Computer Networks 53 (2009) 810–820

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Architecture and scalability of a high-speed traffic measurement platform with a highly flexible packet classification

Detlef Saß *, Simon Hauger, Martin Köhn

Universität Stuttgart, Institute of Communication Networks and Computer Architecture (IKR), Pfaffenwaldring 47, 70569 Stuttgart, Germany

ARTICLE INFO

Article history:
Received 1 May 2008
Received in revised form 16 October 2008
Accepted 6 November 2008
Available online 3 December 2008

Keywords:
Passive traffic measurement
Packet classification
Architecture
Scalability analysis
Prototype
Hardware acceleration

ABSTRACT

Evolving network technologies, new web services and changing usage patterns continuously change traffic characteristics. But a thorough understanding of the traffic is the basis for many applications in networking. Thus, it is crucial to analyze up-to-date traffic traces collected by passive measurements in many relevant network contexts. As the traces' quality defines the significance of such analysis, the measured data is required to be complete, temporally accurate and reliable. This is especially challenging for measurements on high-speed links.

In this paper, we present a scalable architecture for a high-speed, passive measurement platform for obtaining highly accurate packet traces. Its functional blocks are distributed to a specialized hardware unit and a commodity PC unit according to their specific requirements. For pre-processing, the hardware unit integrates a protocol-aware classification and filtering module which allows an easy definition of classification and filtering rules. We analyze the platform's scalability and support this by a implementation with current FPGA technology and a standard server PC.

1. Introduction

In recent years, the penetration of fixed and wireless broad-band Internet access has increased all over the world. While less then 10 years ago, the access to the Internet was dominated by dial-up access with low bit rates, today there is a strong trend towards always-and-everywhere-connected users. Due to this, users change their behavior as well as their usage patterns. On the one hand they still use established services but in a different way, e.g. more frequently. On the other hand, they use new upcoming services with new usage paradigms. For the future, it is according to e.g. [1] foreseeable that such changes will happen due to the introduction of new services and the Web2.0 boom.

Within the networks this shift has changed the traffic characteristics with respect to both the temporal behavior as well as the protocols used and protocol stacks. Examples for this today are the well known self-similarity [2,3] or the long range dependence [4] that is even present in the Internet backbone [5] as well as a partial shift from Peer-to-Peer traffic back to web traffic induced by the broad usage of e.g. video services like YouTube [6].

Clearly, a deep understanding of the traffic characteristics is the basis for many applications in networking. This ranges from the design and evaluation of new network architectures to the management of operational networks. This understanding as well as corresponding models are provided by *traffic characterization* and *traffic modeling*.

Traffic characterization analyzes the current traffic and extracts the relevant characteristics. For this, commonly characteristic properties and metrics are derived from traffic traces that are captured on relevant network links. For generalization, sets of traces are analyzed that are captured at different locations and/or at different instants of time.

* Corresponding author. Tel.: +49 711 68569003; fax: +49 711 68559003.
E-mail address: detlef.sass@ikr.uni-stuttgart.de (D. Saß).

1389-1286/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2008.11.010

Traffic modeling derives from these characteristics mathematical or algorithmic models of the traffic for certain scenarios. These models are capable of being parametrized by the user to adapt them to their needs.

Obviously, the determined characteristics and models can only reflect the behavior of the underlying traces. Accordingly, it is crucial to analyze sufficiently detailed traffic traces collected in many relevant contexts. This leads to a major trade off: to elaborate the balance between traced details and data volume or capturing rate. This is especially important for long-term traces at the packet level in high-speed networks, as simply recording all packets entirely would lead to trace sizes of several terabytes in a short time.

Beyond dumping the raw data, more intelligent solutions pre-process the packet stream before recording in non-volatile memory. This contains e.g. filtering to reduce the captured data volume and rate, or further processing operations to preserve the users' privacy.

In this paper, we present the architecture of our high-speed passive measurement platform, the I^{2}MP (IKR Internet Measurement Platform). We discuss major scalability aspects supported by numerical results of fittings to current FPGAs (Field Programmable Gate Arrays). We tailored the design to support long-term capturing of packet traces on high-speed metro and core network links lasting for days. We realized the performance critical and time critical tasks in a dedicated hardware unit and implemented complex post-processing operations as well as storage on commodity PCs. With this, we enable high-precision and high-performance traffic capturing while keeping the complexity low.

To reduce the data volume on the fly, we integrate a hardware Classification and Filtering unit. This unit extracts arbitrary user-defined byte areas of the packets for recording and discards the remaining data. For this, it automatically decodes the protocol stack of each packet on the basis of a configurable set of arbitrary protocols. It then applies for each protocol layer separately the classification rules as well as the filter rules. Our architecture allows the flexible adaptation of both rule sets as well as of the protocol definitions, even during operation.

The rest of the paper is structured as follows: Section 2 reviews the traffic measurement principles, the relating relevant functional blocks and suitable platforms for measurement systems. Section 3 introduces the architecture of our measurement platform I^{2}MP. In Section 4, the scalability of the architecture with respect to the throughput as well as the number of classification rules is discussed. Realization aspects, implementation results and the deployment are shown in Section 5. Finally, the paper closes with conclusions and an outlook to future work.

2. Traffic measurement

In the literature as well as in practice traffic measurement is widely discussed and used. In the following, we will discuss the different aspects and the scope of traffic measurement, present its common fundamental functional building blocks and review the various approaches for the realization of traffic measurement on different platforms.

2.1. Principles

Traffic measurements differ with respect to the measured data as well as the measurement methods. The measurement purposes span a large scope and determine the methods to obtain the measured data. In the following, we will briefly discuss the fundamentally different measured data types, describe the main measurement methods and, finally, discuss the major measurement purposes.

Measured data is based on packets either from the entire traffic or from a filtered subset of it. It is classified to either unmodified sections of packets, e.g. protocol headers, or to processed values of them. Processed values are for example statistical data calculated from the packet data (e.g. histograms for packet sizes) or data on a higher abstraction level than packet-level (e.g. flow-level).

In addition to the measured data, also time information is recorded when each datum is collected. The required accuracy of this time information varies largely for different measurement purposes. This has to be carefully examined which is a challenging task for many purposes [7].

