Network Processors: Issues and Prospectives

Stamatis Vassiliadis, Stephan Wong, Sorin Coţofană
Computer Engineering Laboratory,
Electrical Engineering Department,
Delft University of Technology,
Delft, The Netherlands
{Stamatis, Stephan, Sorin}@CE.ET.TUDelft.NL

Abstract

In this paper, we first discuss the need for network processors. Consequently, relying on a "program to program" communications model, we briefly describe the functions that are found in the seven OSI layers and classify them using functional requirement criteria for a network processor. Subsequently, we summarize the current state of the art commercially developed network processors. Based on technology trends, we propose a possible network processor architecture and design trajectory in order to tackle the problems current and future functional requirements. Finally, we conclude the paper with a list of important network processor open questions to be addressed in the near future.

1. Introduction

In recent years, we have witnessed an explosive growth of Internet usage which in turn has created the demand for higher transmission speeds as already massive yet still increasing amounts of data need to be transferred within a set time span. In order to meet this demand, wire-speeds have increased explosively and are currently reaching 10 Gigabits per second without any signs of a slowdown. At the same time, more computational power is needed to process data at these speeds. While the increase in computational power of general purpose processors is still obeying the Moore's Law, there is still a "gap" between processor processing and wire speeds (see Figure 1). In the past, this gap between wire-speeds and computational power of processors was bridged by the usage of application-specific integrated circuits (ASICs). They were computationally powerful enough to process the network data at wire-speeds and most likely will be able to do so in the future.

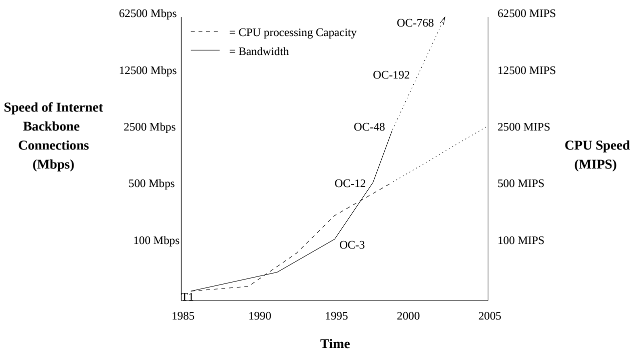

Figure 1. *Bandwidth versus Moore's law [1].*

In recent years, we are also witnessing two trends in the fast-changing environment of network processing:

- Standards within this environment, e.g., routing protocols, are changing quickly and the emergence of new standards in order to meet new demands is commonplace.
- Network equipment vendors need to incorporate increasingly more services and thus functionality into their products in order to distinguish themselves from others.

The usage of ASICs in this environment is becoming rapidly undesirable as they are slow to adapt to new requirements of the network processing environment as their design cycles usually requires at least 18 months. The currently available alternative using general purpose processors is also undesirable, because while general purpose processors provide the desired flexibility, they lack in computational speed as indicated earlier. Consequently, a new kind of processor is needed in order meet the requirements of this network environment. Such a need has sparked the emergence of the network processor (NP).

Currently, when considering network processors, it is immediately verifiable that there is no common agreement of what precisely a network processor is. There is a common agreement that a network processor is a communication integrated circuit, but other than that there is not a common definition of the precise functions it needs to perform. The reasons for such an apparent confusion are several, including: non-existing standards, non-specific functionalities (vast space covered), non-existing benchmarks, etc.. We will try here briefly to put issues in prospective and give some sense of motivation of what the directions such a research area could follow. This paper consists of the following sections. Section 2 sets up the pace of the discussion by briefly describing the 7 layer OSI reference model. Section 3 discusses the requirements of current and future network processors. Section 4 describes several of currently commercially available network processors. Section 5 propose a possible design trajectory and its related open questions.

2. Setting up the pace

Network processors are communication devices that have to operate on both heterogeneous hardware and software (i.e., dissimilar computers and software applications). In order to understand (or better stated: to classify) the network processor requirements, we believe that we should rely on a "program to program" communication model. To this end, we briefly describe the seven layers of the OSI[1] [2][3][4][5] reference model as it describes the modular building blocks and various functions to be performed enabling "program to program" communications in a heterogeneous hardware/software environment. Each layer within this model performs a given set of tasks that is required for communication to take place. We consider the bottom up approach for describing such a model.

[1]There exist other protocol stacks and see for example the TCP/IP protocol stack which only has 4 layers. Such a protocol stack contains functionality that can be found also in the OSI reference model and they are therefore not discussed separately. In all cases, in the design of the network processor and depending on its intended working environment, we must be aware of these other protocol stacks and their related functionalities.

