A Flexible Microprogrammed Packet Classifier for Edge Nodes of Transport Networks*

Simon Hauger, Sascha Junghans, Arthur Mutter, Detlef Sass
University of Stuttgart, Institute of Communication Networks and Computer Engineering (IKR)
Pfaffenwaldring 47, 70569 Stuttgart, Germany
e-mail: {hauger, junghans, mutter, sass}@ikr.uni-stuttgart.de

Abstract

Packet classification is one of the major tasks of packet processing systems at the edge of high-speed transport networks. As existing protocols are evolving and new protocols are constantly being developed, today's network systems have to be highly flexible and adaptable to changing requirements. In this paper we present a scalable architecture for a packet classification system for high speed transport network interfaces. This architecture allows the introduction of new protocols and the update of classification rules during operation by using the concept of microprogramming. In the paper we will give a detailed description of the classifiers microprogrammed architecture. Further, the topics performance data concerning the throughput of the classification unit, realization complexity and resource requirements on the chip are also addressed.

1 Introduction

Packet classification is one of the major tasks of modern packet processing systems imposing challenging requirements, especially at the edge of high-speed transport networks.

Packet classification is the process of mapping packets to certain categories and is needed for most network processing systems. Examples for packet classification are the assignment of packets to MPLS paths, flow identification, classification for QoS and scheduling purposes, observation of firewall rules and detection of intrusions.

The above application areas of packet classification span a wide range of the classification criteria regarding the quantity and complexity. This ranges from classification according to few bits in a packet up to complex and dependent expressions of numerous criteria.

To satisfy the fact that protocol definitions may change and new protocols are developed, a packet processing system should be highly flexible and adaptable.

Packet classification needs to be performed at full line speed to avoid loss, additional delay and additional packet buffers. Modern packet processing systems with link speeds up to 10 Gbps at the edge of transport networks introduce hard constraints to classification speed and it therefore still remains an open networking problem [1]. This is even enlarged by strict design principles resulting from further constraints for edge node systems. For these reasons smart and problem-specific but flexible solutions are required.

In this paper we present an architecture for a packet classification module suitable for high speed transport network interfaces. This architecture allows the flexible adaptation of classification rules by deploying the concept of microprogramming. The usage of microprogramming additionally leads to deterministic processing times. Microprogramming is a concept of enabling the desired flexibility into hardwired logic circuits. It has been widely used in CISC microprocessors and is now being rediscovered for the use in network processing systems [2], [3].

The classification architecture is already successfully implemented and applied to a high-performance measurement platform for network traffic [4].

The paper is structured as follows: Section 2 describes the process of packet classification and suitable platforms for classification systems. Section 3 introduces our architecture for a microprogrammed classification module. The realization aspects and implementation results are shown in Section 4. Finally, the paper closes with conclusions and an outlook to future work.

2 Packet Classification Systems

Packet classification is widely used in many different applications. Generally, packet classification inspects packets with respect to certain criteria and maps them to specific categories [5]. These categories can be interface identifiers for switching or routing decisions, classes of service for QoS support or packet filtering rules in a firewall or intrusion-detection system.

* This work was partly funded within the EIBONE project KOMMT!flexRN by the German Bundesministerium für Bildung und Forschung under contract No. 01BP566.

A classification module obtains a copy of the packet as input and outputs an identifier for the matching category the packet is assigned to.

Depending on the application, the requirements for classification modules vary in a broad range: Very high line rates limit the number of classification rules, while systems with moderate line rates can support high numbers of classification rules.

For the classification process a set of *classification rules* is defined. A classification rule contains of one or several classification criteria. A *classification criteria* describes a certain condition that has to be fulfilled. Conditions can have different characteristics. They can refer to exact values like a specific port number or ranges of values like ranges of target or source IP addresses for the identification of IP subnets or even more complex criteria. Some criteria check for values of certain bits in a distinct protocol header like the *syn* bit of TCP which need the introduction of "care" and "don't care" flags for each bit. This behavior can be achieved with so-called ternary logic and allows the efficient and compact definition of classification criteria.

Depending on the application field, classification is often called differently. In the case of examining the layers two to four to make switching or routing decisions it is often called *lookup*. In the case of examining higher layer protocols the expression *deep packet classification* is used [6].

