Abstract - This paper presents the design and implementation of a network flow identification system. The system identifies data flows based on packet inspection. Both a software and a hardware implementation is presented. The software implementation uses a hash function to reduce the time required for flow retrieval, allowing real-time network packet processing. The hardware implementation significantly reduces the processing time required for the flow identification. For this implementation, a Xilinx Virtex-II Pro FPGA device and the Xilinx Embedded Development Kit (EDK) software are used. This system is intended to be used for designing a reconfigurable router with QoS (Quality of Service) support.

I. INTRODUCTION

Network flow identification is essential to support Quality of Service (QoS) implementation for several reasons. First, it can be used to identify the applications that present certain QoS constraints. Second, it determines the traffic characteristics in order to evaluate and set QoS policies and goals. Finally, it helps network traffic monitoring to examine the changing tendencies that can affect the quality of service. Because of continuing increasing processing requirements at the router level, the flow identification presents real-time constraints that require elaborated software algorithms or hardware implementations.

In this paper we describe a software application and a hardware system for network flow identification. Both the software application and the hardware system identify flows based on five fields in the packet headers. The software application uses a hash function to increase the speed of flow retrieval. This application represents a support for testing packet-processing algorithms, allowing visualization of network flows and of real-time processing performance. The hardware system is implemented in an FPGA device and further increases the processing speed. This system represents the first step for designing a reconfigurable router with QoS support.

The rest of this paper is organized as follows. Section II provides background information related to network flow identification methods and QoS in computer networks. Section III describes the software application for network flow identification. Section IV presents the hardware implementation of the flow identification system. Finally, Section V concludes the paper.

II. BACKGROUND

QoS represents the network capability to deliver better services for the selected flows over different technologies. The main goal of QoS is to provide priority including dedicated bandwidth, controlled latency and jitter, and improved loss characteristics. A service represents the traffic performance provided to the client and is characterized by the QoS parameters. QoS techniques represent the implementation specific methods, including traffic identification and marking, queue and congestion management, link efficiency, traffic shaping and policy, specific protocols. The service and QoS techniques are different but strongly interconnected, since QoS techniques are used for service construction.

The main technologies used for QoS are Integrated Services, Differentiated Services, Multiprotocol Label Switching, and traffic engineering.

Network flow identification can be used for several goals, such as determining the packet processing method and traffic distribution. QoS technologies treat differently packets belonging to different flows. Therefore, it is important to identify various types of packets by inspecting their contents. In case of integrated services, traffic identification consists of traffic flow identification based on the headers’ contents. Typically, the following information is used: source IP address, destination IP address, protocol identifier, source port number, and destination port number. In case of differentiated services, traffic identification represents the process of packet identification based on a set of specified rules.

When there are several paths to the destination station, a traffic distribution system may be used. Traffic distribution should maintain the order of packets per flow, and this is the reason why flow identification should be performed before traffic distribution.

There are several categories of flow identification algorithms:

- Based on data structures: linear search, caching, hierarchical tries, set-pruning tries;
- Geometrical: grid-of-trees, area-based quadtree (AQT), fat inverted segment tree (FIS);
- Heuristics: recursive flow classification (RFC), hierarchical intelligent cuttings, tuple-space search;
- Hardware: CAM (Content-Addressable Memory), bitmap intersection.

Linear search algorithms use a list of rules stored in descending priority order. A packet is compared sequentially to each rule until a rule that complies with all the fields of the packet is found. The hierarchical tries algorithm is based on several multidimensional hierarchical search trees. The set-pruning tries, grid-of-trees [1], AQT [2] and FIS [3] algorithms are also based on search trees. The RFC algorithm [4] maps the fields of packet headers to the classification conditions in several steps, dividing the packet headers into several parts that are used as indexes in several memories in parallel. The hierarchical intelligent cuttings algorithm [5] uses a search tree that contains a small number of rules in the leaves. These
rules can be compared sequentially. The tuple-space search algorithm [6] divides the classification into an exact number of steps. The set of rules mapped to the same field is stored in a hash table. The bitmap intersection method [7] is based on the fact that the set of rules used for packet identification represents the intersection of a number of rules associated to each dimension.

The CAM method uses a ternary associative memory that stores the rules in descending priority order and performs the comparisons in parallel. As opposed to random-access memories (RAMs), in which the stored data are identified by means of a unique address assigned to each data word, CAM words are identified by their content. CAMs are very useful in applications where intensive search operations are to be performed [8]. Based on the values they can store, there are two types of CAMs: binary and ternary [9]. Binary CAMs can only store binary digits (‘0’, ‘1’), while ternary CAMs can store binary digits as well as “don’t care” values (‘X’). Several networking applications have been identified for using CAMs, including Ethernet address lookup, address filtering, routing, security, or information encryption on for high-performance data switches, firewalls, bridges, and routers [10].

