

A Benchmarking Methodology for Network Processors

Mel Tsai¹, Chidamber Kulkarni¹, Christian Sauer², Niraj Shah¹, Kurt Keutzer¹

¹University of California, Berkeley

²Infineon Technologies, CPR ST, Munich

{mtsai, kulkarni, sauer, niraj, keutzer}@eecs.berkeley.edu

Abstract

Due to the heterogeneity of network processor architectures and constantly evolving network applications, it is currently a challenge to compare or evaluate network processors. In this paper, we present principles of a benchmarking methodology that aim to facilitate the realistic evaluation and comparison of network processors. A key aspect of our approach is the separation of function, the environment, and the means of measurement within the benchmark specification. We pay particular attention to system-level interfaces and define a normalizing environment to quantitatively compare network processors. Also, we present a set of representative benchmarks and motivate their selection based on a taxonomy of network equipment. Finally, we provide a proof-of-concept of our methodology by specifying and implementing two benchmarks on a network processor.

1. Introduction

The number of network processors has grown at an astonishing rate; there are currently more than 30 network processor unit (NPU) vendors that have achieved over 500 design wins [1]. Although this rapid growth has spawned many new network processor alternatives, it also presents a host of new problems. Indeed, network processors have widely disparate micro-architectures, memory architectures, and system interfaces that make evaluation and comparison very difficult [2]. Furthermore, NPU vendors target applications with widely different requirements, from low-speed switches to high-end Internet core routers. Because one of the primary tasks of a system architect is to choose the best network processor for a particular application, a fair comparison and evaluation of network processors is needed.

Recent work in network processing benchmarks [3] [4] [5] [6] has mostly focused on functionality, largely neglecting architectural concerns. As a result, these approaches are unable to effectively adapt to rapid changes in network processor architectures. Hence, a methodologi-

cal approach to network processor *benchmarking* is required.

Two important goals of NPU benchmarks are:

- They must provide results that are *comparable* across network processors.
- They must provide results that are *indicative* of real-world application performance.

Though not the focus of this paper, benchmarking also plays a critical role in the architectural development process of application-specific instruction processors (ASIPs). In the development of a network processor, architects can apply a benchmarking methodology that facilitates the evaluation of a prospective architecture and thus enables efficient exploration of the space of potential architectures.

To meet the goals of NPU benchmarking, we present a methodology that allows benchmark results to be comparable and indicative of real-world application performance. Comparability of results is achieved by precise specification and separation of benchmark *functionality*, *environment*, and means of *measurement*. Functionality captures the important aspects of the benchmark's algorithmic core. In contrast, the environment supplies the normalizing test-bench in which the functionality resides and allows results to be compared across NPUs. Finally, guidelines for the measurement of performance ensure that quantitative results are consistently measured across implementations.

To specify our benchmark functionality and environment we augment an English language description with an executable specification in the Click Modular Router framework [7]. Click is a parallel language that naturally communicates the benchmark specification and is superior to approaches that merely include a reference implementation (with little notion of system-level aspects) in a serial programming language.

The specification of benchmark functionality, environment, and means of measurement is only one part of the methodology. For benchmarks to be indicative of real-world applications, the *selection* of benchmarks that will comprise a suite of network processor benchmarks is important. Not only does our methodology describe how to select a realistic suite of benchmarks based on an applica-

tion-domain analysis of network processors, but also how to choose the appropriate granularity of a benchmark.

Network processing is only one point in the space of embedded applications; other application domains (e.g. telecommunications, multimedia, automotive) may also benefit from this methodology.

The rest of this paper is organized as follows: First, we summarize previous and ongoing work relating to NPU benchmarks. Second, we describe our generalized benchmarking methodology and apply it to the network processor domain. We then present our proposed NPU benchmarking suite and include an example benchmark specification for IPv4 packet forwarding. We demonstrate results for the Intel IXP1200 network processor. Next, we summarize and identify some limitations of our benchmarking methodology and conclude with suggested future work.

2. Related Work

Popular benchmarking approaches such as SPEC [8], Dhrystone [9], and MediaBench [10] work well because their intended architectural platform is homogeneous. These and other related approaches focus solely on the functional characteristics of the application and do not emphasize the system-level aspects of the architectural platform. As a result, these approaches do not work well for application domains with widely heterogeneous architectures.

We are aware of four benchmarking efforts related to NPUs: CommBench, EEMBC, NetBench, and the NPF Benchmarking Working Group. Even these approaches do not emphasize the system-level interfaces of a network processor. Because the performance of a network processor is heavily dependent on system-level interfaces such as the network and control interfaces, it is crucial to account for such differences in a network processor benchmarking methodology. For example, how does one compare the Motorola C-Port C-5 (which contains a programmable and flexible on-chip MAC unit) to the Intel IXP1200 (which does not contain an on-chip MAC unit)? Without special consideration of these environmental issues, fair comparisons between the C-Port C-5 and the IXP1200 cannot be drawn.

As shown later, benchmarking also requires a specification methodology, an executable description, a method for performance measurement, and a motivation for selection of benchmarks. In addition, they should also consider design-space exploration.

