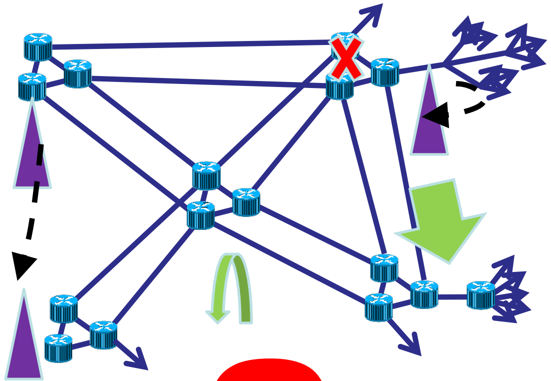
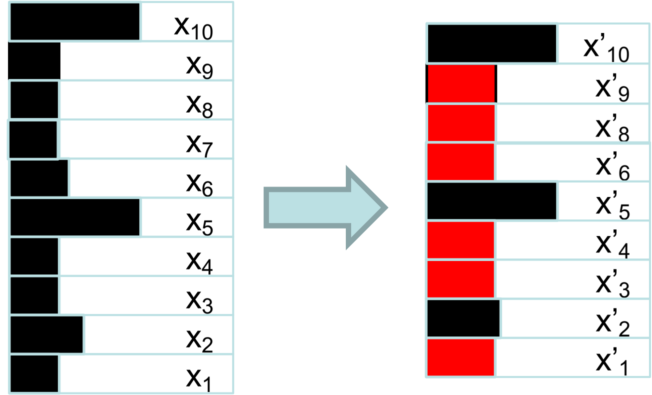
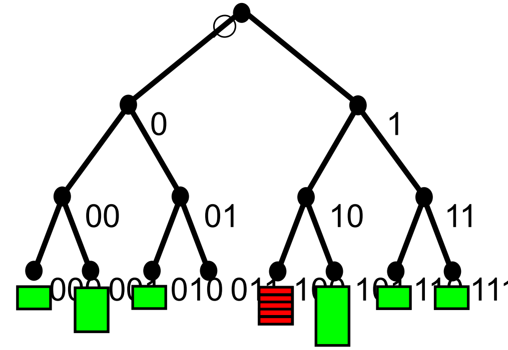
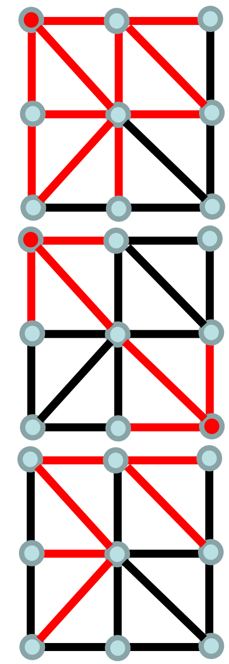
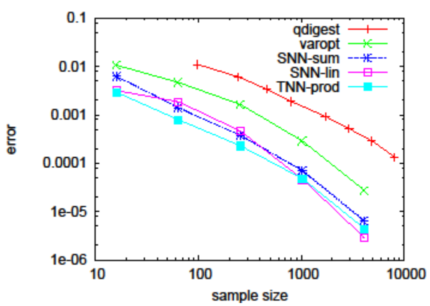






Sampling for Big Data



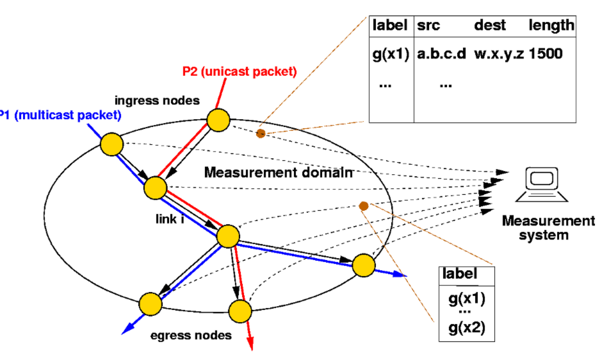


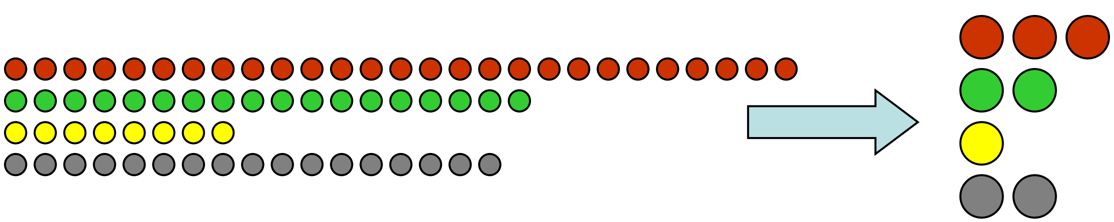
Graham Cormode, University of Warwick

G.Cormode@warwick.ac.uk

Nick Duffield, Texas A&M University

Nick.Duffield@gmail.com



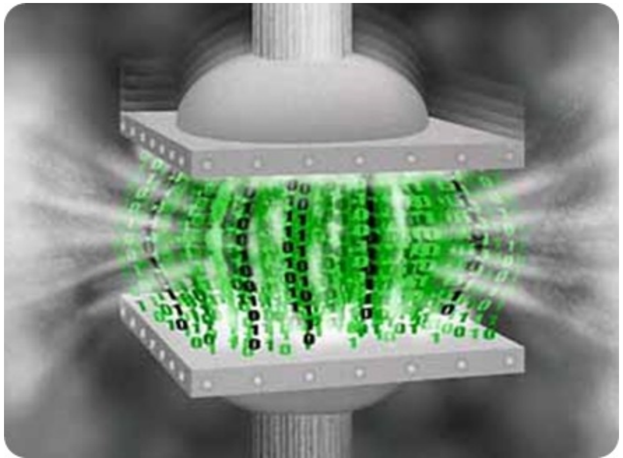


Big Data

- "Big" data arises in many forms:
  - Physical Measurements: from science (physics, astronomy)
  - Medical data: genetic sequences, detailed time series
  - Activity data: GPS location, social network activity
  - Business data: customer behavior tracking at fine detail
- Common themes:
  - Data is large, and growing
  - There are important patterns and trends in the data
  - We don't fully know where to look or how to find them

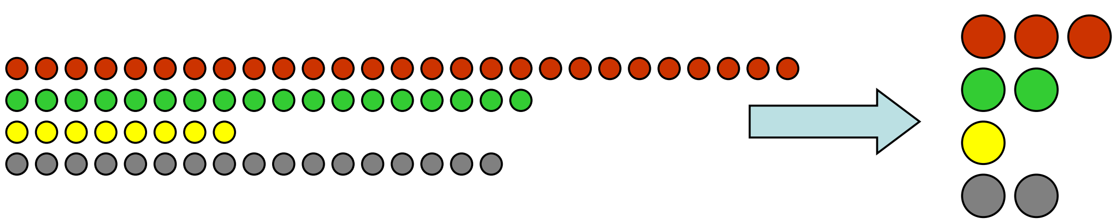
Why Reduce?

- Although "big" data is about more than just the volume…
  …most big data is big!
- It is not always possible to store the data in full
  - Many applications (telecoms, ISPs, search engines) can't keep everything
- It is inconvenient to work with data in full
  - Just because we can, doesn't mean we should
- It is faster to work with a compact summary
  - Better to explore data on a laptop than a cluster



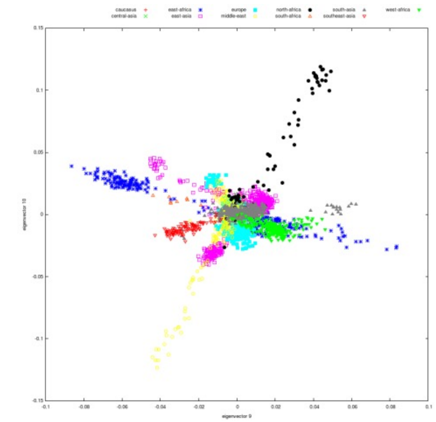
Why Sample?

