Approximate Caches for Packet Classification

Francis Chang, Wu-chang Feng
Systems Software Laboratory
OGI School of Science and Engineering at OHSU
Beaverton, Oregon, USA
{francis, wuchang}@cse.ogi.edu

Kang Li
Department of Computer Science
University of Georgia
Athens, Georgia, USA
kangli@acm.org

***Abstract*—Many network devices such as routers and firewalls employ caches to take advantage of temporal locality of packet headers in order to speed up packet processing decisions. Traditionally, cache designs trade off time and space with the goal of balancing the overall cost and performance of the device. In this work, we examine another axis of the design space that has not been previously considered: accuracy. In particular, we quantify the benefits of relaxing the accuracy of the cache on the cost and performance of packet classification caches. Our cache design is based on the popular Bloom filter data structure. This study provides a model for optimizing Bloom filters for this purpose, as well as extensions to the data structure to support graceful aging, bounded misclassification rates, and multiple binary predicates. Given this, we show that such caches can provide nearly an order of magnitude cost savings at the expense of misclassifying one billionth of packets for IPv6-based caches.**

Keywords— Bloom filter; packet classification; caches; probabilistic algorithms

I. MOTIVATION

Due to the impending explosion of the Internet address space (IPv6), the increasing complexity of networks, and the need to support layer-4 services, we investigate the answer to one question:

What are the quantifiable benefits that relaxing the accuracy of a packet classification cache has on the size and performance of packet classification caches?

Previous studies have shown that on the Internet, TCP checksums will fail for approximately 1 in 1100 to 1 in 32000 packets, even when link-level CRCs should only admit error rates of 1 in 4 billion errors. On average, between 1 in 16 million to 1 in 10 billion TCP packets will contain an undetectable error. We contend that introducing an error in the range of 1 in a billion will not meaningfully degrade network reliability.

II. THEORY

In this body of work, the Bloom filter data structure is analyzed, optimized and adapted for use in a layer-4 network cache. Unlike previous work using Bloom filters, the demands of using a Bloom filter in this context favour maximizing the number of elements (or flow identifiers) that can be stored in the Bloom filter without exceeding a preset misclassification threshold, instead of minimizing the error given a fixed number of elements (or flows).

Given this assumption, we show that:

- The optimal value of L, the number of hash levels in a Bloom filter, is invariant with respect to the size of the Bloom filter, M.
- The maximum number of elements that a Bloom filter can store, k, and the misclassification probability, p, are roughly logarithmically related.
- k is linearly related to M.
- An optimally full Bloom filter has ½ of its bits set.

III. EXTENSIONS TO THE BLOOM FILTER

We propose two extensions to the Bloom filter – one to age a Bloom filter, and one to allow the Bloom filter to store multiple binary predicates.

Bloom filters were originally designed to store large amounts of static data. A new mechanism to evict data stale entries is required to adapt this data structure to support a cache. This work explores two simple options.

The simplest idea is to simply empty the entire cache whenever it becomes "full". Although this algorithm can efficiently use the cache, it is hindered by its need to restart from a "cold-cache", causing spikes in the cache miss rate.

A second aging strategy involves employing two separate Bloom filters – an active cache, and a background cache that is warmed up, in anticipation of switching the active cache and the background cache. Although less efficient in its use of memory, it is effective in removing the performance artefacts associated with zeroing out the cache.

Finally, we explore the effectiveness of employing multiple Bloom filters to store multiple-predicates, necessary for storing information such as a router's outgoing interface number, or a diffserv priority level.

These extensions are analyzed, and performance of the system is evaluated using real-world network traces.

This work supported by the National Science Foundation under Grant EIA-0130344 and the generous donations of Intel Corporation. Any opinions, findings, or recommendations expressed are those of the author(s) and do not necessarily reflect the views of NSF or Intel.

More details of this work are available at http://www.cse.ogi.edu/sysl/projects/ixp/