Network Data Streaming – A Computer Scientist’s Journey in Signal Processing

Jun (Jim) Xu
Networking and Telecommunications Group
College of Computing
Georgia Institute of Technology

Joint work with:
Abhishek Kumar, Qi Zhao, Minho Sung, Jun Li, Ellen Zegura, Georgia Tech.
Jia Wang, Olivier Spatscheck, AT&T Labs - Research
Li Li, Bell Labs

Networking area Qualifying Exam: Oral Presentation
• Motivation and introduction

• Our six representative data streaming works
  – 3 in Single-node single-stream data streaming (like SISD)
  – 1 in Distributed Collaborative Data Streaming (like SIMD)
  – 2 in Distributed Coordinated Data Streaming (like MIMD)
Motivation for network data streaming

Problem: to monitor network links for quantities such as

- Elephant flows (traffic engineering, billing)
- Number of distinct flows, average flow size (queue management)
- Flow size distribution (anomaly detection)
- Per-flow traffic volume (anomaly detection)
- Entropy of the traffic (anomaly detection)
- Other “unlikely” applications: traffic matrix estimation, P2P routing, IP traceback
The challenge of high-speed monitoring

- Monitoring at high speed is challenging
  - packets arrive every 25ns on a 40 Gbps (OC-768) link
  - has to use SRAM for per-packet processing
  - per-flow state too large to fit into SRAM

- Traditional solution using sampling:
  - Sample a small percentage of packets
  - Process these packets using per-flow state stored in slow memory (DRAM)
  - Using some type of scaling to recover the original statistics, hence high inaccuracy with low sampling rate
  - Fighting a losing cause: higher link speed requires lower sampling rate
Network data streaming – a smarter solution

- **Computational model:** process a long stream of data (packets) in one pass using a small (yet fast) memory

- **Problem to solve:** need to answer some queries about the stream at the end or continuously

- **Trick:** try to remember the most important information about the stream *pertinent to the queries* – learn to forget unimportant things

- **Comparison with sampling:** streaming peruses every piece of data for most important information while sampling digests a small percentage of data and absorbs all information therein.
The “hello world” data streaming problem

- Given a long stream of data (say packets), count the number of distinct elements in it
- Say in a, b, c, a, c, b, d, a – this number is 4
- Think about trillions of packets belonging to billions of flows...
- A simple algorithm: choose a hash function $h$ with range $(0,1)$
  - $\hat{X} := 1/min(h(d_1), h(d_2), ...)$
- We can prove $\hat{X}$ is an unbiased estimator
- Then average hundreds of such $X$ up to get an accurate result
**Problem:** To estimate the probability distribution of flow sizes. In other words, for each positive integer $i$, estimate $n_i$, the number of flows of size $i$.

**Applications:** Traffic characterization and engineering, network billing/accounting, anomaly detection, etc.

**Importance:** The mother of many other flow statistics such as average flow size (first moment) and flow entropy.

**Definition of a flow:** All packets with the same flow-label. The flow-label can be defined as any combination of fields from the IP header, e.g., 

$<$Source IP, source Port, Dest. IP, Dest. Port, Protocol$>$.

**Existing sampling-based work is not very accurate.**
Our approach: network data streaming

- **Design philosophy**: “Lossy data structure + Bayesian statistics = Accurate streaming”
  - Information loss is unavoidable: (1) memory very small compared to the data stream (2) too little time to put data into the “right place”
  - Control the loss so that Bayesian statistical techniques such as Maximum Likelihood Estimation can still recover a decent amount of information.
Architectural Overview — Lossy data structure

- Measurement proceeds in epochs (e.g. 100 seconds).
- Maintain an array of counters in fast memory (SRAM).
- For each packet, a counter is chosen via hashing, and incremented.
- No attempt to detect or resolve collisions.
- Each 32-bit counter only uses 9-bit of SRAM (due to [Ramabhadran & Varghese 2003]).
- Data collection is lossy (erroneous), but very fast.
The distribution of flow sizes and raw counter values (both $x$ and $y$ axes are in log-scale). $m = \textit{number of counters}$.
Estimating $n$ and $n_1$

- Let total number of counters be $m$.
- Let the number of value-0 counters be $m_0$.
- Then $\hat{n} = m \times \ln(m/m_0)$.
- Let the number of value-1 counters be $y_1$.
- Then $\hat{n}_1 = y_1 e^{\hat{n}/m}$.
- Generalizing this process to estimate $n_2$, $n_3$, and the whole flow size distribution will not work.
- Solution: joint estimation using Expectation Maximization.
Estimating the entire distribution, $\phi$, using EM

