Brief Announcement: RedRem: A Parallel Redundancy Remover

H. B. Acharya
University of Texas at Austin
acharya@cs.utexas.edu

M. G. Gouda
University of Texas at Austin
National Science Foundation
mgouda@nsf.gov

ABSTRACT

Policies defined by a sequence of predicate-decision rules, with first-match semantics, are widely used; a notable example is their use in firewalls, where the rules are used to decide whether to accept or discard each packet. Owing to the critical importance of correctness of such policies, as well as the need for high performance, they have been the subject of considerable analysis. In earlier work, we have demonstrated that the problem of removing redundant rules from firewalls is theoretically equivalent to verifying that a firewall satisfies a property, and proposed that this theorem be used to build a high performance redundancy remover. In this paper, we realize this promise, and build a fast linear-space redundancy remover, one to three orders of magnitude faster than current approaches. Further, we show that our algorithm is easy to parallelize- there exists a natural way to partition a large instance of the problem into independent small ones.

Categories and Subject Descriptors

C.2.0 [**Computer Systems Organization**]: Computer-Communication Networks

General Terms

Performance

Keywords

Firewall optimization, Redundancy

1. INTRODUCTION

A firewall is a security system that acts as a protective boundary. In the context of computer networks, a firewall is a packet filter that is placed at the point where a private computer network connects to the "outside world", usually the rest of the Internet. The firewall intercepts each packet exchanged between the network and the Internet, examines the headers of the packet, and makes a decision: it may "accept" the packet (allow it to proceed), or it may "discard" the packet.

The decision that a firewall makes when it receives a packet depends on the values in the headers of the packet. A firewall is a sequence of rules; each rule matches certain packets, with specific header values, and specifies the decision (accept or discard) to be made for that packet. When multiple rules match a packet, the first rule (according to the order of the rule sequence), that matches the packet, takes precedence.

Firewalls deal with large volumes of packets, and speed of processing is a serious concern as in the case of a denial-of-service attack a slow firewall will cause a server to fail. There exists an optimal solution to the problem of fast firewall execution: Ternary Content Addressable Memory (TCAM), which returns the decision for a packet in constant time. However, TCAMs have small capacity, and are large custom circuits (and thus very expensive). Thus, while used for extremely high-end routing, they are usually not suitable for firewalls (or packet classifiers in general).

In [3], the authors propose that a packet classifier can be optimized to fit on a TCAM, and develop an algorithm to restructure a given packet classifier and remove all redundant rules. However, applying their method to firewalls reveals a serious disadvantage: the algorithm requires the construction of an all-match tree. An all-match tree has a size at least as large as that of the corresponding Firewall Decision Diagram - which has a space complexity of $(2n)^d$ [4], where n is the number of rules in a firewall and d is the number of fields in a rule. The value of n may be less than a dozen, in small firewalls, but in complex cases it may be over a thousand. d is 4 to 6, with the usual value being 5 (the usual fields checked are source and destination port, source and destination IP address, and protocol).

In this paper, we use our result that firewall verification is equivalent to redundancy detection [1], and build upon the *probe* algorithm of firewall verification [2], to implement RedRem, a new system that detects and removes all redundant rules in a given firewall.

In contrast to prior approaches, RedRem runs in space $O(nd)$ (which is the size of the firewall and thus a lower bound on the space complexity). We also verify experimentally that, in practice, the algorithm requires very little memory to run (around 0.5 kB per rule for large firewalls). Our time complexity is still $O(n^d)$ (indeed, we conjecture that this may be a lower bound for a deterministic algorithm that completely removes redundancies). However, our algorithm runs very fast in practical cases; we show by extensive testing that we greatly outperform earlier redundancy detection algorithms. Furthermore, our algorithm can be used in conjunction with the techniques of projection and division to decompose a problem instance into many smaller, independent instances, so an additional advantage of our algorithm is that it can naturally take advantage of a parallel machine.

SPAA'11, June 4–6, 2011, San Jose, California, USA.
ACM 978-1-4503-0743-7/11/06.

2. RESULTS

Our model of firewalls is that a firewall is a sequence of rules, which consist of predicates and decisions (accept/discard). In each predicate, there are d fields, for each of which there is an interval of natural numbers (a, b). A packet is represented by its header values, which the firewall examines to decide whether to accept or discard it. Hence our model of a packet is a d-tuple of integers, which is said to *match* a rule if every field of the packet is a member of the corresponding field of the predicate of the rule. The first rule in the sequence that is matched by a packet is said to *resolve* the packet, i.e. its decision is the decision of the firewall for the packet.

