Tao Li, Shigang Chen

Traffic Measurement on the Internet

– Monograph –

August 28, 2012

Springer

Chapter 1
INTRODUCTION

Abstract Traffic measurement provides critical real-world data for service providers and network administrators to perform capacity planning, accounting and billing, anomaly detection, and service provision. In many measurement functions, statistical methods play important roles in system designing, model building, formula deriving, and error analyzing. One of the greatest challenges in designing an online measurement function is to minimize the per-packet processing time in order to keep up with the line speed of the modern routers. To meet this challenge, one should minimize the number of memory accesses per packet and implement the measurement module in the on-die cache memory. Hence, it is critical to make the data structures of a measurement module as compact as possible. This book presents several novel online measurement methods that are compact and fast.

Key words: Internet Traffic Measurement, Compact and Fast Solutions

1.1 Online Network Functions

Modern high-speed routers forward packets from incoming ports to outgoing ports via switching fabric, bypassing main memory and CPU. New technologies are pushing line speeds beyond OC-768 (40Gb/s) to reach 100Gb/s or even tera bits per second [14]. The line cards in core routers must therefore forward packets at a rate exceeding 150Mpps [35]; that leaves little time to process each packet. Parallel processing and pipeline are used to speed up packet switching to a few clock cycles per packet [15]. In order to keep up with such high throughput, online network functions for traffic measurement, packet scheduling, access control, and quality of service will also have to be implemented using on-chip cache memory and bypassing main memory and CPU almost entirely [35, 22, 46]. However, fitting these network functions in fast but small on-chip memory represents a major technical challenge today [15, 29].

The commonly-used cache memory on network processor chips is SRAM, typically a few megabytes. Further increasing on-chip memory to more than 10MB is technically feasible, but it comes with a much higher price tag and access time is longer. There is a huge incentive to keep on-chip memory small because smaller memory can be made faster and cheaper. Off-chip SRAM is larger. For example, QDR-III SRAM has 36MB [27]. But it is slower to access. Hence, on-chip memory remains the first choice for online network functions that are designed to match the line speeds.

On-chip memory is limited in size. To make the matter even more challenging, it may have to be shared by security [18], measurement [22], routing [6], and performance [17] functions that are implemented on the same chip. When multiple network functions share the same memory, each of them can only use a fraction of the available space. Depending on their relative importance, some functions may be allocated tiny portions of the limited memory, whereas the amount of data they have to process and store can be extremely large in high-speed networks. The disparity in memory demand and supply requires us to implement online functions as compact as possible [40, 36]. Furthermore, when different functions share the same memory, they may have to take turns to access the memory, making memory access the performance bottleneck. Since most online functions require only simple computations that can be efficiently implemented in hardware, their throughput will be determined by the bottleneck in memory access. Hence, we must also minimize the number of memory accesses made by each function when it processes a packet. The challenge is that compactness (in terms of space requirement) and speed (in terms of memory accesses) are sometimes conflicting objectives.

1.2 Fundamental Primitives

The implementations of many online functions heavily rely on several fundamental building blocks for data processing and storage. This book studies three important fundamental online functions: per-flow size estimators, spread estimators, and origin-destination flow estimators.

Per-flow size estimators are used to measure per-flow information for high-speed links. The goal is to estimate the size of each flow (in terms of number of packets). A flow is identified by a label that can be a source address, a destination address, or any combination of addresses, ports, and other fields in the packet header. Measuring the sizes of individual flows has important applications. For example, if we use the addresses of the users as flow labels, per-flow size measurement provides the basis for usage-based billing and graceful service differentiation, where a user's service priority gracefully drops as he over-spends his resource quota. Studying per-flow data over consecutive measurement periods may help us discover network access patterns and, together with user profiling, reveal geographic/demographic traffic distributions among users. Such information will help Internet service providers

and application developers to align network resource allocation with the majority's needs.

We define a *contact* as a source-destination pair, for which the source sends a packet to the destination. The source or destination can be an IP address, a port number, a combination of address/port together with other fields in the packet header, or even a file name or URL in the payload. The *spread of a source* is the number of *distinct destinations* contacted by the source during a measurement period. Similarly, we can define the *spread of a destination*, which is the number of *distinct sources* that have contacted the destination. Measuring spread values has many applications. Intrusion detection systems can use them to detect port scans [37], in which an external host attempts to establish too many connections to different internal hosts or different ports of the same host. They may be used to detect DDoS attacks when too many hosts send traffic to a receiver [28], i.e., the spread of a destination is abnormally high. They can be used to estimate the infection rate of a worm by monitoring how many addresses each infected host contacts over a period of time. A large server farm may use the spread values of its servers to find how popular the servers' content is, which provides guidance for resource allocation. An institutional gateway may monitor outbound traffic and identify external web servers that have large spread values. This information helps the local proxy learn the popularity of servers and determine the cache priority of web content.

