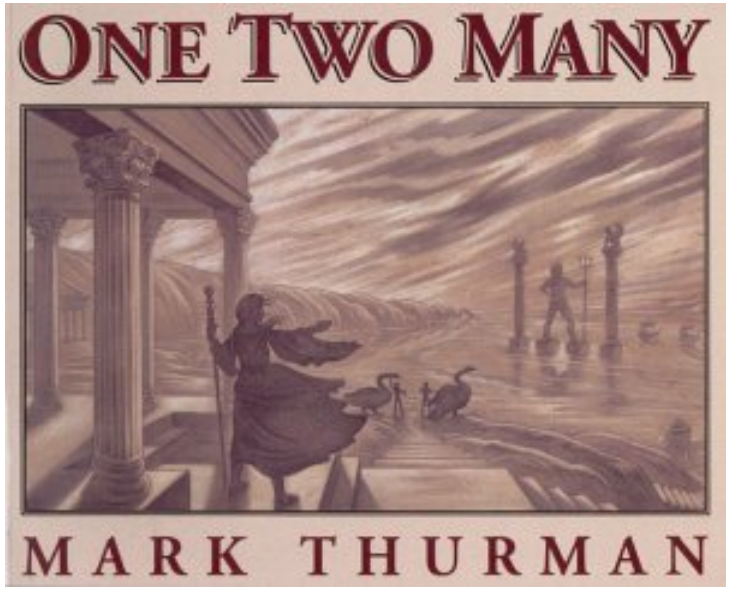
Ideally, all of these should be sublinear in size of input, n

Three settings, depending on number of monitoring places:

- One: a single, centralized monitoring location
- Two: a pair of monitoring locations and we want to compute the difference between their streams
- Many: a large number of monitoring points and we want to compute on the union of all the streams

Outline

- inverse
  defining the inverse distribution
- one
  monitoring occurs at a single centralized location
- two
  monitoring the difference between two locations (eg both ends of a link)
- many
  continuously monitoring multiple locations



The title is a play on words because when Jan's reflection comes to life, Jan discovers that two is one too many.

Motivating Problems

INV – How many people made less than five VoIP calls today?

FWD – Which are the most frequently called numbers?

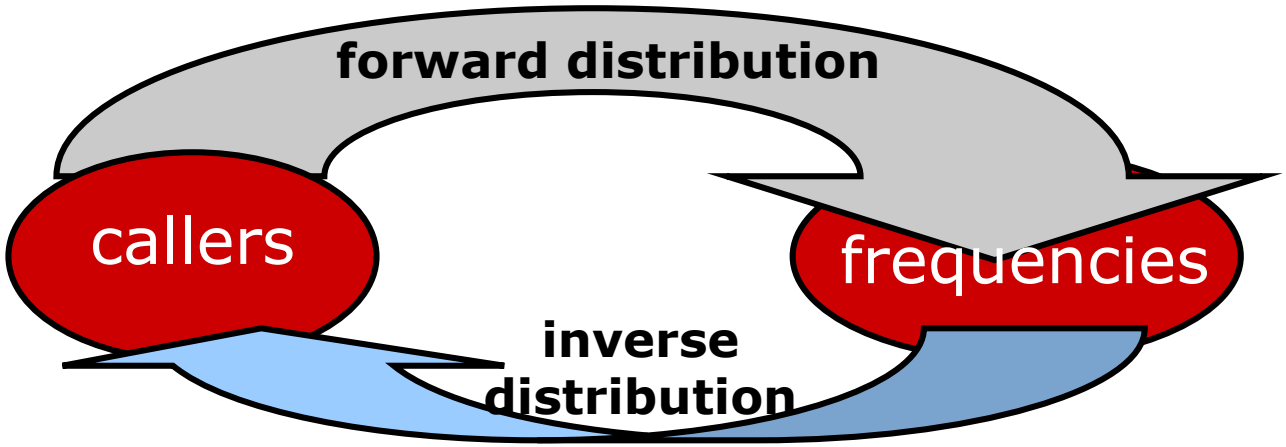
INV – What is most frequent number of calls made?

FWD – What is median call length?

INV – What is median number of calls?

FWD – How many calls did subscriber S make?

Can classify these questions into two types: questions on the *forward distribution* and on the *inverse distribution*.



The Forward Distribution

We abstract the traffic distribution. See one item at a time (eg new call from x to y)

Forward distribution f[0…U],
f(x) = number of calls / bytes / packets etc. from x

How many calls did S make? Find f(S)

Most frequently caller? Find x s.t. f(x) is greatest

Can study frequent items / heavy hitters, quantiles / medians, Frequency moments, distinct items, draw samples, correlations, clustering, etc…

Lot of work over the past 10 years on the forward distribution

The Inverse Distribution

Inverse distribution is $f^{-1}[0\ldots N]$,
$f^{-1}(i)$ = fraction of users making i calls.
$= |\{ x : f(x) = i, i \neq 0\}| / |\{x : f(x) \neq 0\}|$

$F^{-1}(i)$ = cumulative distribution of f^{-1}
$= \sum_{j > i} f^{-1}(j)$ [sum of $f^{-1}(j)$ *above* i]