- Begin with a guess of the flow distribution, $\phi^{ini}$.
- Based on this $\phi^{ini}$, compute the various possible ways of “splitting” a particular counter value and the respective probabilities of such events.
- This allows us to compute a refined estimate of the flow distribution $\phi^{new}$.
- Repeating this multiple times allows the estimate to converge to a local maximum.
- This is an instance of Expectation maximization.
Estimating the entire flow distribution — an example

- For example, a counter value of 3 could be caused by three events:
  - $3 = 3$ (no hash collision);
  - $3 = 1 + 2$ (a flow of size 1 colliding with a flow of size 2);
  - $3 = 1 + 1 + 1$ (three flows of size 1 hashed to the same location)
- Suppose the respective probabilities of these three events are 0.5, 0.3, and 0.2 respectively, and there are 1000 counters with value 3.
- Then we estimate that 500, 300, and 200 counters split in the three above ways, respectively.
- So we credit $300 \times 1 + 200 \times 3 = 900$ to $n_1$, the count of size 1 flows, and credit 300 and 500 to $n_2$ and $n_3$, respectively.
Evaluation — Before and after running the Estimation algorithm

![Graph showing actual flow distribution, raw counter values, and estimation using the algorithm over flow size and frequency axes.](image-url)
Sampling vs. array of counters – Web traffic.

![Graph showing flow distribution and inference methods](image)
Sampling vs. array of counters – DNS traffic.
Extending the work to estimating subpopulation FSD [Sigmetrics 05]

- Motivation: there is often a need to estimate the FSD of a subpopulation (e.g., “what is FSD of all the DNS traffic”).
- Definitions of subpopulation not known in advance and there can be a large number of potential subpopulations.
- Our scheme can estimate the FSD of any subpopulation defined after data collection.
- Main idea: perform both streaming and sampling, and then correlate these two outputs.
Space Code Bloom Filter for per-flow measurement [IMC03, Infocom04]

**Problem:** To keep track of the total number of packets belonging to each flow at a high speed link.

**Applications:** Network billing and anomaly detection

**Challenges:** traditional techniques such as sampling will not work at high link speed.

**Our solution:** SCBF encodes the frequency of elements in a multiset like BF encodes the existence of elements in a set.
Distributed Collaborative Data Streaming

1. Update Packet stream
2. Transmit digest
3. Compose bitmap to stations
4. Analyze & sound alarm

Central Monitoring Station

Feedback to stations

Valued customers
**Traffic matrix** quantifies the traffic volume between origin/destination pairs in a network.

Accurate estimation of traffic matrix $T_{i,j}$ in a high-speed network is very challenging.

Our solution based on distributed collaborative data streaming:

- Each ingress/egress node maintains a synopsis data structure (cost $< 1$ bit per packet).
- Correlating data structures generated by nodes $i$ and $j$ allow us to obtain $T_{i,j}$.
- Average accuracy around 1%, which is about one order of magnitude better than the current approaches.
Distributed coordinated data streaming – a new paradigm

• A network of streaming nodes
• Every node is both a producer and a consumer of data streams
• Every node exchanges data with neighbors, “streams” the data received, and passes it on further
• We applied this kind of data streaming to two unlikely network applications: (1) P2P routing [Infocom05] and (2) IP traceback [IEEE S&P04].
Large-Scale IP Traceback in High-Speed Internet

Jun (Jim) Xu
Networking & Telecommunications Group
College of Computing
Georgia Institute of Technology

(Joint work with Jun Li, Minho Sung, Li Li)
Introduction

- Internet DDoS attack is an ongoing threat
  - on websites: Yahoo, CNN, Amazon, eBay, etc (Feb. 2000)
  - on Internet infrastructure: 13 root DNS servers (Oct, 2002)
- It is hard to identify attackers due to IP spoofing
- IP Traceback: trace the attack sources despite spoofing
- Two main types of proposed traceback techniques
  - Probabilistic Packet Marking schemes: routers put stamps into packets, and victim reconstructs attack paths from these stamps [Savage et. Al. 00] …… [Goodrich 02]
  - Hash-based traceback: routers store bloom filter digests of packets, and victim query these digests recursively to find the attack path [Snoeren et. al. 01]
Scalability Problems of Two Approaches

• Traceback needs to be scalable
  – When there are a large number of attackers, and
  – When the link speeds are high

• PPM is good for high-link speed, but cannot scale to large number of attackers [Goodrich 01]

• Hash-based scheme can scale to large number of attackers, but hard to scale to very high-link speed

• Our objective: design a traceback scheme that is scalable in both aspects above.
Design Overview