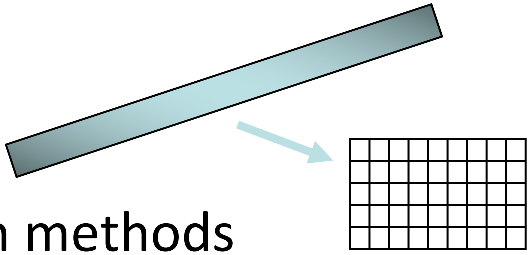
- ◊ Sampling has an intuitive semantics
  - – We obtain a smaller data set with the same structure
- ◊ Estimating on a sample is often straightforward
  - – Run the analysis on the sample that you would on the full data
  - – Some rescaling/reweighting may be necessary
- ◊ Sampling is general and agnostic to the analysis to be done
  - – Other summary methods only work for certain computations
  - – Though sampling can be tuned to optimize some criteria
- ◊ Sampling is (usually) easy to understand
  - – So prevalent that we have an intuition about sampling



Alternatives to Sampling

- Sampling is not the only game in town
  - Many other data reduction techniques by many names
- Dimensionality reduction methods
  - PCA, SVD, eigenvalue/eigenvector decompositions
  - Costly and slow to perform on big data
- "Sketching" techniques for streams of data
  - Hash based summaries via random projections
  - Complex to understand and limited in function
- Other transform/dictionary based summarization methods
  - Wavelets, Fourier Transform, DCT, Histograms
  - Not incrementally updatable, high overhead
- All worthy of study – in other tutorials





Health Warning: contains probabilities

- Will avoid detailed probability calculations, aim to give high level descriptions and intuition
- But some probability basics are assumed
  - Concepts of probability, expectation, variance of random variables
  - Allude to concentration of measure (Exponential/Chernoff bounds)
- Feel free to ask questions about technical details along the way

$$\begin{aligned}
\operatorname{var}\left(\frac{k}{n}\right) &= \mathrm{E}\left[\operatorname{var}\left(\frac{k}{n}\middle|\theta\right)\right] + \operatorname{var}\left[\mathrm{E}\left(\frac{k}{n}\middle|\theta\right)\right] \\
&= \mathrm{E}\left[\left(\frac{1}{n}\right)\theta(1-\theta)\middle|\mu, M\right] + \operatorname{var}\left(\theta|\mu, M\right) \\
&= \frac{1}{n}\left(\mu(1-\mu)\right) + \frac{n-1}{n}\frac{\left(\mu(1-\mu)\right)}{M+1} \\
&= \frac{\mu(1-\mu)}{n}\left(1+\frac{n-1}{M+1}\right).
\end{aligned}$$

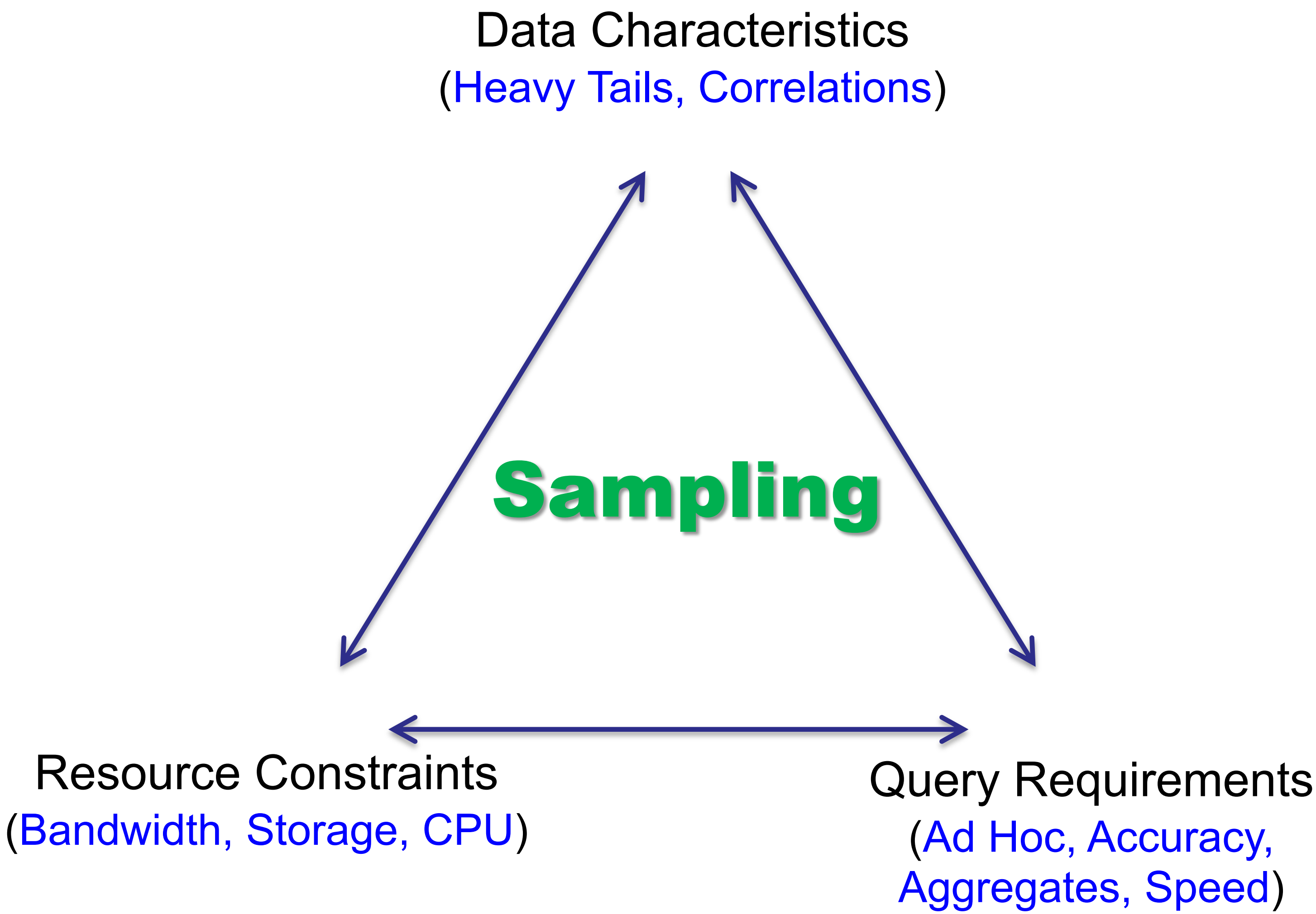
Outline

- Motivating application: sampling in large ISP networks
- Basics of sampling: concepts and estimation
- Stream sampling: uniform and weighted case
  - Variations: Concise sampling, sample and hold, sketch guided

BREAK

- Advanced stream sampling: sampling as cost optimization
  - VarOpt, priority, structure aware, and stable sampling
- Hashing and coordination
  - Bottom-k, consistent sampling and sketch-based sampling
- Graph sampling
  - Node, edge and subgraph sampling
- Conclusion and future directions