Layer 1 (Physical layer): This layer is the foundation of communications between systems and is responsible for transmitting raw bits over a physical link. It specifies: transmission media (e.g., electric/optical cables or unbound media such as radio waves), transmission devices, connector specs, signaling schemes (e.g., analog/digital, signal levels, and signal encoding methods), and the network topology. Furthermore, data from layer 2 is being processed to make it suitable for transmission over the medium defined in this layer.

Layer 2 (Datalink layer): This layer is responsible for the transfer of data between the two ends of a physical link. Data from layer 3 (network layer) are 'framed' into a data organization known as the datalink frame which consists of the following fields: beginning/ending of frame fields, source/destination fields, the cyclic redundancy check field, administration or control field, and the data field. Furthermore, this layer resolves problems due to damaged, lost, or duplicate frames. From the functional point of view, layer 2 is broken down into two logical areas: Media Access Control (MAC) and Logical Link Control (LLC). MAC refers to media access protocol which basically describes how computers on a network gain access to the media and permission to transmit data (contention, token passing, polling). The source and destination fields are used for these purposes. The LLC includes portions of the administration or control fields and CRC field and is responsible for flow control[2], frame length calculations, protocol decisions, frame synchronization, and error checking.

Layer 3 (Network layer): This layer is responsible for the end-to-end routing of data to establish a connection for the transparent delivery of data, i.e., to reliably send and receive data between networks. The data unit in this layer are referred to as packets.

[2]The flow control manages how much data the destination system can handle. Furthermore, the flow control only controls the data flow between the two ends of the physical link. Flow control like routing in a large network is controlled in layer 3.

Each packet contains logical source and destination addresses according to a network protocol using which the packet can be routed through heterogeneous networks. We have to note that in layer 2 *physical* addresses (hardcoded into communication devices) are used to route dataframes and in this layer *logical* addresses are being used for the routing of packets. This layer also provides network routing of packets, flow control monitoring network connections, sequencing of packets, policing, and translation functions. More recent protocols (e.g., IPv6) also address issues such as authentication and security in this layer. Furthermore, the transmission of voice, video, and multimedia content requires the routing of packets without large delays/delay variances and a certain throughput must be ensured. These issues are addressed in the reservation protocol (RSVP) in IPv4 and the Quality of Service (QoS) parameters used in IPv6. Finally, as newer protocols are being rolled out, it is unlikely that the whole world will embrace the new protocols at the same time or with the same speed. Therefore, the coexistence of many protocols requires encapsulation of packets and in some cases translation of packets.

Layer 4 (Transport layer): While layer 3 ensures that data moves from one network to another (even between heterogeneous ones), layer 4 ensures that data arrives (or are reassembled) in one piece and that the data is directed to the appropriate service in higher layers. This layer provides: service addressing by identifying addresses/ports which point to upper layer (network) services, flow control using negotiation for window sizes, disassembling large data units from higher layers (= datagram segmentation), assigning fragmentation and conversation information, and end-to-end error checking.

Layer 5 (Session layer): This layer is responsible for connection establishment, data transfer, and connection release. Mechanisms are provided for establishing, maintaining, synchronizing, and managing communications between computer systems. Furthermore, it provides ID number with layer 4 providing the correct service, it coordinates acknowledgement numbering and retransmission procedures, it tracks order of initialization of conversations, it re-establishes a logical communications session if prematurely terminated (due to lower layer failure), and it releases communication sessions.

Layer 6 (Presentation layer): This layer is responsible for the translation of data into formats, e.g., big-endianness on one system versus little-endianness on another. Furthermore, this layers also encompasses compression and encryption.

Layer 7 (Application layer): This is the layer where network applications related to connectivity resides and service availability and advertisement are managed.

3. Network processing requirements

In this section, we establish the high level network processor requirements. Our goal is to postulate what possibly can be classified as a network processing device function and possibly to help formulate open questions. First, we extract from each layer the possible candidate functions. Then, we explore recent developments in services and their related required functions. Finally, a classification of the discussed functions is given.

Previously, we have discussed the seven layers from the OSI reference model that is required to establish communication between two network devices. Now, we identify the required functions that a network processor (could) need to perform from these layers (see also Figure 2).

Layer 1: Most functions performed in layer 1 are not to be included as they have little to do with actual direct processor operations.

Layer 2: Error checking, framing, table management of all physical addresses in current subnetwork, bitwise compares in order to determine whether address in frame is in current subnetwork.