2.1. CommBench

CommBench [5] specifies eight network processor benchmarks, four of which are classified as “header-processing applications” and the other four as “payload-processing applications”. CommBench motivates their choice of benchmarks, but provides no methodology of specification or consideration of system-level interfaces. In addition, the implementation and analysis of CommBench currently targets general-purpose uniprocessors [5] and it is unclear how their benchmark suite can be applied in the context of network processors.

2.2. EEMBC

The Embedded Microprocessor Benchmark Consortium [4] (EEMBC) defines a set of 34 application benchmarks in the areas of automotive/industrial, consumer, networking, office automation, and telecommunication domains. In the networking domain, EEMBC defines only three simple benchmarks (Patricia route lookup, Dijkstra’s OSPF algorithm, and packet flow between queues) with little notion of a methodology that can be applied to network processors.

2.3. NetBench

NetBench [6] defines and classifies a set of nine network processor benchmarks according to “micro-level,” “IP-level,” and “application-level” granularities. By using this classification, NetBench acknowledges the heterogeneous aspects of network processor micro-architectures and has initial results for the SimpleScalar [11] simulator and three results for the Intel IXP1200. However, as with [4] and [5], NetBench does not provide a methodology that considers the heterogeneity of network processor interfaces.

2.4. NPF Benchmarking Working Group

The NPF Benchmarking Working Group [3] (“NPF-BWG”) defines benchmarks based on “micro-level,” “function-level,” and “system-level” applications. Although NPF-BWG has previewed examples of each type of the above classifications, as of this writing they have produced only an IPv4 routing benchmark. They are currently working on a draft specification for an MPLS benchmark.

To our knowledge, NPF-BWG is the only work thus far that attempts to address environment- and system-related NPU benchmarking issues (through their notion of a “system-test configuration”). However, we have seen little discussion or published work on what defines a system-

test configuration and it is unclear whether network interface issues have been fully addressed.

2.5. Other Work

Nemirovsky [12] recognizes the needs and requirements of network processor benchmarking. Although Nemirovsky provides insight into the methods for quantifying network processor performance, he does not provide precise information about a viable benchmarking approach.

Crowley et. al. [13] present an evaluation of theoretical network processor architectures based on a programmable network interface model and detailed workload analysis. However, their model does not include many aspects of current target systems for network processors. Also, their model does not fully separate functional and environment concerns.

2.6. Summary of Existing Approaches

In Table 1 we compare existing approaches to network processor benchmarking. As this table shows, none of the existing approaches meets all the requirements of a network processor benchmarking methodology. In particular, there are a number of deficiencies including: lack of specification methodology, lack of consideration of interfaces, and little support for design space exploration.

3. A Benchmarking Methodology

In this section, we motivate and present principles of a generalized benchmarking methodology that apply not only to network processors, but to other application domains as well. Next, we demonstrate these principles for network processors. Finally, we illustrate our methodology on an IPv4 packet forwarding benchmark.

Before introducing our methodology, we first define a

benchmark:

A benchmark is a mechanism that allows quantitative performance conclusions regarding a computing system to be drawn.

3.1. Principles of a Generalized Benchmarking Methodology

Complete benchmarking methodologies employ the following four principles: comparability, representative and indicative, granularity, and precise specification method.

3.1.1. Comparability. A quantitative performance claim that a benchmark provides is not enough. Benchmarks also need to be comparable across system implementations with heterogeneous interfaces. Thus, the benchmark specification cannot simply be a functional representation of the application; it must also model the system environment.

- (1) Quantitative benchmark results must be comparable across a range of system implementations.

3.1.2. Representative and Indicative. To properly characterize a particular application domain, a benchmark suite must cover most of the domain and provide results that correlate to real-world performance. While the number of benchmarks chosen should adequately represent the application domain, the usefulness and feasibility of implementation is questionable for large benchmark suites. Thus, a suite of benchmarks must be chosen based on careful *application-domain analysis*.

- (2) A set of benchmarks must be representative and indicative of the application domain.

3.1.3. Granularity. The granularity of a benchmark is its size compared to the largest implementable application of a

Table 1. Characteristics of existing approaches.

	CommBench	NetBench	EEMBC	NPF-BWG
Specification Methodology	No	No	No	Unknown
Executable Description	Yes	Yes	Yes	Unknown
Method for Performance Measurement	No	No	Yes	Yes
Consideration of Interfaces	No	No	No	Unknown
Methodology of Benchmark Choice	Yes	Yes	No	Yes
Facilitates Design Space Exploration	No	No	No	No

particular domain. NetBench and NPF-BWG, for example, implement benchmarks according to three different granularities: small (micro-level), medium (IP- or function-level), and large (application- or system-level).

The right benchmark granularity is determined by the analysis of *performance bottlenecks* of the application. In some cases, a particular subset of the application (i.e. an application kernel) may dominate the performance of the larger application. However, in many cases bottlenecks are not easily identified or are heavily influenced by the architecture on which the application is implemented. In such cases, choosing a benchmark granularity that is too small may lead to performance results that are not indicative of real-world application performance.

- (3) The granularity of a benchmark must properly expose application and architectural bottlenecks of the domain.