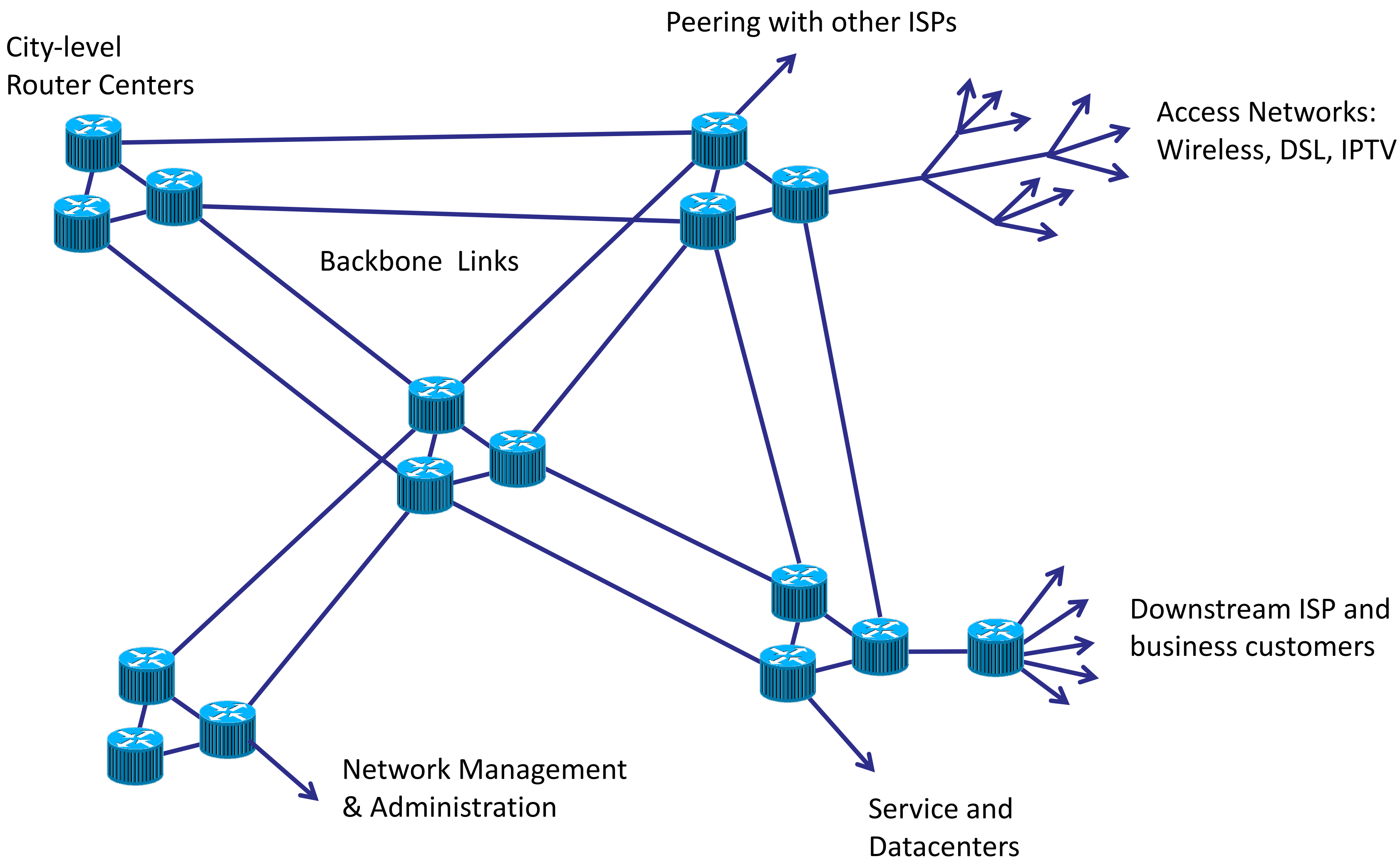
Sampling as a Mediator of Constraints



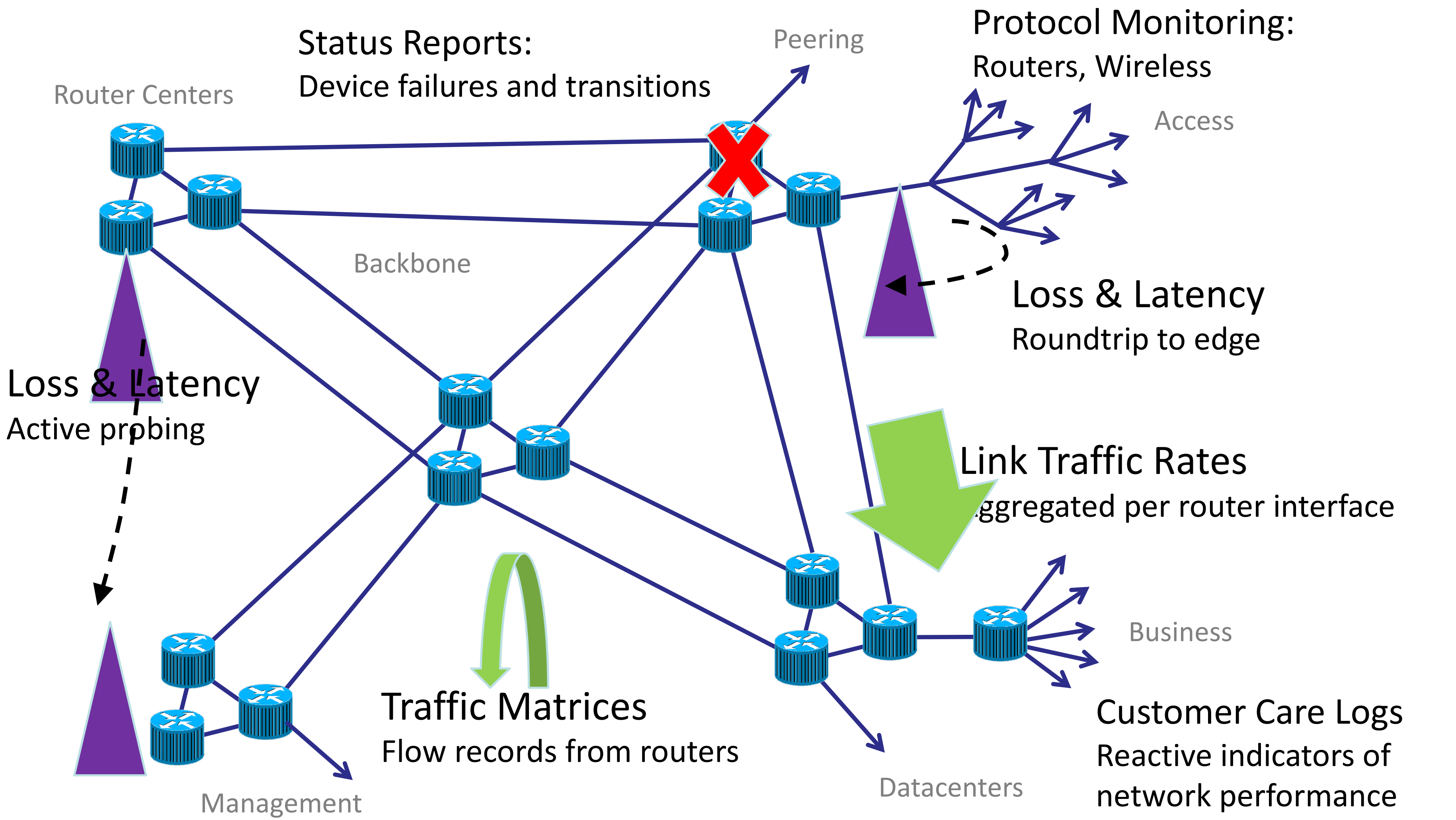
Motivating Application: ISP Data

- Will motivate many results with application to ISPs
- Many reasons to use such examples:
  - Expertise: tutors from telecoms world
  - Demand: many sampling methods developed in response to ISP needs
  - Practice: sampling widely used in ISP monitoring, built into routers
  - Prescience: ISPs were first to hit many “big data” problems
  - Variety: many different places where sampling is needed
- First, a crash-course on ISP networks…

Structure of Large ISP Networks



Measuring the ISP Network: Data Sources



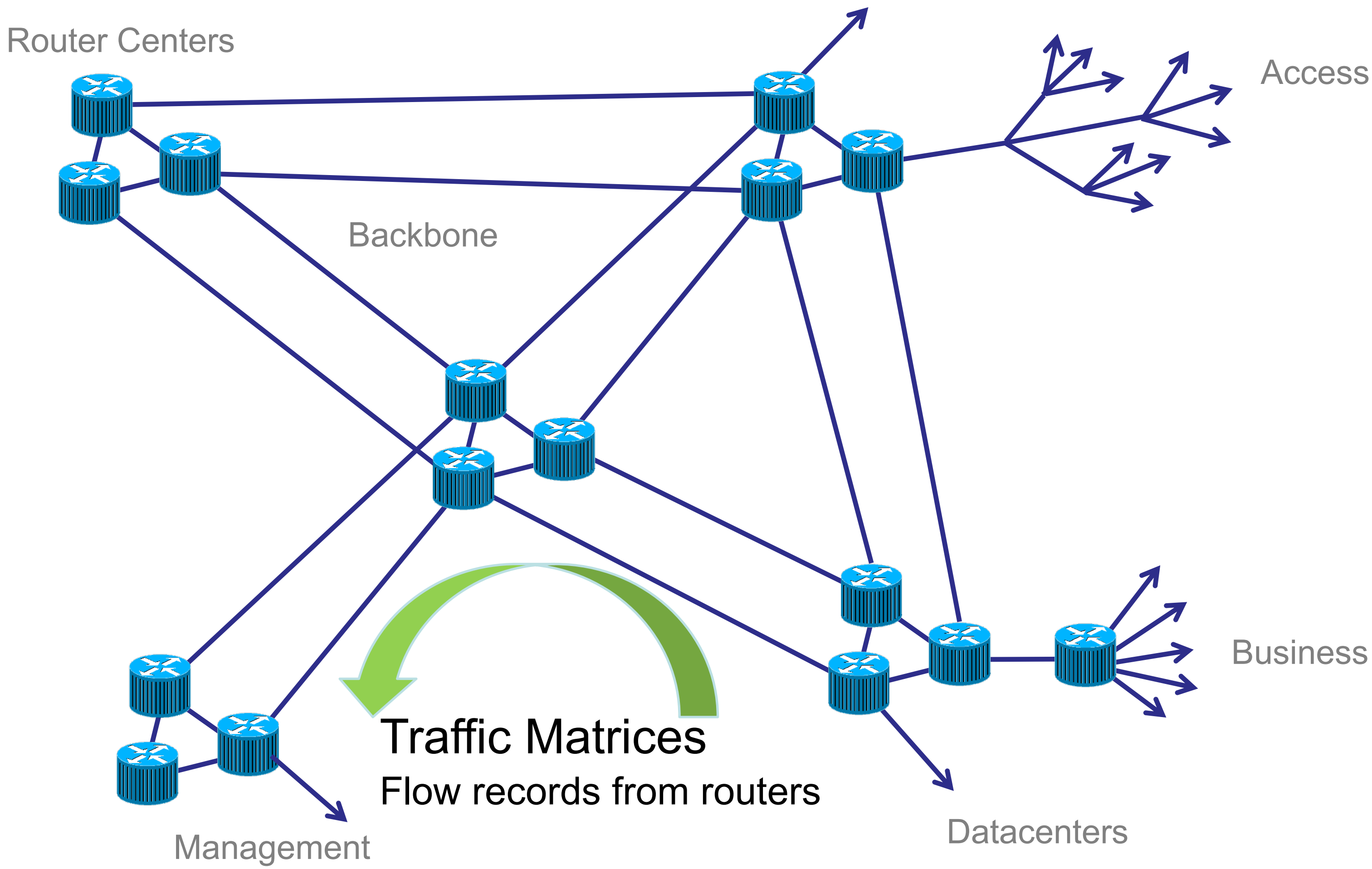
Why Summarize (ISP) Big Data?