Number of people making < 5 calls $= 1 - F^{-1}(5)$

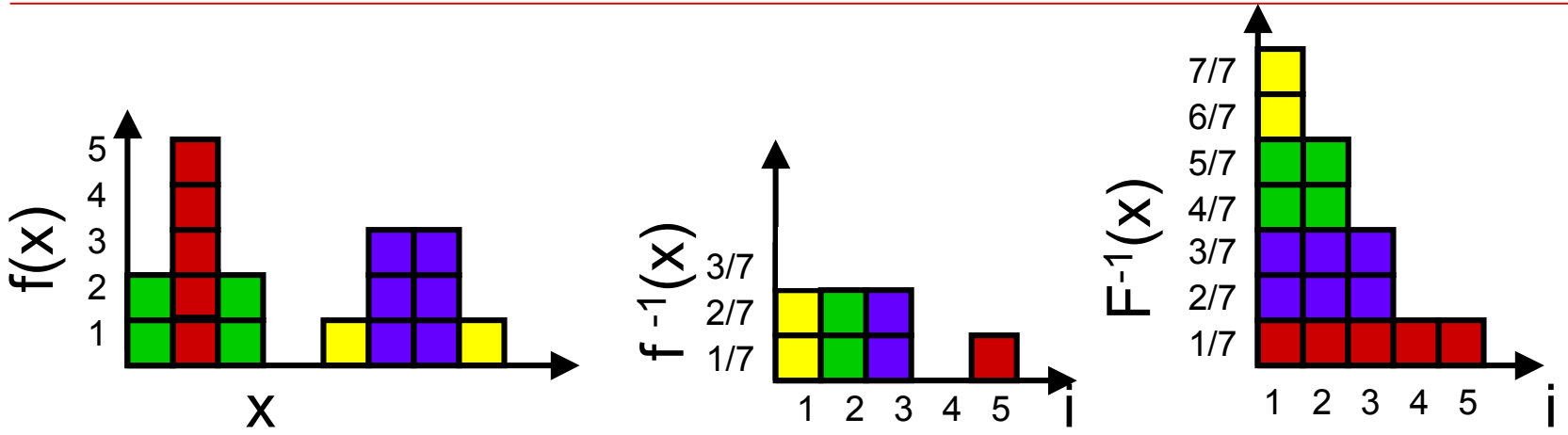
Most frequent number of calls made $= i$ s.t. $f^{-1}(i)$ is greatest

If we have full space, it is easy to go between forward and inverse distribution.

But in small space it is much more difficult, and existing methods in small space don't apply.

Essentially no prior work has looked closely at the inverse distribution in small space, high speed settings.

Example



Separation between inverse distribution:

Consider tracking a simple point query on each distribution. Eg. Find f(9085827700): just count every time a call involves this party

But finding $f^{-1}(2)$ is provably hard: can't track exactly how many people made 2 calls without keeping full space

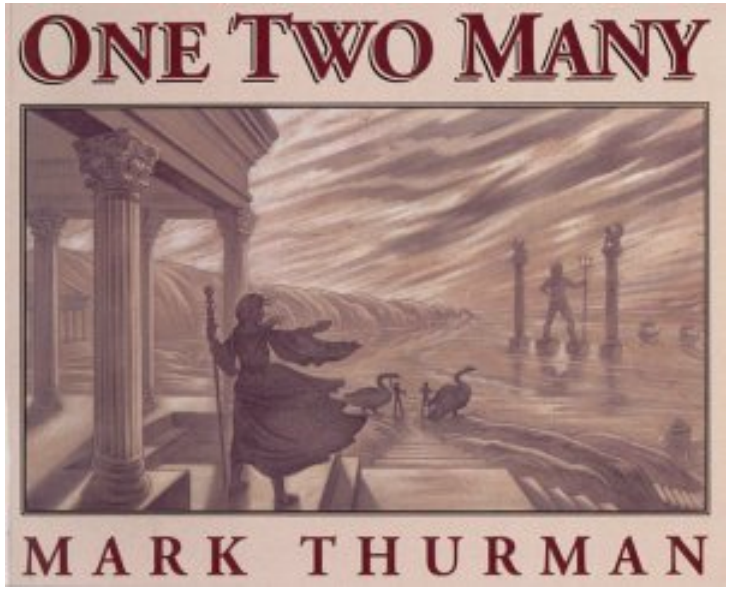
Even approximating up to some constant factor is hard.

Outline

- inverse

  summary: we can map many network monitoring questions onto the inverse distribution. Need new techniques to study it

- one
- two
- many



The title is a play on words because when Jan's reflection comes to life, Jan discovers that two is one too many.

The One and Only

Many queries on the forward distribution can be answered effectively by drawing a sample.

That is, draw an x so probability of picking x is $f(x) / \sum_y f(y)$

Similarly, we want to draw a sample from the inverse distribution in the centralized setting.

That is, draw (i,x) s.t. $f(x)=i$, $i \neq 0$ so probability of picking i is $f^{-1}(i) / \sum_j f^{-1}(j)$ and probability of picking x is uniform.

