Building a Better NetFlow

Cristian Estan, Ken Keys, David Moore, George Varghese

University of California, San Diego

SIGCOMM – September 2, 2004

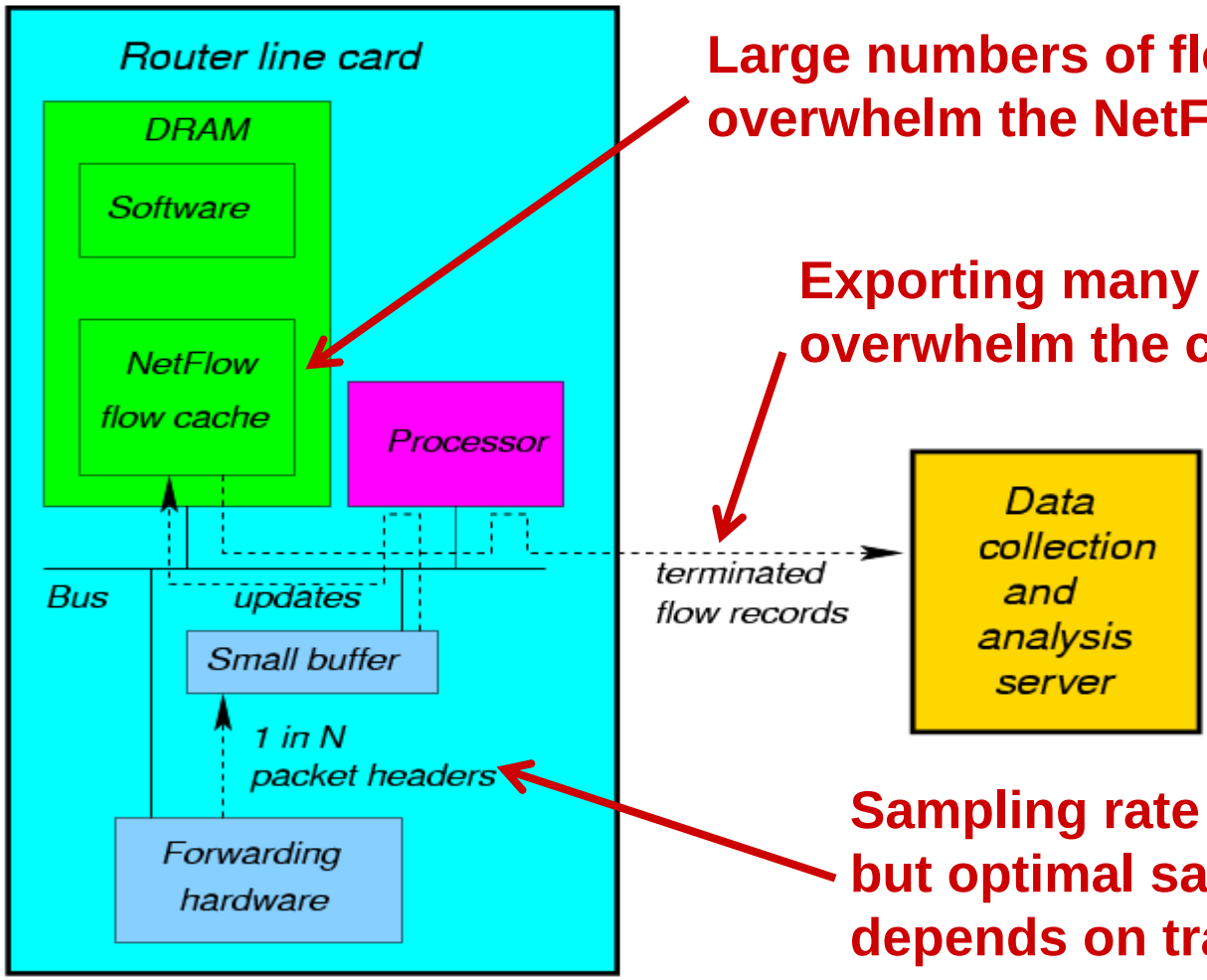
caida

UCSD CSE

Background: Flow measurement

- Flows defined by:
  - <srcIP, dstIP, proto, srcport, dstport, ToS, interface_number>
  - Other definitions possible
- Flow records have:
  - Byte counters
  - Packet counters
  - TCP flag mask
- Typical questions answered through flow data:
  - What is the application breakdown in packets/bytes?
  - How much traffic came from / went to a particular subnet?
  - How many web connections were active? (flow counting)
  - Which of my hosts seem to be spam zombies? (flow counting)

Background: NetFlow System Diagram



Large numbers of flow records can overwhelm the NetFlow cache.

Exporting many records can overwhelm the collection path.

Sampling rate set statically, but optimal sampling rate depends on traffic mix.

Background: Packet Sampling Pros and Cons

- Reduces **processor load**
- Reduces **memory usage**
- Reduces **bandwidth** for reporting

- Results are **less accurate**
