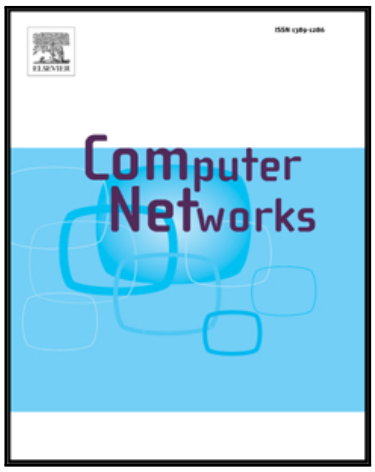

Extensions to Decision-Tree Based Packet Classification Algorithms to Address New Classification Paradigms

Thibaut Stimpfling, Normand Bélanger, Omar Cherkaoui,
André Béliveau, Ludovic Béliveau, Yvon Savaria

PII: S1389-1286(17)30148-2
DOI: 10.1016/j.comnet.2017.04.021
Reference: COMPNW 6160

To appear in: *Computer Networks*

Received date: 1 April 2016
Revised date: 2 December 2016
Accepted date: 7 April 2017

Please cite this article as: Thibaut Stimpfling, Normand Bélanger, Omar Cherkaoui, André Béliveau, Ludovic Béliveau, Yvon Savaria, Extensions to Decision-Tree Based Packet Classification Algorithms to Address New Classification Paradigms, *Computer Networks* (2017), doi: 10.1016/j.comnet.2017.04.021

Extensions to Decision-Tree Based Packet Classification Algorithms to Address New Classification Paradigms

Thibaut Stimpfling[a], Normand Bélanger[a], Omar Cherkaoui[b], André Béliveau[c], Ludovic Béliveau[d], Yvon Savaria[a]

[a] *École Polytechnique de Montréal*
[b] *Université du Québec à Montréal (UQAM)*
[c] *NoviFlow*
[d] *Wind River*

Abstract

The decision-tree based packet-classification algorithm field has seen many contributions since the first algorithm using a geometrical rule representation, HiCuts, has been proposed. While hardware reported implementations for this class of algorithms have proven that a high throughput can be reached, those algorithms are inherently facing a tradeoff between speed and memory consumption.

This paper presents two extensions applicable to decision-tree based algorithms designed to tackle two of their common drawbacks. Applied together, they achieve a reduction of the number of memory accesses, while reducing the data structure size. The first contribution consists of a new rule-clustering method aimed for the reduction of the number of trees built. The second contribution relies on a leaf compression method that allows tackling the problem that stems from linear leaf traversal. Applied together, as shown by simulations, those two new methods alleviate the compromise between search-time complexity and data structure size. These strategies provide gains in many contexts, although they are tailored for handling complex rule sets used in the context of Software Defined Networking. For sets of 100 000 and 10 000 rules, those two strategies reduce the number of memory accesses by a factor of 3 on average, while decreasing the size of the data structure by about 45% over EffiCuts, a well-known decision-tree based algorithm.

Keywords: Decision Tree, Packet Classification, Software Defined Networking, OpenFlow.

1. Introduction

Packet classification is a functionality required by networking devices in a wide range of contexts like Quality of Service (QoS), load balancing, security, monitoring, and network traffic analysis. Its purpose is to match incoming packets with one or more rules, contained in a rule set.

This functionality becomes more challenging to implement as the average link speed increases, while means of classifying improve at a slower pace. The challenge is compounded by the demand put on classifiers to process many more fields and rules. Thereby, classification remains an open research topic.

The networking context is shifting away from the one used in previous work presented in the literature, and a new networking paradigm known as Software Defined Networking (SDN) has emerged in the past few years [1]. SDN has gathered a lot of momentum, and it is now deployed in multiple contexts, from small to nationwide networks, including production networks [2]. SDN supports programmable networks controlled by manipulating a logical map of a slice of the network. Thereby, SDN brings new possibilities, such as optimizing the link utilization rate, getting a unified view of the network fabric, improving failure handling, etc.

A key aspect of SDN is the separation of the data and control planes, while the communication between the logically centralized control plane and the data plane relies on protocol such as OpenFlow. OpenFlow [3] is a widely used protocol that allows communicating between the control plane and the data plane. It does so by setting up "Flow Tables" in the data plane in order to program it. Each flow entry in a flow table contains match fields, counters and a set of instructions to apply to incoming matching packets [3]. The number of match fields used ranges from 12 in version V1.0.0 up to 41 in version V1.3.2. The Data Plane has to support the functionality offered to the Control Plane (through OpenFlow), thereby impacting the classification units that must meet the specification defined in [4].

Hence, the packet classification context is deeply changed and has to handle much more complex rules, with combinations of masks, ranges, prefixes and wildcards over a larger number of fields [3]. Thus, the classical 5-tuple context, which is largely covered in the literature, no longer matches the trends and evolution of the networking field [5, 6].

It is important to note that even if OpenFlow does not force flow entries to use every match fields available, the context of this work is to focus on a worst case scenario, where each

Email addresses: thibaut.stimpfling@polymtl.ca (Thibaut Stimpfling), normand.belanger@polymtl.ca (Normand Bélanger), cherkaoui.omar@uqam.ca (Omar Cherkaoui), andre.beliveau@noviflow.com (André Béliveau), ludovic.beliveau@windriver.com (Ludovic Béliveau), yvon.savaria@polymtl.ca (Yvon Savaria)