DNS OARC 26

public

DNS Traffic Sampling

A HyperLogLog seasoned implementation for dnscap

Madrid
2017-05-14

Alexander Mayrhofer
Head of R&D

DNS Sampling - Background

- Operational Monitoring of DNS traffic
  - Practice of many DNS operators
  - Capture / storage – potentially more resource intensive than actual service
- Solution path: Store a subset
  - Sensible sampling strategies
  - How does sampling affect estimates?
  - Can we work around the caveats?

What is „Sampling“?

*„the **selection of a subset of individuals** from within a statistical population to **estimate characteristics** of the whole population”*

-Wikipedia

- **Application to DNS:** Selecting a subset of messages from a traffic stream / pcap

Which sampling strategy?

- Method?
  - Random Sampling
  - Systematic Sampling
  - Stratified (..) Sampling
- Intensity?
  - 1% … 100% ?
- Existing practices?
  - „Spatial“ / „temporal“ / ?

„DNS Sampling“ @ nic.at R&D

Theory

- Research impact of Sampling on DNS traffic
- Master Thesis
  - Andreas Blatt, Student
  - University of Technology Vienna (Dept. of Statistics and Probability Theory)
- Mentored by nic.at / SIDN Labs

Practice

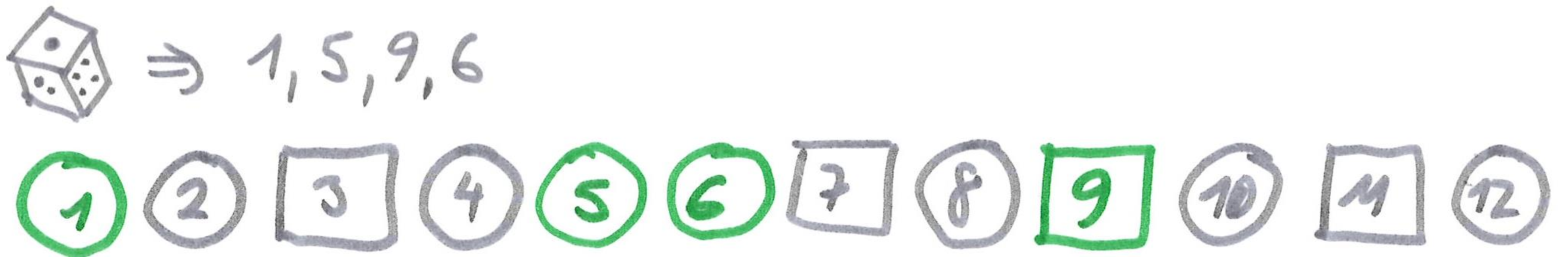
- Implement sampling in a well known tool
- Intern @ nic.at
  - Christian Egger
  - Freshman an University of Technology (Computer Science)
- Mentored by nic.at R&D Team

Sampling Methods

n=12, intensity=1/3 (33.33333%)

Random (probabilistic) Sampling

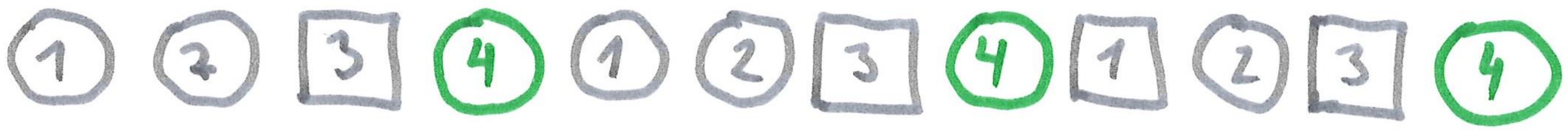
- Pick x% random individuals
  - (or each individual with x% probability)



- **Pro:** Considered the „fairest“ method – each individual has equal chance („no packet left behind“ ;)
- **Con:** requires a source of (pseudo) random numbers
- **Engineer‘s Conclusion:** Hard to implement properly – maybe investigate „pseudo-random“?

Systematic Sampling

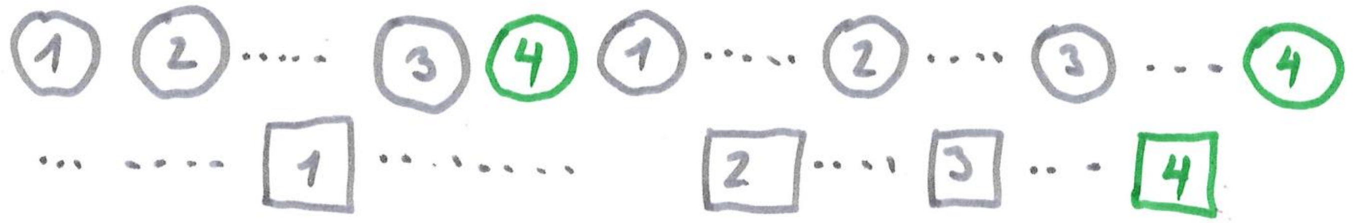
- Pick every nth individual



- **Pro:** no source of entropy required
- **Con:** Most individuals will never be selected (the „ene mene mu“ effect)
  - **Con/pro? side effect:** sampling is reproducible
- **(Lazy) Engineer‘s conclusion:** Looks fast and easy – is it good enough? -> Subject of Andreas‘ paper

Stratified (systematic) Sampling

- Create seperate groups („strata“)
- Sample each stratum individually



- **Pro:** Disproportionate would allow investigating a „rare“ subgroup (TCP?) in greater precision
- **Con:** Results from subgroups are harder to compare
- **Engineer‘s conclusion:** Hard to find a use case - Stratify on which parameter? (Client AS number was considered in hallway discussions..)