Measurement methods are classified into passive and active methods. Passive measurements collect traffic without impacting the traffic of the observed links. Active measurements inject additional traffic on the link. Both methods are performed at either a single link or spatially distributed at multiple links simultaneously.

The major *measurement purposes* are network management, network control and traffic characterization and modeling. These purposes possess rather different requirements on traffic measurement.

For network management, measurements are performed for health monitoring, troubleshooting and accounting. The measured data is mainly counter statistics and flow data from multiple network nodes [8]. For network control various different data is measured by a broad range of methods depending on the specific needs. Common to these applications are that the measured data is usually not stored but immediately used for decisions or further processing steps [9].

The purpose of traffic measurement focused on in this paper is traffic characterization and modeling. Here, the main target is to enlarge the understanding of the traffic and derive appropriate traffic models for analysis, simulation or emulation. The requirements for such measurements vary heavily and depend on context and objectives of the analysis or the modeling. However, for an accurate understanding of the traffic behavior with its underlying principles, sets of measured data are necessary that are rich in detail and that cover a long time span. Such measured data is collected by passive measurements.

These measurements usually result in huge sets of measurement data, especially when recording high-speed links. So, special requirements are imposed on these measurements, which lie in particular on the correctness of the measured data with respect to time, completeness, storage and post-processing.

In the following we will focus on passive traffic measurement at high-speed links for the purpose of traffic characterization and modeling. Subsequently, we will simply use the term traffic measurement for it.

2.2. Functional blocks

Architectures for passive traffic measurement platforms for high-speed links consist in general of functional blocks that can be classified into four groups: blocks for reception of data from the network link, blocks for time stamping, blocks for pre-processing and blocks for storing data. The arrangement and organization of these blocks vary depending on the realization of the measurement platform as well as its purpose.

The first group consists of all blocks that support receiving data from the network by an interface. They are responsible for decoding the line signal including error checks and restoring the packets. Depending on the interface type, they also decapsulate packets from lower layer protocols. Finally, the output is a stream of packets.

The second group of blocks is responsible for time stamping. Hereby, a tag is assigned to each packet reflecting the instant of time at which the packet arrives at the measurement platform or at the time stamping unit.

The scope of such tags differs. In their simplest form, they reflect the time between two successive packets or relative to the start of the measurement. This is sufficient for short term measurements. Long-term measurements will often be correlated to external events like business hours. Thus, their meaning needs to be extended to reflect also the time of day. For distributed measurements at different locations they must be related to a global unique reference.

The achievable precision of the time stamps depends on the architecture of the measurement system as this limits the reachable accuracy. This is due to functional building blocks which are traversed before reaching the time stamping unit and that may introduce jitter. In contrast, the required precision depends on the application of the measurement as this presupposes a certain temporal precision of the entire measurement system.

The functional blocks within the third group (pre-) process passively or actively the packet data stream. Passive processing blocks only analyze the content of the packet while keeping the packet data stream unchanged. Results are commonly metadata or triggers for defined actions. Examples for such tasks are packet classification, or statistical evaluation. The active processing blocks also modify and manipulate the packet data stream. This is possible on data stream level, e.g. by removing unwanted packets, or on the packet level, e.g. by modifying the content of packets through anonymization or by removing unwanted higher layer payload by a filter.

The last group contains all functional blocks that store data. Here, we have to point out the wide ranges of data volume and data rate which must be handled. The data volume ranges from a few hundred bytes for accumulated data to several giga- or even terabytes for long-term traces of high-speed links. Respectively, the data rates range from a few bytes per second to several tens of megabytes per second.

2.3. Platforms

The challenging tasks for a measurement platform are to provide a time resolution in the order of the smallest data unit on the link layer as well as to reliably process and store the incoming data defined by the high-speed measured link. This becomes even more massively challenging as link speeds increase with a growth that exceeds Moore's Law [10].

Traffic measurement at high-speed links requires processing steps of low complexity however on a large amount of data. The described functional blocks in Section 2.2 can be implemented by using several different technologies and in various system configurations fulfilling the above constraint. Both aspects will be discussed in the following.

For processing data in a measurement platform several device technologies with different properties are possible. General-purpose processors (GP-CPU) provide large flexibility. However, they are primarily designed for complex processing on a small set of data and do not fulfill the requirements for high I/O throughput.

Network processors (NP) are devices which are designed for high I/O throughput and optimized for packet processing tasks. They can be programmed with primitive programming languages [11].

Field Programmable Gate Arrays (FPGA) and application specific integrated circuits (ASIC) allow specialized logic designs for the different functional blocks. As the designer can develop system modules on different abstraction layers, he is able to optimize selected modules. Both device types – FPGAs and ASICs – are very well suited to achieve the required performance and accuracy for high-speed links. FPGAs are fast and easily reconfigurable, while ASICs provide the highest processing speeds.

For storing the measured data also different technologies can be employed. The usage of hard disks (HD) provides very large storage capacities in the range of terabytes and the I/O throughput of a single HD is in the area of several hundreds of megabits per second. Random access memory (RAM) modules have a higher I/O throughput of several tens of gigabits per second, however, offer only storage capacity in the range of few gigabytes. This is too little for long-term measurements with measurement data rich in detail.

The above described technologies for the functional blocks can be composed in different ways to measurements platforms. Two classes are deployed in practice: platforms with dedicated measurement hardware (HW) and platforms without dedicated HW.

Commodity PCs equipped with a GP-CPU, a network interface card (NIC) and HDs can be regarded as measurement platforms without dedicated measurement HW. They are often used as they are widely available at moderate prices. This is a very flexible approach with large storage capabilities. Unfortunately, it can only be used for links with small to medium-sized bandwidth or for only very lightly loaded high-speed links [7,10,12]. The main limitations are a rather poor time resolution and time accuracy as well as insufficiently reliable throughput at high-speed links.

Several approaches have been developed to overcome these limitations, while keeping the flexibility of a commodity PC and its GP-CPU. Such systems use additional acceleration HW that adds higher I/O bandwidth and

additional processing power for traffic measurement. These are realized as additional boards equipped with receiving units, little storage memory and processing units implemented in NPs, FPGAs or ASICs. They are commonly connected via the peripheral component interconnect (PCI) bus and write the measurement data via direct memory access.