- Cannot estimate non-TCP **flow counts**

- Sampling is required on high-speed links

- Finding the **sampling rate** that **balances** the pros and cons is **hard**
- The best choice **depends on traffic mix** ☹

Fixes for NetFlow

NetFlow problem	How we solve it
Mismatch between flow report and desired analysis	**Measurement bins (part 1)**
Memory and bandwidth usage strongly depend on traffic mix	Adapting sampling rate (part 2)
Admin must set sampling rate	
Cannot count non-TCP flows	In paper

When are flows reported?

- NetFlow uses timeouts to terminate flows
  - Each flow record active as long as packets arrive
  - Typical value for timeout is 1 minute

- We propose fixed size time bins (say 1 minute)
  - Terminate all flow records at the end of the bin
  - Spread reporting throughout next minute

Why Time Bins?

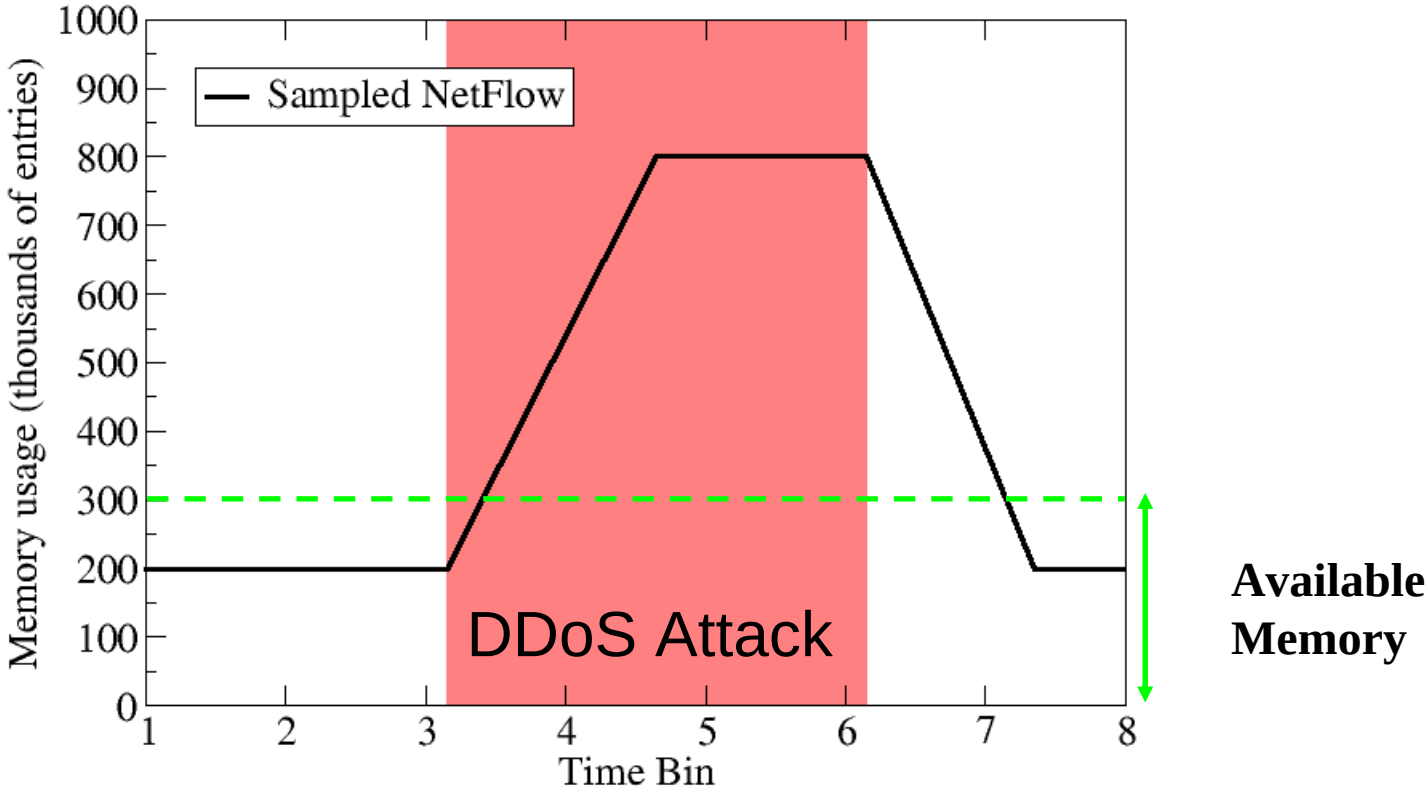
- Operators and researchers often prefer working with fixed time bins
- Most tools convert timeout flows to time bins
- For long-lived connections, the distribution of packets during that time can be important
- Time bins allow reconstruction of flow timeouts