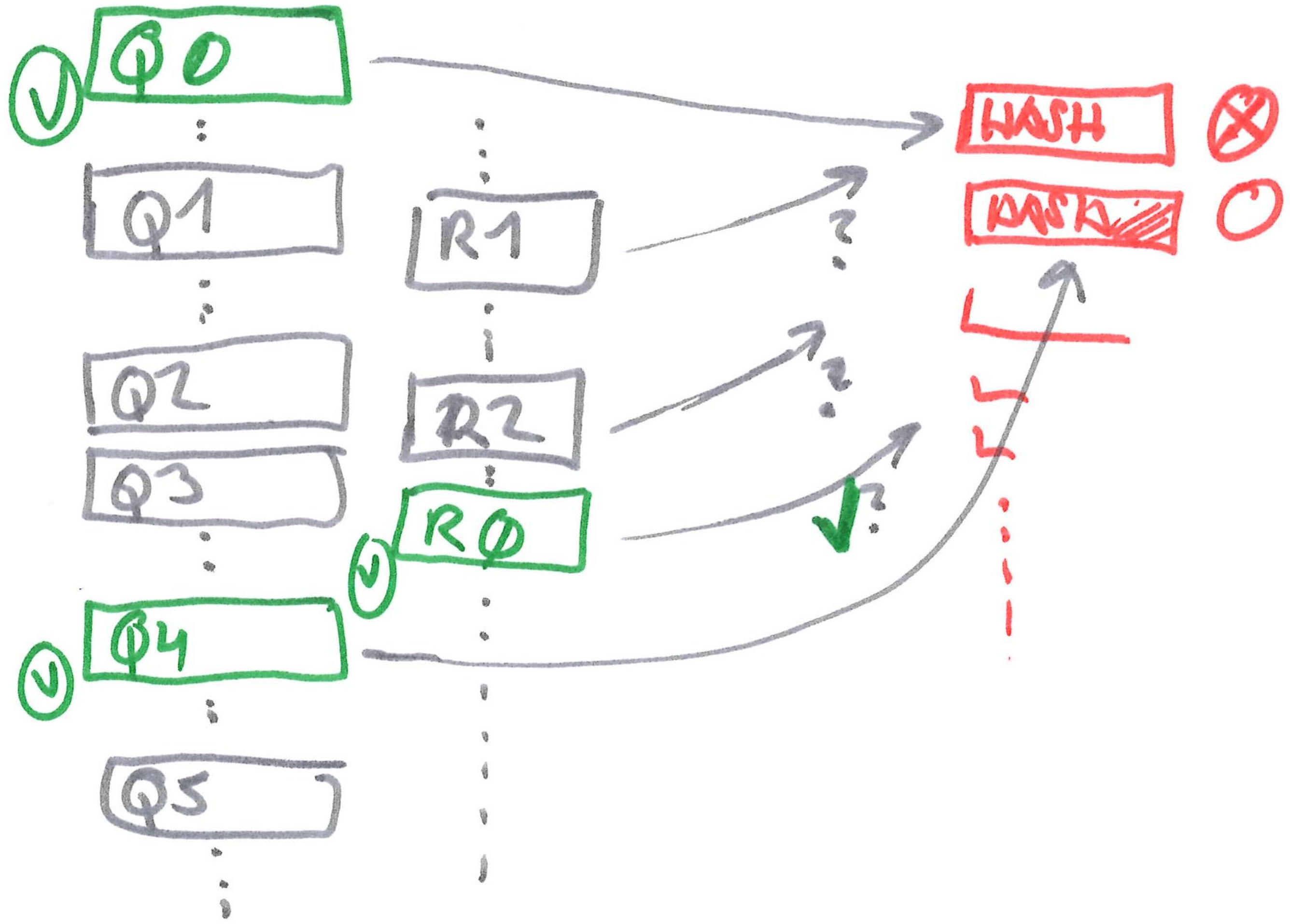
Other forms of „Sampling“

- „Temporal“ Sampling:
  - Based on time
  - First 5 minutes of each hour
  - DITL
- „Spatial“ Sampling
  - Based on geography/topology
  - Eg. 3 out of 7 Nameservers
  - One of 4 bonded network interfaces?

dnscap Implementation

- **Design Choice:** Systematic Sampling
  - Every nth query
  - Based on order of arrival
- Responses?
  - Every nth? Does not correlate to queries!
  - Requirement: Responses matching sampled queries
  - Hash-based correlation