Acceleration HW realized with NPs have only little instruction memory and have to be programmed with primitive programming languages and with insufficient tool support [13]. Platforms based on an additional FPGA board, e.g. DAG cards [14] or SCAMPI adapters [15], are very powerful and very well suitable for long-term passive measurement.

However, common to these approaches is that the additional HW is closely coupled via the PCI bus to the commodity PC. This close coupling results in major difficulties to scale the system to higher link rates and showing higher down-times when replacing parts. Even with the introduction of PCI Express, enabling higher I/O throughput in the area of serval gigabits per second, still the required high-capacity HDs do not support such fast writing throughput. Further, such closely coupled platforms require high-end server components for achieving the needed throughput, thus, resulting in significantly higher costs.

The major limitation of these platforms is their lack of parallelism, thus, having only limited capabilities to scale. The performance is limited by the bottleneck of the overall system. It cannot be replaced by multiple instances to ease this bottleneck as is possible in an inherently parallel solution where parallelism was considered already in very first design phase.

In the following we will introduce an architecture for a scalable, high-speed passive measurement platform which is based on a dedicated HW-unit and a commodity PC. Due to its modularity and the loose coupling of its components, the architecture is easily scalable to higher data rates by increasing the parallelism of its modules.

3. I²MP – concept and architecture

The I²MP is a high-speed, passive measurement platform. In the following we will detail the objectives of the I²MP and its design principles leading to the main architecture of our measurement platform. After that we describe its core components: the HW-unit and the PC-unit.

3.1. Concept and overall architecture

We designed our traffic measurement platform I²MP to capture packet traces on high-speed network links with high time precision. We tailored its concept to long-term measurements lasting hours or even days.

For measurements in high-speed networks, the question arises how much data need to be stored. One extreme is to store the entire packet stream. This is not necessary as payload is rarely analyzed. Furthermore, it is usually not permitted due to privacy issues. The other extreme is to calculate statistics online and to drop all collected packets without recording them. This is not sufficient as often in research, the traffic will be analyzed by many researchers with a broad scope of analysis scenarios and thus detailed traffic traces are needed.

We positioned the I²MP between these two extremes. It captures all incoming packets, however, its design allows arbitrarily selecting byte areas within a packet for further processing or storage while dropping the remaining data. By this the amount of data can be reduced while retaining all relevant information. In order to enable the definition of byte areas relative to the start of protocol headers and independent of their absolute position in the packet, we further designed the selection to be protocol-aware. This especially simplifies handling of packets with protocols with non-constant header length as well as tunneled protocols.

For the realization, one can use the different platforms as introduced in Section 2.3. As explained, customized hardware provides a predetermined temporal behavior which cannot be achieved by pure commodity PC based systems. Furthermore, processing power is tailored to dedicated tasks. Both come at the drawback of a higher implementation complexity. Therefore, in our architecture, we analyzed each functionality with respect to its timing requirements as well as the required processing power. Based on this, we assigned each functionality explicitly to either a hardware module or a software running on a PC. We further separated the functionalities implemented in software according to their execution time to online and offline tasks.

Therefore the I²MP consists of two parts, the HW-unit and the PC-unit. Its basic architecture is depicted in Fig. 1.

In the HW-unit, the performance- and time-critical functions are realized. These are the reception of packets at line speed, high-precision time stamping and intelligent classification and filtering. The filtered packets and their metadata, comprising time stamp and classification results, are assembled to data containers and transmitted to the PC-unit.

The functions of the PC-unit can be divided into online and offline tasks. The former comprise the reception of the data containers and the reliable storage of the contained data records on hard disks. The latter are mainly post-processing functions on the received data records. Apart from this, the PC-unit controls and configures the HW-unit.

This modularization at the hardware and software level necessitates the definition of formats to exchange data. We interconnect the HW-unit and the PC-unit by Ethernet via standard network interfaces. This loose coupling allows a flexible way for optimizing each unit separately. Furthermore, it enables a scalable system where HW-units and/or PC-units can be used to increase the processing/storage capacity, respectively. The software modules exchange data in the widely used pcap format.

Having the entire system running, usability is a critical aspect. On the one hand, users must be able to easily adapt the I²MP to their present scenario. On the other hand, introducing flexibility by means of configuration option increases also the complexity of soft- and hardware modules and makes the entire system more error-prone. This is in

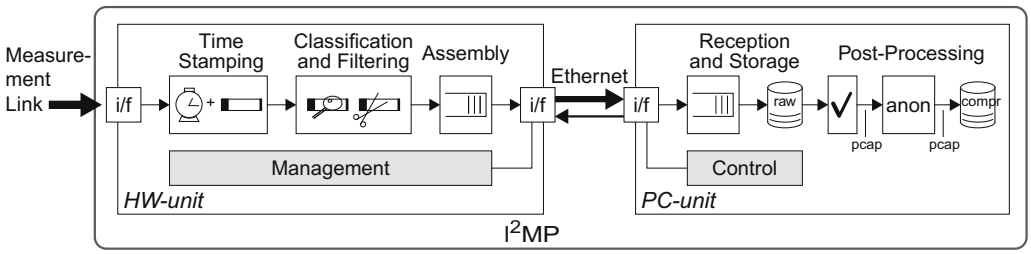

Fig. 1. Architecture of I²MP.

direct contradiction to reliability and stability which is of major impact in long-term measurement scenarios.

Therefore we thoroughly analyzed all relevant use cases. Based on this we defined the set of parameters which can be changed by a user simply, quickly, and without downtime. Such parameters are for example rules for traffic classification and filtering as well as the routing of collected data to storage servers.

Finally, scalability is a major design aspect that impacts on all parts of the design. To allow scaling to higher data rates, we designed all modules such that they need only state information for the packet that is currently processed. By that several modules or groups of modules can be run in parallel to achieve a higher throughput.

3.2. Hardware unit

The HW-unit performs the time-critical tasks of the measurement platform. As depicted in the left part of Fig. 1, it is organized as a functional pipeline. It consists of the line interface to the measured link, the Time Stamping unit, the Classification and Filtering unit, and the Assembly unit with the interface to the PC-unit.

