# Case Study:

# Performance Analysis of a Diversified Router

**Brandon Heller,** brandon.heller@gmail.com

---

**Abstract:**
Routers are responsible for forwarding the packets in today's Internet from source to destination. Unfortunately, the Internet architecture has become resistant to change, in a way that kills new innovation. One potential solution to this problem is the concept of Diversified Networking, which uses virtualization at the network layer to enable multiple concurrent networks with minimal design constraints. Washington University is building a prototype diversified router on network processors, which are highly parallel processors optimized for network tasks. We present a survey of benchmarks for network processors, and describe performance considerations for related systems. We also present an experimental design for evaluating those factors affecting performance, along with preliminary results and analysis, for standard IPv4 packets.

---

**Keywords:** diversified router, diversified networking, network processor, network processor benchmark, router

---

## Table of Contents:

# 1.0 Introduction

When sometime mentions the word "Internet? they're probably referring to today's mix of end systems and routers, all communicating over the IPv4 internetwork protocol. IPv4 is now over thirty years old, and is really starting to show its age. We're running low on unique Internet addresses, have hacked-on mechanisms for ensuring quality of service, and have poor security. The cost to fix these deficiencies is so high that we're forced to live with these issues. Or are we? Network diversification presents one path to change. It enables concurrently running architectures over a fixed substrate, through virtualized resources. Tests for future internet protocols can be done at planet-scale more easily by network researchers, or alternately, a virtualized platform may become the next Internet and enable any number of customized internet protocols.

Section 2 will describe in more detail the concept on Network Diversification. Section 3 will then describe network processing benchmarks, all which could eventually be run in a diversified network. Section 4 will cover related systems, both PC and NP based. Section 5 will describe experiments to analyze the current Diversified Router's performance, and Section 6 will summarize these results.