- When transmission bandwidth for measurements is limited
  - Not such a big issue in ISPs with in-band collection
- Typically raw accumulation is not feasible (even for nation states)
  - High rate streaming data
  - Maintain historical summaries for baselining, time series analysis
- To facilitate fast queries
  - When infeasible to run exploratory queries over full data
- As part of hierarchical query infrastructure:
  - Maintain full data over limited duration window
  - Drill down into full data through one or more layers of summarization

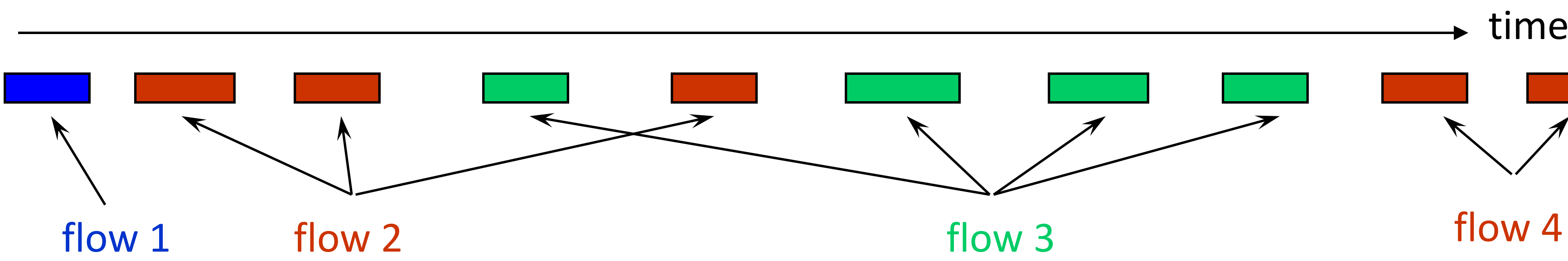
Sampling has been proved to be a flexible method to accomplish this

Data Scale:
Summarization and Sampling

Traffic Measurement in the ISP Network



Massive Dataset: Flow Records



- IP Flow: set of packets with common key observed close in time
- Flow Key: IP src/dst address, TCP/UDP ports, ToS,… [64 to 104+ bits]
- Flow Records:
  - Protocol level summaries of flows, compiled and exported by routers
  - Flow key, packet and byte counts, first/last packet time, some router state
  - Realizations: Cisco Netflow, IETF Standards
- Scale: 100's TeraBytes of flow records daily are generated in a large ISP
- Used to manage network over range of timescales:
  - Capacity planning (months),…., detecting network attacks (seconds)
- Analysis tasks
  - Easy: timeseries of predetermined aggregates (e.g. address prefixes)
  - Hard: fast queries over exploratory selectors, history, communications subgraphs

Flows, Flow Records and Sampling

- ◊ Two types of sampling used in practice for internet traffic:
  1. Sampling packet stream in router prior to forming flow records
     - □ Limits the rate of lookups of packet key in flow cache
     - □ Realized as Packet Sampled NetFlow (more later…)
  2. Downstream sampling of flow records in collection infrastructure
     - □ Limits transmission bandwidth, storage requirements
     - □ Realized in ISP measurement collection infrastructure (more later…)
- ◊ Two cases illustrative of general property
  - – Different underlying distributions require different sample designs
  - – Statistical optimality sometimes limited by implementation constraints
    - □ Availability of router storage, processing cycles

Abstraction: Keyed Data Streams

- ◊ Data Model: objects are keyed weights
  - – Objects (x,k): Weight x; key k
    - □ Example 1: objects = packets, x = bytes, k = key (source/destination)
    - □ Example 2: objects = flows, x = packets or bytes, k = key
    - □ Example 3: objects = account updates, x = credit/debit, k = account ID
- ◊ Stream of keyed weights, $\{(x_i , k_i): i = 1,2,\ldots,n\}$
- ◊ Generic query: subset sums
  - – $X(S) = \Sigma_{i\in S}\, x_i$ for $S \subset \{1,2,\ldots,n\}$ i.e. total weight of index subset S
  - – Typically $S = S(K) = \{i: k_i \in K\}$: objects with keys in K
    - □ Example 1, 2: $X(S(K))$ = total bytes to given IP dest address / UDP port
    - □ Example 3: $X(S(K))$ = total balance change over set of accounts
- ◊ Aim: Compute fixed size summary of stream that can be used to estimate arbitrary subset sums with known error bounds

Inclusion Sampling and Estimation

- Horvitz-Thompson Estimation:
  - Object of size x_i sampled with probability p_i
  - Unbiased estimate $x'_i = x_i / p_i$ (if sampled), 0 if not sampled: $E[x'_i] = x_i$
- Linearity:
  - Estimate of subset sum = sum of matching estimates
  - Subset sum $X(S)= \Sigma_{i \in S} x_i$ is estimated by $X'(S) = \Sigma_{i \in S} x'_i$
- Accuracy:
  - Exponential Bounds: $\Pr[|X'(S) - X(S)| > \delta X(S)] \leq \exp[-g(\delta)X(S)]$
  - Confidence intervals: $X(S) \in [X^-(\varepsilon) , X^+(\varepsilon)]$ with probability $1 - \varepsilon$
- Futureproof:
  - Don't need to know queries at time of sampling
    - "Where/where did that suspicious UDP port first become so active?"
    - "Which is the most active IP address within than anomalous subnet?"
  - Retrospective estimate: subset sum over relevant keyset

Independent Stream Sampling

- Bernoulli Sampling
  - IID sampling of objects with some probability p
  - Sampled weight x has HT estimate x/p
- Poisson Sampling
  - Weight x_i sampled with probability p_i ; HT estimate x_i / p_i
- When to use Poisson vs. Bernoulli sampling?
  - Elephants and mice: Poisson allows probability to depend on weight…
- What is best choice of probabilities for given stream $\{x_i\}$?

Bernoulli Sampling

- The easiest possible case of sampling: all weights are 1
  - N objects, and want to sample k from them uniformly
  - Each possible subset of k should be equally likely
- Uniformly sample an index from N (without replacement) k times
  - Some subtleties: truly random numbers from [1…N] on a computer?
  - Assume that random number generators are good enough
- Common trick in DB: assign a random number to each item and sort
  - Costly if N is very big, but so is random access
- Interesting problem: take a single linear scan of data to draw sample
  - Streaming model of computation: see each element once
  - Application: IP flow sampling, too many (for us) to store
  - (For a while) common tech interview question

Reservoir Sampling

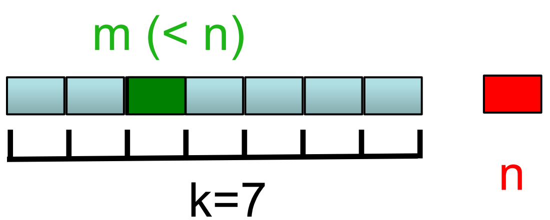
"Reservoir sampling" described by [Knuth 69, 81]; enhancements [Vitter 85]

◊ Fixed size k uniform sample from arbitrary size N stream in one pass

- No need to know stream size in advance
- Include first k items w.p. 1
- Include item $n > k$ with probability $p(n) = k/n$, $n > k$
  - □ Pick j uniformly from $\{1,2,\ldots,n\}$
  - □ If $j \le k$, swap item n into location j in reservoir, discard replaced item