The line interface is connected to the network at the location where the traffic is to be measured. As soon as the line interface unit starts with the reception of a packet, it triggers the succeeding Time Stamping unit to record the exact time of arrival. The interface unit buffers the incoming packet until it is completely received and its integrity is approved. Then the packet is forwarded to the Time Stamping unit. In case of an erroneous packet it notifies the following unit to discard the recorded time stamp.

The Time Stamping unit houses an internal counter that is incremented in constant time-intervals. As soon as the line interface signals the arrival of a packet, it saves the current value of its counter. When this packet is passed from the interface unit, the Time Stamping unit attaches that value (time stamp) to the packet.

The Classification and Filtering unit is the next stage of the functional pipeline. This unit is the core component of the HW-unit and contributes to the highest degree to the outstanding flexibility of our measurement platform. This unit performs three tasks. Firstly, it decodes the protocol stack of each packet. The number and order of the stacked protocols can be arbitrary, as long as all protocols have been specified in this unit. Secondly, it classifies the packet according to a configurable rule set. And finally, this unit filters each packet based on the results of decoding and classification. It discards irrelevant parts of the packet, e.g. parts of the header or the payload, and puts the remaining sections as well as the classification result and the time stamp into a data record. This data record is then forwarded to the following stage.

As protocol definitions are constantly evolving and new protocols are introduced very often, the traffic measurement unit must be adaptable to changing classification and filtering requirements. Hence, the Classification and Filtering unit supports an easy configuration of the classification rules as well as an easy adaptation to new or modified protocols. How this flexibility is achieved is documented in more detail in Section 3.3.

The Assembly unit buffers the data records forwarded by the Classification and Filtering unit. It fills the incoming data records in so called data containers and finally sends them to the PC-unit via the succeeding Ethernet interface. The collected data records are transmitted to the PC not individually but assembled as data containers in order to decrease the critical burden of interrupt handling and context-switching on the PC to a manageable level.

Finally, the interface to the PC-unit is also used to receive control and configuration messages. These messages are forwarded to the Management unit that decodes these messages and configures the addressed units of the HW-unit correspondingly.

3.3. Classification and Filtering unit

The task of the Classification and Filtering unit is to classify each packet according to a given rule set and to extract the relevant parts from the packet headers according to the classification result. The unit comprises the following three blocks, as illustrated in Fig. 2: The Protocol Layering Decoder, the Classifier and the Filter.

Within the entire Classification and Filtering unit the packet data is processed and forwarded in data words of fixed width. The main task of the Protocol Layering Decoder is to interpret the relevant header fields of each protocol in order to determine the contained protocols of the packet and their absolute position within the packet. Furthermore, it aligns each protocol header with a word boundary in order to ease the subsequent classification. The decoder results are forwarded to the Classifier and the Filter. The Classifier maps each packet to a certain category. Based on this, the Filter finally selects specific parts of the packet and discards the rest.

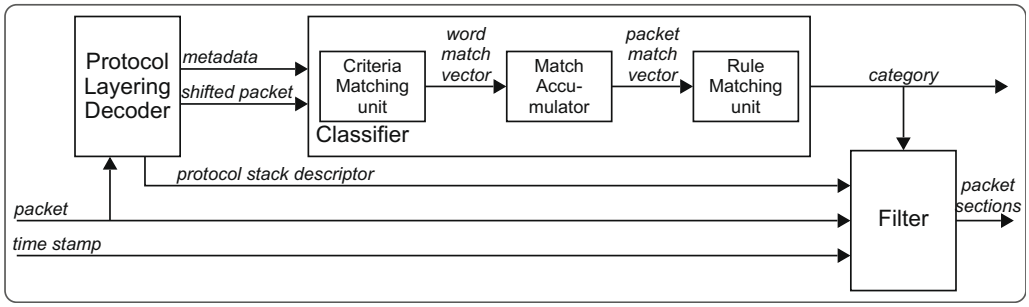

Fig. 2. Basic structure of the classification and filtering unit.

In the following subsections, the Protocol Layering Decoder, the Classifier and the Filter are described in more detail. Besides their concept and functionality also their adaptability to new or changed requirements are presented.

3.3.1. Protocol Layering Decoder

The data of a received packet at the traffic measurement platform can be distinguished into payload and protocol data. The protocol data represents the hierarchical encapsulation of the payload according to the protocol layers, by adding headers (and/or trailers) to the payload.

The main task of the Protocol Layering Decoder is to decode the protocol data in order to determine at which positions in a packet the different, contained protocol headers are located. To achieve this it only requires two pieces of information from each contained protocol: the length of its header and the protocol which it encapsulates. Each protocol must either specify this data in dedicated header fields of each packet or must have fixed, pre-defined values. The Ethernet header has for example a fixed, pre-defined length of 14 Bytes and the encapsulated protocol is specified in Byte 13 and 14 of its header.

The decoding information about each protocol is microprogrammed [16]. Each line of microcode corresponds to one single protocol and contains all necessary information for decoding that protocol. The code specifies if the length is fixed or variable. In the former case it specifies the fixed length, in the latter case the byte position and the width of the length field. Similarly for the encapsulated protocol, the microcode word specifies if it is fixed or variable and, respectively, either a fixed protocol identifier or the position and width of the respective header field.

When decoding a packet, the Protocol Layering Decoder starts with the microcode of the known lowest layer protocol. With this it extracts the required information (length and encapsulated protocol) of the header fields or uses the pre-defined values, respectively. By evaluating the length information it can then determine the start of the next encapsulated protocol. By using the information of the type of the encapsulated protocol, it can determine the line of microcode for this protocol by a fast table lookup, and thus is able to decode the encapsulated protocol layer in the same way. In this way, all stacked protocols can be decoded, as long as the protocols used have been configured in the microcode memory. Even the handling of protocols with a variable header length and tunneled protocols is possible.

As the microcode memory as well as the lookup table mapping the protocol types to their corresponding line of microcode are realized by random access memory, they can be modified by the user. By this the Protocol Layering Decoder is easily adaptable to changing or new protocol definitions, as long as they specify the header length and their encapsulated protocol, as described above.