Hash-based query/response matching



Implementation status

```
dnscap -i eth0 -g -q 5
```

- Samples (UDP) to eg. 20% (1/**5**th)
- Pull request in Github
  - https://github.com/DNS-OARC/dnscap/pull/15
- What doesn‘t work?
  - Does not sample TCP based traffic
  - Does not sample Fragments nor ICMP
  - (limited support in dnscap for those in general)

Sampling TODO

- Get reviewers
- More Testing
  - Performance impact?
  - Fuzzy testing?
- Get patch into upstream *wink*
- Limit hash growth
- Evaluate probabilistic sampling options
  - We have a hash already – but predictable

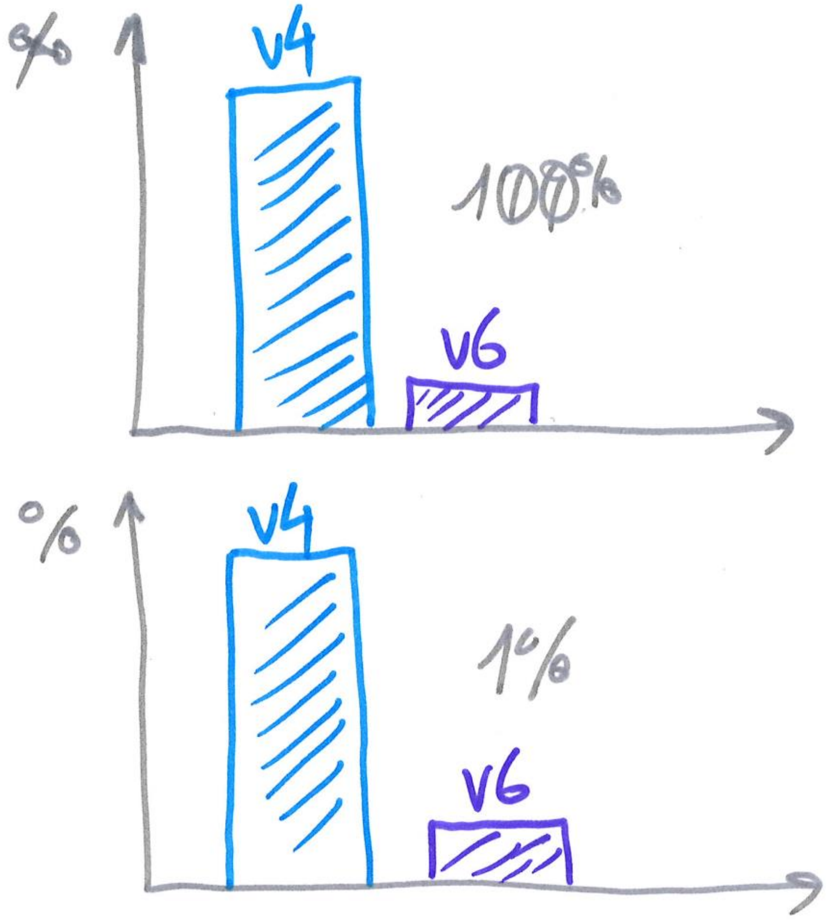
Similar Work

- https://github.com/farsightsec/nmsg
  - „dnsqr“ message module (Query/Response matching, Fragment reassembly)
  - „sample“ filter module (systematic and probabilistic sampling)

