

Measuring Network Traffic

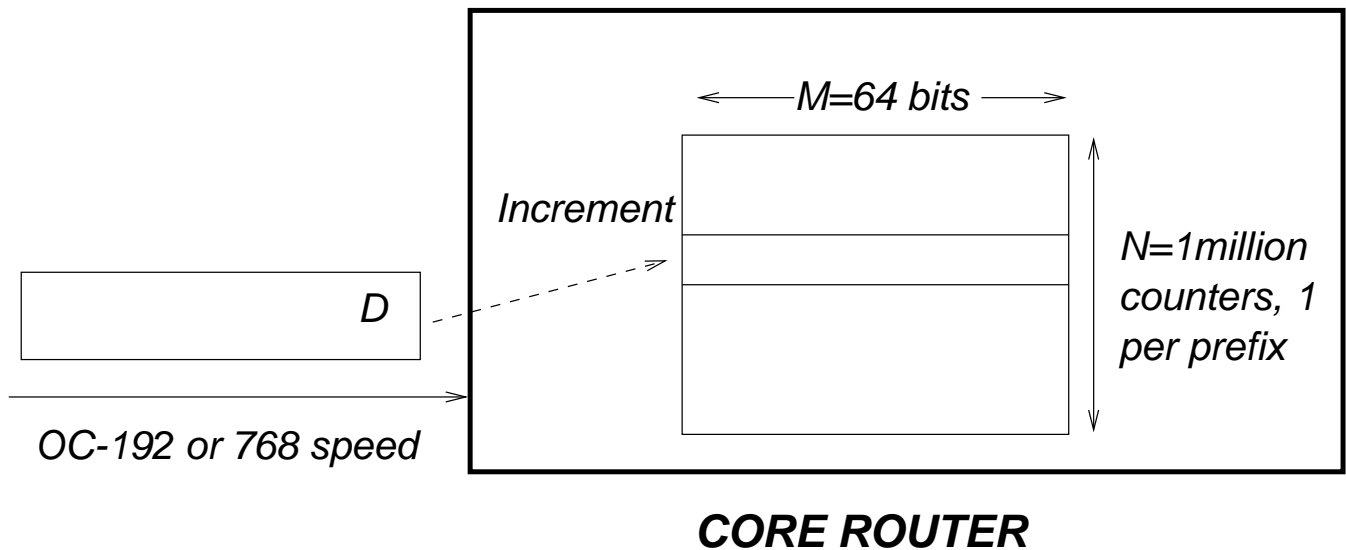
*Not everything that is counted counts, and
not everything that counts can be counted.*

— Albert Einstein.

Measurement Outline

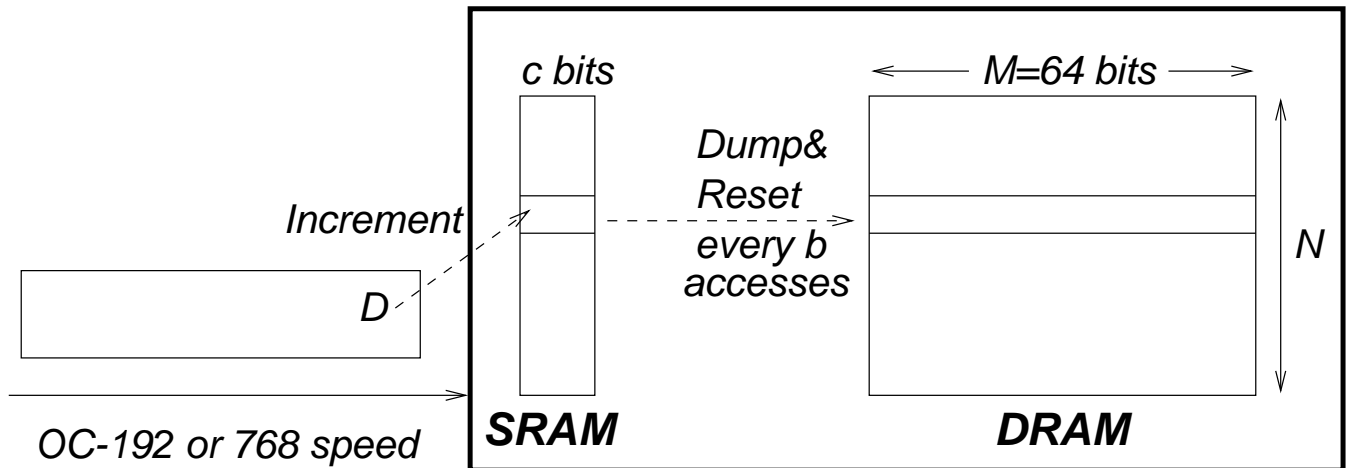
- *16.2* Reducing SRAM Counter width using DRAM backing store.
- *16.3* Reducing Counter widths using Randomized Counting.
- *16.4* Reducing number of counters using Threshold Aggregation.
- *16.5* Reducing number of counters using Flow Counting.
- *16.6* Reducing packet processing using Sampled NetFlow
- *16.7* Reducing NetFlow reporting using Sampled Charging.
- *16.8* Correlating measurements using Trajectory Sampling.