# 2.0 Network Diversification

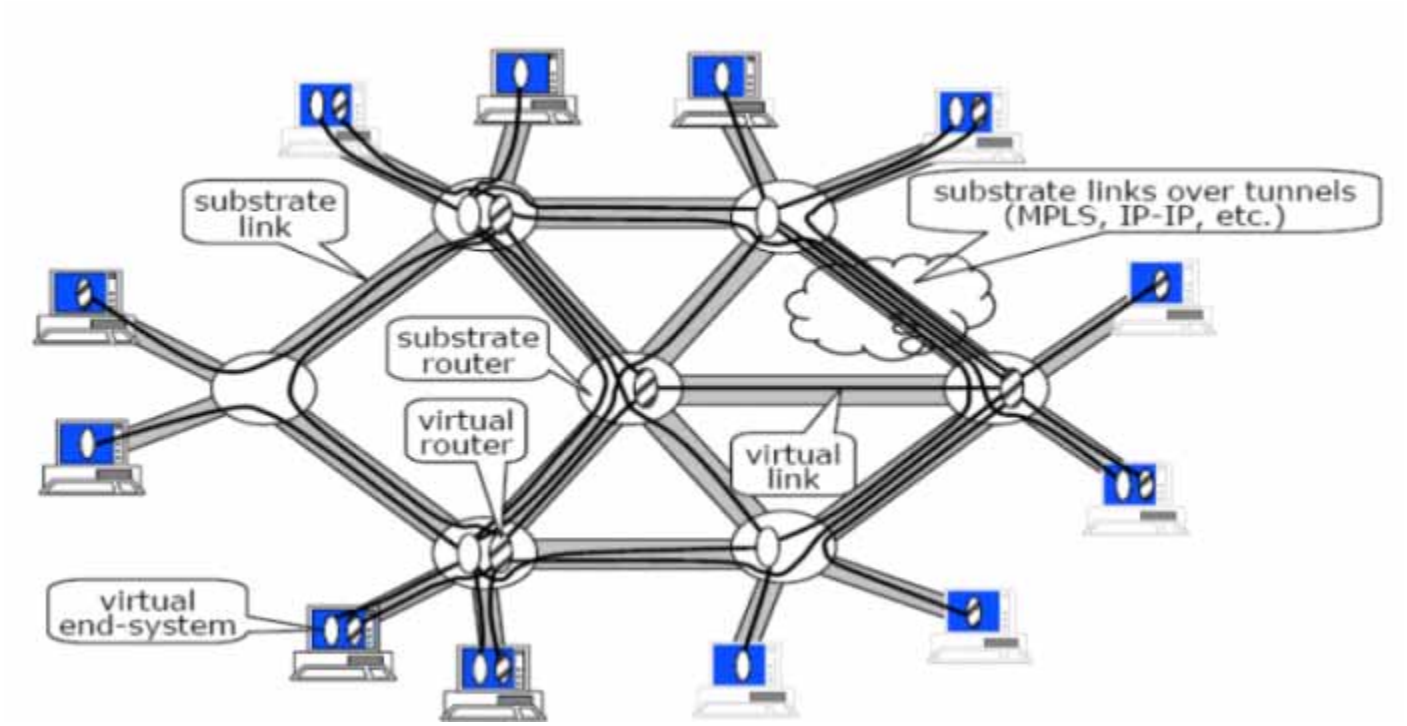


**Figure 1:** Diversified Network

used with permission from Jon Turner

*Diversified networks* consist of a shared physical substrate, virtual routers (metarouters), and virtual links (metalinks), as shown in Figure 1. Virtualizing, or sharing the resources of routers, enables smooth and incremental upgrades to new network services. Although this architecture may sound foreign, the substrate

portion of the network is designed with similar components and techniques to today's networks. Each router in the substrate does the same general operations as a vanilla IPv4 router.

The diversified network idea will be integrated into the upcoming GENI project (Global Environment for Network Innovation), a shared testbed to be available to all network researchers. Before GENI can be built, the substrate components must be demonstrated correct, be of reasonable cost, and perform at high speeds. Washington University is building the Diversified Router to be one such prototype substrate component. By being built on high-speed Network Processors (NPs), the speed should be high, and the cost not significantly higher than a regular multi-core x86 server. Although one of the main selling points of the Diversified Router will be its ability to run multiple "slices" of a network, each with separate bandwidth provisioning and different services, all the tests in this document will focus on a regular IPv4 router. IPv4 is well-known, will remain dominant in the years to come, and provides a nice base of comparison against existing systems.

## 2.1 Typical Router

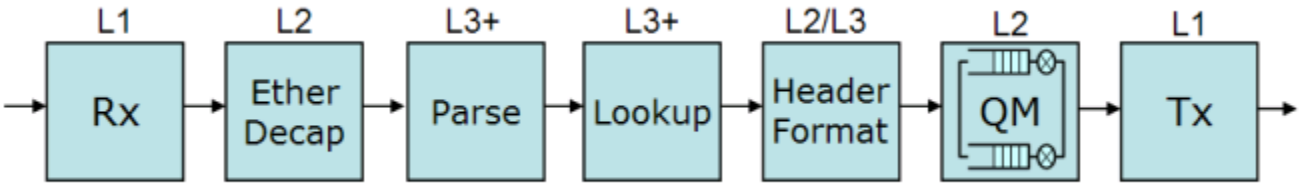


**Figure2:** Typical Router

In general, a router receives a packet, looks up its next hop, and forwards the packet. Figure 2 shows a block diagram of a router's operations. The vast majority of traffic passes through the fast path of a router, also called the data plane. Each packet is received by a mix of hardware and software that converts physical-layer information, such as photons or electrical impulses, to a packet format, typically Ethernet, in the Receive block. The Ethernet header is first validated, then stripped from the packet, in the Ethernet Decapsulate block. Next up is the IP header, which must also be validated, as per RFC1812, Requirements for IPv4 Routers, in the Parse block [RFC1812]. A lookup key is then extracted from the packet, which generally includes the destination IP address and source and destination ports, in the Lookup block. The lookup result includes the port on which the packet may leave, as well as an action to take upon the packet.