3.1.4. Precise Specification Method. The most important tenet of our methodology is the precise specification of a benchmark. A benchmark specification requires functional, environment, and measurement specifications communicated in both an English and an executable description.

To separate concerns of benchmarking, we distinguish between functional, environment, and measurement specifications. The *functional specification* describes the algorithmic details and functional parameters required for the benchmark. This specification is architecture-independent. In contrast, the *environment specification* describes the system-level interface for a specific architectural platform. This ensures comparability of results across multiple architectural platforms. The environment specification also defines the test-bench functionality, which includes traffic generation. The *measurement specification* defines applicable performance metrics and the means to measure them. At a minimum, a method to determine functional correctness should be provided.

These specifications should be communicated in two forms: an English description and an executable specification. The *English description* should provide all of the necessary information to implement a benchmark on a particular system. This description also provides implementation guidelines such as acceptable memory/speed trade-offs (e.g. it may be unreasonable for a benchmark to consume all on-chip memory), required precision of results, and acceptable use of special hardware acceleration units. Besides the English description, the specification includes an *executable description*. This allows rapid initial evaluation, precise functional validation, and facilitates unambiguous communication of the requirements. Note that neither description eliminates the need for the other.

Thus we summarize the final tenet of our benchmarking methodology:

- (4) Each benchmark should include functional, environment, and measurement specifications in an English and executable description.

3.2. A Benchmarking Methodology for NPUs

Because of the heterogeneity of NPU architectures, it is necessary to separate benchmarking concerns to ensure comparability of results. In this section, we present our benchmarking methodology for network processors that separates the functional, environment, and measurement specifications.

3.2.1. Functional Specification. The role of the functional specification for our NPU benchmarking methodology is three-fold.

First, it must specify the benchmark requirements and constraints, such as the minimum allowable routing table size supported by the implementation, the minimum quality of service (QoS) that must be maintained, or restrictions on where tables can be stored. Requirements and constraints may vary significantly across the suite of benchmarks.

Second, the functional specification must describe the core algorithmic behavior of the benchmark along with any simplifying assumptions in an English description.

Finally, the functional specification should include an executable description written in the Click Modular Router Language [7]. While the algorithmic and functional behavior of a benchmark can be unambiguously communicated using standard languages such as C or C++, Click is a parallel, modular, and natural language for the rapid prototyping of network applications. Click allows the programmer to connect network router elements (application building blocks such as IP routing table lookups, queues, and packet sources/sinks) and run simulations on these element collections to design and verify network applications. Click is also an extensible language; new elements can be added into the element database to serve the needs of different applications.

3.2.2. Environment Specification. Previous approaches to NPU benchmarking lack environment specifications that are critical for comparability. We present the NPU line card model to specify the environment. This model overcomes many of the difficulties associated with comparing benchmark results from different network processors.

Currently, a major target of network processors is core and edge network equipment. In these segments, routing and switching are the most common tasks and these tasks are performed by systems based on line cards with

switched back-plane architectures. Thus, a close examination of how a network processor sits in a line card will help us define the boundary between functionality and environment.

Line cards manage the physical network interfaces of a router. Typical router architectures such as the one presented in Figure 1 contain a variable number of router line cards [14] [15]; up to 16 or more line cards are included in larger routers. Each line card is also connected to a switch fabric that allows packets to pass to other line cards and control processors.

Figure 1 demonstrates two different deployment scenarios for line cards. The network processor in line card A is configured in a serial fashion to its surrounding components, while the network processor in line card B is connected to multiple Gigabit Ethernet MAC units in an parallel fashion. Network processors within line cards may have different interface configurations. In our example, line card A supports an OC-192 packet-over-SONET interface, while line card B supports 10x1 Gigabit Ethernet interfaces.

Figure 2 shows the system-level and architectural interfaces of an NPU. The system-level interfaces consist of the network (physical and fabric) and control interfaces; these interfaces directly influence the benchmark specification. The memory and coprocessor interfaces, however, are architectural interfaces that are not visible at the system-level. Nonetheless, they are important for design-space exploration.

For NPUs in line cards, the network interface is the most important consideration due to its tremendous impact on benchmark functionality and performance. Accordingly, this paper focuses on system-level network interfaces of an NPU.

Based on an examination of over twenty NPU interfaces [2], we found two major characteristics: 1) some NPUs integrate network interfaces, while others do not; and 2) of those that do, the supported interfaces vary greatly. For instance, some network processors contain integrated

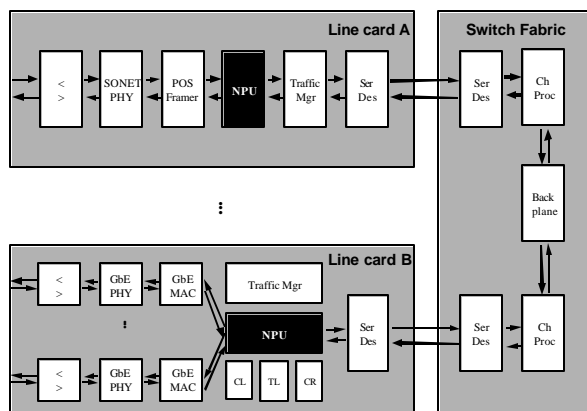


Figure 1. An example router architecture.