16.1 Counting Problem



- Multiple counters per packet with possibly several high speed links per memory requires SRAM for speed.
- Large number of counters (e.g., per prefix or per queue) plus high bit width (i.e., to prevent overflow) requires large amounts of SRAM. Expensive (off-chip) or infeasible (on-chip).

16.2 Reducing SRAM Counter width via DRAM backing store



- *Shah et al*: Store M bit counters in slow DRAM together with $c \ll M$ bit wide SRAM counters. Dump largest SRAM counter every b accesses.

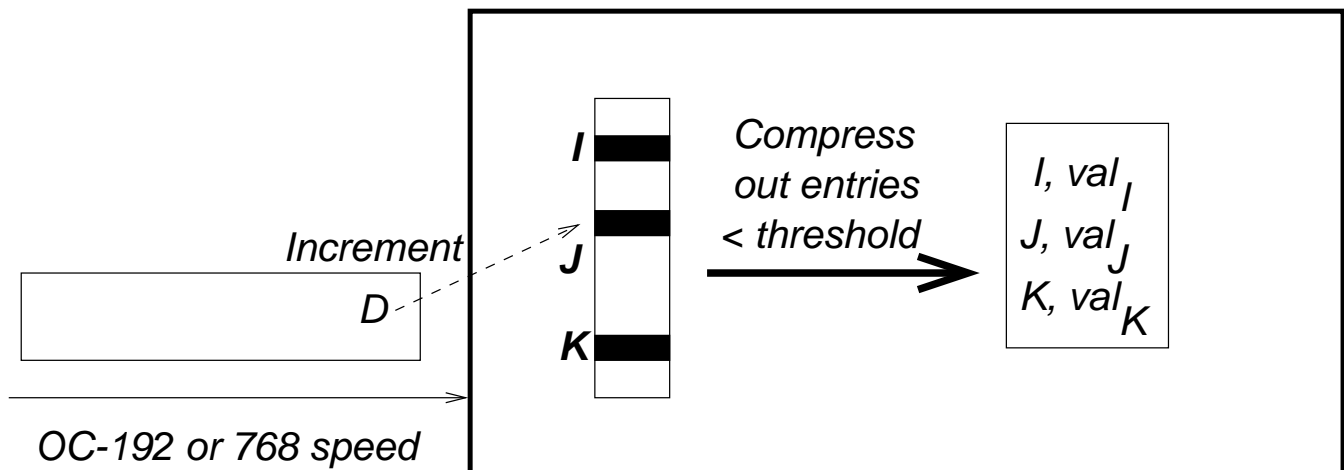
$$2^c \approx \frac{\log_b(N-1)}{\log(b/b-1)}$$

- 3 counters every 8 nsec (OC-768) for $N = 10^6$ requires 8Mbit of 2.5 usec SRAM and 51.2 usec DRAM. But how to find largest counter?