- Robert Edmonds advised when reviewing this talk proposal

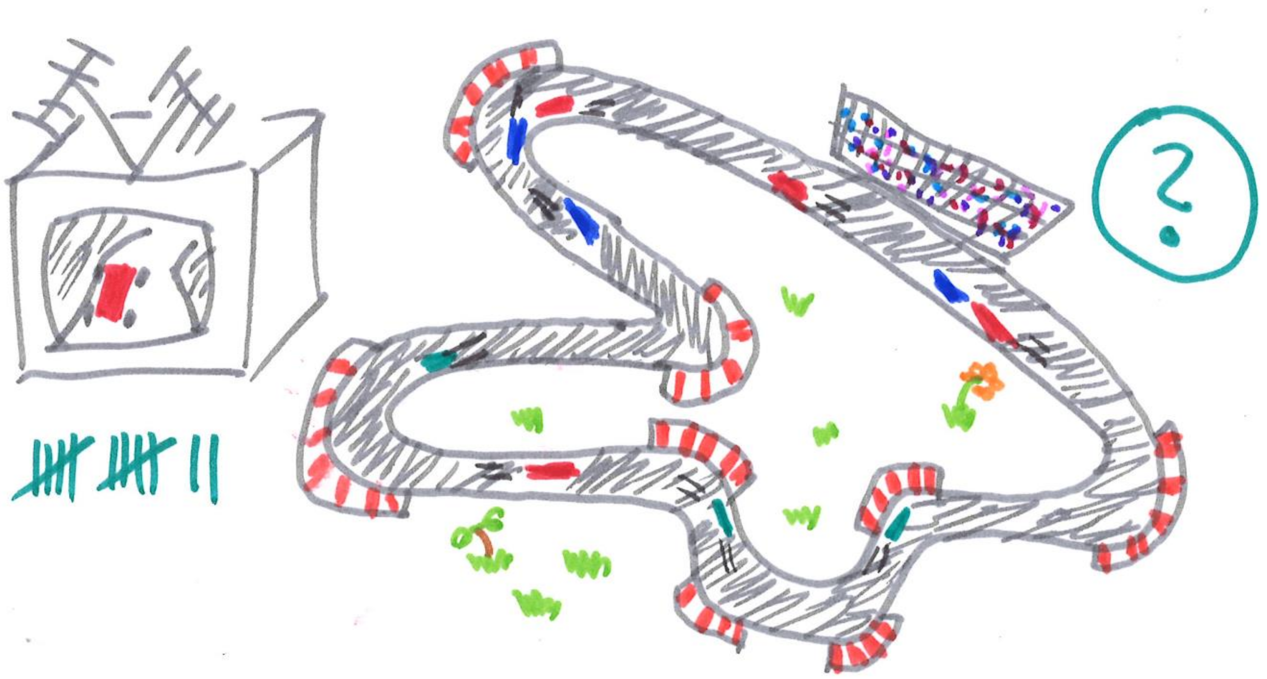
Properties of sampled traffic

- Most aggregates are still very good
  - Qps
  - v4 / v6
  - Source port distribution
  - Avg. QNAME length
  - Top clients
- More details to come in Andreas' master thesis



Problematic: Set Cardinalities

- # of distinct QNAMES
- # of distinct src IP Adresses



How to address this problem?

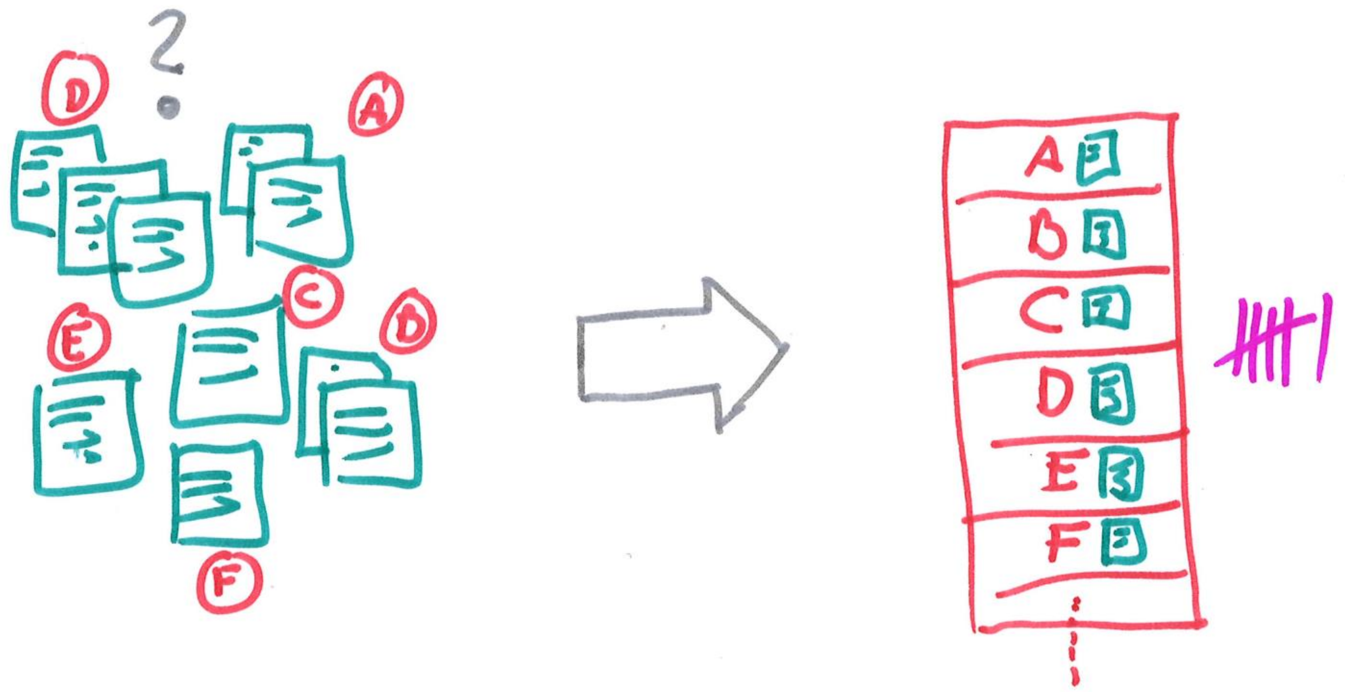
- HyperLogLog
  - Philippe Flajolet, 2007
  - Redis pf_* functions (http://antirez.com/news/75)