Layer 3: Table management as possible routes or 'next hop' decisions for packets must be kept. Buffer management as: multiple buffers must be kept for storing prioritized packets, non-routable packets must be dropped, data must be kept in buffers when entering or leaving, and resources must be reserved in case of certain protocols. Authentication and security related functions. Encapsulation and translation functions. Sequencing functions. Address operations. Error checking related functions.

Layer 4: Error checking related functions. Service addressing. Data segmentation and re-assembly.

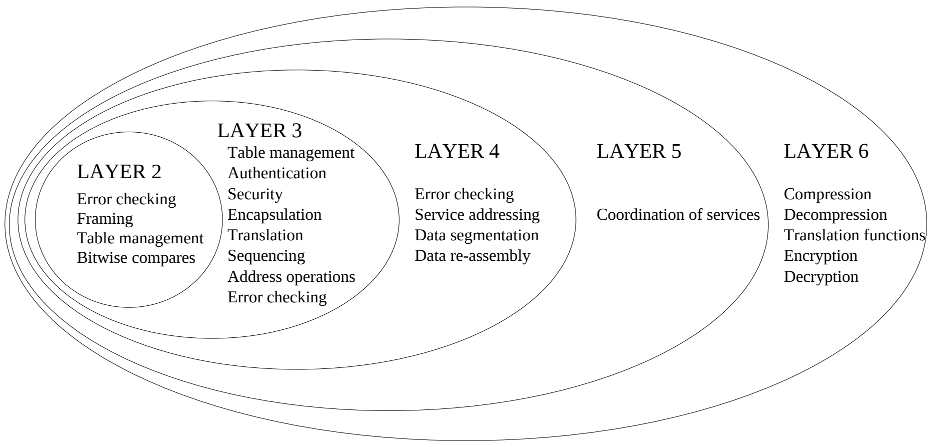

Figure 2. *Network processor requirements in each layer.*

Layer 5: Coordination of services and attributing the correct services can be possibly improved.

Layer 6: Translation functions. Compression and decompression functions. Encryption and decryption functions.

Layer 7: This layer is to be excluded as it is difficult to envision possible direct hardware support for the network applications. Thus, functions may not be included as a potential network processor wanted functionality.

Traditionally, network processing has been limited to purely routing of network packets over an internetwork[3]. In this scenario, information from up to layer 3 is used and their associated functions are supported. However, currently we are witnessing a shift in this paradigm in which increasingly more services need to be provided possibly utilizing information from higher layers. This increases the computational requirements of network processors as they need to support increasingly more functionalities:

- *Deep packet processing* in order to base routing decisions on the packet(s) contents.
- *Statistics gathering* for, e.g., billing services.
- *Authentication* and *encryption* at packet level or even lower levels as increasingly more users are connected to the Internet.
- *Translation* and *encapsulation* functions as the internetwork is becoming increasingly more heterogeneous.
- *Scalability* must be provided at a certain level in order to accommodate the continuing growth of (inter)networks.

[3]The term internetwork is used to describe a network of networks.

In putting all together a network processor, its functions can be categorized as follows[4]:

- Pattern matching: addresses comparisons, etc..
- Data manipulation: segmentation, re-assembly, framing, etc..
- Data management: tables, queues, etc..
- Recognition and classification: protocol determination, port number determination, etc..
- Compression
- Security: authentication, encryption, etc..
- Flow control.
- Routing functions: depending on protocol used.

Before we discuss a possible network processor design method, we discuss in the next section several commercially available network processors according to the vendor descriptions.

4. Existing network processors

Intel IXP 1200[6]: It consists of an integrated StrongARM core complemented with 6 integrated programmable microengines. The StrongARM core is

[4]Other subdivisions could exist. Actually, this categorization could be incorrect as it can NOT be currently substantiated using benchmarks and pre-described computer instructions. Current categorization is an open question.

responsible for the loading the instruction stores of the microengines and instructions for the StrongARM are loaded from memory. Each microengine supports multi-threading with four threads per engine, contains a 2K$\times$32-bit instruction control store, and 128 general-purpose and 128 transfer registers. Furthermore, a 4KB scratchpad is utilized to synchronize and to control the microengines. Additionally, two 16slot$\times$64byte addressable register files can be configured as FIFOs in order to send/receive data to/from the IX bus. The common unit of data transferred is the 64-byte MAC-Packet which is obtained by fragmenting the incoming IP packets. Finally, memory and PCI bus controllers are also integrated.

IBM PowerNP NPe405H[7]: It consists of an IBM PowerPC 405 RISC processor core and the following components: an external peripheral bus, memory and PCI bus controllers, up to 4 Ethernet 10/100 Mbps (full-duplex) units, a 32-channel HDLC controller, and a 8-port HDLC controller.