16.3 Reducing Counter width via Randomized Counting

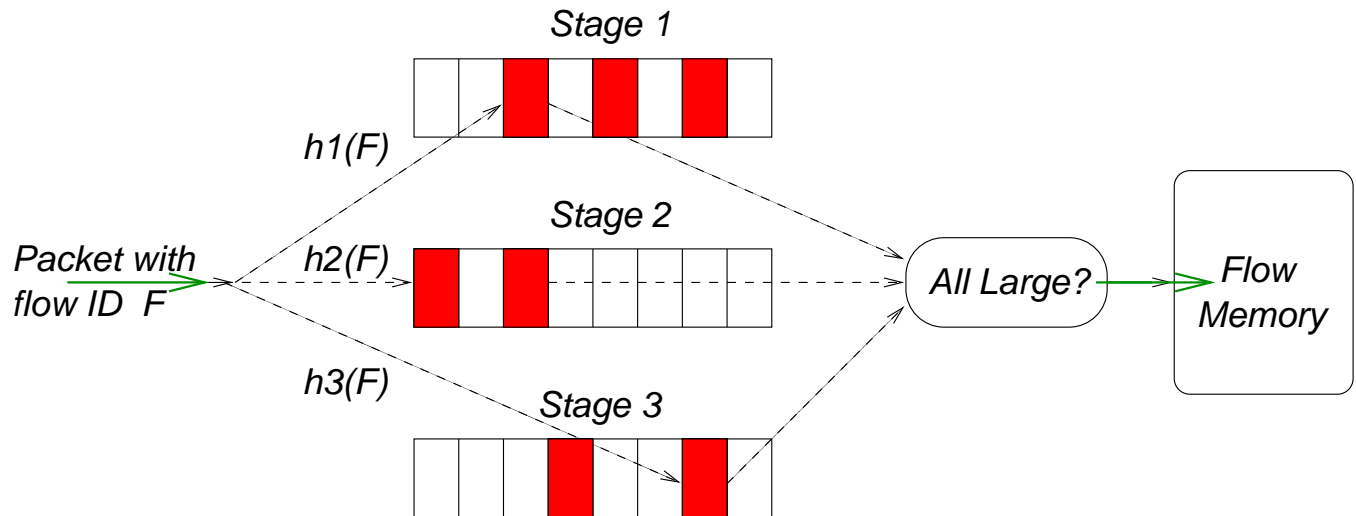
- **Base Idea:** If we increment b bit counter only with probability $1/c$, when counter saturates expected number of counted events is $2^b \cdot c$.
- **Further Opportunity:** In base idea, standard deviation is a few c 's which is small at high counter values $\gg c$. R. Morris's idea: increment counter with *non-constant probability that depends on counter value* so expected error scales with counter size.
- **Algorithm:** Increment counter with probability $1/2^x$ where x is value of counter; counter value of x represents expected value of 2^x .
- **Problems:** High standard deviation, equal to estimate size. Also, hard to pick accurate small random numbers.

16.4 Reducing number of counters using Threshold Aggregation



- *Estan-Varghese*: For applications that only want counters above a threshold (e.g., threshold accounting, hot spot detection) can reduce SRAM height.
- **Problem**: How to detect heavy-hitters above threshold with memory proportional to number above threshold?