◊ Neat proof shows the uniformity of the sampling method:

- Let S_n = sample set after n arrivals



New item: selection probability

$\text{Prob}[n \in S_n] = p_n := k/n$

Previously sampled item: induction

$m \in S_{n-1}$ w.p. $p_{n-1} \Rightarrow m \in S_n$ w.p. $p_{n-1} * (1 - p_n / k) = p_n$

Reservoir Sampling: Skip Counting

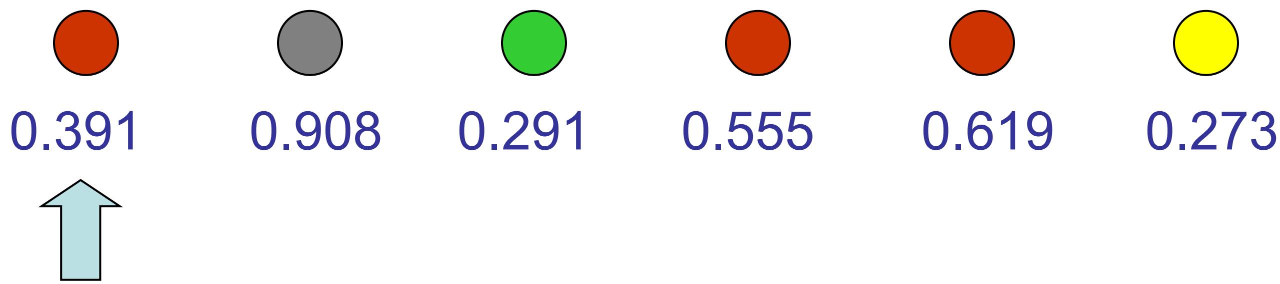
- ◊ Simple approach: check each item in turn
  - – $O(1)$ per item:
  - – Fine if computation time < interarrival time
  - – Otherwise build up computation backlog $O(N)$
- ◊ Better: "skip counting"
  - – Find random index $m(n)$ of next selection $> n$
  - – Distribution: $\text{Prob}[m(n) \le m] = 1 - (1-p_{n+1})*(1-p_{n+2})*\ldots*(1-p_m)$
- ◊ Expected number of selections from stream is

$$k + \Sigma_{k<m\le N}\, p_m = k + \Sigma_{k<m\le N}\, k/m = O(k\,(\,1 + \ln\,(N/k)\,))$$

- ◊ Vitter'85 provided algorithm with this average running time

Reservoir Sampling via Order Sampling

- Order sampling a.k.a. bottom-k sample, min-hashing
- Uniform sampling of stream into reservoir of size k
- Each arrival n: generate one-time random value $r_n \in U[0,1]$
  - r_n also known as hash, rank, tag…
- Store k items with the smallest random tags



- Each item has same chance of least tag, so uniform
- Fast to implement via priority queue
- Can run on multiple input streams separately, then merge

Handling Weights

- So far: uniform sampling from a stream using a reservoir
- Extend to non-uniform sampling from weighted streams
  - Easy case: $k=1$
  - Sampling probability $p(n) = x_n/W_n$ where $W_n = \Sigma_{i=1}^{n} x_i$
- $k>1$ is harder
  - Can have elements with large weight: would be sampled with prob 1?
- Number of different weighted order-sampling schemes proposed to realize desired distributional objectives
  - Rank $r_n = f(u_n, x_n)$ for some function f and $u_n \in U[0,1]$
  - k-mins sketches [Cohen 1997], Bottom-k sketches [Cohen Kaplan 2007]
  - [Rosen 1972], Weighted random sampling [Efraimidis Spirakis 2006]
  - Order PPS Sampling [Ohlsson 1990, Rosen 1997]
  - Priority Sampling [Duffield Lund Thorup 2004], [Alon+DLT 2005]

Weighted random sampling

- Weighted random sampling [Efraimidis Spirakis 06] generalizes min-wise
  - For each item draw r_n uniformly at random in range [0,1]
  - Compute the 'tag' of an item as $r_n^{(1/x_n)}$
  - Keep the items with the k smallest tags
  - Can prove the correctness of the exponential sampling distribution
- Can also make efficient via skip counting ideas

Priority Sampling

- Each item x_i given priority $z_i = x_i / r_i$ with r_n uniform random in $(0,1]$
- Maintain reservoir of $k+1$ items (x_i , z_i) of highest priority
- Estimation
  - Let $z^* = (k+1)^{st}$ highest priority
  - Top-k priority items: weight estimate $x'_I = \max\{ x_i , z^* \}$
  - All other items: weight estimate zero
- Statistics and bounds
  - x'_I unbiased; zero covariance: $\mathrm{Cov}[x'_i , x'_j] = 0$ for $i \neq j$
  - Relative variance for any subset sum $\leq 1/(k-1)$ [Szegedy, 2006]

Priority Sampling in Databases

- ◊ One Time Sample Preparation
  - – Compute priorities of all items, sort in decreasing priority order
    - □ No discard
- ◊ Sample and Estimate
  - – Estimate any subset sum $X(S) = \Sigma_{i \in S} x_i$ by $X'(S) = \Sigma_{i \in S} x'_I$ for some $S' \subset S$
  - – Method: select items in decreasing priority order
- ◊ Two variants: bounded variance or complexity
  1. S' = first k items from S: relative variance bounded $\leq 1/(k-1)$
     - □ $x'_I = \max\{ x_i , z^* \}$ where $z^* = (k+1)^{st}$ highest priority in S
  2. S' = items from S in first k: execution time $O(k)$
     - □ $x'_I = \max\{ x_i , z^* \}$ where $z^* = (k+1)^{st}$ highest priority

[Alon et. al., 2005]

Making Stream Samples Smarter

- Observation: we **see** the whole stream, even if we can't store it
  - Can keep more information about sampled items if repeated
  - Simple information: if item sampled, count all repeats
- Counting Samples [Gibbons & Mattias 98]
  - Sample new items with fixed probability p, count repeats as c_i
  - Unbiased estimate of total count: $1/p + (c_i – 1)$
- Sample and Hold [Estan & Varghese 02]: generalize to weighted keys
  - New key with weight b sampled with probability $1 - (1-p)^b$
- Lower variance compared with independent sampling
  - But sample size will grow as pn
- Adaptive sample and hold: reduce p when needed
  - "Sticky sampling": geometric decreases in p [Manku, Motwani 02]
  - Much subsequent work tuning decrease in p to maintain sample size

Sketch Guided Sampling

- Go further: avoid sampling the heavy keys as much
  - Uniform sampling will pick from the heavy keys again and again
- Idea: use an oracle to tell when a key is heavy [Kumar Xu 06]
  - Adjust sampling probability accordingly
- Can use a “sketch” data structure to play the role of oracle
  - Like a hash table with collisions, tracks approximate frequencies
  - E.g. (Counting) Bloom Filters, Count-Min Sketch
- Track probability with which key is sampled, use HT estimators
  - Set probability of sampling key with (estimated) weight w as $1/(1 + \varepsilon w)$ for parameter ε : decreases as w increases
  - Decreasing ε improves accuracy, increases sample size

Challenges for Smart Stream Sampling

- Current router constraints
  - Flow tables maintained in fast expensive SRAM
    - To support per packet key lookup at line rate
- Implementation requirements
  - Sample and Hold: still need per packet lookup
  - Sampled NetFlow: (uniform) sampling reduces lookup rate
    - Easier to implement despite inferior statistical properties
- Long development times to realize new sampling algorithms
- Similar concerns affect sampling in other applications
  - Processing large amounts of data needs awareness of hardware
  - Uniform sampling means no coordination needed in distributed setting

Future for Smarter Stream Sampling

- Software Defined Networking
  - Current: proprietary software running on special vendor equipment
  - Future: open software and protocols on commodity hardware
- Potentially offers flexibility in traffic measurement
  - Allocate system resources to measurement tasks as needed
  - Dynamic reconfiguration, fine grained tuning of sampling
  - Stateful packet inspection and sampling for network security
- Technical challenges:
  - High rate packet processing in software
  - Transparent support from commodity hardware
  - OpenSketch: [Yu, Jose, Miao, 2013]
- Same issues in other applications: use of commodity programmable HW

Stream Sampling: Sampling as Cost Optimization

Matching Data to Sampling Analysis

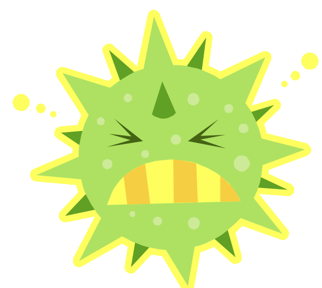
◊ Generic problem 1: Counting objects: weight $x_i = 1$