Origin-destination (OD) flow estimators are used to measure *OD flow sizes*. Consider two routers r_1 and r_2. We define the set of packets that first pass r_1 and then pass r_2 or first pass r_2 and then pass r_1 as an *origin-destination (OD) flow* of the two routers. The cardinality of the packet set is called the *OD flow size*. The OD flow measurement is also an important topic in many network management applications [32, 31, 25, 13, 9]. For example, Internet service providers may use the OD-flow information between points of interest as a reference to align traffic distribution within the network. They may also study the OD-flow traffic pattern and identify anomalies that deviate significantly from the normal pattern. In the event of a persistent congestion, OD-flow data may help point out the source of the congestion.

One of the greatest challenges in designing an online measurement module is to minimize the per-packet processing time in order to keep up with the line speed of the modern routers. To meet this challenge, we should minimize the number of memory accesses per packet and implement the measurement module in the on-die SRAM, which is fast but expensive. Because many other functions may also run from SRAM, it is expected that the amount of high-speed memory allocated for the module will be small. Hence, it is critical to make the measurement module's data structure as compact as possible.

1.3 Per-Flow Size Estimation through Randomized Counter Sharing

This book first presents a particularly challenging problem, the measurement of per-flow sizes for a high-speed link without using per-flow data structures [20]. It has been shown in [10] that maintaining per-flow counters cannot scale for high-speed links. Even for efficient counter implementations [33, 30, 47], SRAM will only be able to hold a small fraction of per-flow state (including counters and indexing data structures such as pointers and flow identities for locating the counters). The *counter braids* avoid per-flow counters and achieve near-optimal memory efficiency [22, 23]. This method maps each flow to a certain number of arbitrary counters; they are all incremented by one for every packet of the flow. Many flows may be mapped to the same counter, which stores the sum of the flow sizes. Essentially, the counters represent linear equations, which can be solved for the flow sizes. Two levels of counters are used to reduce the memory overhead. The counter braids require slightly more than 4 bits per flow and are able to count the exact sizes of all flows. But it also has two limitations. First, it performs 6 or occasionally 12 memory accesses per packet. Second, when the memory allocated to a measurement function is far less than 4 bits per flow, the message passing decoding algorithm of counter braids cannot converge to any meaningful results. When the available memory is just $1\sim2$ bits per flow, the *exact measurement* of the flow sizes is no longer possible. We have to resort to *estimation methods*. The key is to efficiently utilize the limited space to improve the accuracy of the estimated flow sizes, and do so with the minimum number of memory accesses per packet.

We present a fast and compact per-flow size estimation function that achieves three main objectives: i) It shares counters among flows to save space, and does not incur any space overhead for mapping flows to their counters. This distinguishes our work from [33, 30, 47]. ii) It updates exactly one counter per packet, which is optimal. This separates our work from the counter braids that update three or more counters per packet. Updating each counter requires two memory accesses for read and then write. iii) It provides estimation of the flow sizes, as well as the confidence intervals that characterize the accuracy, even when the available memory is too small such that other exact-counting methods including [22, 23] no longer work. We believe this is the first size estimator that achieves all these objectives. It complements the existing work by providing additional flexibility for the practitioners to choose when other methods cannot meet the speed and space requirements.

The design of our size estimator is based on a new data encoding/decoding scheme, called *randomized counter sharing*. It splits the size of each flow among a number of counters that are randomly selected from a counter pool. These counters form the *storage vector* of the flow. For each packet of a flow, we randomly select a counter from the flow's storage vector and increment the counter by one. Such a simple online operation can be implemented very efficiently. The storage vectors of different flows share counters uniformly at random; the size information of one flow in a counter is the noise to other flows that share the same counter. For-

tunately, this noise can be quantitatively measured and removed through statistical methods, which allow us to estimate the size of a flow from the information in its storage vector. We present two estimation methods whose accuracies are statistically guaranteed. They work well even when the total number of counters in the pool is by far smaller than the total number of flows that share the counters. The experimental results based on real traffic traces demonstrate that the new methods can achieve good accuracy in a tight space. We also provide several methods to increase the range of flow sizes that the estimators can measure.

The randomized counter sharing scheme presented in this work for per-flow size measurement has applications beyond the networking field. It may be used in the data streaming applications to collect per-item information from a stream of data items.

