Removing Redundant Rules from Packet Classifiers

Alex X. Liu[*], Mohamed G. Gouda
Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712-1188, U.S.A.

Redundancy Problem: Packet classification is the core mechanism that enables many networking services such as firewall access control and traffic accounting. A packet classifier consists of a sequence of rules whose function is to classify each incoming packet into one of predefined classes. A rule in a packet classifier is redundant iff removing the rule does not change the decision of the packet classifier for each packet. For instance, in the following packet classifier, where each rule only checks one packet field F_1 whose domain is $[1, 100]$, rule r_3 is redundant and rule r_2 is redundant after removing r_3.

$$\begin{array}{ll} r_1: & F_1 \in [1,\ 50] \rightarrow \textit{accept} \\ r_2: & F_1 \in [40,\ 90] \rightarrow \textit{discard} \\ r_3: & F_1 \in [30,\ 60] \rightarrow \textit{accept} \\ r_4: & F_1 \in [51,\ 100] \rightarrow \textit{discard} \end{array}$$

How to detect and remove all redundant rules in a packet classifier is a new problem that has not been scientifically addressed previously.

Why Important: By removing all redundant rules from a packet classifier before a packet classification algorithm starts building data structures from the rules, both classification space and classification time are reduced.

Reducing memory space for packet classification algorithms is of paramount importance because a packet classifier on a core router must use very limited on-chip cache to store complex data structures. Fast packet classification algorithms need $O(n^d)$ classification space, where n is the total number of rules and d is the total number of packet fields that the classifier examines for each packet. On average a packet classifier has more than 15% redundant rules. Figure 1 shows the percentage of classification space that can be saved by removing all redundant rules versus the number of packet fields (assuming only 15% rules are redundant).

Reducing the amount of overlapping among rules or reducing the total number of rules reduces classification time. By removing redundant rules, while other non-redundant rules remain unchanged, both the amount of overlapping of rules and the total number of rules are reduced.

Previous Work: Gupta gave a sufficient but not necessary condition for identifying redundant rules in his PhD thesis. For example, rule r_3 and r_2 in the previous example are not identified as redundant rules by his definition.

Our Contribution: In this paper, we give a necessary and sufficient condition for identifying all redundant rules, based on which we categorize redundant rules into upward redundant rules and downward redundant rules. A rule r in a packet classifier is *upward redundant* iff there are no packets whose first matching rule is r. For example, rule r_3 in the previous example is upward redundant. A rule r in a packet classifier is *downward redundant* iff for each packet, whose first matching rule is r, the first matching rule below r has the same decision as r. For example, rule r_2 in the previous example is downward redundant after r_3 is removed.

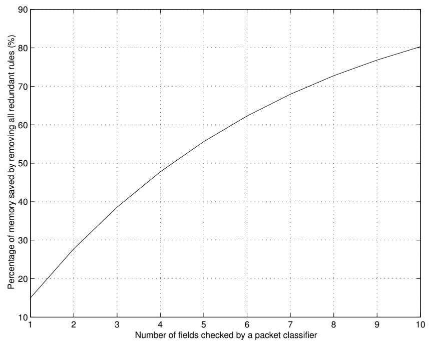

Figure 1: Number of fields vs. Memory Saved

In this paper, we present two efficient graph based algorithms for detecting and removing these two types of redundant rules. The data structure we use for detecting and removing redundant rules is called packet decision diagrams.

Experimental Results: We implemented the algorithms in this paper in SUN Java JDK 1.4. The experiments were carried out on one SunBlade 2000 machine running Solaris 9 with 1Ghz CPU and 1 GB memory. The average processing time for removing all upward and downward redundant rules from a packet classifier versus the total number of rules in the packet classifier is shown in Figure 2.

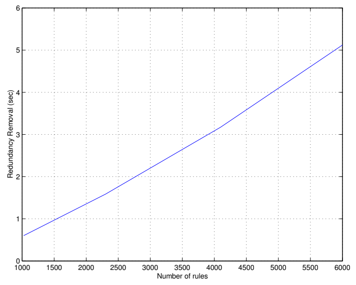

Figure 2: Average processing time vs. Total number of rules

Full Text Available:
http://www.cs.utexas.edu/ftp/pub/techreports/tr04-26.pdf

[1]Corresponding author: Alex X. Liu (alex@cs.utexas.edu)