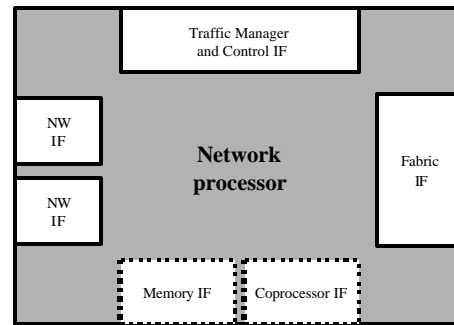


Figure 2. Network processor interfaces.

MAC units for Ethernet interfaces and some do not. If the network processor does not include an integrated MAC unit, one must be added to the hardware or software simulator. If the network processor includes an integrated MAC, it must be configured to conform to the specification. The MAC unit(s) must support the required per-port data rate and number of ports. Further network interface restrictions may be specified by individual benchmarks (e.g. buffer size limits).

The network interfaces in Figure 2 have three important parameters: the number of ports, the type of ports, and the speed of each port. Though defined in the environment specification, the functional specification must also be made aware of these parameters. The number and type of active ports in the system must be known in order to program where, when, and how packets flow within the network processor. In most cases, optimal performance cannot be achieved without customization of the benchmark software to suit a particular port configuration. For example, a programmer of an Intel IXP1200 must treat “slow” ports (i.e. 10/100 Fast Ethernet) significantly differently than “fast” ports (i.e. Gigabit Ethernet) [16].

Line card configurations from router vendors such as Cisco [14] and Juniper Networks [15] have a wide range of performance requirements, as shown in Table 2. Hence, we propose two classes of line card port configurations — one that targets lower performance deployment scenarios and one that targets higher performance scenarios. For this reason, each benchmark should specify two valid port configurations: one that models a “low-end” line card configuration and one that models a “high-end” configuration.

While each benchmark may specify its own port configuration, a uniform configuration should ease implementation complexity. A standardized port configuration also facilitates the understanding of the overall performance of a network processor.

For the low-end configuration, we recommend a standard configuration of 16 Fast Ethernet ports, a relatively low-bandwidth configuration supported by many router

vendors. For the high-end configuration, we recommend a standard configuration of four OC-48 packet-over-SONET ports (10 Gb/s of aggregate bandwidth). OC-48 interfaces are common in high-end line card configurations [17].

Additional considerations of the environment specification are the packet *sources* and *sinks*. The IETF Benchmarking Methodology Workgroup recommends [18] exercising network equipment using the following test setup: a “tester” supplies the device under test (DUT) with packets, which in-turn sends its output back to the tester. In this case, the tester serves as both a packet source and sink. Alternately, a packet “sender” (source) supplies packets to the DUT, whose output is fed into a packet “receiver” (sink) for analysis. Either approach will serve the needs of our benchmarking methodology, as long as the packet traces meet the requirements of the benchmark environment specification.

According to the IETF, the data set for network interconnected devices such as routers and switches must include a range of packet sizes, from the minimum to maximum allowable packet size according to the particular application or network medium (i.e. Ethernet). Specifically, they recommend using evenly distributed Ethernet packet sizes of 64, 128, 256, 512, 1024, 1280, and 1518 bytes [18].

For protocols that use variable size packets, network processor vendors often report packet throughput using minimum-size packets. Because many network applications operate primarily on fields in the packet header, more processing must be performed on a large group of minimum-size packets than on a small group of large packets. Using only minimum-size or maximum-size packets can test the corner cases of benchmark performance, but these are not realistic indicators of overall performance and we defer to the IETF’s recommendations and Newman’s work for packet sizes.

Newman [17] observed packet sizes based on live Internet samples from the Merit network over a two-week period. The top four IP packet sizes, according to the author, were 40, 1500, 576, and 52 bytes (56%, 23%, 17%, and 5% of the total, respectively). Newman uses this proportion of packet sizes to develop a traffic pattern called the “Internet mix” (Imix).

3.2.3. Measurement Specification. A benchmark implementation must be functionally correct before performance can be measured. In many cases, functional correctness can be observed by comparing the trace output of the network processor to the output of a reference implementation (i.e. the Click executable description). However, this may not be sufficient for functional correctness; other measures of functional correctness (such as real-time constraints or allowable packet drop rate) may also be specified by the benchmark.

Table 2. Common router line card configurations.

Number of Ports	Type of Ports	Max Bandwidth (Aggregate), in Mb/s
16	Fast Ethernet	100 (1,600)
24	Fast Ethernet	100 (2,400)
48	Fast Ethernet	100 (4,800)
8	Gigabit Ethernet	1,000 (8,000)
16	Gigabit Ethernet	1,000 (16,000)
8	OC-3 POS	155 (1,242)
4	OC-12 POS	622 (2,488)
4	OC-12 ATM	622 (2,488)
1	OC-48 POS	2,488 (2,488)
4	OC-48 POS	2,488 (9,953)
1	OC-192	9,953 (9,953)

Once a benchmark implementation is shown to be functionally correct, its quality of results can be measured. For many network processor benchmarks, line speed (throughput) is the most important measurement of performance.