On occasion, the lookup will fail, forcing the packet to go through a separate exception path. The exception path leads to a control plane, where higher-level decisions for packet forwarding are made. In the fast-path Header Format block, the IP header is modified with the new destination and a new Ethernet header is constructed. The packet is then sent to the Queue Manager (QM) block, which ensures all output ports are treated fairly, then sent out a physical port by the Transmit Block. The data plane portion of a router must be extremely fast, as routers are performance-specified by the rate at which they forward minimum-sized data packets. In the case of a 10Gb/s router, over 16M minimum-sized Ethernet packets must be forwarded every second! Achieving these speeds requires specialized hardware, which is where network processors come in.

## 2.2 Network Processors

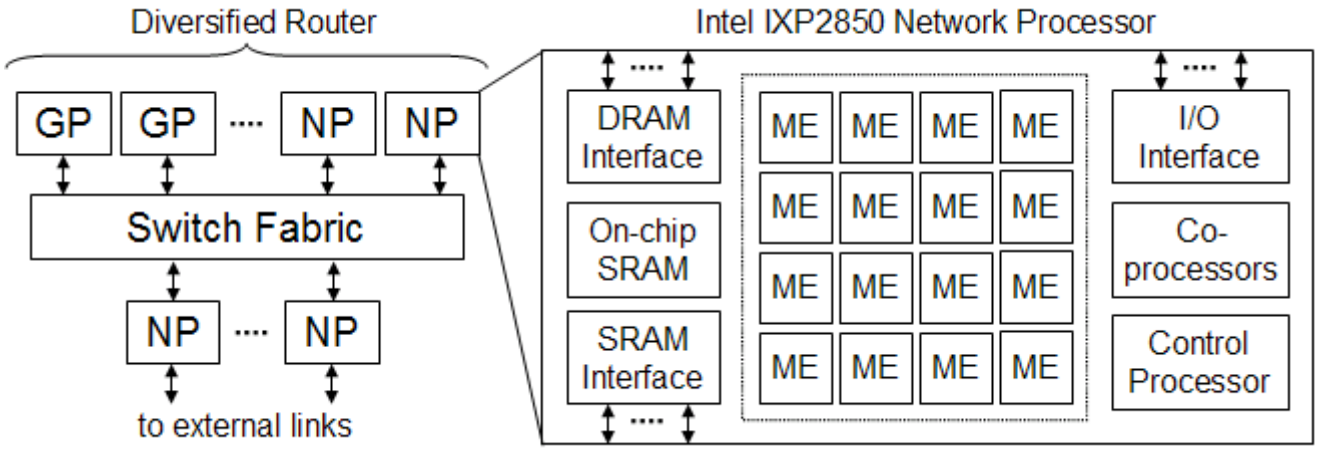


**Figure2:** Diversified Router and Network Processor detail

General-purpose processors (GPPs) like the Intel Pentium enable great flexibility, but are poorly suited for network processing tasks. Five years ago, the common solution to enable high-speed routers was to create an Application-Specific Integrated Circuit (ASIC). ASICs enable line-rate speeds, but are expensive, inflexible, and the years they require to develop places the designer in financial risk. Recently, network processors (NPs), a hybrid approach, have emerged. NPs combine the speed of ASICs with the programmability of GPPs. They have a design optimized for high-throughput data movement and efficiently exploit thread-level parallelism, as opposed to general-purpose processors, which are optimized for instruction-level parallelism. The Intel IXP2850 is one example, as shown in Figure 3. It features multiple high-speed memory interfaces, 16 (!) multithreaded RISC data plane processing engines called MEs (Microengines), and on-die coprocessors. All this power doesn't come cheap; a dual IXP2850 system runs in the neighborhood of $10K - but it is the lowest-cost way to get flexible 10 Gb/s processing power, even for minimum-size packets. The prototype diversified router is built on one of these systems. Proper evaluation of its performance requires knowledge of the structure of NPs, as well as benchmarks of the applications the system will run.