Idea: Augment sampled traffic with on-the-fly counters for QNAME and Client IP

- The well known „Set Cardinality“ problem

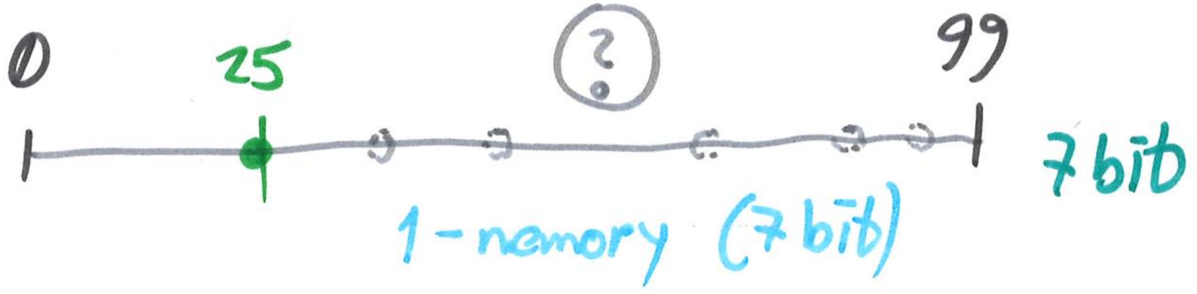
Set Cardinality Algorithms

- Storing each unique element
- Storing a hash (collisions!)

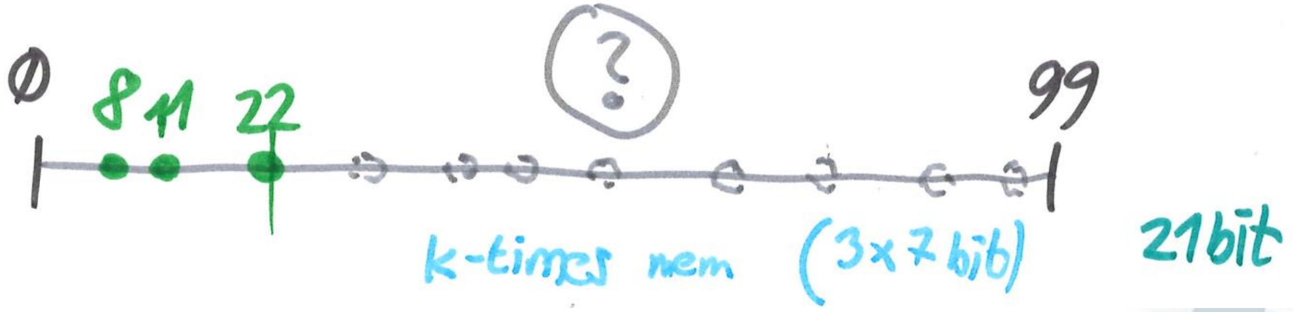


Set Cardinality *Estimation*

- Remembering only the „lowest“ element



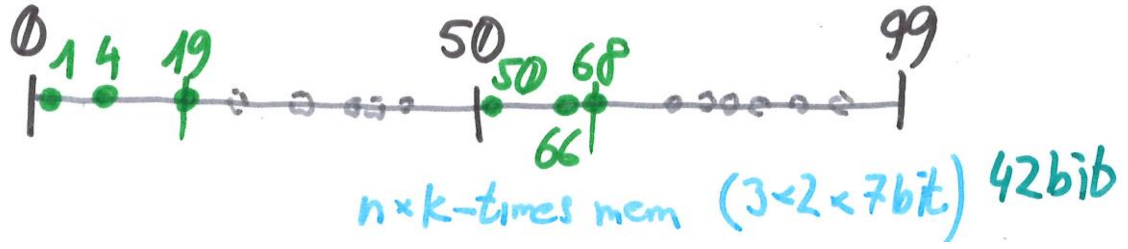
- Or a few of them („k-minimum“)



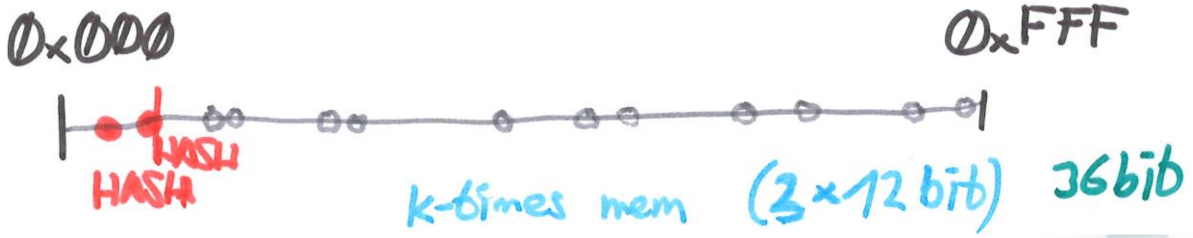
https://research.neustar.biz/2012/07/09/sketch-of-the-day-k-minimum-values/

Precision / Non-numeric data

- More precision? Use multiple „windows“ of k-min values (memory complexity!)