There are two different units used to measure line speed: packets per second and bits per second. While the former provides insight into computational performance, the latter provides a clearer understanding of throughput. However, with knowledge of the packet size distribution, one can be converted to the other.

According to RFC 1242 [19], throughput is defined as “the maximum rate at which none of the offered [packets] are dropped by the [device].” However, for our methodology, we extend this definition to include the possibility that some packets *should* be dropped in accordance with the benchmark specification.

The measurement of throughput is challenging. According to [18], throughput is measured by sending packets to the DUT at a specific rate. If packets are incorrectly dropped by the DUT, the rate is successively throttled back until packets are no longer dropped. Using this procedure in our methodology ensures that the measurement of throughput is consistent across implementations.

Besides line speed, there are other useful performance metrics. For example, in a benchmark with real-time constraints, packet latency may be an important performance metric. Derived metrics such as cost effectiveness (i.e. performance/cost) may also be important. The notion of time is central to the measurement of many of these metrics. At the very least, a cycle-accurate software simulator of the network processor architecture is required.

<i>Functional Specification</i>	
Requirements and Constraints	
Behavior	
Click Implementation	
<i>Environment Specification</i>	
Network Interface	
Control Interface	
Traffic Mix and Load Distribution	
Click Implementation	
<i>Measurement Specification</i>	
Functional Correctness	
Quality of Results	

Figure 3. Template for NPU Benchmarking.

First introduced by NPF-BWG [3] and discussed by Nemirovsky [12], the concept of *headroom* is loosely defined as the amount of available processing power that could be used by other tasks in addition to the core application. In theory, headroom is useful to evaluate support for additional functionality on top of the core application. Unfortunately, as recognized by Nemirovsky, headroom is difficult to define and measure and we agree.

3.3. An Example Benchmark Specification

To illustrate our methodology we present a benchmark specification for IPv4 packet forwarding. Using our template for network processor benchmarks shown in Figure 3, we motivate and summarize the specification for this benchmark.

3.3.1. Functional Specification. Our functional specification of this benchmark is based on RFC 1812, Requirements for IP Version 4 Routers [20]. Because the entire English description is too long to fit in this paper, we list the main points:

- A packet arriving on port P is to be examined and forwarded on a different port P' . The next-hop location that implies P' is determined through a longest prefix match (LPM) on the IPv4 destination address field. If $P = P'$, the packet is flagged and forwarded to the control plane.
- Broadcast packets, packets with IP options, and packets with special IP sources & destinations are forwarded to the control plane.
- The packet header and payload are checked for validity and packet header fields checksum and TTL are updated.
- Packet queue sizes and buffers can be optimally configured for the network processor architecture unless large buffer sizes interfere with the ability to measure sustained performance.

- The network processor must maintain all non-fixed tables (i.e. tables for the LPM) in memory that can be updated with minimal intrusion to the application.
- Routing tables for the LPM should be able to address any valid IPv4 destination address and should support up next-hop information for up to 64,000 destinations simultaneously.

In addition to the written description, Figure 4 shows the graphical representation of the accompanying Click description. This Click description also includes components of the environment specification (see below).

3.3.2. Environment Specification. For the network interfaces, we require our standard 16 Fast Ethernet ports for the low-end configuration and 4 OC-48 POS ports for the high-end configuration. A simplifying assumption of our benchmark is that it is a stand-alone configuration. As a result, we do not require a fabric interface.

We define a simple control interface that drops all incoming control packets and does not generate any control packets.

The traffic mix for our benchmark contains destination addresses evenly distributed across the IPv4 32-bit address space. Packet sizes are evenly distributed across 64, 128, 256, 512, 1024, 1280, and 1518 bytes and 1% of the packets are hardware broadcast packets [18]. In addition, 1% of the packets generate IP errors.

There is a single packet source for each input port that generates an evenly distributed load. Also, the range of destination addresses and associated next-hop destinations provide an evenly distributed load on every output port.

Figure 4 contains packet sources and sinks that model network interfaces of the low-end configuration.

3.3.3. Performance Measurement. To prove the functional correctness of a benchmark implementation, one must compare all output packets (including error packets) from the Click description to that of the DUT.

Overall performance should be measured as the aggregate packet-forwarding throughput of the network processor such that no valid packets are dropped. This is measured by successively lowering the bandwidth at which packets are sent to each MAC port simultaneously until the network processor does not drop valid packets. This measurement is obtained by using packet traces and port loads specified by the environment.

4. The Benchmark Suite

Before we present our suite of benchmarks for network processors, we first discuss benchmark granularity for our methodology.

4.1. Granularity

The appropriate benchmark granularity must be chosen after careful application-domain analysis and the identification of performance bottlenecks. Previous benchmarking approaches often choose benchmarks with relatively small granularities, e.g. “micro-level.” In our experience, typical applications running on a network processor must perform a large number of diverse tasks on packets. Nearly all such applications must perform micro-level operations such as input queuing, table lookups, bit-field replacements, filtering rule application, appending bytes to packets (resizing), and checksum calculations. While it is possible that any one of these functions may become a performance bottleneck, it is virtually impossible to determine *a priori* which “micro-level” tasks will dominate a particular application on a particular network processor. As an example, Figure 5 shows a micro-level breakdown of IPv4 packet forwarding benchmark on the XPI200 for packet traces consisting only of 64-byte packets. Even when processing small packets (which heavily stresses the header processing ability of a network processor), LPM routing lookup consumes only 18% of the processing time on an IXP1200 microengine. As Figure 5 shows, no single micro-level task dominates execution time, thereby supporting the chosen granularity of this benchmark.