1.4 Spreader Classification

It is very costly to measure the spread of each source (or destination) precisely. When a router measures the spread of a source, it has to remember the destinations that the source has contacted so far. Future packets from the source to the same destinations do not increase the spread value. The spread is increased only when a packet is sent to a new destination. The problem is that it takes too much memory to store all destination addresses that every source has contacted.

To solve this problem, various techniques such as sampling [39], probabilistic counting [16], Bloom filters [46], and bitmaps [11, 3, 40] are used to reduce memory overhead at the expense of measurement accuracy. The rationale is that absolutely precise measurement of spread values may not be necessary for most applications. It is often practically sufficient to estimate spread values with a certain level of accuracy. Moreover, many applications only require us to classify spreaders into categories, such as (1) *heavy spreaders*, i.e., sources (or destinations) whose spread values are large, and (2) *non-heavy spreaders*. This further lowers the accuracy requirement and allows additional room for memory saving. For example, in scan detection, we want to identify heavy spreaders (scanners) that have contacted a lot of destinations. In the previous server-farm example, we want to know the set of servers with large spread values. Even if we do not identify all such servers, it is very helpful in resource allocation if we can identify most of them.

This book addresses the *spreader classification* problem. *Single-objective spreader classification* is to identify the set of heavy spreaders. *Multi-objective spreader classification* places sources (or destinations) into more categories based on their spread values. We present an efficient spreader classification scheme based on a new storage method, called *dynamic bit sharing*, which stores contact information of all sources in a compact format. The level of compactness is so deep that the total number of available bits is less than one twentieth of the number of sources in some of our experiment cases: On average, just one bit is available for every twenty sources. Yet still we are able to make spreader classification with predictable accuracy. We

employ a maximum likelihood estimation method to extract per-source information from the compact storage and determine the heavy spreaders. It ensures that false positive/negative ratios are bounded. Moreover, given an arbitrary set of false positive/negative bounds, we develop a systematic approach to determine the optimal system parameters, such that the amount of memory needed to satisfy the bounds is minimized. We carry out experiments based on a real traffic trace and demonstrate that, using these optimal parameters, we can reduce the memory consumption by three to twenty times when comparing with other existing work.

1.5 Origin-Destination Flow Measurement

When we solve the problem of *origin-destination (OD) flow measurement* [21], the goal is to design an efficient method to measure the number of packets that traverse between two routers during a measurement period. It generally consists of two phases: One for online packet information storage and the other for offline OD-flow size computation. In the first phase, routers record information about arrival packets. In the second phase, each router reports its stored information to a centralized server, which performs the measurement of each OD flow based on the information sent from the origin/destination router pair.

Measurement efficiency and accuracy are two main technical challenges. In terms of efficiency, we want to minimize the per-packet processing overhead to accommodate future routers that forward packets at extremely high rates. More specifically, the function should minimize the computational complexity and the number of memory accesses for each packet.

Accuracy is another important design goal. In high-speed networks, we have to deal with a very large volume of packets. And it is unrealistic to store all packet-level information in order to achieve 100% accuracy. To solve this problem, some past research [42, 43, 41] uses data such as link load, network routing, and configuration data to indirectly measure the OD flows. Cao, Chen and Bu [2] propose a quasi-likelihood approach based on a continuous variant of the Flajolet-Martin sketches [12]. However, none of them is able to achieve both efficiency and accuracy at the same time.

To meet these challenges, we present a novel OD flow measurement method, which uses a compact bitmap data structure for packet information storage. At the end of a measurement period, bitmaps from all routers are sent to a centralized server, which examines the bitmaps of each origin/destination router pair and uses a statistical inference approach to estimate the OD flow size. The new method has three elegant properties. First, its processing overhead is small and constant, only one hash operation and one memory access per packet. Second, it is able to achieve excellent measurement results, which will be demonstrated by both simulations and experiments. Finally, its data storage is very compact. The memory allocation is less than 1 bit for each packet on average.

Traffic measurement is an important subject of Internet technologies. In the broad context of computer networks, there are many other topics such as QoS and maxmin routing [34, 38, 24, 8, 26, 7, 5], P2P networks [19, 45, 44], distributed computing [1, 4], etc. Although we do not address these topics, they may interact with traffic measurement under certain scenarios where new research problems and applications may sprout.

1.6 Outline of the Book

The rest of the book is organized as follows: Chapter 2 presents a fast and compact per-flow size estimator based on *randomized counter sharing*. In this chapter, we provide of a novel data encoding/decoding scheme, which mixes per-flow information randomly in a tight SRAM space for compactness. Chapter 3 presents an efficient spread estimation scheme based on *dynamic bit sharing*, which optimally combines probabilistic sampling, bit-sharing storage, and maximum likelihood estimation. Chapter 4 gives a novel method for OD flow measurement which employs the bitmap data structure for packet information storage and uses statistical inference approach to compute the measurement results.