Drawing from forward distribution is "easy": just uniformly decide to sample each new item (connection, call) seen

Drawing from inverse distribution is more difficult, since probability of drawing $(i,1)$ should be same as $(j,1000)$

Sampling Insight

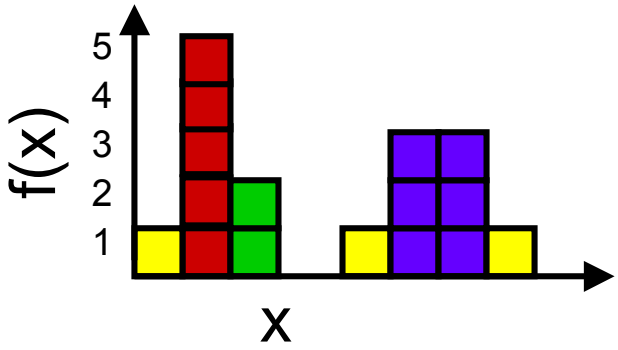
Each distinct item x contributes to one pair (i,x)

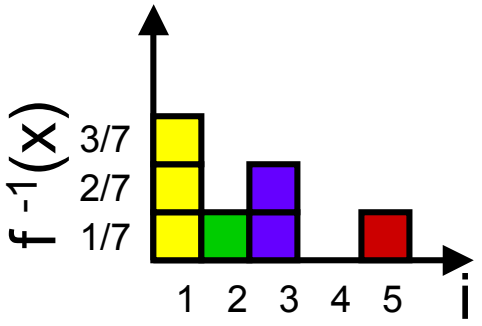
Need to sample uniformly from these pairs.

Basic insight: sample uniformly from the items x and count how many times x is seen to give (i,x) pair that has correct i and is uniform.

How to pick x uniformly before seeing any x?

Use a randomly chosen hash function on each x to decide whether to pick it (and reset count).





Hashing Technique

Use hash function with exponentially decreasing distribution:

Let h be the hash function and r is an appropriate const < 1

$$Pr[h(x) = 0] = (1-r)$$

$$Pr[h(x) = 1] = r\,(1-r)$$

...

$$Pr[h(x) = l] = r^{l}(1-r)$$

Track the following information as updates are seen:

- x: Item with largest hash value seen so far
- $uniq$: Is it the only distinct item seen with that hash value?
- $count$: Count of the item x

Easy to keep $(x, uniq, count)$ up to date as new items arrive

Hashing analysis

Theorem: If uniq is true, then x is picked uniformly.
Probability of uniq being true is at least a constant.

(For right value of r, uniq is almost always true in practice)

Proof outline: Uniformity follows so long as hash function h is at least pairwise independent.

Hard part is showing that uniq is true with constant prob.

- Let D is number of distinct items. Fix l so $1/r \cdot Dr^l \cdot 1/r^2$.
- In expectation, Dr^l items hash to level l or higher
- Variance is also bounded by Dr^l, and we ensure $1/r^2 \cdot 3/2$.
- Analyzing, can show that there is constant probability that there are either 1 or 2 items hashing to level l or higher.

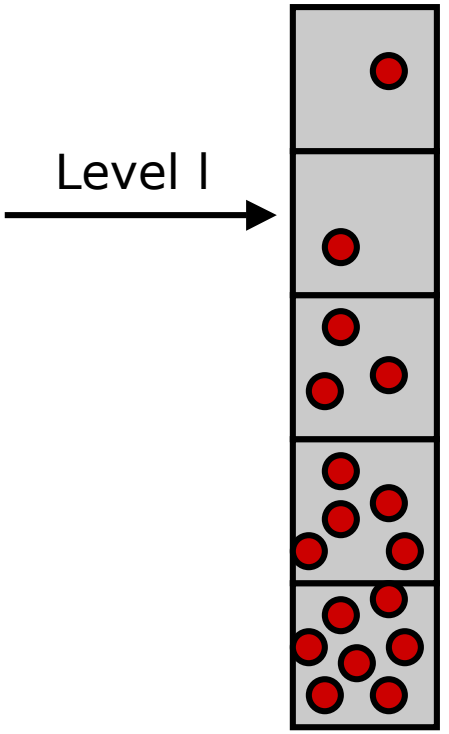
Hashing analysis

If only one item at level l, then uniq is true

If two items at level l or higher, can go deeper into the analysis and show that (assuming there are two items) there is constant probability that they are both at same level.

If not at same level, then uniq is true, and we recover a uniform sample.

- Probability of failure is p = r(3+r)/(2(1+r)).
- Number of levels is O(log N / log 1/r)
- Need 1/r > 1 so this is bounded, and $1/r^2$ ¸ 3/2 for analysis to work
- End up choosing r = p(2/3), so p is < 1



Sample Size

This process either draws a single pair (i,x), or may not return anything.

In order to get a larger sample with high probability, repeat the same process in parallel over the input with different hash functions $h_1 \ldots h_s$ to draw up to s samples (i_j,x_j)

Let $\varepsilon = p(2 \log (1/\delta)/s)$. By Chernoff bounds, if we keep $S = (1+2\varepsilon)\, s/(1 - p)$ copies of the data structure, then we recover at least s samples with probability at least $1-\delta$

Repetitions are a little slow — for better performance, keeping the s items with the s smallest hash values is almost uniform, and faster to maintain.

Using the Sample

A sample from the inverse distribution of size s can be used for a variety of problems with guaranteed accuracy.

Evaluate the question of the sample and return the result.

Eg. Median number of calls made: find median from sample

Median is bigger than ½ and smaller than ½ the values.