In both cases the rules can be valid for a long period of time like the mapping of packets into MPLS paths at the edge of a transport network or dynamically changing like in QoS-aware routers which react on newly identified flows. In the latter case not only the fast processing of packets is important but also the update process for new classification rules.

As many criteria can match for a single packet, several rules can match as well. At the end, one rule must be selected to determine the corresponding system action. This needs a prioritization of the classification rules which can be seen as a point location problem in a multi-dimensional space [1].

The criteria rules can be represented by multi-dimensional trees which must be traversed during the classification process or they can be ordered in a grid of tries. Further hash values can be used with lookup tables for ordering the identification process of the rule with the highest priority.

As some protocol headers can have variable lengths the definition of classification criteria can not be defined by providing a fixed position in the data stream but must be defined with relative positions like "Byte 3 and 4 of the UDP header". As new protocols and services are introduced from time to time, the different protocols and headers should not be hard-wired or hard-coded in the classifier but should be easily adaptable to new requirements.

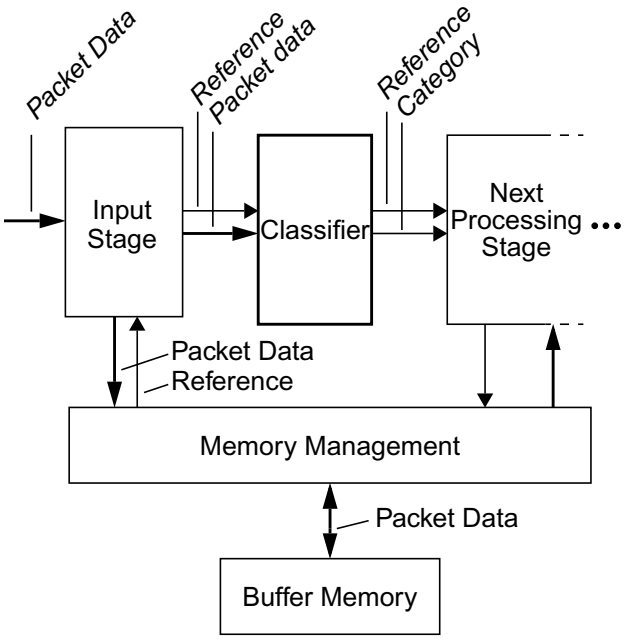

Fig. 1 Typical environment for a classification module

For the implementation of packet processing systems several basic methods and technologies can be used. General-purpose processors provide the flexibility for achieving highly adaptable systems but can not fulfil the high processing requirements and the high I/O bandwidth required at the edge of transport networks. Network processors (NPs) are devices which are designed for high I/O throughput and provide dedicated functions for packet processing tasks. They can be programmed with specialized programming languages which allow the efficient usage of all NP components [5]. As NPs usually have a dedicated lookup engine as a monolithic block, the programmer can not influence the lookup process in detail.

Field programmable gate arrays (FPGAs) and application specific integrated circuits (ASICs) allow more specialized logic designs for the processing modules. While FPGAs provide a very high flexibility because the logic design can be updated in the system, ASICs are cheaper if the number of produced devices is high enough. As the designer can develop system modules on different abstraction layers und make them very specific, these devices are well suited to achieve the required performance.

Classification modules can determine the fulfillment of the rules in different ways. If the packet data is sent to the classifier as a data stream the classifier often scans the whole packet and identifies all matching classification criteria. Others read out the buffer memory at those addresses which are important for their rules.