### 2.3 Diversified Router

The Diversified Router can seen in the left side of Figure 3. Linecards connect to the outside world, initially the Internet. Once traffic has been validated by a Linecard NP, it is sent to either a general-purpose processor router or a router NP, by way of a non-blocking crossbar switch. The router NPs are where the action is, and where the performance analysis will focus. Each NP is a blade in shared ATCA (Advanced Telecommunications Architecture) chassis, and connected to an external switch over 1 Gb/s ports.

## 3.0 Network Processing Benchmarks

Since network processing generally happens on network processors, the benchmarks for NPs and NP apps are often one and the same. The relative youth (less than 7years) of NPs has led to multiple benchmarks, none of which is a definitive standard. It is important to understand these, even if they're not used yet on the Diversified Router. They, or other such multi-application benchmarks, will eventually need to be implemented on it.

### 3.1 CommBench

CommBench (CB) was the first NP-specific benchmark, created while the field was in a state of flux [Wolf00]. CB defines two groups of network applications, Header Processing Applications (HPA) and

Payload Processing Applications (PPA), each with four selected workloads. Each workload focuses on a small, computationally intense program kernel used in a network app, typically in the data plane portion of a router. The benchmarks are intended both for traditional routers, where little processing is performed for each packet, as well as active routers, where a program may be executed along each router hop.

expand on programs

CB looks at three metrics:

- instruction mix: static and dynamic code and kernel sizes
- computational complexity: instructions at least once, instructions for 99% coverage, instructions for 90% coverage
- cache performance: hit rate

These metrics enable a comparison to more traditional SPEC embedded benchmarks intended for workstation and personal computers, like those with MIPS, PowerPC, and ARM processors. The SPEC embedded benchmark lacks a focus on I/O, which is extremely important in the network environment, where we must consider a wide range of input sizes. Another issue with SPEC is that performance for its constituent benchmarks is unrelated to real-time constraints. A router that can handle packets at line rate, regardless of packet distribution is a hard real-time system; a new packet must enter and exit every n microseconds, regardless of instruction or data cache state. In a workstation environment, throughput and response time of complex applications is the key, rather than real-time processing of relatively simple kernels. The last issue with SPEC for NPs is that it assumes a static environment. In a complex NP, the application has a dynamic and unpredictable input stream. NPs often used a fixed control store rather than an instruction cache, resulting in less space for data plane programs. As a result, CB programs are an order of magnitude smaller than SPEC programs. CommBench is compared to MediaBench, a more recent benchmark that focuses on media encoding and kernels. They both address input/output concerns, but transcoding is a small part of the set of network apps, and control operations are excluded, which makes MediaBench weakly representative of NPs. The authors note that new requirements push processing down closer to the network layer, and argue that CB is a useful tool for evaluating and designing telecom network processors. They show examples for deriving computational and I/O requirements given a CommBench output of one network computation kernel. The authors also note that benckmarks can be controversial and change rapidly, a claim supported by the fact that another NP benchmark came out the following year.

## 3.2 NetBench

NetBench (NB), released in 2001, covers a set of 9 applications representative of NPs [Memik01]. The applications cover all levels of packet processing, from low-level code fragments to larger application-level programs. NB uses the following metrics:

instructions per cycle: instructions, cycles, IPC
instruction distribution
branch prediction accuracy: address prediction rate, direction prediction rate
cache behavior: data/instruction cache total accesses, data/instruction cache miss ration, L2 miss ratio

Three separate benchmark levels are defined:

- low or micro: operations near the link layer
- routing: operations like IP routing
- application level

A comparison with MediaBench leads the authors to conclude that separate benchmark for NPs are a

necessity. They use a 4-way superscalar processor model, similar to the Alpha, to characterize the workloads in SimpleScalar. The comparison shows that media and communications systems have similar benchmark results in a few metrics, however, there are stastically significant differences. For example, at the 90% confidence interval, NetBench has a higher IPC level. Overall, the differences seems sufficient to warrant a new benchmark. The test applications are implemented on an IXP1200 processor simulator, and show that a 6-core IXP1200 running at 200 MHz is significantly faster than a Pentium 1 GHz in most benchmarks, and in some up to 50% faster.