References

1. Basu, A., Buch, V., Vogels, W., von Eicken, T.: U-Net: a user-level network interface for parallel and distributed computing. Proc. of ACM SOSP pp. 40–53 (1995)
2. Cao, J., Chen, A., Bu, T.: A Quasi-Likelihood Approach for Accurate Traffic Matrix Estimation in a High Speed Network. Proc. of IEEE INFOCOM (2008)
3. Cao, J., Jin, Y., Chen, A., Bu, T., Zhang, Z.: Identifying High Cardinality Internet Hosts. Proc. of IEEE INFOCOM (2009)
4. Chen, S., Deng, Y., Attie, P., Sun, W.: Optimal Deadlock Detection in Distributed Systems based on Locally Constructed Wait-for Graphs. Proc. of the 16th International Conference on Distributed Computing Systems pp. 613–619 (1996)
5. Chen, S., Fang, Y., Xia, Y.: Lexicographic Maxmin Fairness for Data Collection in Wireless Sensor Networks. IEEE Transactions on Mobile Computing **6**(7), 762–776 (2007)
6. Chen, S., Nahrstedt, K.: Maxmin Fair Routing in Connection-oriented Networks. Proc. of Euro-Parallel and Distributed Systems pp. 163–168 (1998)
7. Chen, S., Shavitt, Y.: SoMR: A Scalable Distributed QoS Multicast Routing Protocol. J. of Parallel and Distributed Computing **68**(2), 137–149 (2008)
8. Chen, S., Song, M., Sahni, S.: Two Techniques for Fast Computation of Constrained Shortest Paths. IEEE/ACM Transactions on Networking **16**(1), 105–115 (2008)
9. Erramilli, V., Crovella, M., Taft, N.: An independent-connection model for traffic matrices. Proc. of Internet Measurement Conference (IMC) (2006)
10. Estan, C., Varghese, G.: New Directions in Traffic Measurement and Accounting. Proc. of ACM SIGCOMM (2002)
11. Estan, C., Varghese, G., Fish, M.: Bitmap Algorithms for Counting Active Flows on High-Speed Links. IEEE/ACM Transactions on Networking (TON) **14**(5), 925–937 (2006)
12. Flajolet, G.: Probabilistic counting. Proc. of Symp. on Fundations of Computer Science (FOCS) (1983)
13. Fortz, B., Thorup, M.: Optimizing OSPF/IS-IS weights in a changing world. IEEE JSAC Special Issue on Advances in Fundamentals of Network Management (2002)
14. Gardner, W.D.: Researchers Transmit Optical Data At 16.4 Tbps. InformationWeek (2008)
15. Hermsmeyer, C., Song, H., Gemelli, R., Bunse, S.: Towards 100G Packet Processing: Challenges and Technologies. Bell Labs Technical J. **14**(2), 57–80 (2009)
16. Hwang, K., Vander-Zanden, B., Taylor, H.: A Linear-time Probabilistic Counting Algorithm for Database Applications. ACM Transactions on Database Systems **15**(2), 208–229 (1990)
17. Jian, Y., Chen, S.: Can CSMA/CA Networks Be Made Fair? Proc. of the 14th ACM international conference on Mobile computing and networking pp. 235–246 (2008)
18. Jung, J., Paxson, V., Berger, A., Balakrishnan, H.: Fast Portscan Detection Using Sequential Hypothesis Testing. Proc. of IEEE Symposium on Security and Privacy (2004)
19. Kamvar, S., Schlosser, M., Garcia-Molina, H.: The Eigentrust algorithm for reputation management in P2P networks. Proc. of the World Wide Web Conference (2003)
20. Li, T., Chen, S., Ling, Y.: Fast and Compact Per-Flow Traffic Measurement through Randomized Counter Sharing. Proc. of IEEE INFOCOM (2011)
21. Li, T., Chen, S., Qiao, Y.: Origin-Destination Flow Measurement in High-Speed Networks. Proc. of IEEE INFOCOM, mini-conference (2011)
22. Lu, Y., Montanari, A., Prabhakar, B., Dharmapurikar, S., Kabbani, A.: Counter Braids: A Novel Counter Architecture for Per-Flow Measurement. Proc. of ACM SIGMETRICS (2008)
23. Lu, Y., Prabhakar, B.: Robust Counting Via Counter Braids: An Error-Resilient Network Measurement Architecture. Proc. of IEEE INFOCOM (2009)
24. Lui, K., Nahrstedt, K., Chen, S.: Hierarchical QoS Routing in Delay-bandwidth Sensitive Networks. Proc. of 25th Annual IEEE Conference on Local Computer Networks pp. 579–588 (2000)
25. Medina, A., Taft, N., Salamatian, K., Bhattacharyya, S., Diot, C.: Traffic matrix estimation: Existing techniques and new directions. Proc. of ACM SIGCOMM (2002)

