AFBV: A Scalable Packet Classification Algorithm

Ji Li, Haiyang Liu, Karen Sollins
MIT Laboratory for Computer Science
{jli, hyliu, sollins}@ mit.edu
http://www.mit.edu/~jli/afbv

Packet classification is a central function of a number of network activities, such as routing and firewall management. The packet classifier has a ordered rule database, one rule for each type of packets, and all arriving packets are classified into equivalent classes based on the first rule they match. With the development of the Internet, the database size can only grow with time. Consequently, scalability becomes an important issue for packet classification algorithm. This project is addressing the scalability problem in the selection of rules for packet classification.

There is significant previous work in this area. Much previous work executed in linear time with the number of rules. Our work is an improvement on the ABV scheme of Baboescu and Varghese [1], which in turn is an improvement on the straightforward linear Lucent bit vector scheme (BV) [2]. The BV scheme produces bit vectors of all rules for each field, and selects the rules applicable to the complete header by first finding the bit vectors corresponding to each field, then intersecting them, and the first set bit in the consequent bit vector implies the position of the rule in the database. ABV made two observations: first the set bits in the bit vectors are sparse and second a packet matches no more than a few rules. Hence, ABV improves BV to achieve logarithmic time in two steps. First, ABV rearranges the rules by sorting, so that those rules that match specific prefixes are near each other in the list. Second, it aggregates bits in each field to reduce memory accesses, but increasing the possibility of false matches. The rule rearrangement mitigates the false matches to some extent, but also incurs additional cost. First, rule rearrangement makes it necessary to find all matches, instead of first match in BV. Second, it is not easy to find an effective rearrangement method. Third, rules need to be mapped back to the original order so as to find the rule with the lowest order.

Our scheme, the Aggregated and Folded Bit Vector algorithm (AFBV) inherits the idea of bit vector and aggregation from BV and ABV, but discards rule rearrangement and introduce a new concept: folding. All bits in a predefined range *r* (*r* is called *folding range*, usually equal to multiple aggregate sizes), whose positions have the same mod-f value are OR-ed together, the result occupying that position mod *f*, thus folding the bit vector (*f* is called *folding size*). Only when there are at least one bit set at the same mod-f position can one get false matches, while in ABV, a match among any bits within an aggregation group can lead to a false match. Once there is a match in the folded bit vector at position k (k is inclusively between 0 and f-1), the possible real match position must take the form ($I \cdot f + k$). Therefore, folding helps not only filter out false matches but also locate the real match positions. This scheme will not produce any false negatives. Thus, correctness is guaranteed.

Besides the folding range and the folding size, the number of folding is another important parameter in AFBV. By using multiple folding with different sizes, we can enhance the detecting and locating ability of the folded vectors. First, only if there are matches in all the intersections of the folded vectors may there be real matches in the original vectors. The gain here is that mostly we do not need to fetch the whole original bit vector to conclude a false match as required in BV and ABV. Second, with proper selection on the folding sizes, multiple folding greatly reduces the number of the possible match positions than single folding. The gain here is that in case of real matches, we only need to fetch parts instead of the whole original bit vectors.

In addition to the aggregate bit vectors, AFBV needs to store the folded bit vectors, which incurs about 10% additional storage requirements in our case.

AFBV offers the following benefits: Only the first matched rule need to be found; no mapping back cost is needed; without rule rearrangement, pre-processing is much simple; a large aggregate size can be applied because the folded vectors help to reduce false matches.

In our simulations of relative performance we compare the Lucent BV algorithm, the ABV algorithm and our AFBV algorithm, and evaluate both performance and space utilization. Preliminary results on different-sized databases show that AFBV outperforms ABF and BV on average, and can achieve the similar level of performance in worst cases, with about a 10% increase in memory requirements, for the additional, folded vectors.

* This work was funded by the National Science Foundation under grant number CCR-0122419.

References:

1. F. Baboescu, G. Varghese. Scalable Packet Classification. In Proc. ACM SIGCOMM'01, Aug. 2001.
2. T. Lakshman and D. Stidialis. High speed policy-based packet forwarding using efficient multi-dimensional range matching. In Proc. ACM SIGCOMM'98, Sept. 1998.