Threshold Compression via Multiple Hashed Counters



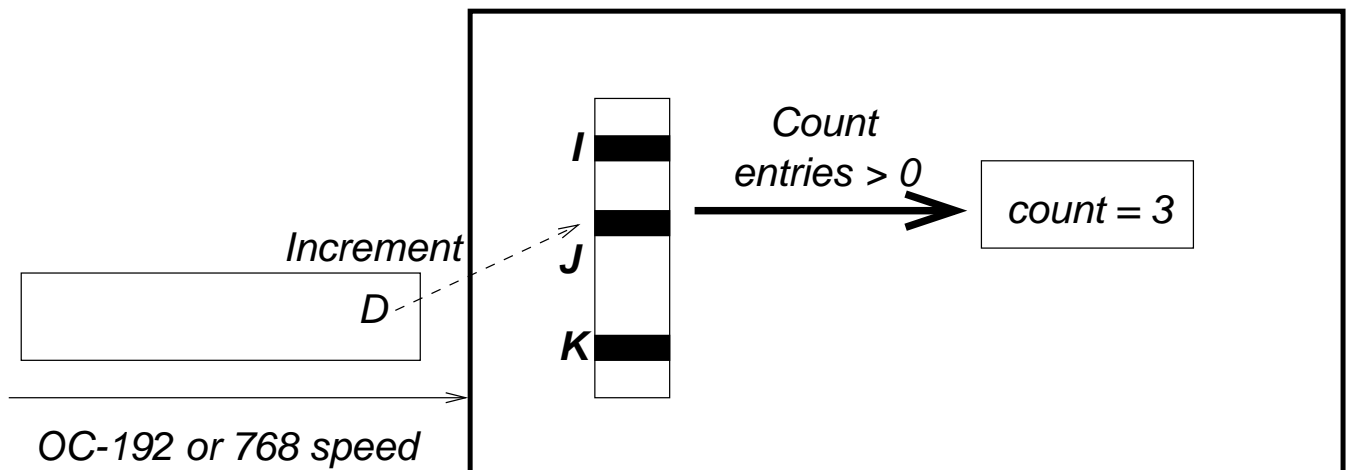
- No false negatives but can cause false positives if a small flow hashes onto several “heavy” buckets.

Analysis of Multiple Hashing

Assume 1 percent threshold. Bound probability that a flow F of 0.1% or less gets through 6 stages of size 1000 each.

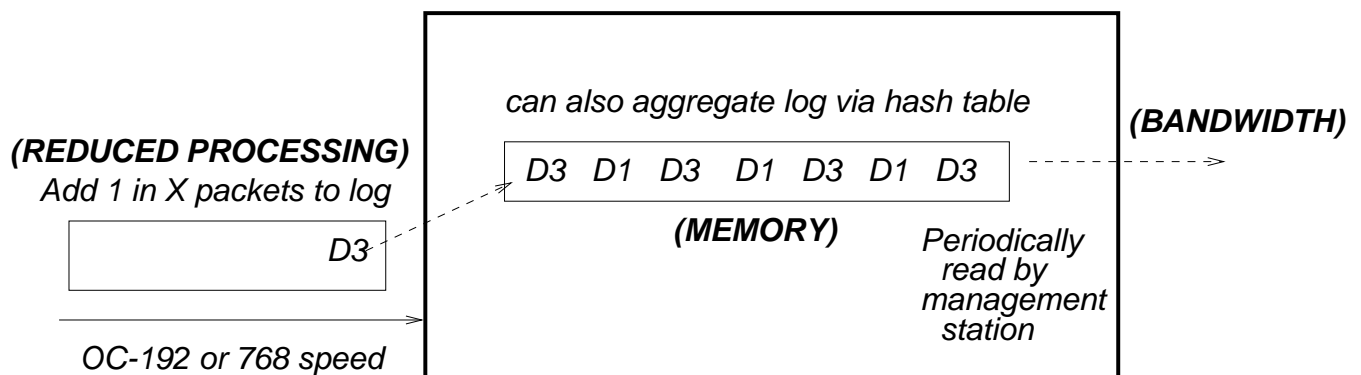
- *Why trouble?*: F can fall into a “hot” bucket if and only the sum of traffic of all other flows in that bucket is more than 0.9%
- *Single stage probability*: At most $100/0.9 = 111$ buckets that can be over 0.9% before we bring on F . Thus probability F falls in a “hot” bucket is less than $111/1000 = 0.111$
- *Overall False negatives*: To be branded, F must be unlucky in all 6 stages with a probability of no more than $(0.111)^6$ which is very small. Thus at most 1000 false positives with very high probability.