We believe the performance bottlenecks of network

applications are not appropriately represented by micro-level or function-level benchmarks. Hence network processor benchmarks should be specified at the application-level.

4.2. Choosing the Benchmarks

Existing approaches to NPU benchmarking define criteria for differentiating the characteristics of benchmarks (i.e. header vs. payload) and choose benchmarks based on these criteria. However, we strongly believe that the choice of benchmarks should be driven by an application-domain analysis. For this analysis, we created a classification of applications based on network equipment and chose benchmarks that were characteristic of these classes. This equipment-centric view provides a representative set of applications to evaluate system architectures.

We identify three major segments within the network application domain. First, equipment designed for the Internet *core* segment are generally the highest throughput devices found in networking. Core equipment is usually responsible for routing high-speed WAN/MAN traffic and handling Internet core routing protocols such as BGP.

Second, the network *edge* segment comprises equipment that operates at lower speeds (with shorter links) than equipment in the core. These devices include a variety of mid-range MAN packet processors such as routers,

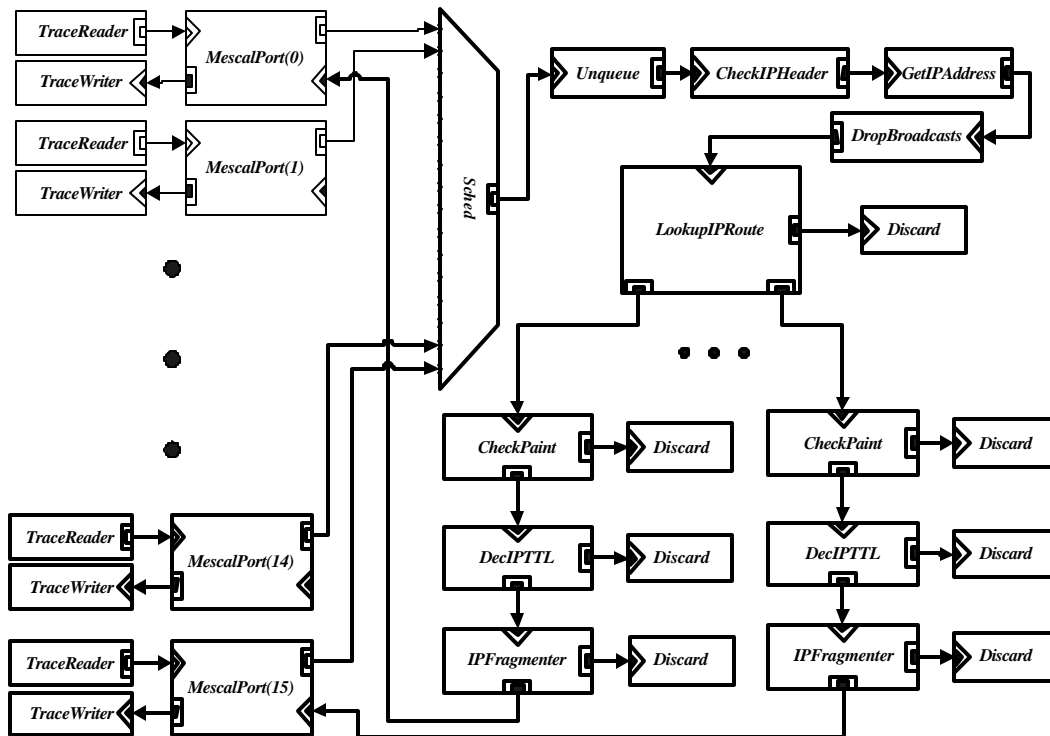


Figure 4. IPv4 packet forwarding Click diagram (low-end configuration).

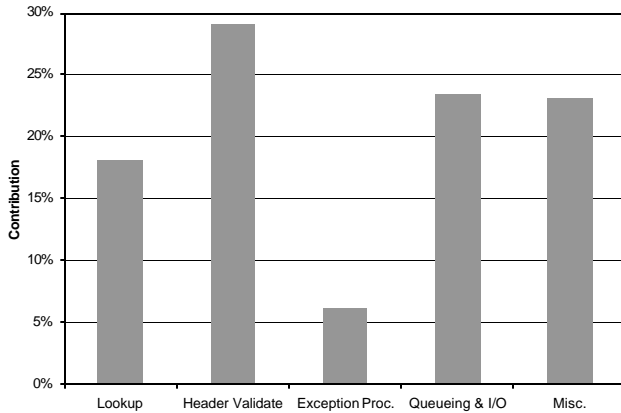


Figure 5. Profile for IPv4 packet forwarding on the IXP1200 (64-byte packets).

switches, bridges, traffic aggregators, layer 3-7 devices (such as web switches and load balancers), and VPNs/firewalls.