26. Nahrstedt, K., Chen, S.: Coexistence of QoS and Best-effort Flows-routing and Scheduling. Proc. of 10th IEEE Tyrrhenian International Workshop on Digital Communications: Multimedia Communications (1998)
27. Pearson, M.: QDRTM-III: Next Generation SRAM for Networking. http://www.qdrconsortium.org/presentation/QDR-III-SRAM.pdf (2009)
28. Plonka, D.: FlowScan: A Network Traffic Flow Reporting and Visualization Tool. Proc. of USENIX LISA (2000)
29. Qiao, Y., Li, T., Chen, S.: One Memory Access Bloom Filters and Their Generalization. Proc. of IEEE INFOCOM (2011)
30. Ramabhadran, S., Varghese, G.: Efficient Implementation of a Statistics Counter Architecture. Proc. of ACM SIGMETRICS (2003)
31. Ringberg, H., Soule, A., Rexford, J., Diot, C.: Sensitivity of PCA for traffic anomaly detection. Proc. of ACM SIGMETRICS (2007)
32. Roughan, M., Thorup, M., Zhang, Y.: Traffic engineering with estimated traffic matrices. Proc. of Internet Measurement Conference (IMC) (2003)
33. Shah, D., Iyer, S., Prabhakar, B., McKeown, N.: Maintaining Statistics Counters in Router Line Cards. Proc. of IEEE Micro **22**(1), 76–81 (2002)
34. Shi, Y., Hou, Y.T.: Theoretical results on base station movement problem for sensor network. Proc. of IEEE INFOCOM (2008)
35. Song, H., Hao, F., Kodialam, M., Lakshman, T.: IPv6 Lookups Using Distributed and Load Balanced Bloom Filters for 100Gbps Core Router Line Cards. Proc. of IEEE INFOCOM (2009)
36. Staniford, S., Hoagland, J., McAlerney, J.: Practical Automated Detection of Stealthy Portscans. J. of Computer Security **10**(1-2), 105–136 (2002)
37. Staniford, S., Hoagland, J., McAlerney, J.: Practical Automated Detection of Stealthy Portscans. Journal of Computer Security **10**, 105 – 136 (2002)
38. Tang, Y., Chen, S., Ling, Y.: State Aggregation of Large Network Domains. Computer communications **30**(4), 873–885 (2007)
39. Venkatataman, S., Song, D., Gibbons, P., Blum, A.: New Streaming Algorithms for Fast Detection of Superspreaders. Proc. of NDSS (2005)
40. Yoon, M., Li, T., Chen, S., Peir, J.: Fit a Spread Estimator in Small Memory. Proc. of IEEE INFOCOM (2009)
41. Zhang, Y., Roughan, M., Duffield, N., Greenberg, A.: Fast accurate computation of large-scale ip traffic matrices from link loads. Proc. of ACM SIGMETRICS (2003)
42. Zhang, Y., Roughan, M., Lund, C., Donoho, D.: An informationtheoretic approach to traffic matrix estimation. Proc. of ACM SIGCOMM (2003)
43. Zhang, Y., Roughan, M., Lund, C., Donoho, D.: Estimating Point-to-Point and Point-to-Multipoint Traffic Matrices: An Information-Theoretic Approach. IEEE/ACM Transactions on networking **13**(5), 947–960 (2005)
44. Zhang, Z., Chen, S., Ling, Y., Chow, R.: Capacity-aware Multicast Algorithms on Heterogeneous Overlay Networks. IEEE Transactions on Parallel and Distributed Systems **17**(2), 135–147 (2006)
45. Zhang, Z., Chen, S., Yoon, M.: MARCH: A Distributed Incentive Scheme for Peer-to-peer Networks. Proc. of IEEE INFOCOM pp. 1091–1099 (2007)
46. Zhao, Q., Xu, J., Kumar, A.: Detection of Super Sources and Destinations in High-Speed Networks: Algorithms, Analysis and Evaluation. IEEE J. on Selected Areas in Communications (JASC) **24**(10), 1840–1852 (2006)
47. Zhao, Q., Xu, J., Liu, Z.: Design of a Novel Statistics Counter Architecture with Optimal Space and Time Efficiency. Proc. of ACM Sigmetrics/Performance (2006)