A rule in a firewall is redundant iff removing it from the firewall does not change the decision of the firewall for any packet. In our earlier paper [1], we demonstrate that the necessary and sufficient condition for a rule R_j in firewall F to be redundant is that it resolves no packet p that is resolved in firewall $F - R_j$ by a rule R_k, whose decision is opposite to the decision of R_j. Every packet that matches R_j is resolved by either R_j or some preceding rule R_i, so if we change the decisions of all rules R_i that precede R_j in firewall F to the decision of R_j, thereby creating firewall F', then in firewall $F' - R_j$ there is no packet matching R_j that is resolved by some rule R_k that conflicts with R_j (i.e. whose decision is opposite to that of R_j). In other words, $F' - R_j$ satisfies property R_j - a firewall verification problem.

Further, this algorithm can be speeded up by taking notice of the fact that if there is indeed even one conflicting R_k that resolves a packet matching R_j in $F' - R_j$, rule R_j is not redundant in F. Hence we can parallelize the search for such a packet (the *signature packet* of R_j) by dividing the firewall $F' - R_j$ into slices. Each slice consists of one conflicting rule R_k, preceded by the compliant (non-conflicting with R_j) rules that precede R_k in $F' - R_j$. We further optimize the search by projecting the slice over R_k, i.e. removing any portion of the predicate of any rule that does not belong to the predicate of R_k. Finally, we verify, using the probe algorithm of [2], that each slice satisfies the projection of R_j over the corresponding R_k. Such verification can be executed in parallel for all the conflicting rules R_k that are preceded by R_j in F. If there exists a slice which does not satisfy the property, R_j is non-redundant.

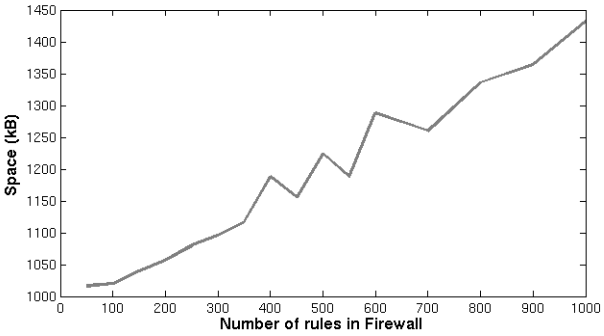
In this paper, we present a system, which we name RedRem, that uses the above algorithm to identify and remove all redundant rules from a large sample set of firewalls. As RedRem is faster for smaller firewalls, we were able to test a larger number of short firewalls. We tested $100,000$ firewalls of each length varying from 50 to 250 rules in steps of 50, and $10,000$ firewalls of each length from 300 to 1000 in steps of 50. In total, we tested our algorithm on $650,000$ firewalls, which vary in length from 50 to 1000. Our test firewalls have the number of fields $d = 5$.

Our results are presented in Figures 1(a) and 1(b), respectively. Our algorithm is clearly superior to the fastest previous algorithm, all-match trees [3], as we demonstrate in Table 1. (Unfortunately, the authors published their performance test with only three firewalls so we cannot provide a more comprehensive comparison.) In addition, our algorithm has several other important advantages. Firstly, as we have demonstrated, it can be naturally decomposed into subproblems that can be solved in parallel. Secondly, on a sequential computer it uses $O(nd)$ space, as opposed to $O(n^d)$ for all-match trees.

Length	RedRem	First-match tree	All-match tree
42	0.44	491	171
87	1.5	179	47
661	69	1105	750

Table 1: Running time (ms.)

(a) Execution time of Algorithm.



(b) Space required by Algorithm.

3. REFERENCES

[1] H. B. Acharya and M. G. Gouda. Firewall Verification and Redundancy Checking are Equivalent. In *Proceedings of IEEE INFOCOM*, 2011.

[2] H. B. Acharya and M. G. Gouda. Projection and division: Linear-space verification of firewalls. In *Proceedings of ICDCS*, 2010.

[3] C. R. Meiners, A. X. Liu, and E. Torng. Tcam Razor: A systematic approach towards minimizing packet classifiers in tcams. In *Proceedings of the IEEE Conference on Network Protocols (ICNP)*, pages 266–275, 2007.

[4] A. X. Liu and M. G. Gouda. Diverse firewall design. *IEEE Transaction on Parallel and Distributed Systems*, 19(9):1237–1251, 2008.