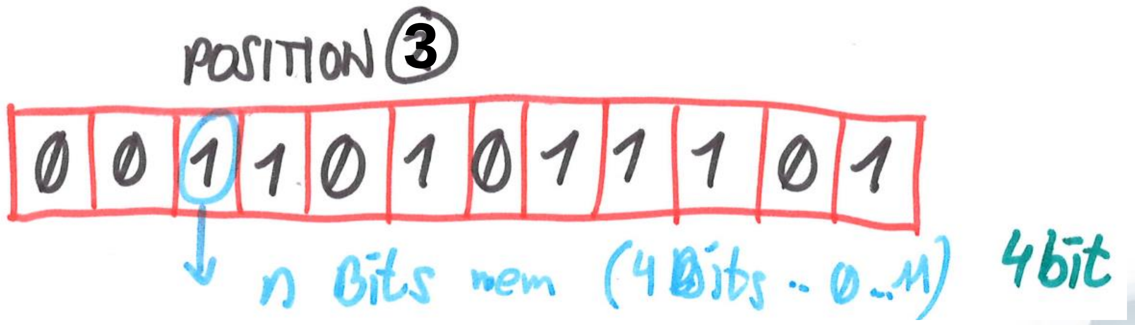


- More complex elements? Use (uniformly distributed) hashes



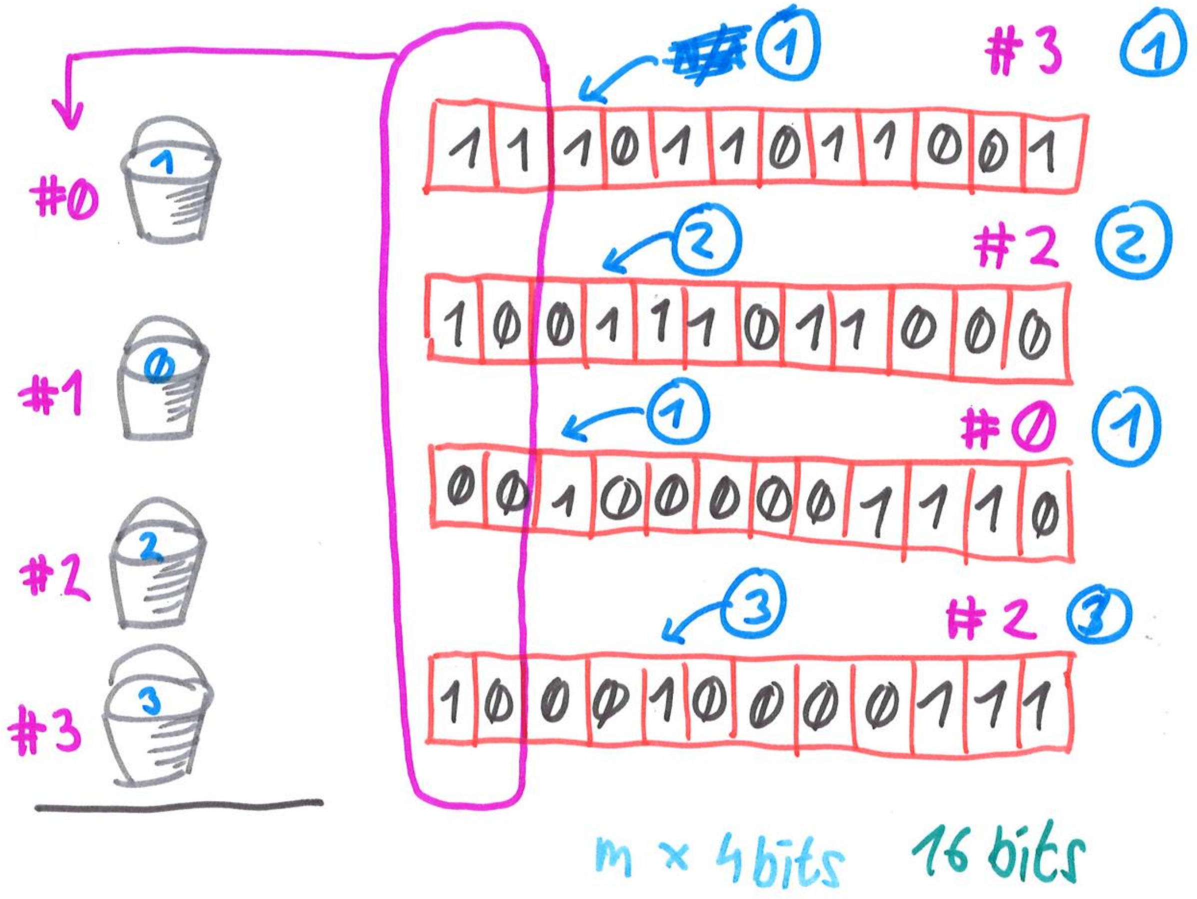
Reduce memory complexity

- Don‘t store the values themselves
- remember the greatest position of the first „1“ bit across the set