16.5 Reducing number of counters using Flow Counting



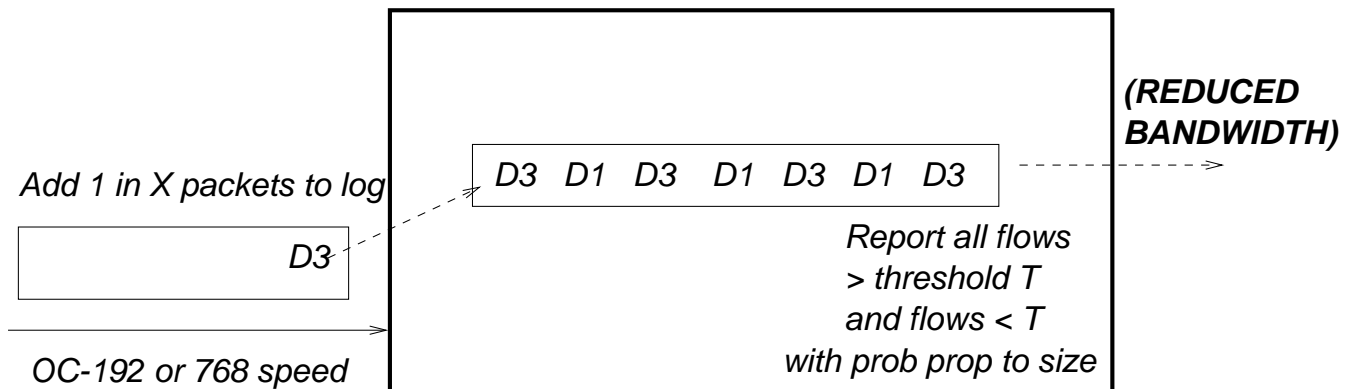
- Many applications only require an active flow count. Snort IDS port scan plugin counts destinations sent *per source*
- Classical solution (*Flajolet-Martin*): compute X , highest number of trailing zeroes in hash of all flow IDs; Return 2^X . Improve accuracy by averaging X over trials with independent hashes.

16.6 Reducing packet processing using Sampled NetFlow



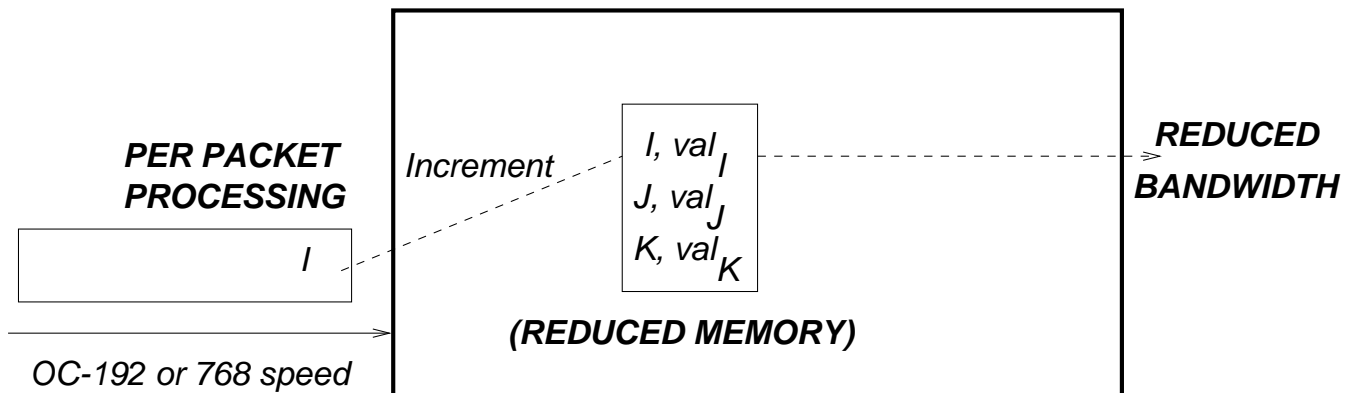
- Besides say per prefix counters, may want to log TCP microflows for later analysis (charging, security etc).
- **Problems:** Massive DRAM logs and bandwidth to send logs to Management station.