Answer has some error: not ½, but $(½ \S \varepsilon)$

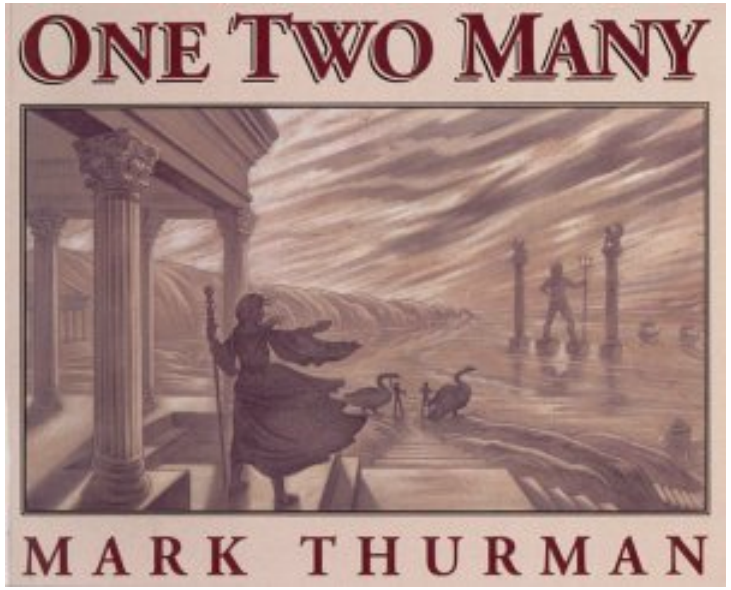
Theorem If sample size $s = O(1/\varepsilon^2 \log 1/\delta)$ then answer from the sample is between $(½-\varepsilon)$ and $(½+\varepsilon)$ with probability at least $1-\delta$.

Proof follows from application of Hoeffding's bound.

Outline

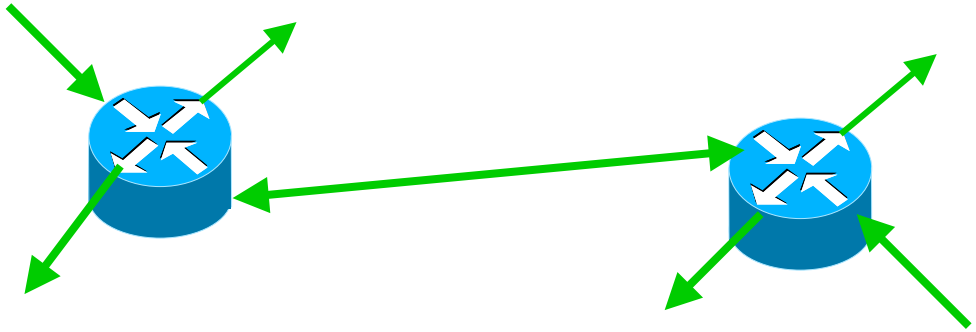
- inverse
- one

  summary: can use hashing approach to draw a uniform sample from inverse distribution. Using the sample we can answer many questions.
- two
- many



The title is a play on words because when Jan's reflection comes to life, Jan discovers that two is one too many.

The Power of Two



We often want to compare two massive streams and look at their difference.

Examples: what's the difference between yesterday and today; what's the difference between Router A and Router B etc.

Formally, we want to ask the same questions as before but on the the *difference distribution:*

$$(f\text{-}g)(x) = f(x) - g(x)$$

How to handle the inverse of the difference distribution, $(f\text{-}g)^{-1}$?

Extended Hashing Approach

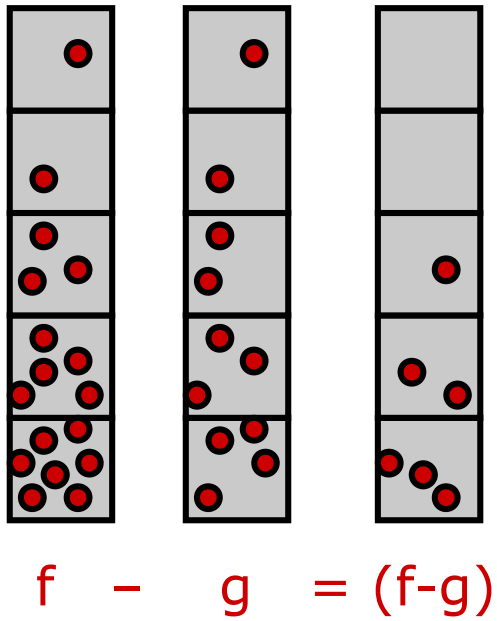
Take the hashing approach, and combine two summaries to get a summary of the difference.

Direct combination is not easy: what if the item at highest level occurs same number of times in both summary? Then it will cancel out. More generally, is result uniform?

Sample (i,x) uniformly from (f-g) so x is chosen uniformly from x where $(f-g)(x) \neq 0$.

Idea: track info about all levels. Ensure when combining two synopses result is uniform over $(f-g)^{-1}$

Ensure that combining info about f and g has duplicate items exactly canceling out.