- Coarse estimator: Set cardinality is > 2^p

HyperLogLog concept



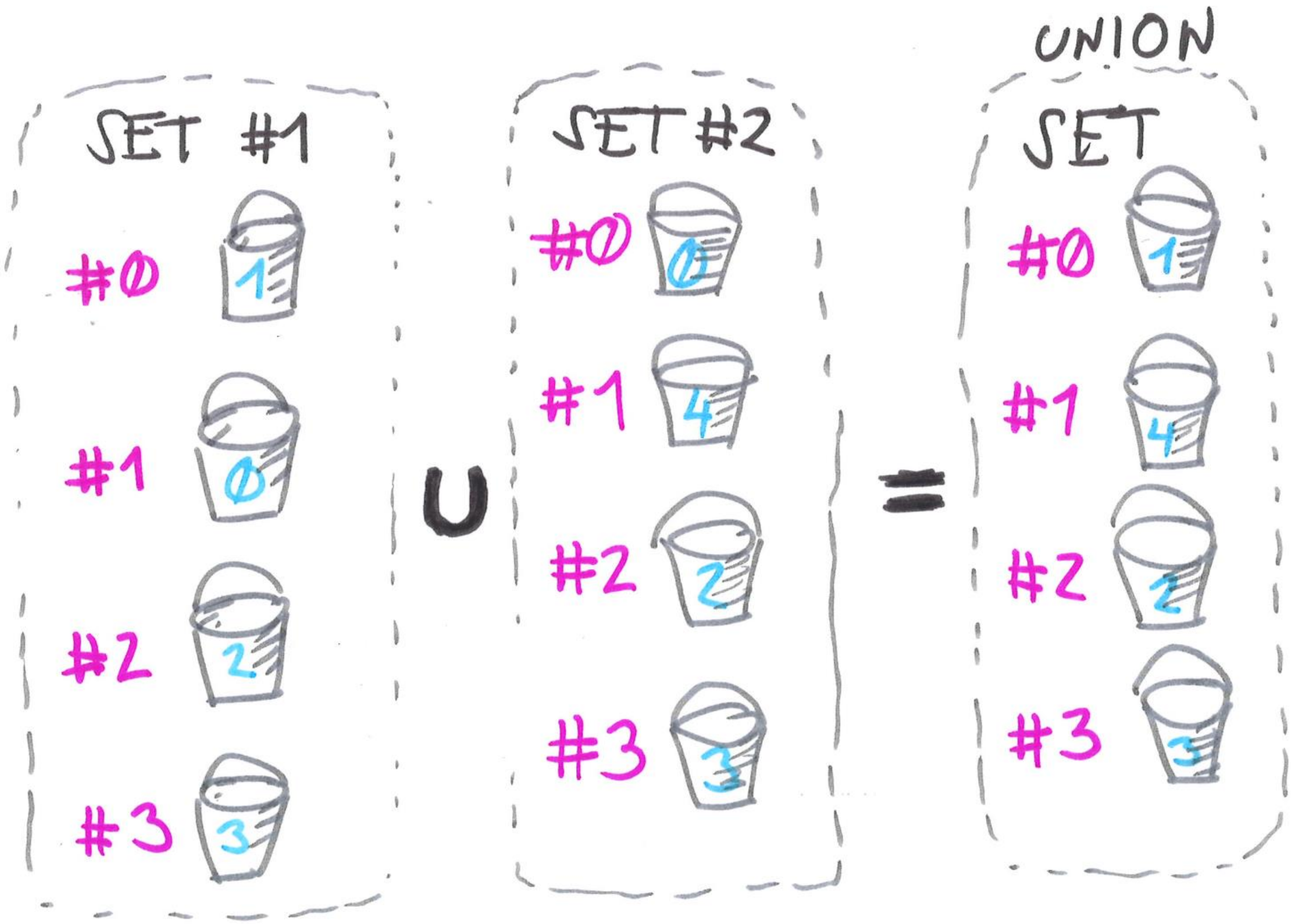
http://algo.inria.fr/flajolet/Publications/FlFuGaMe07.pdf

HyperLogLog details

- The magic is in the aggregation function
  - Harmonic mean
- 32-bit Hash function
  - Typically 12-16 bit Bucket ID
  - Leaves 16-20 bits
  - Requires storing 4-6 bit per bucket