Finally, the network *access* segment is usually comprised of low- to high-speed LAN equipment. This segment includes equipment such as LAN switches (Fast Ethernet, Gigabit Ethernet, FDDI, Token Ring, etc.), wireless devices (802.11 a & b), integrated access devices, access concentrators, and cable modems.

Based on this classification, we identify five major application categories in which network processors are likely to play a role. Ten benchmarks were chosen based on their relative significance within the network.

- LAN/WAN Packet Switching
 1. Ethernet Bridge
 2. IPv4 Packet Forwarding
 3. ATM Switch
 4. MPLS Label Edge Router
- Layer 3+ Switching
 5. Network Address Port Translation (NAPT)
 6. HTTP (Layer-7) Switch
- Bridging and Aggregation
 7. IP over ATM
 8. Packet over SONET Framers
- QoS and Traffic Management
 9. IPv4 Packet Forwarding with QoS and Statistics Gathering
- Firewalls and Security
 10. IPv4 Packet Forwarding with Tunneling and Encryption

5. Preliminary Results

To date, we have completed three benchmark specifications: IPv4 packet forwarding, MPLS, and NAPT. We im-

plemented the first two benchmarks in assembly language on the Intel IXP1200 network processor.

In our experience, describing network processor benchmarks using Click is advantageous. First, the intended functionality of the benchmark can be verified using Click’s Linux-based simulation tools. Second, the extensibility of Click allowed us to write a number of new elements and tools. For example, we have written a “port” element that models the ports of a router line card. Port elements are used in conjunction with new elements that read and write packet trace files (in a format similar to TCPDump [21]) and supply the Click simulator with realistic packet data from our packet-generation utility (written in C).

The same packet trace interface used by the Click simulator is used for the network processor simulation environment, which aids verification of benchmark functionality. In our benchmark experiments with the IXP1200, a Windows dynamic-linked library was written to interface with the IXP1200 Developers Workbench tools.

Performance on the IXP1200 was measured using version 2.0 of the Developer Workbench software assuming a microengine clock rate of 200 MHz and an IX Bus clock rate of 83 MHz. Intel’s IXF440 MAC devices are modeled within the Developer Workbench and configured for 16 Fast Ethernet ports.

Figure 6 shows the results for IPv4 packet forwarding. A variety of packet sizes were tested, including the mix of packet sizes recommended by the IETF Benchmarking Methodology Workgroup (BMWG) in [18]. With 16 100 Mb/s ports, the maximum achievable throughput is 1600 Mb/s, yet for the range of packet sizes the IXP1200 cannot sustain this throughput due to the computational complexity of the benchmark specification.

As shown in Figure 7, the IXP1200 achieves better performance on our MPLS benchmark. This is because the computational requirements of our MPLS specification are

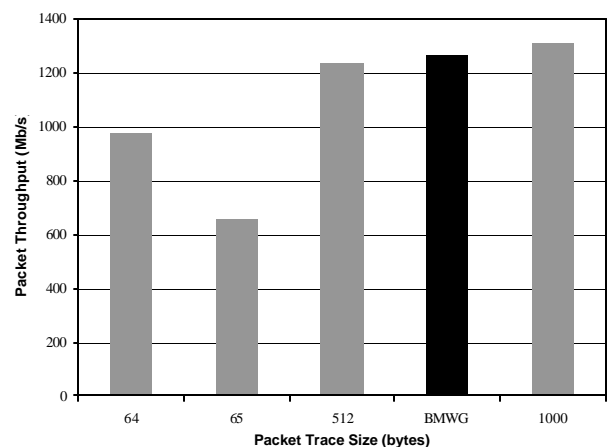


Figure 6. IPv4 packet forwarding on the IXP1200.

lower than with IPv4 packet forwarding. We believe our MPLS benchmark achieves performance that is indicative of a real-world implementation. However, the wide parameter space of MPLS must be closely examined before its performance can be generalized.

In theory, as packet size increases, packet throughput should also increase. In practice, we observe a reduction in throughput between 64- and 65-byte packets, 128- and 129-byte packets, etc. This is due to the 64-byte alignment of the receive and transmit FIFOs on the IXP1200. Extra processing and transmission time are required for non-aligned packet segments. This type of information provides insight into the architectural nuances of network processors.

6. Summary

In this paper we present four principles of a generalized benchmarking methodology. We tailor this methodology to the specific requirements and goals of network processor benchmarking. The methodology defines a template for a benchmark consisting of a functional, environment, and measurement specifications.

Our key contributions are the emphasis of the network-level interfaces for NPUs based on our line card model, the utility of a Click executable description, motivation for application-level benchmarks, and a disciplined approach to identifying a set of benchmarks.