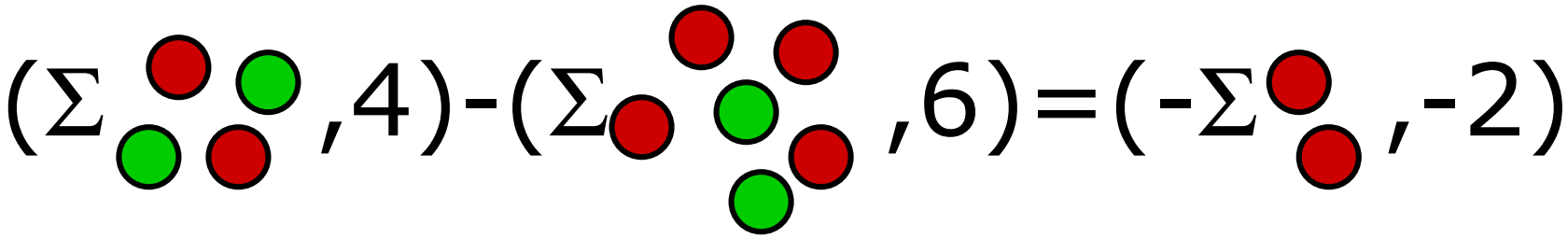
Details

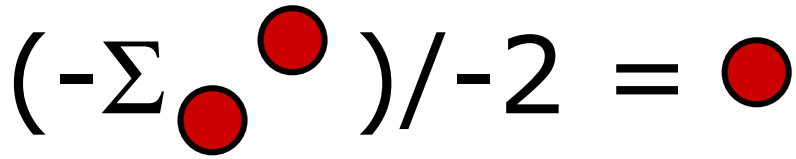
For each level, keep sum of item identifiers that hash there (sumx), and sum of their counts (count).

To combine f and g, compute $sumx_f - sumx_g$ and $count_f - count_g$ for every level.

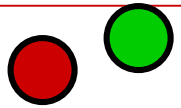
If they are same, they will cancel out (result is zero)

If one item is left over, we have its exact count, and can recover its identity: $(sumx_f - sumx_g)/(count_f - count_g)$





Verification

But we can get fooled: $(\Sigma$  $)/2 =$  ?

How do we know that there is one item?
(equivalent to the uniq flag from the centralized case)

Solution: Use additional counters based on bit wise representation of each item: keep c(b) = number of times item with bit b=1 has been seen.

If $c(b)_f - c(b)_g = \{0,(count_f - count_g)\}$ for all b, item is unique.

If item is not unique, then this test will fail for some b value.

2	0	0	0
0	2	2	2

uniq=true

2	1	0	1
0	1	2	1

uniq=false

Variation: updating all these c(b) counts could be slow (32 bit IP address pairs?) so use speed-ups based on hashing.

Result

Can draw a uniform sample from $(f\text{-}g)^{-1}$ by keeping concise synopses of f and g, and combining them by subtraction.

For each level, recover $(x, count, uniq)$ as before:

$x = (sumx_f - sumx_g)/(count_f - count_g)$

$count = (count_f - count_g)$

$uniq = \Pi_b\, c(b)_f - c(b)_g\ 2\ \{0,(count_f - count_g)\}$

Correctness follows from the centralized case, by linearity: it's as if we are seeing pairs (i,x) $(i \neq 0)$ arriving and choosing whether to sample them based on $h(x)$.

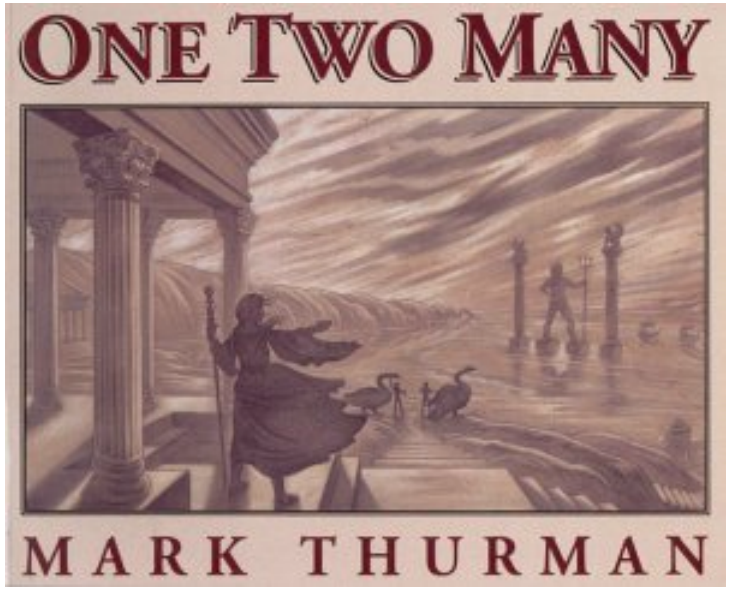
Probability of uniq being true is same as before.

Hence we draw (i,x) uniformly from $(f\text{-}g)^{-1}$ so $(f\text{-}g)(x) = i$

Outline

- inverse
- one
- two

  summary: computing difference can be done with care. Using linear composition of synopses allows differences to precisely cancel out.
- many



The title is a play on words because when Jan's reflection comes to life, Jan discovers that two is one too many.

Many Rivers to Cross

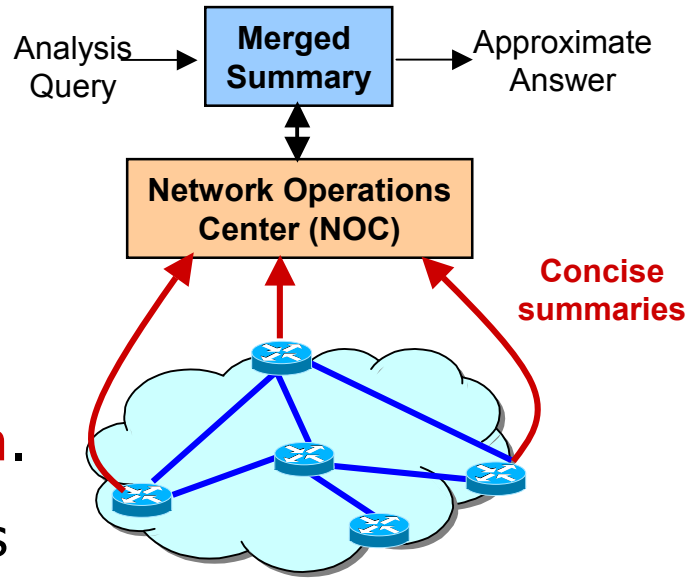
Models many situations: large network monitoring, sensor networks etc.