Fig. 3 illustrates the processing of a packet by the Protocol Layering Decoder. It signals each protocol used as well as its absolute start and end position within the packet to the Filter unit. Furthermore, it labels each data word of the packet with an identifier of the protocol it belongs to and aligns it such that each protocol starts at a word boundary. Additionally, it numbers each data word with its position relative to the beginning of that protocol. The shifted data words and its metadata are then output to the subsequent Classifier. More technical details about the Protocol Layering Decoder can be found in [17].

3.3.2. Classifier

Packet classification is one of the major tasks of modern packet processing systems and it is widely investigated by researchers. Different classification methods have been introduced in recent years. They can be coarsely divided into three groups: approaches using decision trees, searches in tuple spaces and decomposition techniques [18]. We used an approach of the third group that is similar to the Bit Vector scheme [19].

In our Classifier, the determination of the category is based on rules. Such a rule combines several criteria that a packet must fulfill to be mapped to a certain category. Each criteria corresponds to a matching expression of one or several bytes within a certain word of a header. This is easy to realize due to the shifting and labeling of the data words by the Protocol Layering Decoder. For the user, this makes the definition of criteria easy and compact. The classification criteria are defined on behalf of different characteristics. They can refer to specific bit patterns or byte values, e.g. a port number or protocol type, but also to numerical ranges of certain byte fields, e.g. ranges of IP addresses or subnets.

The packet data is processed word by word, as illustrated in Fig. 4. By taking the metadata of each word into account, the Criteria Matching unit of the Classifier

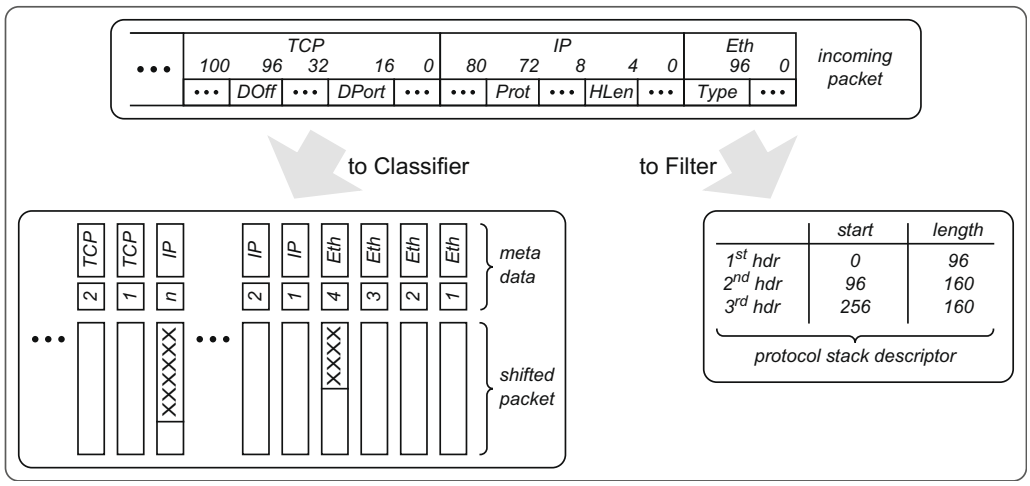

Fig. 3. Generation of metadata by decoding the protocol data.

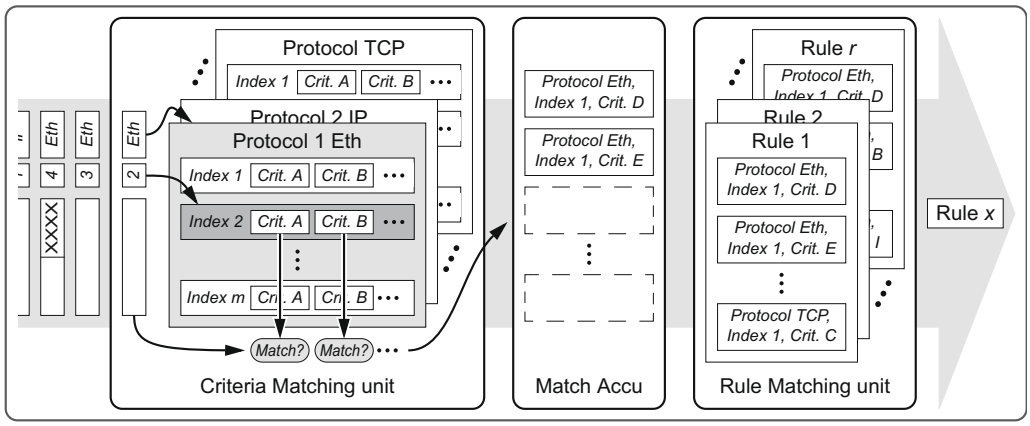

Fig. 4. Functionality of the Classifier: criteria and subsequent rule matching.

compares each word only to those criteria that correspond to the current protocol and position within the current header. All relevant criteria are compared in parallel to the data word. This can be done by using TCAM (ternary content addressable memory) technology. While comparing all data words of the packet to their corresponding criteria, the Match Accumulator records all matching criteria.

After this criteria check the Rule Matching unit of the Classifier compares the record with all matching criteria to all rules in the rule database, again using TCAM technology. By this it finds the rule with the highest priority that is fulfilled by the packet. The category associated to this rule is then forwarded to the Filter unit as well as to the following stage of the HW-unit, the Assembly unit, to be recorded for post-processing steps. For a more technical description of the Classifier and its classification process please refer to [17].

3.3.3. Filter

Based on the found category of the Classifier and the metadata from the Protocol Layering Decoder the Filter reduces the amount of packet data to the required data blocks. The Filter selects specific parts of the packet to be further processed in succeeding stages of the measurement platform and discards the rest.

The classification category is used to look up which parts of the packet to select. The table containing the selection ranges is realized as random access memory and thus easily configurable by the user of the measurement platform. The definitions which parts of the packet to select are expressed as Byte positions relative to protocol boundaries. This has the advantage of being independent of variable header lengths, inserted shim headers or protocol tunnels. The exact location of these protocol headers within the packet is conveyed by the metadata from the Protocol Layering Decoder.

3.4. PC-unit

The main tasks of the PC-unit are the reception of the data containers, post-processing of the data records and storage of the final trace. As these different tasks underlie different time and throughput constraints the PC-unit works in different modes: online and offline.