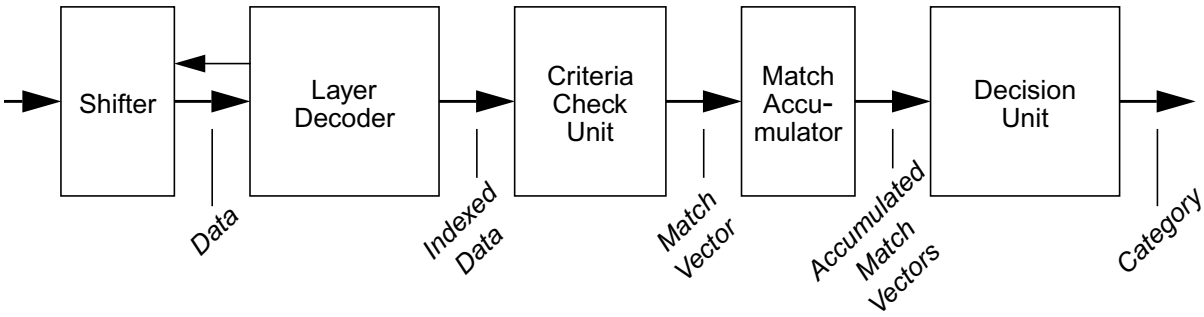

Fig. 2 Basic structure of the packet classifier

The constraints concerning processing delays for the scheduling modules are very strict at the edge of a transport network with high data rates as all packets must be classified at line speed. This is why classification sytems at this location do not handle references to buffer addresses which can lead to long memory access times. Instead, architectures are used which scan copies of the packets on the fly without time costly memory accesses.

Figure 1 shows a typical environment for a classification module with on the fly processing in a logic design.

The packet data enters the system via an input stage. This input stage forwards the data to the memory management to store the data into the buffer memory. The memory management returns a unique reference to the stored data. Upon this the input stage forwards a copy of the packet data as well as the corresponding reference to the classifier. The copied packet is scanned by the classifier and dropped after the classification process. The classifier forwards the reference and the identified category to the next packet processing stage. Following stages, processing the determined category, use the reference to access the packet data in memory.

We introduce a classification module for the edge of transport networks. It therefore processes the packets on the fly as motivated above. The interpretation of the protocol layers is performed by a microprogrammed system. Its knowledge of the protocols and the classification rules are stored in fast accessible internal memory. This allows the easy introduction of new protocols and updates of classification rules.

3 Concept and Architecture

The classification module processes a copy of the incoming packet as a stream of data words. In the following section the basic structure and the functionality of the units of the classification module are described. The most challenging unit, the Microprogrammed Layer Decoder, is explained in more detail, in order to show the high flexibility and adaptability to arbitrary new protocols or protocol extensions. Finally there is a short section regarding the realization aspects of this classifier on an FPGA.

3.1 Basic structure

The classification unit is built up by several functional units. These units form a pipeline structure as shown in Figure 2, consisting of a Shifter, a Layer Decoder, a Criteria Check unit, a Match Accumulator and finally a Decision unit. Each packet is continuously processed in 32 bit wide data words.

The first unit of the pipeline is the *Shifter.* This unit aligns the incoming data in a way that each protocol header starts in a new data word, reducing the complexity in following functional units. This unit is controlled by the adjacent Layer Decoder.

The *Layer Decoder* unit owns the knowledge about all the different protocol layers, incoming packets might consist of. It is completely microprogrammable, so that the classification unit can easily be adapted to changes in protocol definitions or even new protocols. This unit is described in depth in the next section. The main task of this unit is interpreting the relevant fields (e.g. header length field) of each protocol in order to control the Shifter and to label each data word with a distinct index. This index states the current protocol and the relative position of this data word within that protocol layer (e.g. word 2 within protocol UDP).

The indexed data words are forwarded to the *Criteria Check Unit*. Here each data word is compared to a database consisting of all criteria that are needed for the classification rules. Because of the appended index of each word only those criteria that are relevant for the current protocol and the current position within that protocol can lead to a match. The result of this comparison is output in the form of a match vector, consisting of zeros at those positions where no match was found and ones at those positions where a match was found at the according positions in the database.