Since packet identification for integrated services, differentiated services and traffic distribution is different for each of these technologies, some algorithms are used with predilection for each of them. In case of integrated services, several packet identification algorithms are used, each representing a compromise between area and speed [11]: CAM-based search, hashing-based schemes and binary search. Hashing-based schemes involve calculation of a hash function and further comparisons if there is collision. When a reservation is made for a flow, a router applies a hash function to the information used for flow identification. If the output hash value has not been used by other reserved flows, the hash value is linked to the state of this flow in the reservation table. If other flows have the same output hash value, a collision occurs and a bit is set to indicate this collision. The hash value points to a collision resolution table that contains the information used for flow identification of all reserved flows with the same output hash value.

For differentiated services, three types of packet identification algorithms are used: caching, geometrical, and tries. The caching method uses a cache memory to store the information that defines the last flows. The problem of this method is that performance is not deterministic. The geometrical method involves localization of a point in a multidimensional space. The number of dimensions is determined by the number of conditions that must be fulfilled by a packet [7]. The tries method is based on a binary tree with each branch labeled with 0 or 1 [12].

III. NETWORK FLOW IDENTIFICATION APPLICATION

In order to test the network traffic identification algorithms, we implemented a software application for traffic flow identification. The flow identification is performed based on five fields contained in the packet header. These fields are the following: source IP address, destination IP address, protocol identifier, source port number, and destination port number. The values of these five fields from a packet are compared against the values of the same fields that define the previously identified flows to determine whether the packet belongs to a flow that is already stored or it belongs to a new flow.

The main design problem of a real-time traffic flow identification application is the comparison of the captured packet information against the information stored about the already identified flows. At the router level, the number of flows is very high and the processing time must be extremely low.

For real-time flow retrieval, a method based on hash function has been used. For this method, an optimized algorithm has been implemented, which allows to reduce the time required for flow retrieval due to the small number of iterations. After testing various algorithms, we draw several conclusions that allow the optimization of flow retrieval. The larger the hashing table is, the smaller the collision rate. A collision appears when the resulted hash value is equal to the hash value calculated for another flow. In case of a collision, the information that identifies the captured packet must be compared against the information that identifies the stored flows. The smaller the number of hash values for which collision appears, the smaller the time required for comparison. Depending of traffic characteristics, the hash function considered may use only some of the five fields previously mentioned.

The application identifies the flows, stores them, and elaborates statistics about the characteristics of these flows. In order to capture the packets from the entire subnet, the application should run on the router. However, since the router is usually a dedicated equipment, a hub or switch configuration may be used, as illustrated in Fig. 1.

A hub transmits each received packet to all the other ports. A switch must be configured to transmit the packets from all the other ports to the port to which the computer that runs the application is connected.

In order to capture all the packets transferred in the subnet, the network interface card is set to promiscuous mode. Each packet is inspected and the fields that identify the flow to which the packet belongs are extracted from its headers. The next step is to compare the identified flow against the existing flows that are stored in a table. If the flow is new, it is inserted into the table. If the packet belongs to an existing flow, the statistic data of the flow are updated.

The graphical interface of the application displays the flow number as it was identified and the values of the fields used for flow identification.

![Figure 1. Network configuration for testing the software application.](Image 348x75 to 494x173)
Other information displayed by the application is the maximum, average, and minimum length of the packets, and the total number of packets belonging to each flow. The information required to evaluate the performance of the flow identification algorithm, represented by the number of iterations required to retrieve the flow, is also displayed, as shown in Fig. 2.

The application is developed for the .NET platform in the C# language. It allows visualizing the network flows, their characteristics and the performance of retrieval methods. The application also allows testing various algorithms and optimization techniques that allow for real-time processing of data flows in computer networks in order to support QoS implementation.

IV. EMBEDDED SYSTEM FOR NETWORK FLOW IDENTIFICATION

To increase the speed of packet processing, we implemented an embedded system for network flow identification. For the implementation we used a Xilinx XC2VP30 FPGA device from the Virtex-II Pro family. This device contains an array of 80x46 configurable logic blocks and integrates two PowerPC 405 processors.

As hardware development platform, we used a Xilinx XUP Virtex-II Pro board [13], which allows implementing complex hardware systems, containing one or more central processing units (CPUs), configurable logic and a large number of peripheral devices that communicate with the CPUs.

For the implementation we chose a combined hardware/software solution, which allows exploiting the advantages of both solutions: the flexibility of the software solution and the high performance of the hardware solution. The time-critical operations are implemented in hardware, while the operations that are less demanding are implemented in software.