Fixes for NetFlow

NetFlow problem	How we solve it
Mismatch between flow report and desired analysis	Measurement bins (part 1)
Memory and bandwidth usage strongly depend on traffic mix	**Adapting sampling rate (part 2)**
Admin must set sampling rate	
Cannot count non-TCP flows	In paper

NetFlow: Simulated memory usage under DDoS

- Timeout based NetFlow with fixed sampling rate



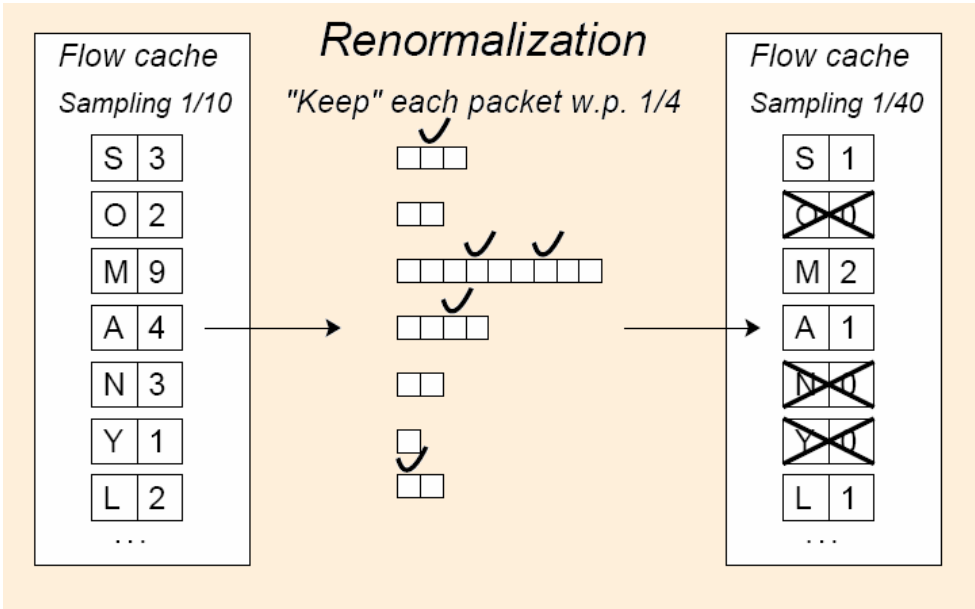
Adaptive NetFlow

- Goals:
  - Guaranteed accuracy under any traffic mix
  - Graceful response to adverse traffic
  - More meaningful tuning knob: # of desired records