IBM PowerNP NPr2.7[7]: It consists of an integrated PowerPC processor core complemented by: a PHY interface unit, a segmentation and reassembly unit, a memory interface unit, a PCI bus controller, and transmit/receive queuing interfaces. This network processor is an interface and translator between a PCI bus and an ATM UTOPIA or similar interface to an ATM PHY. The segmentation and reassembly functions are performed by the packet/frame memory.

IBM32NPR161EPXCAC133[7]: It consists of an embedded PowerPC processor complemented by: two physical MAC multiplexers and an embedded processor complex. One MAC multiplexer receives frames from the Ethernet or POS physical layer devices while the other transmits them to the same devices. The embedded processor complex consists of up to 16 programmable protocol processors with each having the following characteristics: 3 stage pipeline, general-purpose registers, special purpose registers, eight instruction cache, dedicated ALU, and co-processors. Two of the protocol processors are special as one is for handling Guided Frames and the other is for building lookup data in control memory. Additionally, each protocol processor also contains 7 co-processors and they are named as follows: data store, checksum, enqueue, interface, string copy, counter, and policy. Finally, this network processor can operate as a standalone device or can be placed as a component in a large system and thus allowing scalability.

Chameleon Systems CS2000 Family[8]: It consists of a 32-bit ARC processor core complemented by: a 32-bit reconfigurable processing fabric, integrated PCI and memory controllers, and 160 programmable I/O pins. The processing fabric consists of slices (up to four in the CS2112) with each slice containing 3 tiles. Each tile consists of: 4 local memory store units (32-bit$\times$128 each), 7 32-bit datapath units, 2 16$\times$24 multipliers, and a control logic unit.

Agere "Payload Plus"[9]: The functional processor is segmented into two components, FPP and RSP, and complemented by a third called ASI. Data from the PHY chip enters the Fast Patter Recognition Processor (FPP) which performs protocol recognition and classification as well as reassembly. The FPP can classify traffic based on information contained at Layer 2 through 7. The Routing Switch Processor (RSP) takes data from the FPP and handles queuing, packet modification, traffic shaping, application quality of service tagging, and segmentation. The Agere System Interface (ASI) is the management component of the processor and provides support for RMON and tracks state information. The ASI also controls the data movement between the FPP and RSP to ensure that it moves at wire-speeds.

EZChip[10]: It performs basically four basic tasks that are related to packet processing, namely parse, search, resolve, and modify. The TOPcode (Task Optimized Processing core) technology entails the usage of task-optimized processors (TOPs) which use a customized instruction set and data path for each processing task. This resulted in four TOPs: TOPparse, TOPsearch, TOPresolve, TOPmodify. The processing of packets is actually pipelined since the basic four task are performed after each other and thus in a pipelined fashion. Furthermore, the TOPs can also be superscalar-ed, i.e., many of the same TOPs can be placed in parallel.

Vitesse IQ2000[11]: It consists a control plane processor with four integrated 200 MHz fully programmable, multi-context packet processing engines complemented by: a flexible, high bandwidth I/O interface, specialized co-processors, and hardware support for Quality of Service of service building blocks. The co-processors are intended for look-up, order management, multicast support, DMA management, and context management.

5. Future directions and concluding remarks

We conclude this paper by describing some possibilities for the general network processor design framework. We suggest a potential architecture trajectory and describe what we perceive to be important open questions. Clearly, the existing general network processor organization paradigm is based on having a general-purpose processor surrounded by ASICs. It is commonly accepted in the network processing community that the direction to be followed is the programmable processor paradigm that performs ”well” in networks at the exclusion of ASICs[5]. We foresee an in-between hybrid possibly multi-threaded processor organization with the additional use of reconfigurable hardware. The reasons for including reconfiguration are the following:

- There is evidence of substantial improvements in reconfigurable technologies[12].
- They are programmable, flexible, and can improve general purpose performance substantially if reconfiguration times are reduced.
- Regarding this issue, it has been shown for example that a microcoded reconfigurable engine can provide substantial reduction of reconfiguration times[13].
- Non-existing new network related developments can be easily incorporated using reconfiguration with advantages over general purpose processors and with faster development times than ASICs.