Want to track the union of their distributions:
$(S_1 [S_2 [\ldots [S_n) (x) = \sum_{j=1}^{n} S_j(x)$

And the global inverse distribution:
$(S_1 [S_2 [\ldots [S_n)^{-1}$

Most important resource in this distributed model is communication.

Want to guarantee accurate solutions while minimizing communication cost.



New Challenges

Monitoring is Continuous…

- Need real time tracking, not one-shot query/response

…Distributed…

- Many remote sites, connected over a network but with communication constraints

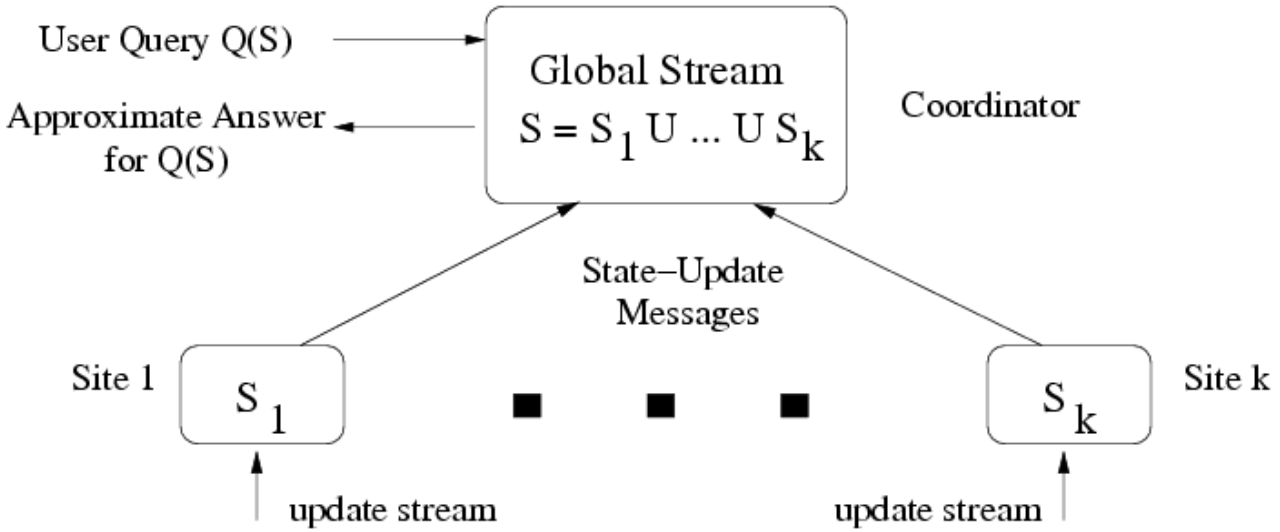
…Streaming…

- Each site sees a high speed stream of data, and may be resource (CPU/Memory) constrained.

…Holistic…

- Queries over whole distribution

Distributed Model



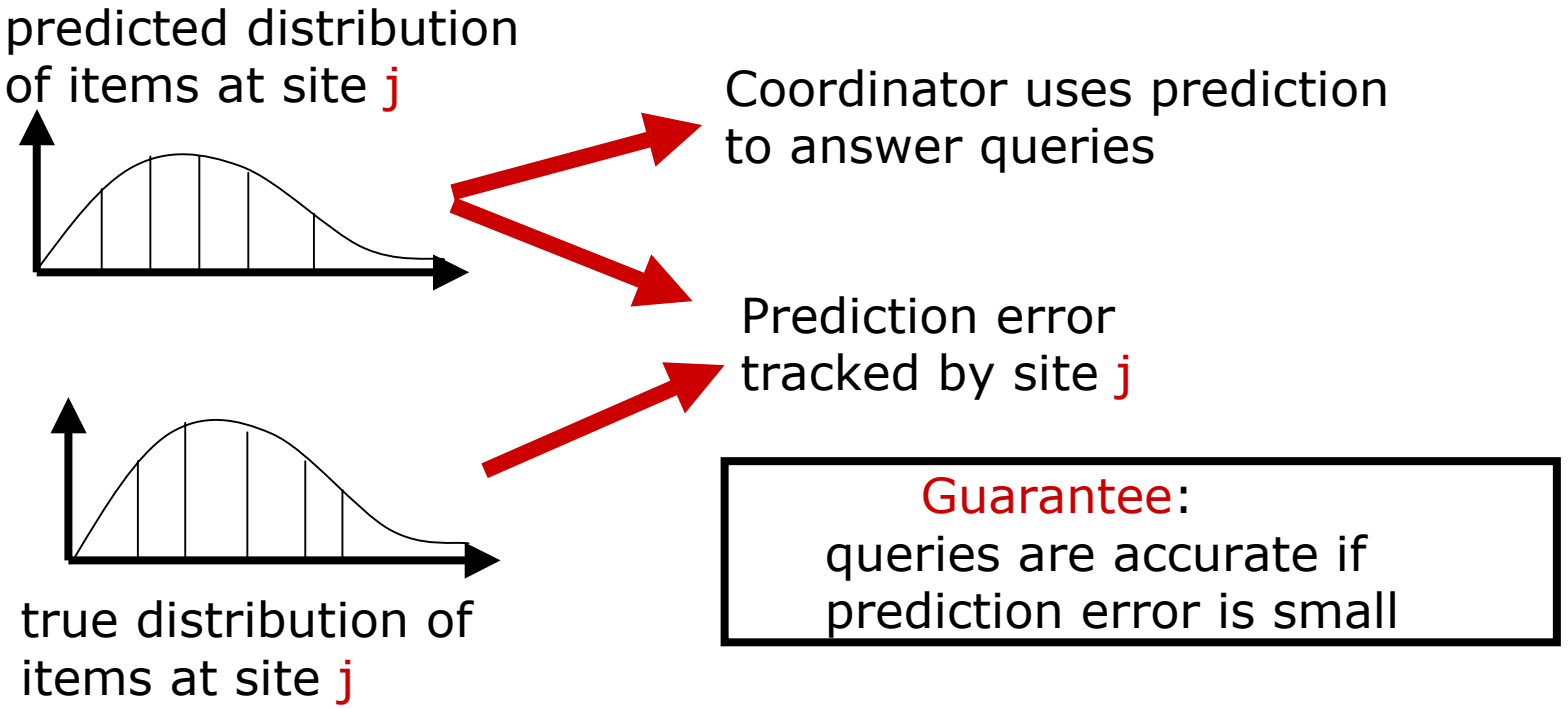
Streams at each site add to distributions S_j