- Choose the sampling rate based on traffic:
  - Use a high sampling rate when traffic allows
  - Within each time bin, reduce the rate when necessary:
    - Ensure we never overload CPU
    - Ensure we never run out of memory
  - Keep counters meaningful as sampling rate varies

Adaptive NetFlow: Renormalizing counters

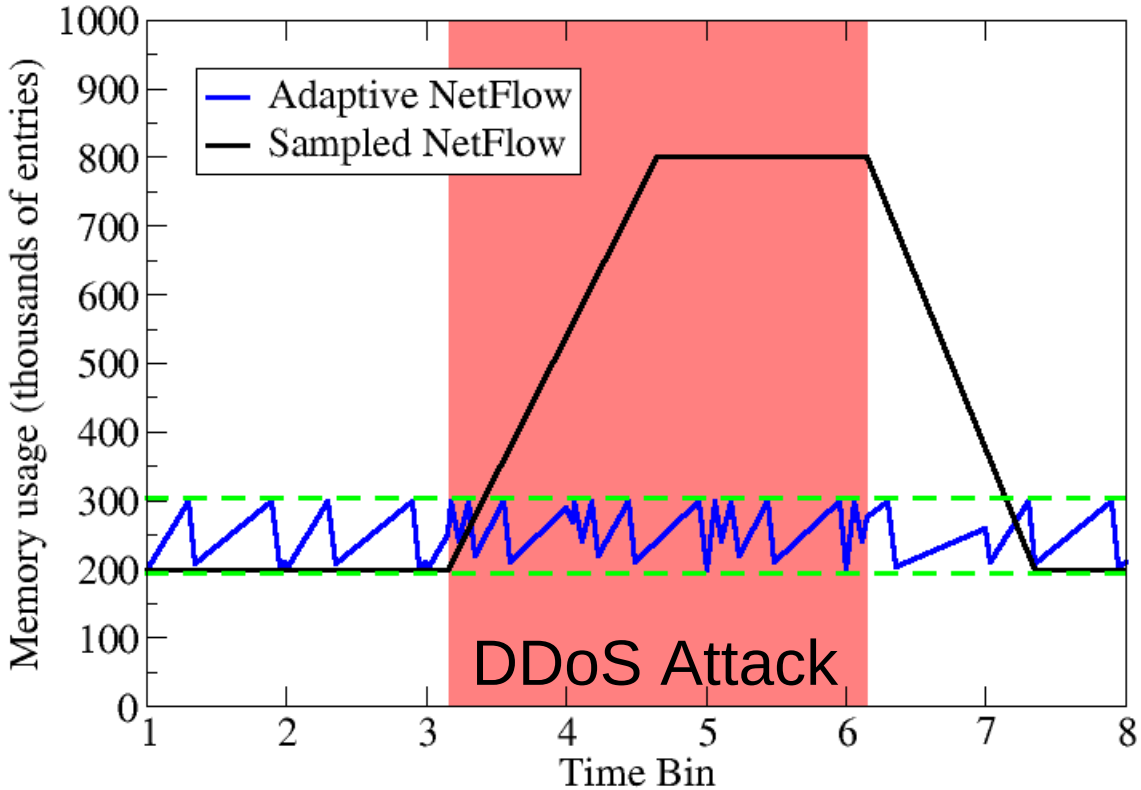
- Decreasing sampling rate
  - pretend to throw away previously observed packets
- Increasing sampling rate
  - information has already been discarded
  - would increase error
- Start each measurement bin with optimistically aggressive sampling