Bernoulli (uniform) sampling with probability p works fine

- Estimated subset count $X'(S) = \#\{\text{samples in } S\} / p$
- Relative Variance $(X'(S)) = (1/p - 1)/X(S)$
  - □ given p, get any desired accuracy for large enough S

◊ Generic problem 2: x_i in Pareto distribution, a.k.a. 80-20 law

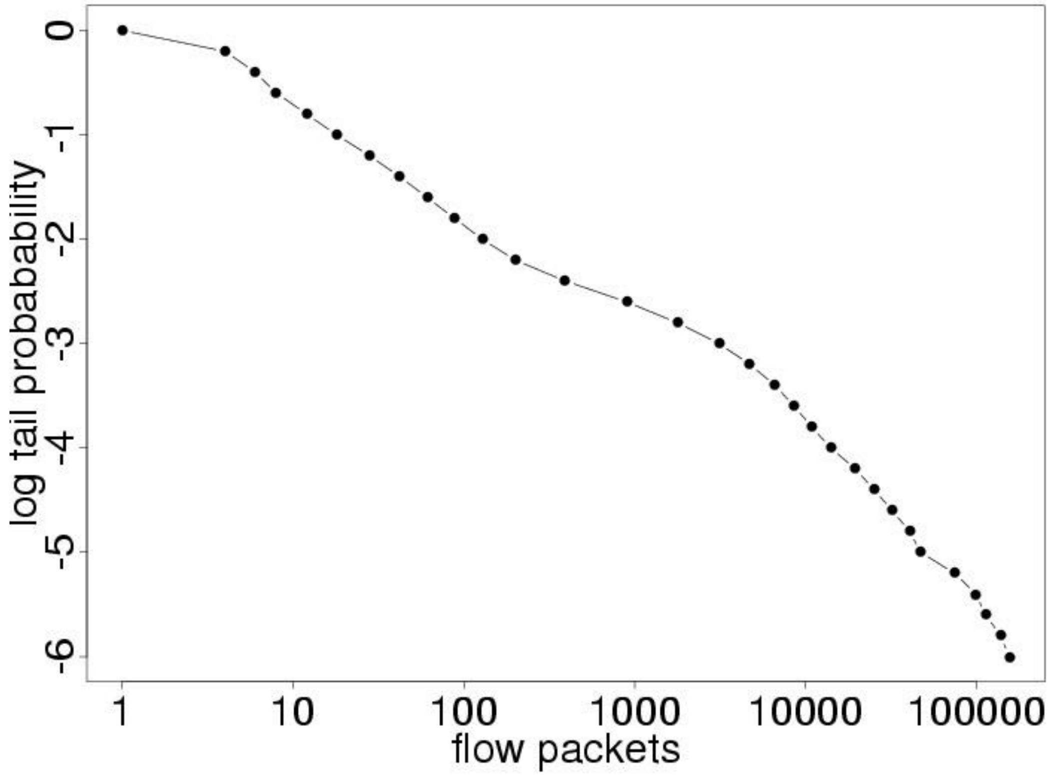
- Small proportion of objects possess a large proportion of total weight
  - □ How to best to sample objects to accurately estimate weight?
- Uniform sampling?
  - □ likely to omit heavy objects $\Rightarrow$ big hit on accuracy
  - □ making selection set S large doesn't help



- Select m largest objects ?
  - □ biased & smaller objects systematically ignored

Heavy Tails in the Internet and Beyond

- Files sizes in storage
- Bytes and packets per network flow
- Degree distributions in web graph, social networks



Non-Uniform Sampling

- Extensive literature: see book by [Tille, “Sampling Algorithms”, 2006]
- Predates “Big Data”
  - Focus on statistical properties, not so much computational
- IPPS: Inclusion Probability Proportional to Size
  - Variance Optimal for HT Estimation
  - Sampling probabilities for multivariate version: [Chao 1982, Tille 1996]
  - Efficient stream sampling algorithm: [Cohen et. al. 2009]

Costs of Non-Uniform Sampling

- Independent sampling from n objects with weights $\{x_1, \dots, x_n\}$
- Goal: find the "best" sampling probabilities $\{p_1, \dots, p_n\}$
- Horvitz-Thompson: unbiased estimation of each x_i by

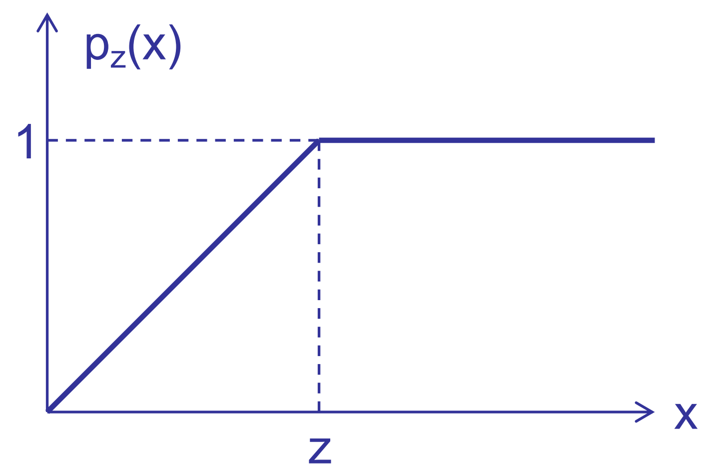
$$x'_i = \begin{cases} x_i / p_i & \text{if weight } i \text{ selected} \\ 0 & \text{otherwise} \end{cases}$$

- Two costs to balance:
  1. Estimation Variance: $\mathrm{Var}(x'_i) = x^2_i\,(1/p_i - 1)$
  2. Expected Sample Size: $\sum_i p_i$
- Minimize Linear Combination Cost: $\sum_i (x_i^2(1/p_i - 1) + z^2 p_i)$
  - z expresses relative importance of small sample vs. small variance

Minimal Cost Sampling: IPPS

IPPS: Inclusion Probability Proportional to Size

- Minimize Cost $\Sigma_i (x_i^2 (1/p_i - 1) + z^2 p_i)$ subject to $1 \geq p_i \geq 0$
- Solution: $p_i = p_z(x_i) = \min\{1, x_i /z\}$
  - small objects $(x_i < z)$ selected with probability proportional to size
  - large objects $(x_i \geq z)$ selected with probability 1
  - Call z the “sampling threshold”
  - Unbiased estimator $x_i/p_i = \max\{x_i , z\}$
- Perhaps reminiscent of importance sampling, but not the same:
  - make no assumptions concerning distribution of the x



Error Estimates and Bounds

- Variance Based:
  - HT sampling variance for single object of weight x_i
    - $Var(x'_i) = x^2_i (1/p_i - 1) = x^2_i (1/\min\{1,x_i/z\} - 1) \leq z\, x_i$
  - Subset sum $X(S)= \Sigma_{i\in S}\, x_i$ is estimated by $X'(S) = \Sigma_{i\in S}\, x'_i$
    - $Var(X'(S)) \leq z\, X(S)$
- Exponential Bounds
  - E.g. $Prob[X'(S) = 0] \leq exp(- X(S) / z)$
- Bounds are simple and powerful
  - depend only on subset sum $X(S)$, not individual constituents

Sampled IP Traffic Measurements

- ◊ Packet Sampled NetFlow
  - – Sample packet stream in router to limit rate of key lookup: uniform $1/N$
  - – Aggregate sampled packets into flow records by key
- ◊ Model: packet stream of (key, bytesize) pairs $\{ (b_i, k_i) \}$
- ◊ Packet sampled flow record (b,k) where $b = \Sigma \{b_i : i \text{ sampled} \wedge k_i = k\}$
  - – HT estimate b/N of total bytes in flow
- ◊ Downstream sampling of flow records in measurement infrastructure
  - – IPPS sampling, probability $\min\{1, b/(Nz)\}$
- ◊ Chained variance bound for any subset sum X of flows
  - – $\mathrm{Var}(X') \leq (z + N b_{max}) X$ where b_{max} = maximum packet byte size
  - – Regardless of how packets are distributed amongst flows

