CS 559: Algorithmic Aspects of Computer Networks		Fall 2007
Lecture 10 — October 9		
Lecturer: John Byers	*BOSTON UNIVERSITY*	*Scribe: Michel Machado*

Today's lecture covered Estan-Varghese paper: "New directions in Traffic Measurement and Accounting." This paper purposes two measurement techniques that do much better than Cisco's sampled NetFlow to track "elephants" (heavy flows), namely, "Sample and Hold" and "Parallel Filter."

10.1 The problem

Tracking flows over a measurement epoch is necessary for auditing, billing, anomaly detection (i.e. hot spots and denial-of-service attack), provisioning, and finding problems in a network.

However, this problem becomes a heavy burden in the presence of routers connected to very high speed links because to accurately track all of flows in this context requires a huge amount of high-expensive, high-speed memory (SRAM). If cheaper, slower memory (DRAM) were used to track all flows, the link would not be fully used since significant amount of time would be spent on handling the memory.

By the time the paper was written, it is known that a small number of flows consumes most of the links' capacity. These large flows were named "elephants" and strictly defined in the paper as a flow consuming at least something like 0.1% or 1% of a given link capacity.

Once the definition of flow is application dependent, any function that computes an ID, the flow ID, over a given packet is enough to define flow. A natural definition is the tuple $< IP_{source}, Port_{source}, IP_{destination}, Port_{destination} >$, but any other pattern easily computable at packet level is valid. Wild cards may also be used to create aggregated flows.

Cisco's sampled NetFlow was the state of the art when this paper was first published in 2001. NetFlow randomly samples packets at rate M/T where M is the amount of memory allocated to hold NetFlow's accounting data and T the epoch time. More precisely, M is the number of flows NetFlow can track within an epoch. NetFlow stores its accounting data in DRAM; It is possible because NetFlow samples only a small fraction of the packets.

Problems of NetFlow's approach:

- Even with infinite memory, NetFlow can not precisely track all flows because DRAM is too slow (it would limit the link capacity). Using SRAM is not an option since small amount of SRAM is expensive.
- It produces many false positives, that is, many "mice" (non-elephants) are counted.
- It produces false negatives, that is, elephants are missed.
- The accuracy of the estimates are poor, counters may differ dramatically from actual value.

- Large reports have to be transfered from the routers to a management station. Since DRAM is cheap, M is often large to improve accuracy.

The reader should note that NetFlow and the two techniques described on the next sections are designed to run in line in the routers. It is different from IP Back Track paper that purposes a not in line solution.

10.2 Sample and Hold

Sample and Hold limits the amount of used memory to fits all its accounting data (flow memory) in SRAM. It randomly samples like NetFlow, but after a flow is identified by sampling all following traffic from that flows is counted. Thus, when a packet arrives, it checks if the packet belongs to a flow that has been already seen, if so, the respective counters are updated. Otherwise, it decides or not to sample that packet with a probability p.

In order to simplify the following explanation of the method, some concrete numbers are used (they may be easily changed to variables if wished).

Defining elephants as flows that take at least 1% of the link, there are at the most 100 elephants. Instead of fixing the flow memory with just 100 entries, sample-and-hold technique allows oversampling by a factor of 100. Thus, $M = 100 * 100 = 10,000$.

If C bytes can be transmitted in the measurement interval, $p = 10,000/C$. An elephant sends at least 1% of C, that is $100/C$ bytes. Therefore, the probability of sample-and-hold method misses an elephant is as follow:

$$Pr[\text{miss elephant}] = (1-p)^{(elephant\_size)} \tag{10.1}$$

Considering the smallest possible elephant, this probability is at most:

$$Pr[\text{miss elephant}] = \left(1 - \frac{10,000}{C}\right)^{\frac{C}{100}} \approx e^{-\frac{10,000}{C}\cdot\frac{C}{100}} = e^{-100} \tag{10.2}$$

The previous approximation can be made because C is large (this technique is designed to very high speed links).

Using the same analysis, the probability of missing an elephant after sending 5% of its traffic is at most e^{-5}, that is less than 1%.

One should notice that the previous analysis does not count packets, but bytes. Moreover, discarding flows that do not classify as an elephant at the end of the epoch avoids reporting false positives at all.

10.3 Parallel Filter

Parallel Filter also limits the amount of used memory to fits all its data structures in SRAM. However, its memory is divided between a counting bloom filter and accounting data.

The flow ID is hashed with every hash function and each hashed counter is updated by packet size. If all counters are bigger than the definition of elephants, it is copied to the flow memory with the smallest number in the hashed counters.