(More generally, can have hierarchical structure)

Use summaries to communicate…

Much smaller cost than sending exact values

Prediction



Remote sites monitor local stream, compare certain local information to *predicted values*

Stability through prediction

If behavior is as predicted, no communication

Inverse Distribution Tracking

Try to run the same algorithm at the central site. Remote sites send up new information when needed.

Allow some amount of "lag" when sending: instead of ensuring that count is accurate, can tolerate error up to $(1+\theta)$count for some fixed θ.

Three basic approaches:

Local Count Only (LCO): sites send when

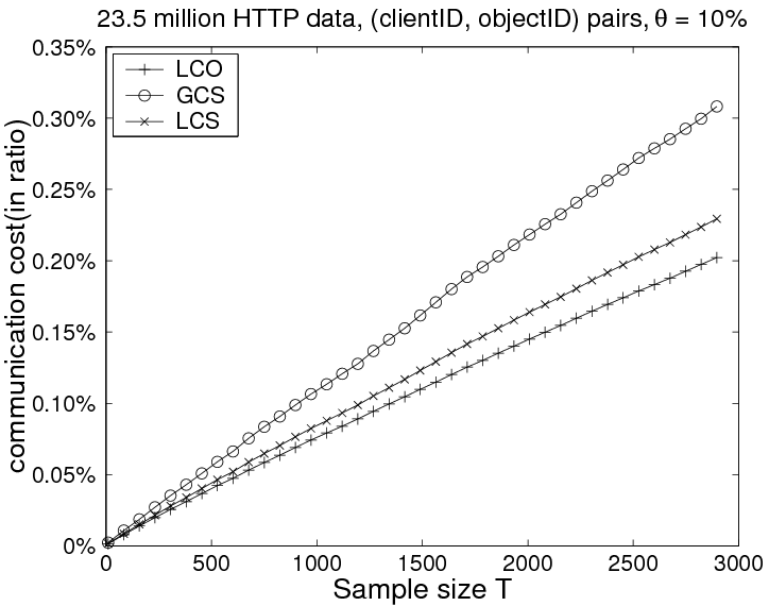
$$\text{count}_j > (1+\theta)\ \text{oldcount}_j$$

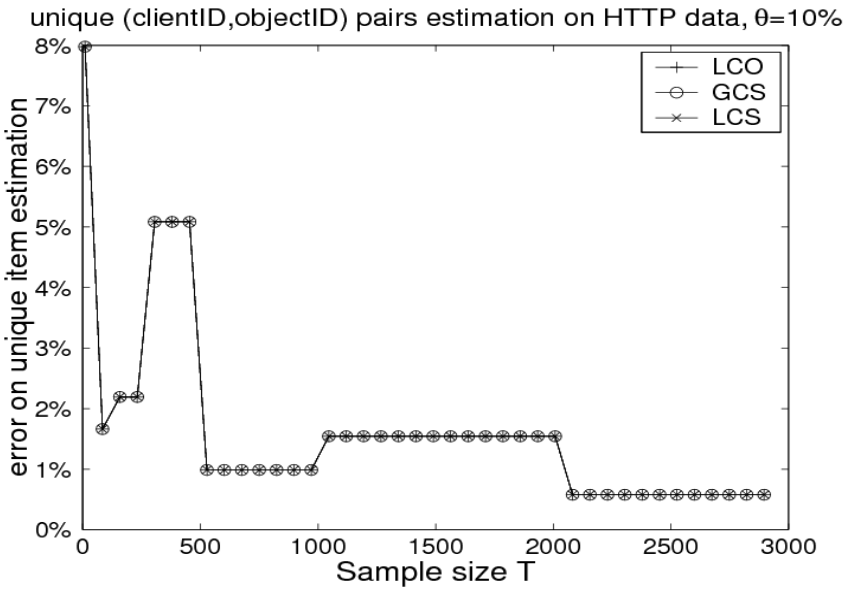
Global Count Sharing (GCS): sites share count, send when

$$\text{count}_j > \text{oldcount}_j + \theta\text{count}/n$$

Local Count Sharing (LCS): instead of broadcasting count, sites receive new value of count when the update.