During the online mode, the PC-unit receives the data containers sent by the HW-unit and stores them without any processing. Furthermore, several statistics and counters are maintained to validate the completeness, integrity as well as to prepare plausibility checks of the data records which will be done in the offline mode.

In the offline mode, the PC-unit post-processes the stored data in order to build the compressed final trace. This contains validation and plausibility checks, format conversion and anonymization of the data stored during the online mode. These post-processing steps are configurable and adaptable to the needs of the current measurement scenario, especially anonymization and format conversion. The plausibility checks are performed to validate the completeness and correctness of the trace mainly by means of sequence numbers of the data containers and byte counters of processed data-records at different processing points.

The checked records with their corresponding metadata are converted to the widely used trace format pcap [20]. The pcap format enables the application of a large number of different, widely available post-processing tools working on the trace.

The anonymization ensures that the final trace contains no payload beyond the transport layer. Furthermore, the addresses (IP addresses and transport layer port numbers) are anonymized configurably either in prefix-preserving manner or according to their appearance in the trace. The prefix-preserving anonymization enables a consistent anonymization across multiple measurement sessions and is based on a keyed-hash method [21]. Finally, the checked, converted and anonymized trace is compressed and stored on hard disks.

Beyond its main tasks, the PC-unit controls the HW-unit during online and offline modes. It starts and stops the collection of data records of the HW-unit, reads out statistic counters and configures the Classification and Filtering unit by setting the criteria and rule definitions as well as filtering sections.

4. Scalability

Network data rates will continue to increase so it is important to examine the presented architecture with respect to its capability to support higher data rate processing. Also, more and more protocols and network services emerge. This means more filtering categories and consequently more criteria to check in the Classifier of our measurement platform.

In the following we show the theoretical scalability of our architecture concerning both the above mentioned issues. After that, results from a numerical scalability analysis are shown, that support the theoretical results.

4.1. Theoretical scalability

In the following we will first discuss the theoretical scalability of the throughput. In the second subsections we detail the effects when scaling the number of classification rules within the Classification and Filtering unit of our measurement platform.

4.1.1. Throughput

The throughput of a measurement platform characterizes the data rate, measured in bits per second. Both the HW-unit and the PC-unit (in online mode) of the I^{2}MP have to support the required throughput.

The PC-unit is easily scalable to higher data rates, by simply attaching several PC-units to the HW-unit using a switch. In this configuration the HW-unit has to distribute its data containers evenly between the attached PC-units, e.g. in a round robin fashion, and recombined in the offline mode based on the containers' sequence numbers.

For scaling the HW-unit to a higher throughput three approaches or a combination of them can be considered. In the following we summarize the results of our detailed scalability analysis presented in [17].

One approach to speed up the throughput of the system is to increase the clock rate f_{clk} of the system. An ASIC solution would allow clock rates up to one GHz whereas future high performance FPGAs support several hundreds of MHz.

The other approach is to increase the number of bytes processed in each clock cycle. This can be done in two ways: increasing the internal word width or parallelization of entire units.

The possibility to use a larger internal word width is not indefinitely, reasonably applicable, especially for the Classification and Filtering unit. The reason for this is, that this unit processes the packet data of two consecutive protocols in two consecutive words, even if they would fit into one single word. Therefore, increasing the word width is only reasonable up to a width comparable to the length of usual packet headers (e.g. 20 Bytes).

The other possibility to process more data per clock cycle and therefore to increase the system's throughput is to replicate the entire system (or those parts of it that limit the overall throughput) on the chip and to distribute the packets evenly between the replicated units. No changes have to be made to the processing units as the processing of each packet is independent of the other packets. This approach would only necessitate additional buffers after splitting the packet stream and before recombining the measured data.

Another possibility to increase the supported line rate of the platform, would be to only process the first part of each packet (e.g. the first 64 Bytes). By this the internal data rate does not have to be as high as the external line rate. However, this would only increase the average throughput of the platform, not its worst case performance, when only small packets are received.

All of the above approaches are good candidates for increasing the system's throughput. However increasing the parallelism by replication is the most promising approach to scale the throughput of the measurement platform to very high data rates, as this approach is neither limited by the clock speed, nor architecturally, nor statistically.

4.1.2. Classification criteria and rules

Another important aspect, when evaluating the scalability of a measurement platform is its ability to support a high number of classification rules, and thus also a high number of classification criteria.

Our detailed scalability analysis in [17] shows that the architecture of the I²MP can be easily scaled with respect to the number of classification rules and criteria. Increasing those parameters does not deteriorate the system's throughput. Its delay increases only slightly.

We also analyzed the effect on the required hardware resources when scaling to more rules and criteria. We found that the resource utilization of the Classification and Filtering unit scales linearly with the number of rules or criteria, respectively.

4.2. Numerical scalability analysis

In order to validate the scalability analysis of our Classification and Filtering unit we configured several differently dimensioned variants of our VHDL implementations of the Protocol Layering Decoder and the Classifier. We synthesized, mapped, placed, and routed these variants for the configuration of a Xilinx Virtex IV FPGA [22] and compared their resource utilizations.

In our experimental implementations we varied the number of classification criteria between 64 and 512 and the number of rules between 16 and 256. In order to achieve a high throughput we used a word width of 16 Bytes in all tests.

The results we obtained support our theoretical statements in Section 4.1. The resource utilization of the Classification and Filtering unit increases linearly with the number of supported criteria. Also, the resource utilization shows a linear dependency on the number of supported rules. For more details refer to [17].

All our experimental implementations supported a maximum clock rate f_{clk} of 60 MHz. This also supports our claim from Section 4.1.

This clock rate results in an internal data rate of approximately 7 Gb/s. Assuming Internet traffic with a mean packet size of 200 Bytes our traffic measurement platform I²MP would support a throughput of approximately 20 Gb/s, when processing only the headers (Ethernet, IP, TCP) of the packets [17]. Furthermore, by replicating the slowest modules within the measurement platform even higher throughputs should be achievable.

5. Prototype and deployment

We built up the described measurement platform I²MP and used it for traffic measurements in one of our campus student networks. In the following we will first detail the prototype implementation of the HW-unit on our FPGA-based hardware platform. Then we describe the realization of the PC-unit using a PC and several hard disks. Finally we give a short overview about the deployment of the I²MP.