While this approach addresses many of the weaknesses of previous and ongoing projects, it is a work in progress. That our benchmarks are *indicative* and that our approach enables *comparability* still needs to be demonstrated. In addition, we have only specified and implemented a fraction of our benchmark suite. Finally, our method of measuring functional correctness requires further refinement. In summary, we believe we have identified a number of key issues in benchmarking NPUs and have

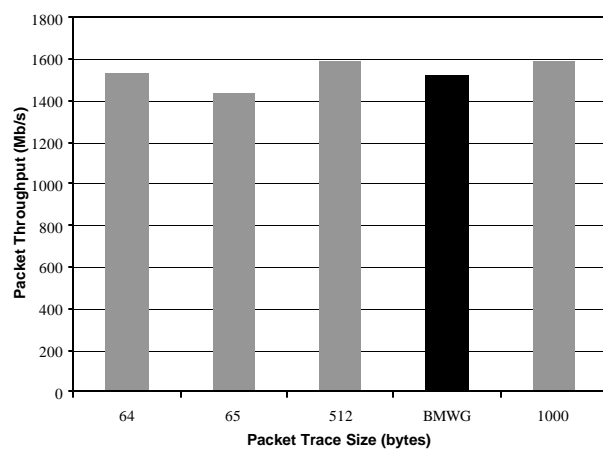


Figure 7. MPLS on the IXP1200.

made steps toward embodying them in a benchmarking methodology.

7. Future Work

For our benchmark methodology to show NPU comparability, we need to develop benchmark implementations for network processors other than Intel's IXP1200. We are working to develop IPv4 packet forwarding and MPLS benchmark implementations on other network processors, including the Motorola C-Port C-5. We also need to correlate our benchmark results with entire application implementations on NPU-based line cards to illustrate the indicativeness of our benchmark suite. Third, we have to develop specifications for the remaining benchmarks in our suite. Based on further implementation insights, additional issues related to control, memory, and co-processor interfaces need to be investigated.

Finally, it would also be interesting to determine other metrics and measurements such as robustness. This issue gains further significance as our methodology is used in the design of network processor architectures.

8. References

- [1] C. Matsumoto, "Technical trial-by-fire awaits NPUs," EE Times, CMP Media LLC, December 27, 2001.
- [2] N. Shah, *Understanding Network Processors*, master's thesis, Dept. of Electrical Engineering & Computer Sciences, Univ. of California, Berkeley, 2001.
- [3] S. Audenaert, P. Chandra (NPF Benchmarking Working Group co-chairs), "Network Processors Benchmark Framework," *NPF Benchmarking Workgroup*, <http://www.npforum.org/>.
- [4] Embedded Microprocessor Benchmark Consortium (EEMBC), <http://www.eembc.org/>.
- [5] T. Wolf, M. Franklin, "CommBench — A Telecommunications Benchmark for Network Processors," *IEEE International Symposium on Performance Analysis of Systems and Software*, Austin, TX, Apr. 2000, pp. 154-162.
- [6] G. Memik, B. Mangione-Smith, W. Hu; "NetBench: A Benchmarking Suite for Network Processors," *International Conference on Computer-Aided Design (ICCAD)*, Nov. 2001.
- [7] E. Kohler, R. Morris, B. Chen, J. Jannotti, M. Kaashoek; "The Click Modular Router," *ACM Transactions on Computer Systems* (18)3, Aug. 2000, pp. 263-297.
- [8] Standard Performance Evaluation Corporation (SPEC), <http://www.spec.org/>.

- [9] R. Weicker, "Dhrystone Benchmark: Rationale for Version 2 and Measurement Rules," *SIGPLAN Notices*, Vol 23, No 8, August 1988.
- [10] C. Lee, M. Potkonjak, W. Mangione-Smith, "MediaBench: a tool for evaluating and synthesizing multimedia and communications systems," *Proceedings of the 30th Annual International Symposium on Microarchitecture*, December 1997, pp. 330-335.
- [11] D. Burger, T. Austin, "The SimpleScalar Tool Set, Version 2.0," *Technical Report CS-TR-97-1342*, University of Wisconsin, June 1997.
- [12] A. Nemirovsky, "Towards Characterizing Network Processors: Needs and Challenges," *XStream Logic* (now *Clearwater Networks*), whitepaper, November 2000.
- [13] P. Crowley, M. Fiuczynski, J. Baer, B. Bershad, "Characterizing Processor Architectures for Programmable Network Interfaces," From the Proceedings of the *2000 International Conference on Supercomputing*, Sante Fe, N.M., May 2000.
- [14] Cisco Systems, "Technology of Edge Aggregation: Cisco 1000 Series Edge Services Router," *Product Datasheet*, March 2001.
- [15] Juniper Networks, "M160 Internet Backbone Router," *Product Datasheet*, December 2001.
- [16] Intel Corporation, "Intel IXP1200 Network Processor Family: Hardware Reference Manual", Revision 8, pp. 225-228, August 2001.
- [17] D. Newman, "Internet Core Router Test," *Light Reading* (<http://www.lightreading.com/>), March 2001.
- [18] S. Bradner, J. McQuaid, "A Benchmarking Methodology for Network Interconnect Devices," *Request for Comments - 2544*, Internet Engineering Task Force (IETF), March 1999.
- [19] S. Bradner, "Benchmarking Terminology for Network Interconnection Devices," *Request for Comments - 1242*, Network Working Group, July 1991.
- [20] F. Baker, "Requirements for IP Version 4 Routers," *Request for Comments - 1812*, Network Working Group, June 1995.
- [21] S. McCanne, C. Leres, and V. Jacobson, *Tcpdump 3.4 Documentation*, 1998.