Adaptive NetFlow: Main tuning knob: # of records M

- User configures number of records to be exported for each measurement bin
  - Memory in router, resources for data collector
  - Accuracy of results
  - Independent of traffic mix
- Relative error in estimating an aggregate that is a certain fraction of the traffic depends on ***M***
- Dropping random records worse than generating fewer records by using lower sampling rate [DL03]

Adaptive NetFlow: Simulated memory usage under DDoS



M=200,000 records, 1 minute time bins

Adaptive NetFlow: CPU usage

- Renormalization in parallel with operation
- Efficient renormalization – for most records only simple integer arithmetic, no random numbers
  - Updating 1 entry 3.4 μs
  - Renormalizing 1 entry 1.5 μs

- Initial sampling rate chosen to allow update and renormalization with worst-case traffic mix

Adaptive NetFlow: Picking the sampling rate

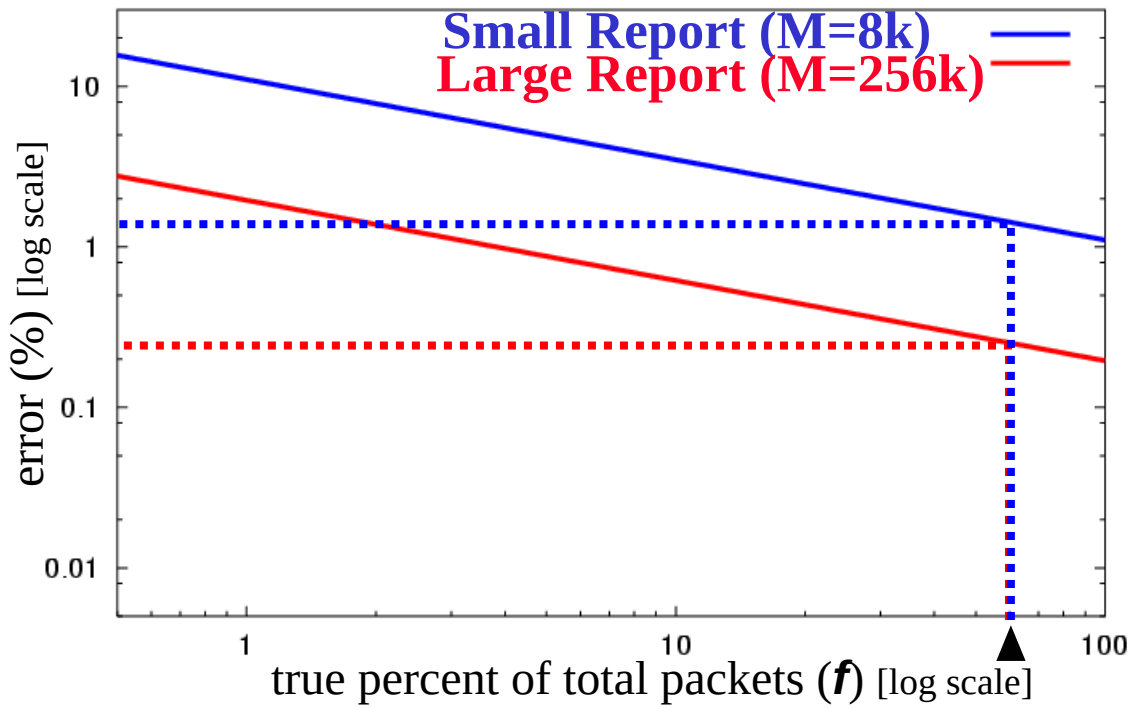
- Chose new sampling rate to leave M flow records after renormalization
- If traffic slows, the sampling rate is not too low
  - The M flow records accurately describe the traffic
- If traffic increases, the sampling rate is not too high
  - Renormalization frees space faster than new entries appear
  - Each time that the sampling rate is reduced, the worst case rate of new entries decreases