5.1. Implementation of the Hardware unit

The functionality of the HW-unit has been implemented in VHDL. We developed the VHDL design to be as generic and modular as possible. This allowed us to easily synthesize differently scaled versions for the numerical scalability analysis (see Section 4.2). Further, we could synthesize the design for the FPGA families from both major vendors Altera and Xilinx.

We built up the HW-unit on our FPGA-based universal hardware platform (UHP) [23]. The version of the platform used provides a rather old FPGA from Altera (APEX20K 400CB652-C7, [24]) and several communication interfaces, like electrical and optical Gigabit Ethernet interfaces. We used two Gigabit Ethernet interfaces, one for capturing the traffic and one for communicating with the PC-unit for the transmission of the captured measurement data and control tasks.

The time stamping unit in the HW-unit uses a 64-bit counter, incrementing at a frequency of 125 MHz. This results in a time resolution of 8 ns which corresponds to the byte-time of Gigabit Ethernet.

The following parts of the measurement platform, i.e. the Classification and Filtering unit and the Assembly unit, operate at a clock frequency of 35 MHz and use a word width of 32 bits. The Classifier supports 32 classification categories based on 32 different criteria, which is an ample amount for the pre-processing within a measurement platform.

With this configuration more than 2 million minimum-sized Ethernet/IP packets can be recorded per second. Thus with our architecture of the measurement platform it is still possible to perform all processing tasks, including the classification, at more than full Gigabit Ethernet line speed, despite the rather old FPGA technology. The complete HW-unit fits well even on the outdated FPGA we used, only occupying two thirds of the available resources.

As modern Altera FPGAs do not support an efficient realization of TCAMs within the FPGA anymore, higher throughput Classifiers cannot be implemented easily on Altera FPGAs. So future versions of the I²MP will be based on Xilinx FPGAs, that we have already used successfully in our numerical scalability analysis.

5.2. Implementation of the PC-unit

On the PC-unit, a stripped Linux distribution runs where only the needed services and applications are active. Also, we adapted kernel and NIC variables mainly to enlarge the networking related kernel buffers. The PC-unit used is equipped with an 2.53 GHz CPU, 3 GB RAM, multiple removable HDs and a Gigabit Ethernet NIC. We use a DCF77 receiver for time synchronization.

The online tasks, i.e. the reception, processing and storage of the data containers, are performed by in-house, thread-based software. The reception thread receives the data containers sent by the HW-unit via a low-level packet socket and places them in a large cyclic buffer located in the RAM. To each HD in the PC-unit a separate dumping thread is associated. Each dumping thread writes the data containers contained in the cyclic buffer to the associated HD where the buffer is processed in a round robin fashion. We use Linux raw-devices to write the data block-oriented and unbuffered to the HDs, thus, avoiding additional delay and potential loss by the file system.

The offline tasks are correctness checks, format conversion and anonymization which are realized by a tool chain producing a pcap [20] trace file. We realized the first two

tasks by in-house software. This software reads the previously dumped data records from the HDs and performs checks to verify the completeness of the data records. Further, the software converts data records to the pcap format where pcap time-stamps are used in nanosecond resolution. For anonymization, we used an in-house modified tool based on TCPdpriv [25]. We extended the software to add prefix-preserving IP address anonymization based on a keyed-hash method [21] and the additional possibility to apply different anonymization methods independently to source and destination addresses. Finally, the resulting trace in pcap format is compressed by a standard compression tool of the Linux distribution.

5.3. Deployment

We deployed the measurement platform I²MP in the large dormitory network "Selfnet" of the University of Stuttgart with approximately 1000 actively connected students. The students extensively use a wide variety of services (e.g. web, email, e-learning, gaming, e-commerce, voip, video conferencing) and new emerging Internet services are quickly adopted. Each student has a 100 Mbit/s access rate and, at the time of the collection, the uplink bandwidth was 100 Mbit/s. We measured the traffic at the Internet uplink of the dormitory network over multiple days. These captured traces where also used in the IST project NOBEL [26].

6. Conclusion

We presented a scalable architecture for a high-speed, passive measurement platform that can be used for obtaining highly accurate packet traces for traffic characterization and modelling.

In order to achieve an optimal performance, we allocated the required functional blocks for traffic measurement to a specialized HW-unit and a commodity PC-unit according to the requirements concerning temporal accuracy, throughput, flexibility and storage capability. Both units are interconnected by standard network interfaces.

The HW-unit features high-precision time stamping as well as highly adaptable classification and filtering. The latter is used for mapping each packet to a category and, based on this, to select certain sections of the packet while discarding the rest for reducing the data volume to be stored. This unit is easily adaptable to new or changed protocol definitions as well as user-defined classification categories and filter rules. Its protocol awareness allows the easy classification of arbitrarily tunneled protocols and protocols with variable header lengths. Furthermore, it simplifies the definition of classification and filter rules as they can be described relative to protocol boundaries instead of absolute positions within the packet. The PC-unit receives and post-processes the measured data, ensures its anonymity, and stores the results on hard disks in a standard trace format.

We implemented the measurement platform using an FPGA-based hardware board and a standard server PC. The logic design of the HW-unit could be easily fitted even into an outdated FPGA and supports the measurement of Gigabit Ethernet links at full line speed. The platform was deployed for long-term measurements of the traffic of a large local dormitory network.

The architecture presented is easily scalable to higher data rates. Furthermore the classification unit is also scalable to larger numbers of classification categories and is therefore also suited for other classification purposes besides its current application in the measurement platform.

Our numerical scalability analysis by exemplary configurations of the HW-unit for higher data rates and different numbers of classification categories supports the results of our scalability analysis. Furthermore it shows that with current FPGA technology the architecture presented is in principle capable of handling data rates of up to 20 Gigabit/s.

Therefore an implementation of the measurement platform for measuring links with 10 Gigabit Ethernet is planned. Furthermore we see no principle limitation of our architecture to be scalable even to 100 Gigabit/s data rates.

Acknowledgements

The authors would like to thank Sascha Junghans for his support, contributions, and the valuable discussions. This work was partly funded by the European commission through IST NOBEL (FP6 507509) and by the German Ministry for Research and Education (BMBF) within the EIBONE Project Under Contract 01BP566.