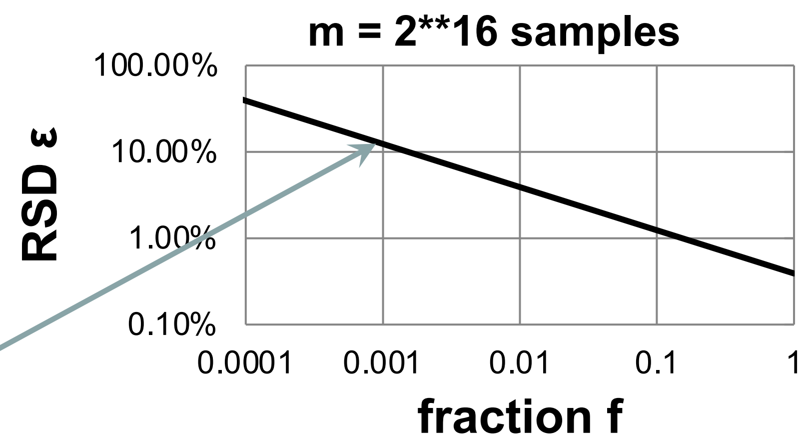
[Duffield, Lund, Thorup, IEEE ToIT, 2004]

Estimation Accuracy in Practice

- Estimate any subset sum comprising at least some fraction f of weight
- Suppose: sample size m
- Analysis: typical estimation error ε (relative standard deviation) obeys

$$\varepsilon \leq 1 / \sqrt{f\,m}$$

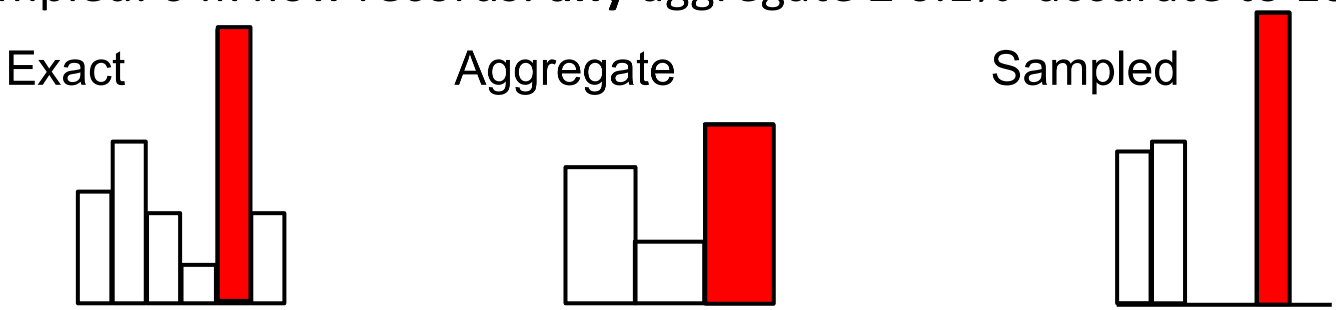
Estimate fraction f = 0.1% with typical relative error 12%:



- 2*16 = same storage needed for aggregates over 16 bit address prefixes
  - But sampling gives more flexibility to estimate traffic within aggregates

Heavy Hitters: Exact vs. Aggregate vs. Sampled

- Sampling does not tell you where the interesting features are
  - But does speed up the ability to find them with existing tools
- Example: Heavy Hitter Detection
  - Setting: Flow records reporting 10GB/s traffic stream
  - Aim: find Heavy Hitters = IP prefixes comprising ≥ 0.1% of traffic
  - Response time needed: 5 minute
- Compare:
  - Exact: 10GB/s x 5 minutes yields upwards of 300M flow records
  - 64k aggregates over 16 bit prefixes: no deeper drill-down possible
  - Sampled: 64k flow records: **any** aggregate ≥ 0.1% accurate to 10%



Cost Optimization for Sampling

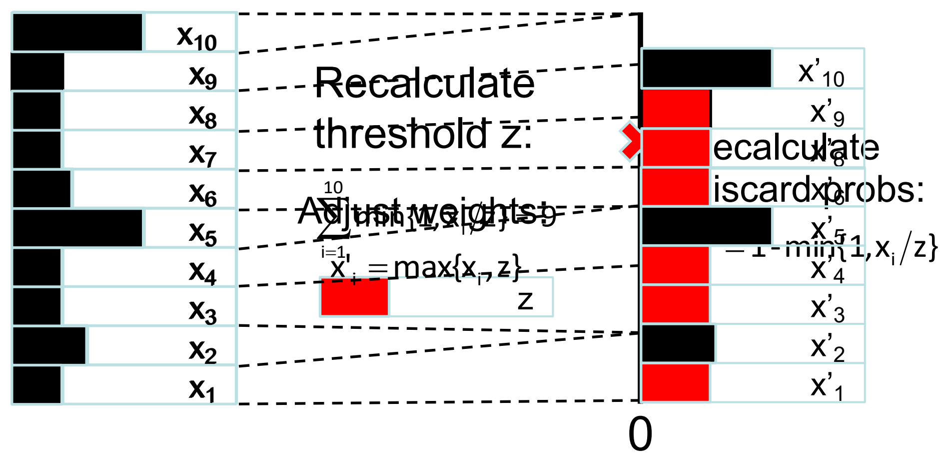
Several different approaches optimize for different objectives:

1. Fixed Sample Size IPPS Sample
   - Variance Optimal sampling: minimal variance unbiased estimation
2. Structure Aware Sampling
   - Improve estimation accuracy for subnet queries using topological cost
3. Fair Sampling
   - Adaptively balance sampling budget over subpopulations of flows
   - Uniform estimation accuracy regardless of subpopulation size
4. Stable Sampling
   - Increase stability of sample set by imposing cost on changes

IPPS Stream Reservoir Sampling

- Each arriving item:
  - Provisionally include item in reservoir
  - If $m+1$ items, discard 1 item randomly
    - Calculate threshold z to sample m items on average: z solves $\Sigma_i\ p_z(x_i) = m$
    - Discard item i with probability $q_i = 1 - p_z(x_i)$
    - Adjust m surviving x_i with Horvitz-Thompson $x'_i = x_i / p_i = \max\{x_i, z\}$
- Efficient Implementation:
  - Computational cost $O(\log m)$ per item, amortized cost $O(\log \log m)$

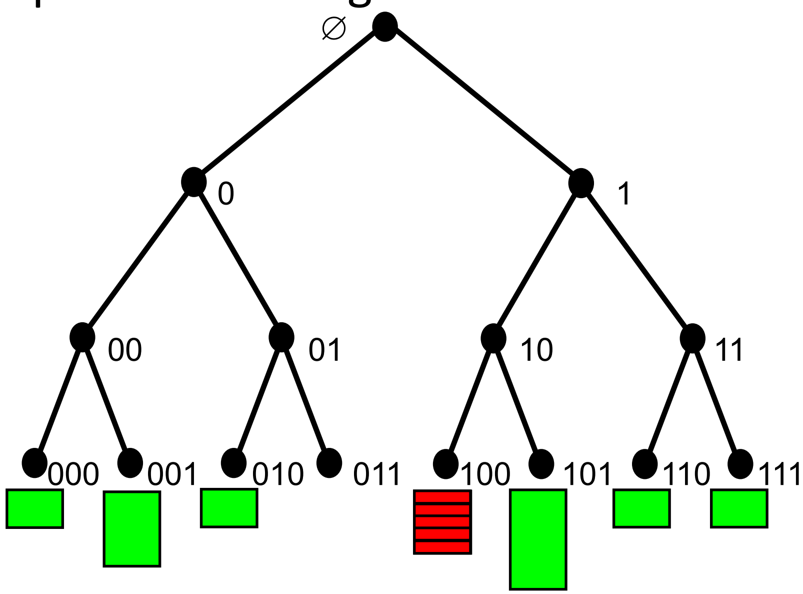
Example:
m=9



[Cohen, Duffield, Lund, Kaplan, Thorup; SODA 2009, SIAM J. Comput. 2011]

Structure (Un)Aware Sampling

- Sampling is oblivious to structure in keys (IP address hierarchy)
  - Estimation disperses the weight of discarded items to surviving samples