Adaptive NetFlow guarantees

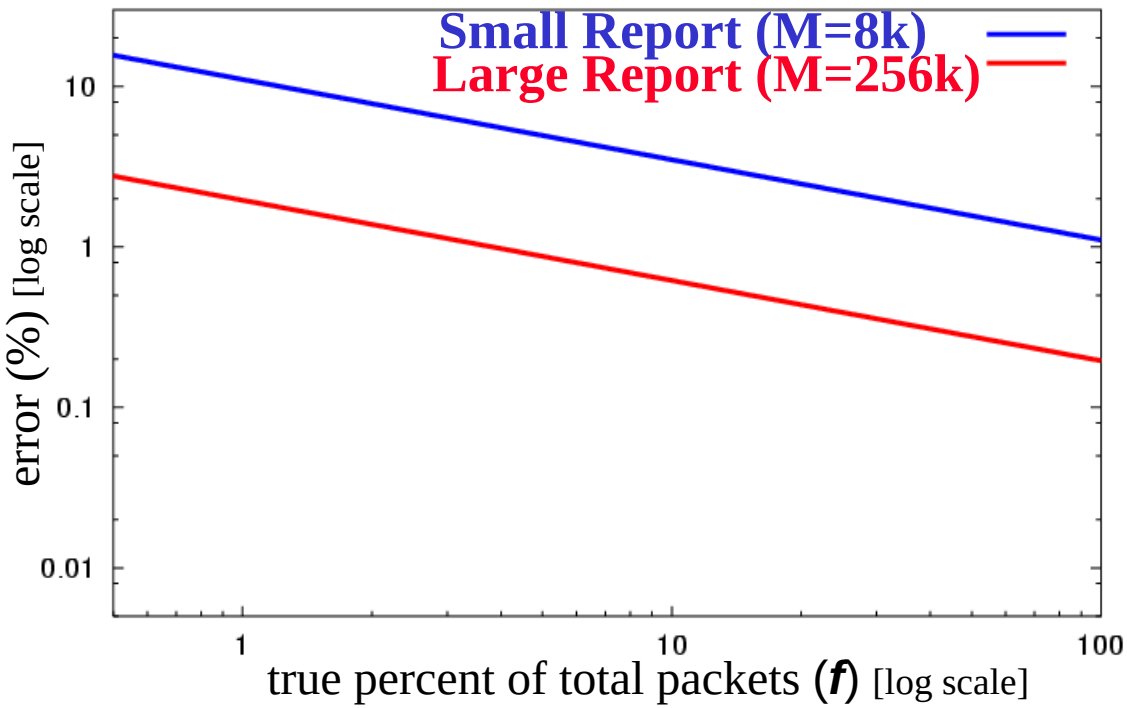
If ANF generates ***M*** entries, the relative standard deviation for an aggregate that is fraction ***f*** of the traffic is at most $\sqrt{1/(Mf)}$ in packets and $\sqrt{s_{max}/(s_{avg}Mf)}$ in bytes (for any the traffic mix).

Example:

If HTTP traffic represents 60% of the actual total packets,
then the expected error is less than 2% for M=8k,
and less than 0.3% for M=256k.