The results of the study must taken with a grain of salt. In the early 2000 s there were up to 40 new network processors released, some of which has superscalar organization, some which were chip multiprocessors (CMP), some very-long instruction word (VLIW), and so on; any choice would have been reasonable. Since then, the market has shaken out most of the competitors, leaving only CMPs, most with in-order RISC cores. Another benchmark to include both the data and control plane portions of router applications was NpBench. The authors claim that 10Gbps forwarding results in 19.5M packets per second. This is wrong; Ethernet packets are 64B minimum with 12B required inter-frame spacing. But they re close enough, and right that the processing requirements are demanding and must be understood better to successfully engineering router apps.

## 3.3 NpBench

NpBench (NPB) is another NP benchmark, but sliced differently from the others [Lee03]. It splits NP apps into three functional groups: traffic-management and quality of service, security and media processing group, packet processing. NPB is motivated by more complicated protocols and network services that now require additional processing power, to the point that the communication bottleneck may have moved into the network nodes, rather than the old bottleneck of links between them. GPPs are flexible but cannot support full line rates for small packets.

It evaluates the following metrics:

- instruction mix
- parallelism
- cache behavior
- required processing capability per packet

NpBench one is compared to other benchmarks. The main difference between it and CommBench is that like most some NP benchmark suites, it does not focus on control plane tasks, those defined for flow management, signaling, higher-level protocols and other control tasks. NPB looks are both the control and data planes. EEMBC and MiBench have some network apps, but only look at routing and encryption. The benchmark results are all based on C language implementations and yield instructions required per packet at different line rates and processor frequencies.

## 3.4 PacketBench

The bench?in PacketBench refers to workbench, rather than benchmark [Ramaswamy03]. It is a framework for implementing network processing applications and obtaining an extensive set of workload characteristics? In other words, it does not define a specific benchmark or set of benchmarks, but creates a set of tools to more easily create and run higher-level simulations. Those simulations can guide the choice of a processor selection, application selection, or even be used to guide the creation of a new network processor engine. The tools are designed not only for routers, but also for packet classification and payload modifiers. The authors use the now-familiar argument that the workload on network nodes is totally different from traditional server or workstation loads. Those workloads are generally very simple tasks operating on small

chunks of data, and imply that SPEC or TPC are not applicable to the domain. PacketBench creates a programming and simulation environment built on top of SimpleScalar, as well as easily implemented packet processing functions. According to the authors, the result is an easier-to-program system that results in realistic system behavior. One should be wary of this claim. NPs are selected for applications where speed matters. This speed constraint is a strong motivator for assembly programming and highly optimized communication functions. Almost by definition, one cannot recreate realistic system behavior, because the compiler and low-level optimizations have such an influence on the performance. PacketBench outputs enable detailed and processor-specific analysis, and provide input for analytic performance models. Its metrics include:

- instructions vs packet size (in bytes)
- accesses to packet and non-packet memory
- instruction/data memory sizes

# 4.0 Related Router Systems

The diversified router is being created for an upcoming NSF grant to create a national-scale testbed for networking research. Critically evaluating its design requires, at minimum, comparing it to the prototype to existing testbeds. For additional evaluation ideas, we'll also look at other systems running on an IXP.

## 4.1 PC-based

PCs are often used for research routers, due to their flexibility and low cost.

### 4.1.1 PlanetLab

The leading testbed for network research is currently PlanetLab, a series of PC-based routers connected over the Internet [Peterson06]. Its use of general-purpose processors limits network speeds and its ability to provide hard resource guarantees for repeatable experiments, but its distributed-service model is flexible and useful for network research, and many researchers have used it as a base for distributed processing.

### 4.1.2 PL-VINI

The Virtual Internet Infrastructure (VINI) is a project to extend the reach of PlanetLab from an overlay of PCs to a full substrate, i.e. a platform for simulating the entire internet, with all its protocols [Bavier06]. It will let network researchers deploy and evaluate their ideas with real routing software, traffic loads, and network events. PL-VINI is an initial implementation, designed to demonstrate the viability of using off-the-shelf software components, including the Click modular router, XORP router control plane, OpenVPN, VServers, VNET traffic isolation, and Linux.

