Research statement for Cristian Estan

One of the main reasons for the success of the Internet is its service model that emphasizes flexibility: any computer can send any type of data to any other computer at any time. While this freedom enabled the unhindered deployment of the applications popular today such as email and the World Wide Web, it also made the job of administering the network harder. In the phone network, exhaustive call logs are readily available, whereas the Internet lacks built-in traffic measurement features that would help the operators figure out how the network is used or misused. Existing solutions either lack the necessary detail, do not scale up to the speeds of today's networks, fail to extract the meaningful information from raw data or are not flexible enough to keep up with the ever changing traffic mix. The goal of my research at UCSD was to **build a scientific foundation** for systems relying on **traffic measurement and analysis in the Internet**.

There is a wide variety of systems solving various traffic measurement problems, yet all of them share a common structure: at one end they have a module observing the packets in the network and at the other the final module reports some higher level results to a human operator or another application. Between the two ends, there are a varying number of modules that process the data gradually reducing its size and raising its level of abstraction. I adopted the approach of identifying **building blocks** common to many measurement systems and finding good **algorithmic solutions** for them.

The **constraints** these building blocks operate under can be **severe**. For example a router module working with an OC-768 link (40 gigabits/second) has 8 nanoseconds to process a minimum sized packet, thus operations must be implemented in hardware, using expensive SRAM memory limited in size. Therefore, algorithmic solutions must have low worst-case bounds on the per-packet processing and memory usage.

Identifying heavy hitters

Many systems are interested in the "heavy hitters" sending large amounts of traffic. Knowing the pairs of networks exchanging most traffic helps ISPs decide where to add capacity to their network, knowing the destinations receiving most traffic can help identify victims of flooding attacks, knowing which customers generate most traffic during times of congestion can be important for billing and accounting. Typically we can afford enough memory to track the heavy hitters, but we need a mechanism that decides which of the millions of active flows is large and thus worth tracking. In ***[EV02]*** and ***[EV03]*** we propose and evaluate two algorithmic solutions for identifying heavy hitters.

For each packet, with a small probability, **sample and hold** allocates an entry to the flow the packet belongs to and once the entry is created all packets belonging to the flow are counted. A simple analysis shows that the algorithm is very likely to detect the flows with many packets while the number of entries used likely stays below a limit. Since the worst-case number of entries created is much larger than what we observed on typical traffic mixes, we use a simple adaptive feedback loop to find the highest sampling probability at which the number of entries created is safely within the available memory. Analysis shows that sample and hold is more accurate than taking a random sample of the

traffic and extrapolating: accuracy increases linearly with the available memory instead of increasing with its square root.

The second solution improves accuracy further at the cost of a slight increase in implementation and configuration complexity. A **multistage filter** enforces that all flows whose size is above a threshold are tracked while for flows with few packets the probability of being tracked is very low. Flows hash to a small table of counters. Usually more than one flow will hash to a counter. The algorithm allocates per-flow entries only for packets that hash to counters that reach the threshold. Flows above the threshold are guaranteed to be noticed while flows much below it can get an entry only if the counters they hash to are close to the threshold due to the traffic of other flows. We can further decrease this probability by using multiple stages of counters using independent hash functions operating in parallel. A large flow will push all the counters it hashes to above the threshold while the probability for small flows to hash to large counters at all stages (and thus consume an entry) decreases exponentially with the number of stages. **Conservative update** is a simple optimization to the multistage filter that improves its performance significantly. When a packet hashes to counters with different sizes, the smallest of them is an upper bound on the number of packets the flow sent so far. Even if we only increment the smallest counter(s), we maintain this invariant. Therefore, even with conservative update, large flows are guaranteed to push all their counters to the threshold. On traffic traces, multistage filters with conservative update give results up to 7 times more accurate than sample and hold using the same amount of memory.

Counting active flows

Internet traffic is most often measured in packets or bytes, yet often measuring the traffic in flows (a flow is roughly equivalent to one conversation), is useful. Counting flows is fundamentally different from counting bytes or packets: it is algorithmically equivalent to other problems that we need to solve often such as **counting the number of distinct** source and destination addresses in the traffic mix. For example one can detect network scans (malicious hackers looking for vulnerabilities to exploit) by finding in the traffic mix the foreign addresses with many flows. One can distinguish between a computer receiving large amounts of legitimate traffic and a victim of a flooding attack by looking at the number of distinct sources of traffic (attackers forge source addresses at random to make it more difficult to filter out attacks).

