A two-level packet classification

Ons Jelassi , Olivier Paul
INT, National Institute of Telecommunication
Evry, France
Email: {ons.jelassi, olivier.paul}@int-evry.fr

***Abstract*— Packet classification is a central function in firewalls, intrusion detection mechanisms and monitoring architectures. Network elements assuming these techniques operate on packet flows to insure access control. A large variety of multi-fields packet classification techniques were reported in litterature but it remains difficult to find a packet classification solution that represents a good tradeoff between classification times, fast updates, memory requirements and scalability to large filters database. In this paper, we introduce a new five-fields classification concept, the two level classification algorithms based on an architecture that can be applied to any decision tree-based packet classification algorithm, we test it with a well-known algorithm the Extended Grid-of-Tries and present performance measurments. In this paper, we show how our algorithm improves search times.**

I. Introduction

Security and monitoring applications require packet classification. Packet classification implies finding out in a set of filters the highest priority filter matching the packet. The growth of in demand bandwidth has driven the throughput over classification engines to increase exponentionally. The classification task has become more complex and is based on at least five packet header fields: source and destination IP adresses and ports and protocol identifier. The type of fields values are typically prefixes for IP adresses, ranges for ports and exact values or wildcard for protocol identifier.
Filtering in firewalls generates security classes. A firewall processes incoming packets based on a filter database. Entities such as MIDCOM (MIDdlboxes COMmunication) agents configure dynamically the rules in firewalls or NATs (Network Adress Translator) [1]. This communication is managed by the MIDCOM protocol [2].
QoS monitoring needs led to the expansion of monitoring frameworks with PSAMP (Packet SAMPling) [3] or IPFIX (IP Flow Information eXport) [4] proposals. These techniques configure some functions of capture, stamping, aggregation and compute QoS indicators within networks. In order to perform this supervision, monitoring techniques need to achieve packet classification to select packets in the monitored traffic. In this paper, we present a novel two-level classification algorithm. It's illustrated through a use case.

II. Problem statement

A network element offering traffic differentiation maintains a database of rules. Each rule associates an action to a processed packet. A rule R consists in a filtering part based on one or more packet fields, an associated action to process if the packet matches the rule and a given priority.

Matching packets with corresponding actions is a bottleneck for the intermediate network equipment. The aim of the work presented in this paper is to make this correspondence faster with responding to the needs of upper-layers protocols and architectures in terms of fields format and fields number.

III. Previous work

Many implementations of firewalls use, mostly, a simple linear search. The data structure associated is a simple chained list containing the N rules sorted by decreasing priority. The search is sequential with a time of O(N). The storage of the list is also O(N). The update time is O(log N). It's obvious that the search time is too important.
Other packet classification algorithms use heuristics and present very good time and memory complexities. But due to data structure precomputation, algorithms such as RFC (Recursive Flow Classification) algorithm described in [5] are unable to handle incremental updates. We also cite HICuts (Hierarchical Intelligent Cuttings) presented in [6], an algorithm for two-fields classification which was extended by HyperCuts [7] to multi-fields classification.
In [8], a scheme called Tuple Space Search was proposed, it can handle any type of filters and has a good update time. However, the time complexity of searches and updates are non deterministic. Enhancements using precomputation and markers were proposed in [9].

A. Positioning our work

In this paper, we present an improvement to the Extended Grid-of-Tries algorithm presented in [10].
One of most cited proposals in the packet classification literature is the grid of tries algorithm. This algorithm, proposed in [11], was originally designed to be applied for two fields classification.
This algorithm uses a prefix representation for classification conditions. We notice that the ranges can be converted into prefixes. Splitting a range in a set of minimal ranges satisfying this property produces a set of prefixes corresponding to the initial range. A W-width range can be split into at most 2*W-2 prefixes.
A binary trie is a binary tree where branches are labeled according to bits values, the left branch is labeled '0' and the right one is labeled '1'. A node represents the concatenation of the labels of all the branches between the root and this node. For example the prefix 0* is represented by the left root's child.

Rule	Field 1	Field 2
F1	0*	10*
F2	0*	01*
F3	0*	1*
F4	00*	1*
F5	00*	11*
F6	10*	1*
F7	*	00*

TABLE I
AN EXAMPLE OF A CLASSIFIER

A 2 fields grid-of-tries is an hierarchical structure. We first, construct the first level, a 1-dimensional trie which corresponds to the first field. Then, for each node labeled with a prefix, p, in this trie, we construct a trie on the second field for the rules of classifier for which the first field is p.