16.7 Reducing NetFlow reporting using Sampled Charging



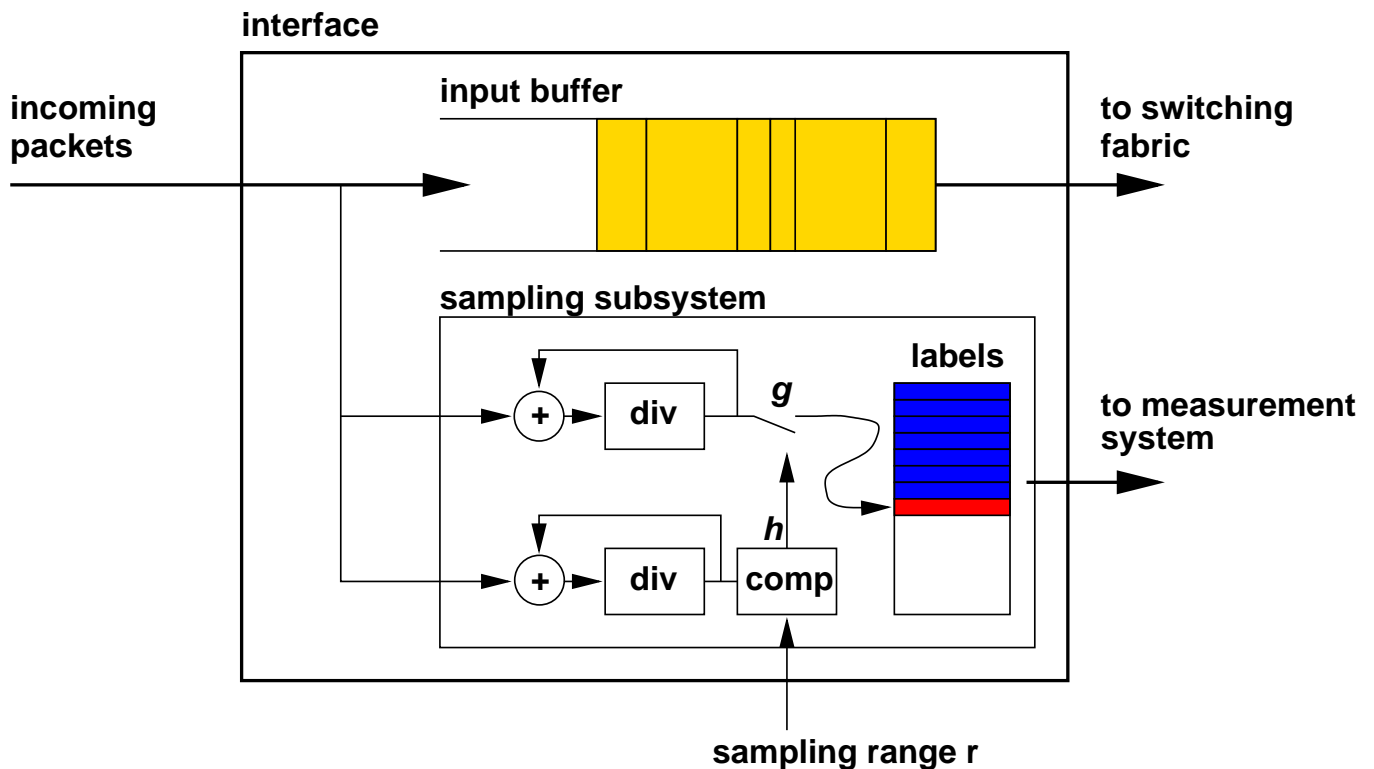
- *Duffield, Lund, Thorup*: Reduced bandwidth but not size of DRAM log, via sampled transmission of DRAM log.

Reducing NetFlow Log via Threshold Compression



- **Tradeoff:** Requires \sqrt{M} memory where M is memory required by NetFlow for same relative error, but requires per packet processing; processing feasible using small SRAM.
- **Problem:** Less flexible than NetFlow and SampledCharging.

16.8 Correlating measurements using Trajectory Sampling



- Correlates flows on different links using a consistent decision to sample a packet at each router, and by using a consistent label for each packet. 2 hash implementation.
- Unlike NetFlow which uses independent sampling decisions. Thus can see effects like packet looping, packet drops, multiple shortest paths.