• **Our idea:** same as hash-based, but store bloom filter digests of sampled packets only
  - Use small sampling rate $p$ (such as 3.3%)
  - Small storage and computational cost
  - Scale to 10 Gbps or 40 Gbps link speeds
  - Operate within the DRAM speed

• **the challenge of the sampling**
  - Need many more packets for traceback
  - Independent random sampling will not work: need to improve the “correlation factor”
Overview of our hash-based traceback scheme

• Each router stores the bloom filter digests of sampled packets
• Neighboring routers compare with each other the digests of the packets they store for the traceback to proceed
  – Say P is an attack packet, then if you see P and I also see P, then P comes from me to you …
• When correlation is small, the probability that both see something in common is small
One-bit Random Marking and Sampling (ORMS)

- ORMS make correlation factor be larger than 50%
- ORMS uses only one-bit for coordinating the sampling among the neighboring routers

\[
\text{correlation: } \frac{p}{2} + \frac{p}{2} \cdot \frac{p}{2 - p} = \frac{p}{2 - p}
\]

\[
\text{total sampling probability: } \frac{p}{2} + \left(1 - \frac{p}{2}\right) \cdot \frac{p}{2 - p} = p
\]

\[
\text{correlation factor (sampled by both): } \left(\frac{p}{2 - p}\right)/p = \frac{1}{2 - p}
\]
Traceback Processing

1. Collect a set of attack packets $L_v$
2. Check router $S$, a neighbor of the victim, with $L_v$
3. Check each router $R$ (neighbor of $S$) with $L_s$
4. Pass $L_v$ to $R$ to be used to make new $L_s$
5. Repeat these processes

"Have you seen any of these packets? "yes"
"You are convicted! Use these evidences to make your $L_s$"

Victim
A fundamental optimization question

• Recall that in the original traceback scheme, the router records a bloom filter of 3 bits for each and every packets

• There are many different ways of spending this 3 bits per packet budget, representing different tradeoff points between size of digest and sampling frequency
  – e.g., use a 15-bit bloom filter but only record 20% of digests (15*20% =3)
  – e.g., use a 12-bit bloom filter but only record 25% of digests (12*25% =3)
  – Which one is better or where is the optimal tradeoff point?

• Answer lies in the information theory
Intuitions from the information theory

• View the traceback system as a communication channel
  – Increasing the size of digest reduces the false positive ratio of the
    bloom filter, and therefore improving the signal noise ratio (S/N)
  – Decreasing sampling rate reduces the bandwidth (W) of the
    channel
  – We want to maximize $C = W \log_2 (1+S/N)$

• C is the mutual information – maximize the mutual
  information between what is “observed” and what needs to
  be predicted – or minimize the conditional entropy

• Bonus from information theory: we derive a lower bound
  on the number of packets needed to achieve a certain level
  of traceback accuracy through Fano’s inequality
The optimization problem

\[ k^* = \arg\min_k H(Z \mid X_{t1} + X_{f1}, Y_t + Y_f) \]

subject to the resource constraint \( s = k \times p \)

- **s**: average number of bits “devoted” for each packet
- **p**: sampling probability
- **k**: size the bloom filter digest
Applications of Information Theory

Resource constraint: $s = k \times p = 0.4$
Verification of Theoretical Analysis

- Parameter tuning

Parameters: 1000 attackers, \( s = k \times p = 0.4 \)
Lower bound through Fano’s inequality

• $H(p_e) \geq H(Z \mid X_{t1} + X_{f1}, Y_t + Y_f)$

Parameters: $s = 0.4$, $k = 12$, $p = 3.3\% \ (12 \times 3.3\% = 0.4)$
Simulation results

- False Negative & False Positive on Skitter I topology

Parameters: $s=0.4$, $k=12$, $p=3.3\% \times 0.4$
Verification of Theoretical Analysis

- Error levels by different $k$ values

Parameters: 2000 attackers, $N_p=200,000$
Future work and open issues

1. Is correlation factor $1/(2-p)$ optimal for coordination using one bit?

2. What if we use more than one bit for coordinating sampling?

3. How to optimally combine PPM and hash-based scheme – a Network Information Theory question.

4. How to know with 100% certainty that some packets are attack packets? How about we only know with a certainty of $p$?
Conclusions

• Design a sampled hash-based IP traceback scheme that can scale to a large number of attackers and high link speeds

• Addressed two challenges in this design:
  – Tamper-resistant coordinated sampling to increase the “correlation factor” to beyond 50% between two neighboring routers
  – An information theory approach to answer the fundamental parameter tuning question, and to answer some lower bound questions

• Lead to many new questions and challenges
Related publications