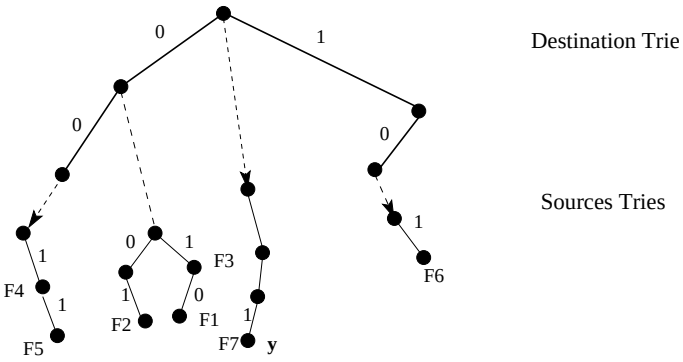

Fig. 1. Grid-of-Tries structure

The data structure for the classifier in Table I is shown in Fig.1. To extend the Grid-of-Tries to handle more then two

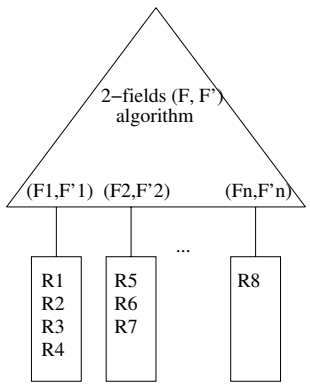

Fig. 2. Extending the Grid-of-Tries structure

fields, [10] propose to use a linear search on the rest of the fields after using the source-destination matching (see Fig.2).

IV. A TWO-LEVEL CLASSIFICATION ALGORITHM

A. Motivation

In this section, we describe our two-level classification approach and we apply it to the Extended Grid-Of-Tries.
The idea on which is based this approach is that a classification engine treats packets headers that often share the same fields values, and the fields may share the same values for a certain number of consecutive bits. The number of this recurrent packet headers values is obviously more important when packets belong to flows from applications such as VPN (Virtual Private Networks) where packets share at least source and destination IP adresses, in a RTP (Real Time Protocol) connexion, or in a HTTP/FTP connexion.
To justify this assertion, we studied filter files given by the benchmark developped in the Washington university [12]. These files represente real filter files used in access lists in entreprise edge and backbone routers (ACL), access lists and security filters in firewalls (FW), VPN and NAT filters in software-based systems (IPC). The study of redundancy depends on number of considered bits. We tested the number of redundant filters in different filter files. For example an ACL file of 5000 filters is composed of 1990 filters begining with'0' and 2110 of filters begining with '1', of 334 of filters begining with '01'and 1596 files begining with '00', etc., of 540 filters with '*' in the IP destination adress and 167 filters with '206.*.*.*' in the IP destination adress, etc.

Our approach is totally different from the flow-based classification where packets are accepted or rejected depending on the nature of the flow ie the normal sequence of packets (the packet following a request must be a response). Our optimisation reduces search times for packets belonging to different flows as long as packets share any bits considered from the packet header.

B. Implementation

We propose to have a two-level structure for packet classification. The main structure contains the classification structure constructed by the classification algorithm mainly the Extended Grid-of-Tries.
The first structure is an hash table which is used to store last searches results in order to help next searches. The hash function is the CRC (Cyclic Redundancy Check) on the header bits. In the case of collision, a linear search in the results with the same hash key is performed.
If we have the same packet header, we find immediatly the best-matching result. If the new packet header shares some fields values with an entry in the first structure, we return a pointer to the result of the best matching for the first structure entry and we continue the search begining from that node. This reduces the time of search since we don't begin the search from the root node of the basic structure. In case we can't find correspondance in the first structure, we do a normal exploration in the basic Extended Grid-of-Tries structure.
We notice that the pointer contained in the first level of structure can contain any node in the search path depending on

the treated flow nature and on the number of bits considered. Typically, in the Extended Grid-of-Tries, if we consider the first header field for first level correspondance, the pointer can contain the node terminating the search in the first trie corresponding to the first field, and if we consider the concatanation of the two first header fields the pointer contains the node terminating the search in the second trie corresponding to the matching for the two first headers.
The first level of the data structure is refreshed periodically, the timeout for an entry depends on the nature of the flow.