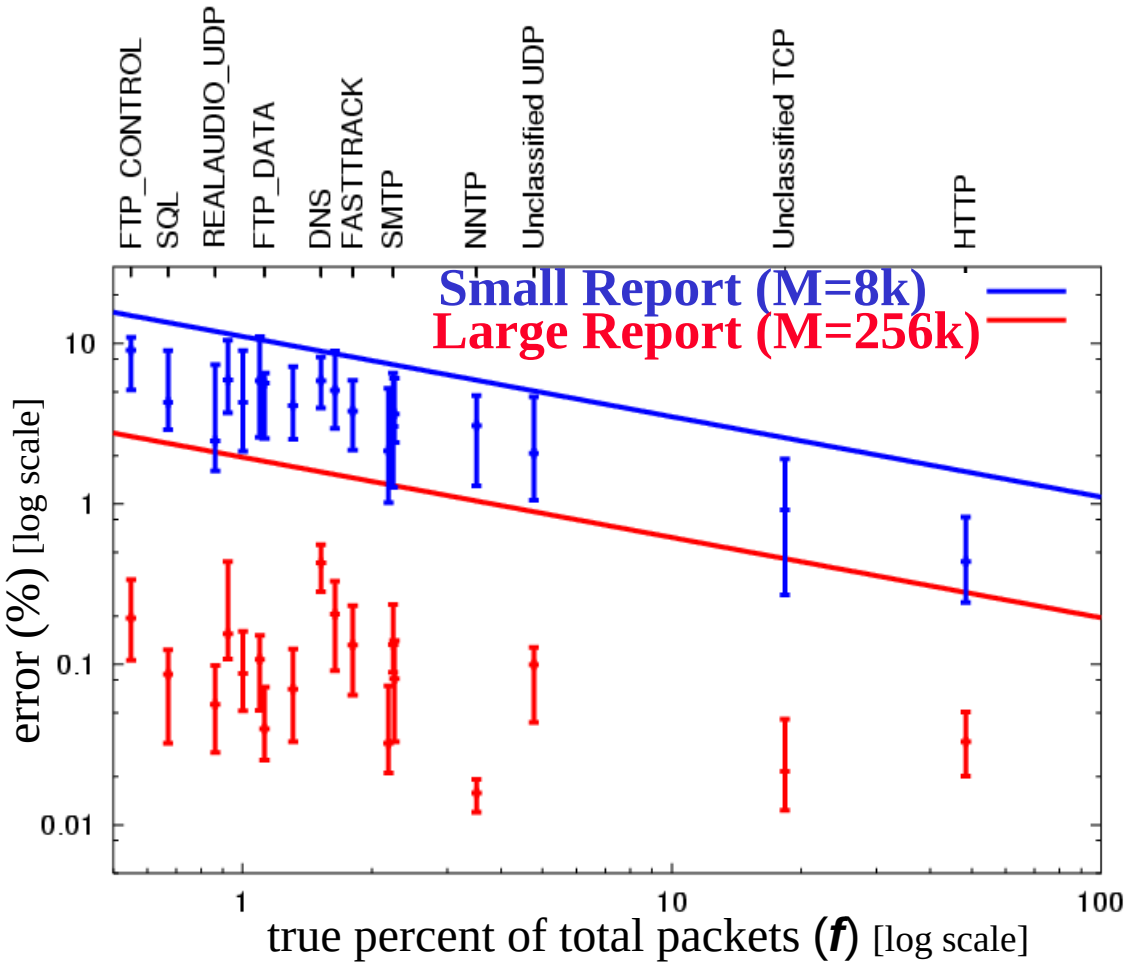


Adaptive NetFlow results



Adaptive NetFlow results

- Measured error is significantly better than theoretical results for normal traffic mixes.



Fixes for NetFlow

NetFlow problem	How we solve it
Mismatch between flow report and desired analysis	Measurement bins (part 1)
Memory and bandwidth usage strongly depend on traffic mix	Adapting sampling rate (part 2)
Admin must set sampling rate	
Cannot count non-TCP flows	**In paper**

Conclusions

- Binned measurement better matches analysis
- Adaptive NetFlow improves NetFlow
  - Guaranteed accuracy under any traffic mix
  - Graceful response to adverse traffic
  - More meaningful tuning knob: # of records *M*
- Flow Counting Extension
  - Accurate flow counts for both TCP and non-TCP flows
  - Hardware extension required
- Impact
  - Changes to IPFIX RFC drafts
  - Cisco already expressed interest
  - Software implementations

Questions?

Extended Technical Report

- http://www.caida.org/outreach/papers/2004/tr-2004-03/