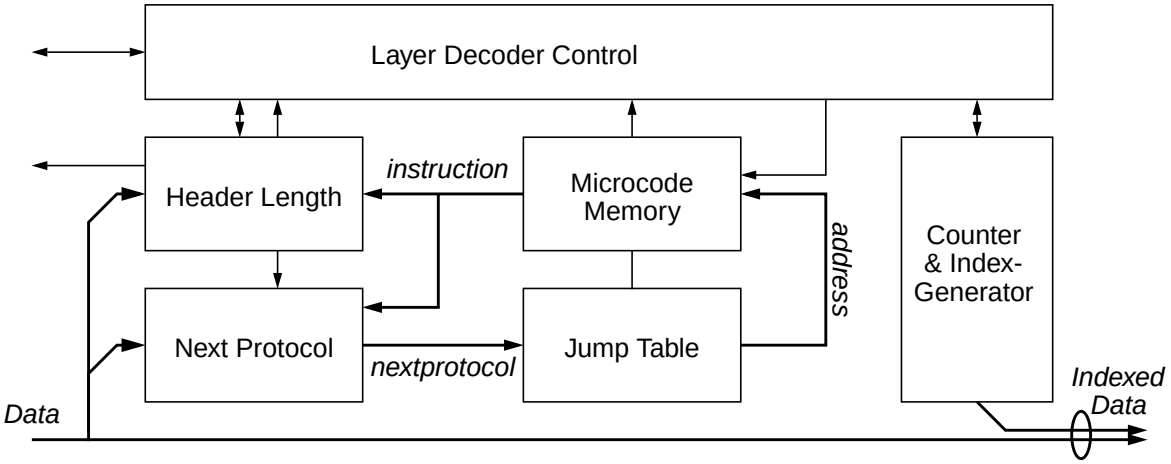

Fig. 3 Structure of the microprogrammed Layer Decoder

The match vectors of each data word of a packet are combined in the *Match Accumulator* unit. This unit basically combines the match vector of each data word with all previous match vectors of this packet by a bit by bit logical OR. When all words of a packet have been processed, the accumulated match vector shows a set of all criteria this packet complies with.

The *Decision Unit* uses this accumulated match vector to come to a decision and assigns the respective category to the processed packet. This is done by comparing the fulfilled criteria with all classification rules, and classifying the packet according to the classification rule with the highest priority that is adhered to.

3.2 Microprogrammed Layer Decoder

In order to achieve the required flexibility to current and future protocols our classifier contains a microprogrammed Layer Decoder.

The Layer Decoder utilizes the fact that all protocols must contain a field specifying the protocol contained in its payload, unless this is statically defined. Further, all protocols have either a fixed header length or contain a field in their protocol header defining their header length.

The structure of the Layer Decoder is depicted in Figure 3. The *Microcode Memory* is the central unit in the Layer Decoder. It contains microcoded instructions for all supported protocols. The microcoded instructions contain information about the position of the header length field or a fixed length as well as the position of the field specifying the next protocol. These instructions are interpreted by two functional units aptly named *Header Length* unit and *Next Protocol* unit. These units extract, process and buffer the respective fields of the incoming packet data words. The header length is used to determine when the next microcoded instructions for the following protocol have to be loaded as well as to compute the shift value for the preceding Shifter. The extracted header field containing the information about the next protocol layer is forwarded to a *Jump Table*, where the address of the corresponding microcode in the *Microcode Memory* is determined. The *Counter- & Index Generator* keeps track of the position of the current data word within the current protocol. This information is used in the Header Length and Next Protocol units in order to extract the correct byte fields of the protocol header. Furthermore the Counter- & Index Generator uses this information to label each data word with an index stating the protocol this data word is part of and the word position within this protocol. This index is used in following Criteria Check unit as described in the previous section. The *Layer Decoder Control* consists of a finite state machine controlling the cooperation of all units.

As the Layer Decoder as well as the Jump Table are completely user-programmable this classification module is adaptable to arbitrary new or changed protocol definitions. There are no constraints about the structure of the protocol headers except for mandatory fields specifying the following protocol and the current header length unless those are not fixed values.

3.3 Realization Aspects

This module was realized in a field-programmable gate array (FPGA). The design extensively utilizes small content-addressable memory (CAM) blocks within the FPGA.

Both the Criteria Check unit and the Decision unit mainly consist of a CAM. Also the Jump Table within the Layer Decoder uses a CAM.

The first two units use ternary CAMs (TCAMs) that allow the usage of a third value ‘X’ meaning "don’t care" besides the logical ‘1’ and ‘0’. With this third state irrelevant bit fields within a data word can be masked out. This is an efficient way to check for different criteria or to find a final decision in those units.

The Jump Table uses an ordinary binary CAM, looking for exact matches when looking for the address for the next protocol instructions.