In ***[EVF03]*** we present **a family of bitmap algorithms for counting active flows** that can be implemented at the highest speeds in computer networks. The core idea behind these algorithms is to map flow identifiers to bits in a bitmap using a hash function. The number and position of bits set is used to compute an estimate of the number of active flows. By using various mappings from flow identifiers to bits and by varying other details, we derive a family of algorithms that exploit the idiosyncrasies of various traffic measurement settings. Compared to prior algorithms, using two kilobytes of memory, adaptive bitmap gives results four times more accurate. The publicly available **bmpcount** library implements the entire family of bitmap algorithms.

Traffic clusters

When analyzing the traffic one is often interested in the “heavy hitters”, but the definition of how we group packets into aggregates varies depending on what we are

looking for. When looking for flooding attacks we group traffic into aggregates by destination address, when looking for dominant applications we group it by protocol and source or destination port, when looking at the traffic matrix, we group it by source and destination network. In ***[ESV03]*** we propose reports based on traffic clusters as a method for presenting to the network administrator **a single, combined view** that summarizes concisely the traffic mix across all relevant dimensions for aggregation. The report gives the traffic of clusters above a certain threshold. A range for each important packet header field defines each traffic cluster. The ranges are from natural hierarchies associated with the respective fields (e.g. the IP prefix hierarchy contains individual addresses, networks of various sizes up to the entire address space). Listing all large traffic clusters above a threshold generates much redundant data. We can reduce the size of the reports by orders of magnitude by applying the **compression** rule: omit from the report the clusters whose traffic can be inferred accurately enough from more specific clusters in the report.

AutoFocus is a network traffic analysis application using multidimensional traffic cluster analysis with traffic measured as packets, bytes and flows. Feedback from network administrators using it in production networks is extremely favorable.

The method underlying traffic cluster analysis can also be applied to other data. The algorithmic core of AutoFocus was applied successfully to a log of ebay auctions and I am currently working on applying traffic cluster analysis to BGP routing updates.

Future directions

The two areas I am particularly interested working on in the near future are building robust systems to improve **network security** using scalable traffic measurement building blocks and **extending traffic cluster analysis**. An example of the first area is the EarlyBird project that uses traffic measurement to recognize unknown worms based on their traffic patterns. By not requiring human involvement for triggering the countermeasures, the system might be effective at stopping worm epidemics before they compromise a significant fraction of the vulnerable computers.

There are many ways I would like to extend traffic cluster analysis beyond the obvious extension of applying it to datasets other than network traffic. Extending it so that it is aware of time and network topology is one. Using it to give a view of the traffic of the whole network instead of just one vantage point is another. A closer look at the algorithmic underpinnings of traffic cluster analysis shows that the problem is related to the well-studied **data cube** used in On-Line Analytical Processing in the database context and **association rules** used in data mining. The differences among the three settings lead to distinct solutions, but I believe that relating traffic cluster analysis to the bigger picture can lead to exciting new discoveries with applications beyond computer networking.

The Internet is a complex mix of technologies, devices, standards, protocols and people with wildly differing capabilities and often diverging interests. As the network grows and new uses arise, its very open and permissive service model keeps surprising us with the complexity of the interactions it makes possible and often with the unintended consequences of earlier decisions. Insufficient support for traffic measurement is one of these undesirable consequences, worms, denial of service attacks and spam are others, and I am convinced more will arise. It is my **long-term goal** to find solutions that counteract these unintended consequences – where possible by incremental additions where not, by fundamental changes to the Internet service model.

[EV02] Cristian Estan and George Varghese, "New Directions in Traffic Measurement and Accounting", *SIGCOMM, August 2002*

[EV03] Cristian Estan and George Varghese, "New Directions in Traffic Measurement and Accounting: Focusing on the Elephants, Ignoring the Mice", *ACM Transactions on Computer Systems, August 2003*

[ESV03] Cristian Estan, Stefan Savage and George Varghese, "Automatically Inferring Patterns of Resource Consumption in Network Traffic", *SIGCOMM, August 2003*

[EVF03] Cristian Estan, George Varghese and Mike Fisk, "Bitmap Algorithms for Counting Active Flows on High Speed Links", *Internet Measurement Conference, October 2003*