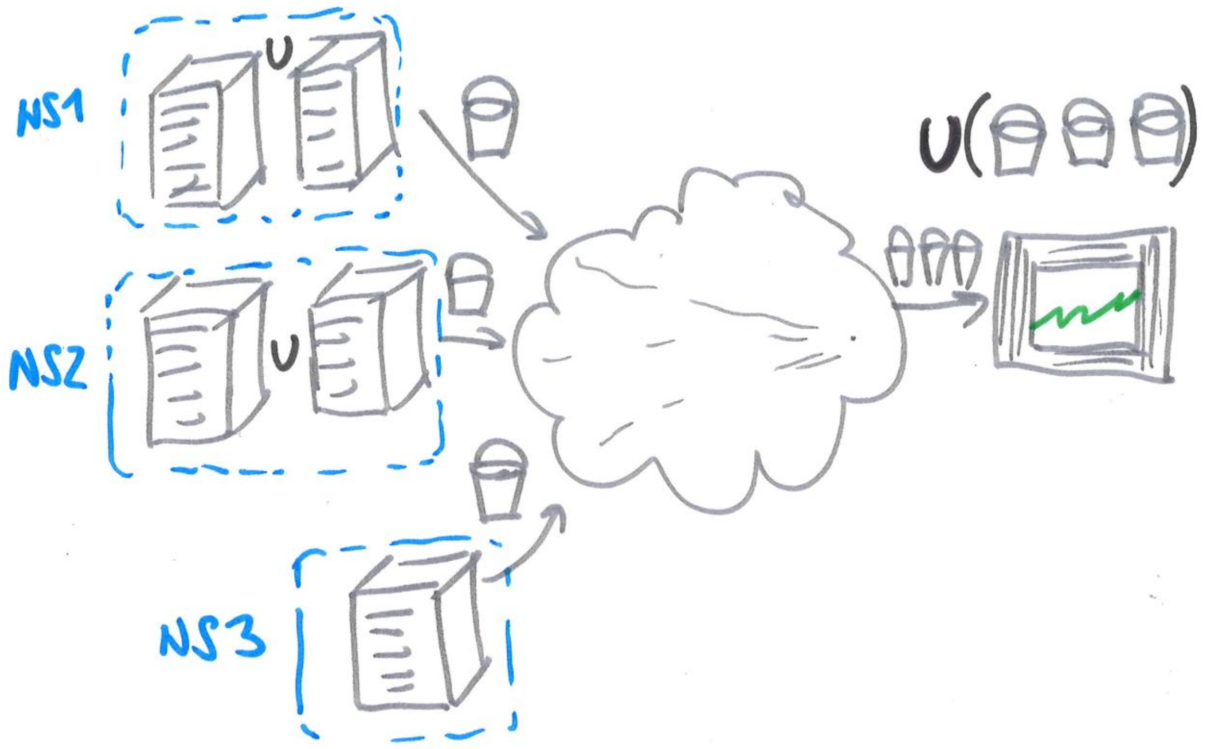
- Accuracy ~1.04/sqrt(m)
  - Eg. 0.8% with 16k buckets, ~12k mem

More magic – Unions!



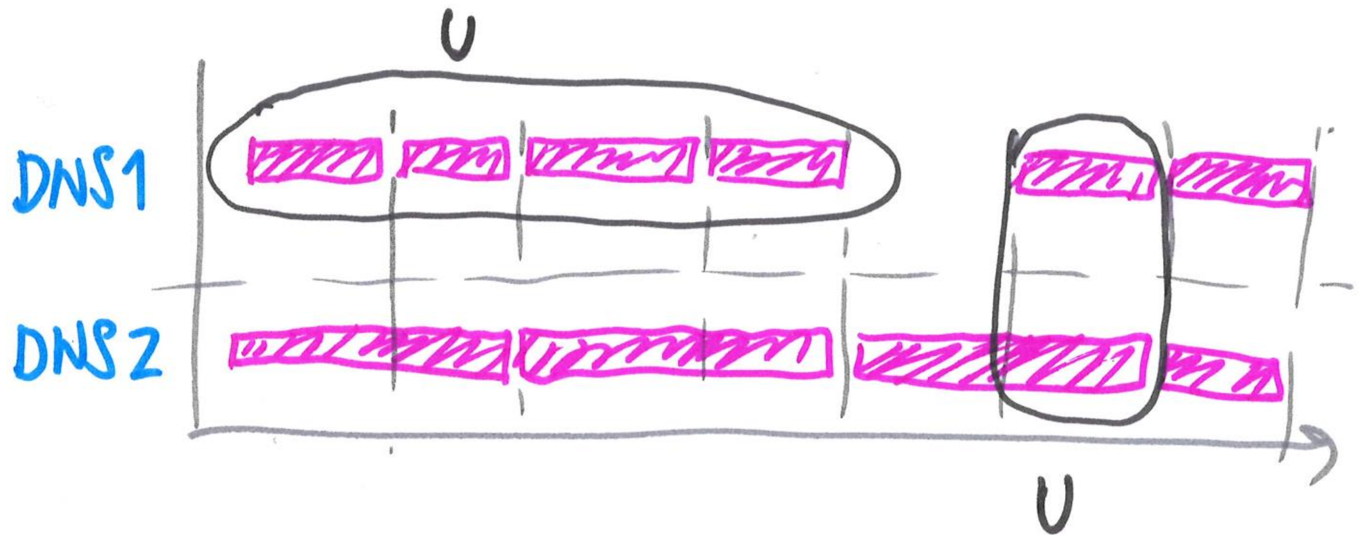
DNS infrastructure and Unions

- HyperLogLog‘s „Union“ property fits the DNS operations model perfectly



More Unions!

- Time-based
  - Eg. Aggregate 5min-intervals to hours
  - Sliding window!



HyperLogLog in dnscap

- Implemented as a rough first prototype
- Outputs estimates on exit

```
This is the v4 card: 104
This is the v6 card: 0
This is the Qname card: 147
```

https://github.com/chegger/HyperLogLog

HLL TODO

- A proper implementation
- Count other sets?
- Truncate v6 addresses to /64?
- HyperLogLog++ instead?
  - 64 bit hashes
  - Significant precision improvements

https://stefanheule.com/papers/edbt13-hyperloglog.pdf

Summary

- Systematic Sampling patch for dnscap
- Most estimates survive the sampling
- Set Cardinalities are badly affected
  - HyperLogLog could be used to augment sampled traffic with those cardinalities
  - The properties of HyperLogLog perfectly fit the DNS model
  - Rough dnscap HLL prototype exists

Questions? Message <alexander.mayrhofer@nic.at>