4 Application and Results

The classification module we introduced in this paper is applied in a high-performance Internet measurement platform, the I^2MP [4]. The measurement platform is built up on the UHP (Universal Hardware Platform) [7] of the authors’ institute. This platform provides a rather old FPGA from Altera (APEX20K 400CB652-C7) and several communication interfaces, like electrical and optical Gigabit Ethernet interfaces.

The measurement platform records packet header information according to filters that are realized using this classification module. Despite the rather old FPGA technology used for this application, it is still possible to perform all packet processing tasks, including classification utilizing the introduced module, at full Gigabit Ethernet line speed.

The classification module within the measurement platform is operated with a clock frequency of 31.25 MHz. Timing analyses of the Quartus place-and-route software from Altera show however that even clock frequencies up to 42 MHz can easily be supported by this FPGA.

With this clock frequency classification can be performed with up to 1.3 Gbps with large packets and up to 2.2 million minimum-sized Ethernet packets can be processed in one second.

The classification module has a latency of only 16 clock cycles, resulting in a delay of 0.38 µs with a clock frequency of 42 MHz.

This performance is achieved with only using few resources on the FPGA. A classification module with 32 classification rules using a total of up to 64 criteria only uses 10% of the logic elements and 5% of the available memory bits of the used FPGA. A module with 128 rules using 256 classification criteria still uses only 27% of the total number of logic elements and 19% of the available memory.

The used FPGA is outdated. Modern FPGAs contain significantly more logic elements and memory bits at processing speeds being two to three times faster. These facts enable the adaptation of the proposed classification module to processing rates of 10 Gbps.

The only drawback, however, is that the new Stratix II FPGA family from Altera does not support CAMs using the internal memory blocks. However building a CAM with logic elements is very inefficient and resource-wasting. So we are going to extend the classification module in order to support an external CAM device. With this extension the presented packet classifier should be able to process packets with up to 10 Gbps when realized on the Stratix II from Altera.

5 Conclusion and outlook

In this paper we presented a scalable architecture for a high speed classification module. It enables a network node to scan data packets on the fly with very low latency at line speed. With the microprogrammed layer decoder, the module is easily adaptable to new protocols and services on any layer during operation. This flexible classifier was realized on an FPGA-based platform as part of a traffic measurement system providing important traces for traffic characterization. The module needs only few FPGA resources without critical timing performance in the Gigabit Ethernet environment.

In our future work we are going to extend the design for line rates up to 10 Gbps. An additional scan module will be designed which allows the identification of signatures at arbitrary positions in the packets. This module will support the easy identification of higher layer services like peer to peer traffic or typical patterns for intrusion-detection systems.

6 Acknowledgement

The authors would like to thank Damir Ferenci for his contributions to the development, implementation and test of the I^2MP measurement platform containing the classification module.

7 References

[1] Kounavis, M., Kumar, A., Yavatkar, R. & Vin, H., *Line Rate Packet Classification and Scheduling*, Tutorial at Symposium on Architectures for Networking and Communications Systems (ANCS), Princeton, New Jersey, USA, October 2005

[2] Vassiliadis, S., Wong, S., Cotofana, S., *Microcode Processing: Positioning and Directions*, IEEE Micro, Vol. 23, Issue 4, 2003, pp. 21-30

[3] Semeria, C., *Implementing a Flexible Hardware-based Router for the New IP-Infrastructure*, Juniper Networks Inc., White Paper, 2001

[4] Sass, D., Junghans, S. *I^2MP - An architecture for hardware supported high-precision traffic measurement*, Proceedings of the 13th GI/ITG Conference on Measurement, Modeling, and Evaluation of Computer and Communication Systems (MMB), Nürnberg, Germany, March 2006

[5] Comer, D. E., *Network systems design using network processors*, Prentice Hall International, 2006

[6] Lekkas, P. C., *Network processors - Architectures, Protocols and Platforms*, McGraw-Hill, 2003

[7] Institute of Communication Networks and Computer Engineering (IKR), University of Stuttgart, *The Universal Hardware Platform (UHP)*, http://www.ikr.uni-stuttgart.de/Content/UHP/