As software development system, we used the Xilinx Embedded Development Kit (EDK), version 7.1i [14], which allows to implement complex systems containing both hardware and software modules, and provides tools for testing and debugging the system. The EDK also requires the corresponding version of the Xilinx ISE design package to generate the configuration bitstream for the FPGA device. The EDK provides support for using the tools from the GNU Compiler Collection (GCC), such as the C compiler, the assembler, and the software debugger.

The block diagram of the embedded system for network flow identification is illustrated in Fig. 3. The main components of the system are the MicroBlaze processor, the dual-port BRAM (Block RAM), the BRAM controllers, the EMAC (Ethernet Media Access Controller), the INTC interrupt controller, the UART controller for the serial interface, the CAM, and the MDM (Microprocessor Debug Module).

The MicroBlaze is a 32-bit soft processor core provided by Xilinx EDK. This processor is organized as a Harvard architecture with separate bus interface units for data and instructions. The on-chip dual-port BRAM is connected to the processor using an Instruction-side Local Memory Bus (ILMB) and a Data-side Local Memory Bus (DLMB). These buses use a simple synchronous protocol to provide single-cycle access to the on-chip BRAM. The peripheral modules are connected to the processor using an On-chip Peripheral Bus (OPB). These buses are part of the CoreConnect architecture standard specified by IBM [15] [16].

The CAM performs the comparison between the fields of data packets that uniquely identify a certain flow between two computers. The comparison is performed in parallel for all the words in the CAM, which allows obtaining the result of the comparison in a single clock cycle. An extra clock cycle is required to write the contents of the packet fields into the argument registers. The CAM word size is 104 bits: 2 x 32 bits for the source and destination IP addresses, 2 x 16 bits for the source and destination port numbers, and 8 bits for the protocol identifier.

Fig. 4 illustrates the block diagram of the CAM. The processor accesses the CAM through five 32-bit software-addressable registers, Reg0 to Reg4. Before the search operation, the source and destination IP addresses are written into Reg0 and Reg1, the source and destination port numbers are written into Reg2, and the protocol ID is written into Reg3. For a write operation, the write address is written into Reg3 and the WE bit of this register is set. Reg4 contains the result of the search and it can only be used for reading. Bit 31 of this register represents the match bit.

The CAM is connected to the on-chip peripheral bus through an interface called OPB IPIF (OPB Intellectual Property Interface). Part of this interface, the OPB IPIC (OPB Intellectual Property Interconnect) is generated automatically by the Core Generator module. The other part of the interface has been written in VHDL.
For the CAM, a software driver has been developed, which has been written in C. The main functions that can be performed with this driver are the following: searching an argument word in the CAM, reading from the CAM and writing into the CAM.

The MicroBlaze processor executes a program that initializes and controls the hardware modules. In order to be able to detect all the packets that travel in the network, the program sets the EMAC into promiscuous mode. To initialize the interrupt system, the program uses the EMAC module’s driver to register the callback functions required to service the interrupts. When a frame is sent or received, or when an error occurs, the corresponding interrupt function is called. In the main loop, the program extracts the required fields from the data packets, stores them into the argument registers, and sends the command to the CAM. The program also performs the required operations for managing the data flows and displaying the statistics about the identified flows.

For testing the embedded system, we connected the development board to a hub and monitored the traffic between a server and a PC connected to the same hub. This connection allows monitoring all the packets that travel between the server and the PC.

V. Conclusions

In this paper, we presented the design and implementation of a network flow identification application and the implementation of a hardware system with the same function. The software application performs packet inspection by using a hash function to identify traffic flows. These flows are identified based on five fields contained in each packet header. We used an algorithm that allows real-time identification and processing of traffic flows.

For the hardware implementation of the network flow identification system, we used a Xilinx Virtex-II Pro FPGA device. We designed an embedded system containing a CAM that performs the comparison between the five fields of incoming packets and the previously stored information about the identified flows. Only two clock cycles are required to write the contents of the packet fields into the argument registers and to retrieve the result of the comparison. The main advantage of the hardware implementation is that it reduces significantly the time required for flow identification. This implementation, however, does not represent a pure hardware system, but rather a combined hardware/software design. Only the time-critical operations are implemented in hardware, while other operations are implemented in software. The advantage of this combined solution is that flexibility is increased and the hardware resources required are reduced.

The hardware implementation of the network flow identification system represents the first step for designing a reconfigurable router with QoS support. The next step would be the design of an embedded system that determines the network traffic characteristics, and the last step would be the design of the reconfigurable router that performs the constraint-based routing.

REFERENCES