References

[1] G. Eilenberger (Ed.), Multi-layer transport networks with integrated control, Position paper by the EIBONE working group on network aspects, 2008.
[2] W.E. Leland, M.S. Taqqu, W. Willinger, D.V. Wilson, On the self-similar nature of ethernet traffic (extended version), IEEE/ACM Transactions on Networking 2 (1) (1994) 1–15.
[3] D. Veitch, N. Hohn, P. Abry, Multifractality in TCP/IP traffic: the case against, Computer Network Journal, special issue Long-Range Dependent Traffic 48 (2005) 293–313.
[4] C. Park, F. Hernandez-Campos, J.S. Marron, F.D. Smith, Long-range dependence in a changing Internet traffic mix, Computer Networks 48 (3) (2005) 401–422.
[5] T. Karagiannis, M. Molle, M. Faloutsos, Long-range dependence: ten years of Internet traffic modeling, IEEE Internet Computing 8 (5) (2004) 57–64.
[6] M. Burke, Ellacoya data shows web traffic overtakes peer-to-peer (P2P) as largest percentage of bandwidth on the network, Media Alert, June 2007. <http://www.ellacoya.com/news/pdf/2007/NXTcommEllacoyaMediaAlert.pdf>.
[7] J. Micheel, S. Donnelly, I. Graham, Precision timestamping network packets, in: Proceedings of the ACM SIGCOMM Internet Measurement Workshop (IMW-01), ACM Press, New York, 2001, pp. 273–280.
[8] H.-G. Hegering, S. Abeck, B. Neumair, Integrated Management of Networked Systems: Concepts, Architectures, and their Operational Application, Morgan Kaufman Publishers Inc., San Francisco, CA, USA, 1998.
[9] L. Burgstahler, Bewertung von Mess- und Abschätzverfahren zur Unterstützung dienstgüteorientierter Verkehrslenkung in verbindungslosen Datennetzen, Dissertation, University of Stuttgart, Stuttgart, 2007.
[10] J. Coppens, S.V. den Berghe, H. Bos, E.P. Markatos, F.D. Turck, A. Øslebø, S. Ubik, SCAMPI: a scalable and programmable architecture for monitoring gigabit networks, in: A. Marshall, N. Agoulmine (Eds.), MMNS, Lecture Notes in Computer Science, vol. 2839, Springer, 2003, pp. 475–487.
[11] D.E. Comer, Network Systems Design using Network Processors, firstst ed., Prentice Hall International, 2006.
[12] L. Deri, Improving passive packet capture: beyond device polling, August 12, 2004.

[13] L. Deri, Passively monitoring networks at gigabit speeds using commodity hardware and open source software, in: Proceedings of the Passive and Active Measurement Workshop (PAM), 2003.
[14] Endace Measurement Systems, DAG cards. <http://www.endace.com/>.
[15] J. Coppens, E. Markatos, J. Novotny, M. Polychronakis, V. Smotlacha, S. Ubik, SCAMPI – a scalable monitoring platform for the internet, in: Second International Workshop on Inter-Domain Performance and Simulation (IPS 2004), 2004.
[16] S. Vassiliadis, S. Wong, S. Cotofana, Microcode processing: positioning and directions, IEEE Micro 23(4) (2003) 21–30.
[17] S. Hauger, S. Junghans, M. Köhn, D. Sass, A scalable architecture for flexible high-speed packet classification, Technical Report, Universität Stuttgart, Institute of Communication Networks and Computer Engineering (IKR), 2006.
[18] D.E. Taylor, Survey and taxonomy of packet classification techniques, ACM Computing Surveys 37 (3) (2005) 238–275.
[19] T.V. Lakshman, D. Stiliadis, High-speed policy-based packet forwarding using efficient multi-dimensional range matching, SIGCOMM Computer Communication Review 28 (4) (1998) 203–214.
[20] V. Jacobson, C. Leres, S. McCanne, pcap – Packet Capture library, 2003. <http://www.tcpdump.org>.
[21] J. Xu, J. Fan, M. Ammar, S. Moon, Prefix-preserving ip address anonymization: measurement-based security evaluation and a new cryptography-based scheme, in: Proceedings of the 10th IEEE International Conference on Network Protocols 2002, 12–15 November, 2002, pp. 280–289.
[22] Xilinx, Inc., Virtex-4 Family Overview, 2007. <http://www.xilinx.com/support/documentation/data_sheets/ds112.pdf>.
[23] Institute of Communication Networks and Computer Engineering, The Universal Hardware Platform (UHP), 2005. <http://www.ikr.uni-stuttgart.de/Content/UHP/>.
[24] Altera Corporation, Apex 20K Programmable Logic Device Family, 2004. <http://www.altera.com/literature/ds/apex.pdf>.
[25] G. Minshall, TCPdpriv Command Manual, 1996.
[26] IST Project NOBEL. <http://www.ist-nobel.org/>.

Detlef Saß received his Diploma degree (dipl.math.) in mathematics from the University of Stuttgart in 2000. During 2001–2002, he was with DeTeLine (a subsidiary of Deutsche Telekom), as a member of the technical and consulting staff focused on the area of IP telephony systems. Since 2003 he joined the Institute of Communication Networks and Computer Engineering, University of Stuttgart, as member of the research staff. His major research interests include the modelling, charaterization and measurement of network traffic in transport networks and in the Internet.

Simon Hauger received his MSc degree on Artificial Intelligence and Signal Processing from the University of Surrey, UK in 2003 and his Diploma degree (Dipl.-Ing.) in Electrical Engineering and Information Technology from the University of Stuttgart, Germany in 2004. Since then he is a member of the research staff at the Institute of Communication Networks and Computer Engineering, University of Stuttgart. His major research interests include the realization of high-speed network nodes, in particular with Field Programmable Gate Arrays and network processors.

Martin Köhn received his Diploma degree (Dipl.-Ing.) in Electrical Engineering and Information Technology from the University of Stuttgart in 2002. Since then he is a member of the research staff of the Institute of Communication Networks and Computer Engineering, University of Stuttgart. Since 2006, he is head of the research group on optical high-speed networks. His major research interests include traffic engineering and network dimensioning of dynamic multi-layer transport networks.