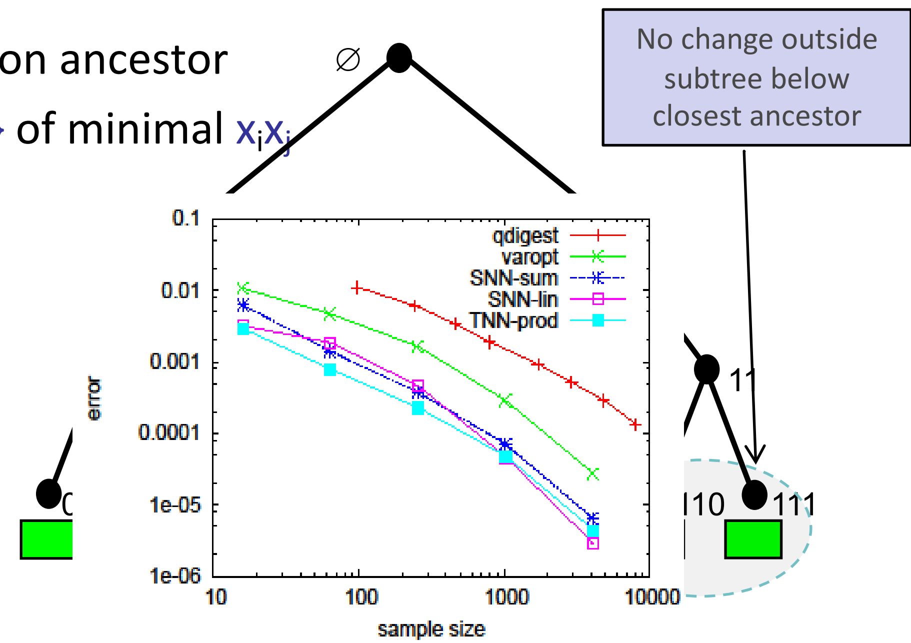


- Queries structure aware: subset sums over related keys (IP subnets)
  - Accuracy on LHS is decreased by discarding weight on RHS

Localizing Weight Redistribution

- Initial weight set $\{x_i : i \in S\}$ for some $S \subset \Omega$
  - E.g. Ω = possible IP addresses, S =observed IP addresses
- Attribute "range cost" $C(\{x_i : i \in R\})$ for each weight subset $R \subseteq S$
  - Possible factors for Range Cost:
    - Sampling variance
    - Topology e.g. height of lowest common ancestor
  - Heuristics: R^* = Nearest Neighbor $\{x_i , x_j\}$ of minimal $x_i x_j$
- Sample k items from S:
  - Progressively remove one item from subset with minimal range cost:
  - While$(|S| > k)$
    - Find $R^* \subseteq S$ of minimal range cost.
    - Remove a weight from R^* w/ VarOpt

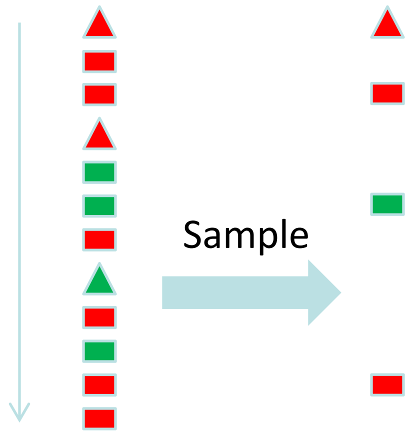
[Cohen, Cormode, Duffield; PVLDB 2011]



Order of magnitude reduction in average subnet error vs. VarOpt

Fair Sampling Across Subpopulations

- Analysis queries often focus on specific subpopulations
  - E.g. networking: different customers, user applications, network paths
- Wide variation in subpopulation size
  - 5 orders of magnitude variation in traffic on interfaces of access router
- If uniform sampling across subpopulations:
  - Poor estimation accuracy on subset sums within small subpopulations



Color = subpopulation

▲ , ▲ = interesting items

- occurrence proportional to subpopulation size

Uniform Sampling across subpopulations:

- Difficult to track proportion of interesting items within small subpopulations: ▲

Fair Sampling Across Subpopulations

- Minimize **relative** variance by sharing budget m over subpopulations
  - Total n objects in subpopulations $n_1,\ldots,n_d$ with $\Sigma_i n_i=n$
  - Allocate budget m_i to each subpopulation n_i with $\Sigma_i m_i=m$
- Minimize average population relative variance R = const. $\Sigma_i 1/m_i$
- Theorem:
  - R minimized when $\{m_i\}$ are Max-Min Fair share of m under demands $\{n_i\}$
- Streaming
  - Problem: don't know subpopulation sizes $\{n_i\}$ in advance
- Solution: progressive fair sharing as reservoir sample
  - Provisionally include each arrival
  - Discard 1 item as VarOpt sample from any maximal subpopulation
- Theorem [Duffield; Sigmetrics 2012]:
  - Max-Min Fair at all times; equality in distribution with VarOpt samples $\{m_i$ from $n_i\}$

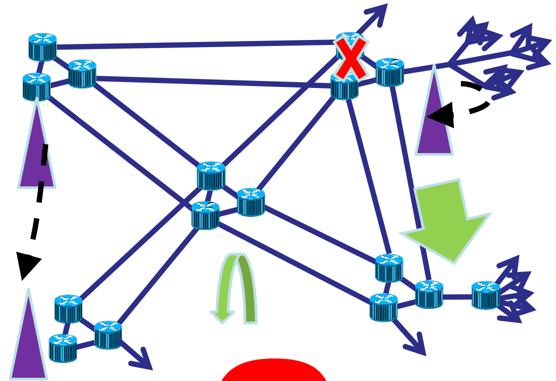
Stable Sampling

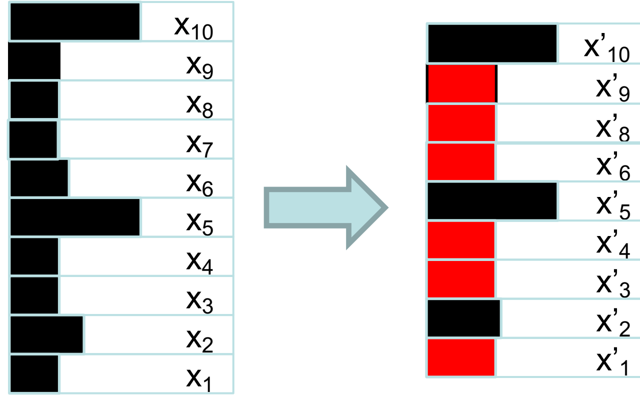
- Setting: Sampling a population over successive periods
- Sample independently at each time period?
  - Cost associated with sample churn
  - Time series analysis of set of relatively stable keys
- Find sampling probabilities through cost minimization
  - Minimize Cost = Estimation Variance + z * E[#Churn]
- Size m sample with maximal expected churn D
  - weights $\{x_i\}$, previous sampling probabilities $\{p_i\}$
  - find new sampling probabilities $\{q_i\}$ to minimize cost of taking m samples
  - Minimize $\Sigma_i x^2_i / q_i$ subject to $1 \geq q_i \geq 0$, $\Sigma_I q_i = m$ and $\Sigma_I \mid p_i - q_i \mid \leq D$

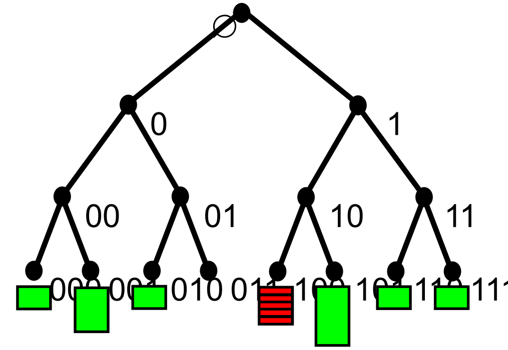
[Cohen, Cormode, Duffield, Lund 13]

Summary of Part 1

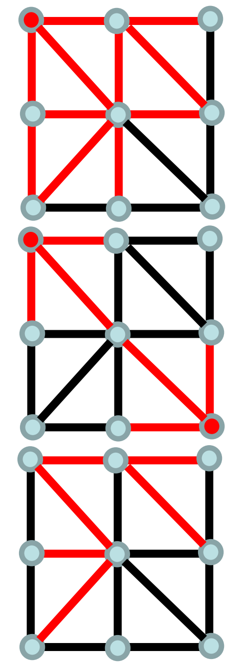
- Sampling as a powerful, general summarization technique
- Unbiased estimation via Horvitz-Thompson estimators
- Sampling from streams of data
  - Uniform sampling: reservoir sampling
  - Weighted generalizations: sample and hold, counting samples
- Advances in stream sampling
  - The cost principle for sample design, and IPPS methods
  - Threshold, priority and VarOpt sampling
  - Extending the cost principle:
    - structure aware, fair sampling, stable sampling, sketch guided

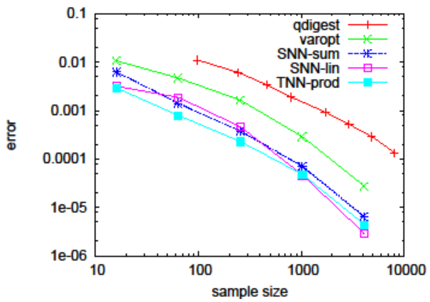






Sampling for Big Data





Graham Cormode, University of Warwick

G.Cormode@warwick.ac.uk

Nick Duffield, Texas A&M University

Nick.Duffield@gmail.com