C. Use Case Study

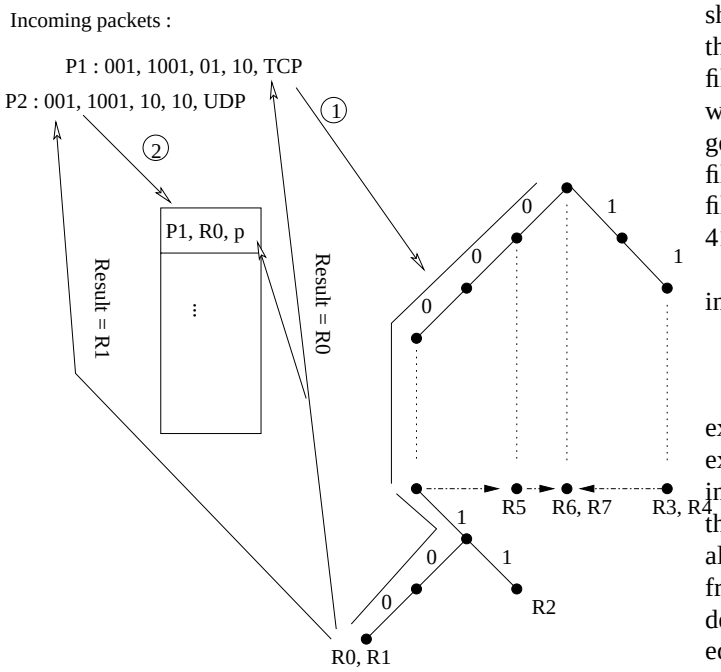

Fig. 3. Search example

For example, let's take two incoming packets with headers P_1 (001,1001,01,10,TCP) and P_2 (001,1001,10,10,UDP).
After doing a normal best-matching search in the Extended Grid-of-Tries structure for P_1 (Fig.3), we store the result of the search, the P_1 header values and a pointer of the node terminating the search in the first level of the two-level classification algorithm data structure.
To find the best-matching filter corresponding to P_2, we do an hash search in the first level structure, we find that P_2 has the same two first fields as P_1. Since, the pointer pointing on the result of the search corresponding to the two first fields in the Grid-of-Tries is stored in the first level of our data structure, we go immediatly to that node. Arriving there, a simple search in the bucket containing the filters allows us to find the best-matching filter for P_2.
We can see on this example that the practical time search for packet header P_2 is optimized using our proposal.

	Basic EGT	Two-level EGT
ACL	0.253	0.186
FW	0.689	0.510
IPC	0.742	0.493

TABLE II
THE AVERAGE BEST-MATCHING TIME SEARCH

D. Performance results

In this section we present measurements from our implementations of basic EGT and the two-level EGT in C++ on a Linux machine (Intel Pentium 2 Ghz, 256 Mo RAM). We make a comparaison between the two implementations and show that our proposal gives better processing times. We use the washington university benchmark. The three measurement file sets contain filter files composed of 5 000 filters. This size was choosen because it is representative of real filters sets. We generate, using the trace generator of the benchmark the packet files corresponding to these filter files, the size of the packet files vary depending on the formats (see justifications in [13]): 41.815 packets for ACL, 50.192 for IPC, 63.729 for FW.

As reported in Table II, we have an improvement of 30% in best-matching search times.

V. CONCLUSION

In this paper, we presented a novel idea based on an extension to tree based packet classification algorithms. This extension allows them to take advantage of redundancies in packet headers. We show an application of this idea to the well-known Extended Grid-of-Tries packet classification algorithm. We believe such an extension can be interesting from a practical point of view since it can be optimized depending on the type of flows treated by the classification equipement. Optimizations and performance tests on memory usage and updates support will be investigated in further works.

REFERENCES

[1] P. Srisuresh et M. Holdrege, *IP Network Adress Translator (NAT) Terminology and Considerations*. IETF 1999.
[2] M. Barnes, *RFC 4097: Middlebox Communications (MIDCOM) Protocol Evaluation*. IETF 2005.
[3] T. Zseby, M. Molina, N. Duffield, S. Niccolini, F. Raspall, *Sampling and Filtering Techniques for IP Packet Selection*. IETF 2005.
[4] J. Quittek, T. Zseby, B. Claise, S. Zander, *Requirements for IP Flow Information Export (IPFIX)*. IETF 2004
[5] P. Gupta, *Algorithms for routing lookups and packet classification*. Thesis of the university of Standford 2000.
[6] N. McKeown, P. Gupta *Packet Classification Using Hierarchical Intelligent Cuttings*. Hot Interconnects VII 1999.
[7] G. Varghese , J. Wang, S. Singh, F. Baboescu *Packet Classification Using Multidimensional Cutting*. SigComm 2003.
[8] V. Srinivasan, S. Suri and G. Varghese, *Tuple Search for Fast Layer-4 Packet Classification*. SigComm 1999.
[9] V. Srinivasan, *A Packet Classification and Filter Management System*. InfoCom 2001
[10] F. Baboescu, S. Singh, G. Varghese, *Packet classification for Core Routers: Is there an alternative to CAMs?*, Infocom 2003.
[11] V. Srinivasan, G. Varghese, S. Suri and M. Wadvogel, *Fast and Scalable Layer Four Switching*. SigComm 1998.

[12] D.E. Taylor, J.S. Turner, *ClassBench : A Packet Classification Benchmark*. IEEE Infocom 2005
[13] D.E. Taylor, J.S. Turner, *ClassBench : A Packet Classification Benchmark*. Technical Report 2004