It is our contention that in order to achieve reconfigurable programmable processing with high performance, the following has to be achieved:

1. Once the area is defined, we need to identify functions that are implementable in commonly available technologies that are more complex than existing instructions. These functions are candidates for reconfiguration. For an example of such functions, see Sum of Absolute Differences (SAD) expressed by:

$$SAD(x,y,r,s) = \sum_{i=0}^{i=15}\sum_{j=0}^{j=15} |A_{(x+i,y+j)} - B_{((x+r)+i,(y+s)+j)}|$$

which has been shown to be executed by an augmented multiplier without cycle time extensions[14]. This augments the general purpose processing to close the gap between programmability and ASICs.

2. After functions have been selected, such functions should be ”reconfigured” in firmware. The intend here is to shorten the reconfiguration time and to allow reduced hardware and dynamic reconfiguration.

3. Evaluate the gains to be expected by the incorporation of such units in reconfigurable hardware.

[5]Reason for such direction include among others the impossibility of general purpose computing to follow the demands and the inflexibility of ASICs.

Thus, a possible overall “fixed” hardware organization incorporates:

- A programmable core(s) (think of general purpose processor core(s)) that can be used for non time critical functions;
- Specialized hardware for the most common well established network functions (preferably programmable ones (e.g., utilizing programmable microcode) as protocols/functions keep changing);
- Buffers that store incoming and outgoing packets or frames. These buffers might or might not already perform a limited array of functions;
- Memory in which tables are stored, preferably organized in an optimized way for accessibility and speed.

In addition, the possibility must be presented to replicate the (firmwared) specialized hardware units several times if it is found to be needed for parallel executions.

While the potential general characteristics of a NP can be extrapolated, there are several still open questions that need be addressed including the following:

- How can the characteristics (described previously) actually be translated into precise requirements for network processor architectures with “manageable” organizations?
- Are requirements found in several layers that are apparently doing similar functions also similar in terms of architectural constraints? If so, how can we exploit such similarities in a transparent way?
- What are the architectural implications of implementing QoS in a network processor? How can an NP deal with future services and/or not standardized features?
- Given that standards are evolving and thus not stable, what are the performance metrics for such processors?
- What are the consequences to NP of including integrated services (e.g., voice, video, data) over a “single” connection?
- How can we support the wide range of protocol stacks and their variations using one NP architecture or without major architectural changes for each variation?
- How do we provide scalability as wire-speeds continue to outpace “computer speeds”?
- Is it possible to apply this scalability (if possible) of NPs in systems that are used in different environments, e.g., a router located inside a homogeneous low-speed network has different requirements than a router located in a heterogeneous high-speed network?
- What is the optimal or a good division in terms of software and hardware for the requirements?
- Is it possible to derive a generic overall processor organization for a network processor?

It is evident that many open question needs to be resolved before one can arrive at a network processor solution that is able to address all of current and future (still unknown) functional requirements. We believe that the approach based on reconfiguration is capable of addressing all these requirements at reasonable fast enough speeds assuming that the reconfigurable technologies continue to improve at the current rate.

References

[1] T. Cwynar and S. Lima. http://www.ini.cmu.edu/~servio/telecom/project/, August 2000.

[2] J. P. Slone, ed., *Local Area Network.* Auerback Publications, 1999.

[3] M. A. Gallo and W. M. Hancock, *Networking Explained.* Digital Press, 1999.

[4] J. Y. Hsu, *Computer Networks: Architecture, Protocols, and Software.* Artech House, 1996.

[5] S. Saunders, *Gigabit Ethernet Handbook.* McGraw-Hill, 1998.

[6] “Intel IXP1200 Network Processor.” http://www.intel.com/design/network/products/npfamily/ixp1200.htm.

[7] “IBM PowerNP Network Processors.” http://www.chips.ibm.com/products/wired/communications/network_processors.html.

[8] “Chameleon Systems CS2000 Family.” http://www.chameleonsystems.com/what/whatWeDo.html.

[9] “Network and Communication ICs.” http://www.lucent.com/micro/netcom/platform/npu.html.

[10] “EZChip Network Processors.” http://www.ezchip.com/.

[11] “Vitesse IQ2000.” http://www.vitesse.com/products/categories.cfm?family_id=5&category_id=16.

[12] “Virtex-II 1.5V FPGA Family: Detailed Functional Description .” http://www.xilinx.com/partinfo/databook.htm.

[13] S. Vassiliadis, S. Wong, and S. Cotofana, “The MOLEN rm-coded Processor,” Tech. Rep. 1-68340-44(2001)-01, Computer Engineering Laboratory, TU Delft, the Netherlands, 2001.

[14] S. Vassiliadis, E. Hakkennes, S. Wong, and G. Pechanek, “The Sum-Absolute-Difference Motion Estimation Accelerator,” in *Proceedings of the 24th Euromicro Conference*, 2000.