Experimental Study



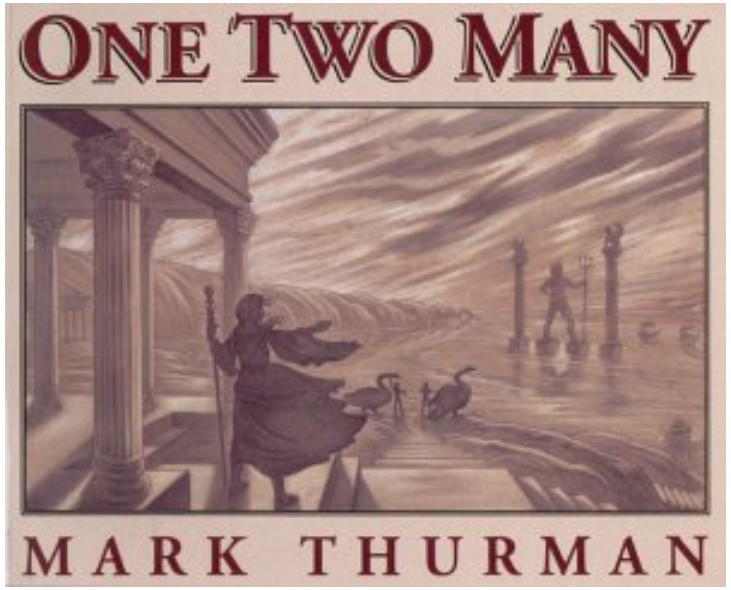


- BIG savings over sending every update, ¿ 1% cost
- Local is better than global information: LCS and LCO consistently beat GCS on different data sets.
- Accuracy improves with sample size, about 1% error on querying $f^{-1}(1)$

Outline

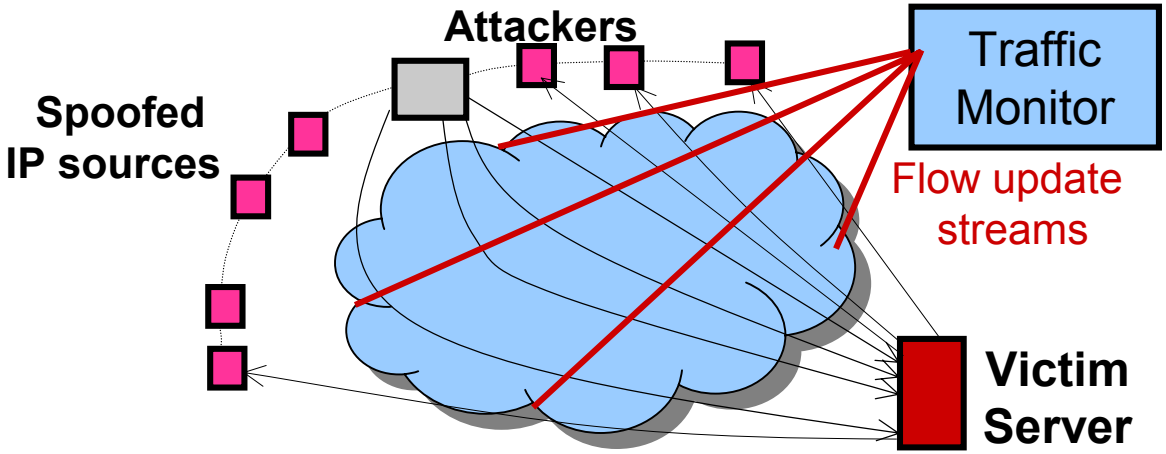
- inverse
- one
- two
- many

  summary: distributed setting gives new challenges to minimize communication overhead. Avoiding global information helps.



The title is a play on words because when Jan's reflection comes to life, Jan discovers that two is one too many.

Going forward… applications



Building "Bloodhound System": distributed high speed monitoring for network security applications.

Apply these and other high speed monitoring techniques deep inside network to track anomalies and threats.

Goal is to be able to monitor approximately many parameters when exact approaches break down.

Going forward… research

Many more problems on high speed network data remain unanswered.

Many problems on the inverse distribution still open. Eg. Sample based approach typically gives additive error ε with a sample of size $1/\varepsilon^2$. Many problems on forward distribution can be answered using space $1/\varepsilon$ or better. Can the bounds be improved here?

Problems that are well understood in the "one" case are less well understood in the "two" and "many" cases.

A solid theoretical basis (a new continuous communication complexity) needed for lower bounds in the "many" model we use here.

References

TWO Summarizing and mining inverse distributions on data streams via dynamic inverse sampling. In *VLDB*, 2005

MANY What's Different: Distributed Continuous Monitoring of Duplicate-Resilient Aggregates on Data Streams, under submission, 2005.

TWO What's new: Finding significant differences in network data streams. *Transactions on Networking*, Feb 2006.

MANY Sketching streams through the net: Distributed approximate query tracking. In *VLDB*, 2005.

ONE Space efficient mining of multigraph streams. In *ACM Principles of Database Systems*, 2005.

MANY Holistic aggregates in a networked world: Distributed tracking of approximate quantiles. In *ACM SIGMOD*, 2005

Joint work with Minos Garofalakis, Rajeev Rastogi (Bell Labs)
S. Muthukrishnan, Wei Zhuang, Irina Rozenbaum (Rutgers)