I. INTRODUCTION

Network packet classification is one of the most fundamental and performance critical tasks that is performed by a broad range of security appliances, such as firewalls, intrusion detection systems, and application-level gateways. Basically, the goal of packet classification is to map inspected network packets to a corresponding rule in a predefined rule set that specifies how to treat different kinds of traffic. This task, however, is difficult to achieve in environments with large traffic volumes and/or complex security policies. In order to process network packets at line rate under such circumstances, fast classification systems are required, which implement sophisticated search algorithms in software or even dedicated hardware. These systems have to process two kinds of input data: first, the network packets to be classified, and second, the rule set that specifies how the packets should be discriminated. However, treating the security rule set as a system input parameter can have negative implications for the classification system performance: in software, the rule set interpretation can lead to additional runtime overhead [1], as the traversal of the rule set data structure during packet classification requires additional expensive memory accesses. Furthermore, systems based on dedicated hardware, such as Ternary Content-Adressable Memories (TCAMs), cannot efficiently store a wide range of security policies with respect to the hardware resource utilization [2].

In the HARDFIRE research project [3], we address these issues and present methodologies to specialize network classification systems with respect to the used security policy. These methodologies are based on partial evaluation of generic classification systems on the rule set input. To this end, we integrate the semantics of the specified rule set into the classification system implementation itself. For software-based systems, this specialization happens through native code generation which accelerates the well-known bit vector classification algorithm [4]. We evaluate this approach in simulated scenarios as well as with real network traffic in the OpenFlow reference switch. Our evaluation results show that the packet processing performance can be increased significantly when using the specialized code instead of the generic bit vector algorithm.

II. SCOPE

We review our research results in the area of software- [1] and hardware-based [2] packet classification in the context of the HARDFIRE research project. In these works, we demonstrated the feasibility of classification system specialization on CPU- and FPGA-based systems. In particular, we focus on the different specialization techniques that we used, as briefly outlined in Sections II-A and II-B. Furthermore, we discuss potential use cases for the presented classification techniques.

A. SOFTWARE-BASED SYSTEM SPECIALIZATION

In [1], we use a just-in-time compiler in order to generate rule set-specific x86 machine code for packet matching which accelerates the well-known bit vector classification algorithm [4]. We evaluate this approach in simulated scenarios as well as with real network traffic in the OpenFlow reference switch. Our evaluation results show that the packet processing performance can be increased significantly when using the specialized code instead of the generic bit vector algorithm.

B. HARDWARE-BASED SYSTEM SPECIALIZATION

In certain environments (e.g., with data rates of 40 Gbps or higher), even the fastest software-based packet classification systems can hardly meet line speed. Therefore, these environments require specialized hardware such as Application Specific Integrated Circuits (ASICs) or FPGAs which offer low processing latencies and high parallelism. In [2], we exploit the ability of FPGAs to implement an arbitrary user-specified circuit in order to generate rule set-tailored packet matching circuitry. That is, the security policy is integrated into the circuit itself rather than stored in a configuration memory. Our evaluation results reveal that the generated circuitry can be heavily optimized with respect to the layout of the specified rule set, which cannot be achieved with common, generic matching engines with an equivalent degree of parallelism.

REFERENCES