The performance numbers for PL-VINI are disappointing, and really highlight the performance differences between GPPs and NPs. Specifically, PL-VINI reports a maximum throughput of 86.2 Mbps at 40% CPU usage. This is an improvement over the 22.5 Mbps sent by a stock PlanetLab node. The measurements were taken on a roughly 1-GHz PC, and it is highly likely that maximum-size Ethernet packets (~1500B) were used. If we assume that per-packet costs dominate, a reasonable assumption, then throughput on PL-VINI for min-size packets plummets to around 10Mb/s. The PL-VINI results are reported for real-time scheduling priority, which is entirely unreasonable when multiple virtual routers are run on a node. These throughput numbers would drop as more virtual routers are loaded. Jitter results are also reported, and come out favorably. PL-VINI seems to have a reasonable jitter of 1.3 ms on a network link with 24.7ms delay. The thing to take away from the PL-VINI discussion is the use of simple TCP throughput, jitter, and utilization as

primary metrics. The NP-based Diversified Router will ideally have numbers for the same metrics.

### 4.1.3 Click Modular Router

The Click Modular Router is a set of Linux kernel modules to simplify the process of creating custom routers [Kohler00]. Click includes a library of elements that can be flexibly connected to describe an application. The built-in elements includes IPv4 forwarders, packet schedulers, lookup engines, and receive and transmit blocks. To enable faster speeds, Click runs in kernel space and synthesizes its own scheduler. The scheduler uses polling rather than OS signals to reduce latency and increase throughput. The journal paper on Click reports that a configuration can forward up to 333,000 minimum-size 64B IP packets, equal to 170 Mb/s, not bad for year-2000 hardware.

## 4.2 NP-Based

NP-based systems are often faster that general-purpose routers, but the flexibility comes at a price: high architectural exposure makes programming NPs extremely difficult . The following two systems attempt to strike the right balance between programmability and performance.

### 4.2.1 NP-Click

NP-Click, unsurprisingly, is an extension of Click's ideas to the Network Processor field [Shah04]. It uses the same application descriptions as Click, but to cope with the lack of C++, it implements a precompiler to generate code suitable for the IXP C compiler. As a result, the element implementations look very different. For performance reasons, NP-Click lets the application developer define thread boundaries over elements, as well as data specifiers.

The most novel idea in NP-Click, and hardest to get right, is its ability to automatically map a Click description onto a set of processing engines in an NP. The mapping problem is formulated as an integer linear program, which is an NP-hard technique, but one that results in reasonable times of around a second on modern hardware. States and tasks are profiled for execution times, memory usage, and access time, then optimized for throughput with a solver. The process produces an optimal assignment of states to memories, tasks to processing elements, and links to communication resources. The strength is that one can change thread boundaries without having to change the application description, while working at a higher-level that gets to correct code more quickly. The weakness is that it is not good for resource constrained applications, and lacks performance guarantees.

### 4.2.2 ShaRE

ShaRE is an acronym for Shangri-La Runtime Environment, and was built as part of Ravi Kokku's PhD Thesis at UT Austin [Kokku05]. Similar to the Diversified Router, it is a system to manage multiple virtual routers on a shared NP. If NP-level sharing is eventually desired on the Diversified Router, ShaRE provides a nice model of how to partition tasks between processors and map them to an NP.

The Baker language used in ShaRE defines Packet Processing Stages (PPSes), which each consist of a finite amount of processing code, and an infinite dequeue-process-enqueue stage. Another component is the Resource Abstraction Layer, which abstracts a programmer from the details of which processors their program is mapped onto, as well as which memories their data is mapped onto. A pipeline compiler enables the PPSes to be mapped onto multiple processing engines. The last component is a runtime system that performs adaptive reconfiguration. The runtime processor scheduler, Everest, is designed to be both agile and wary. It is agile, in that it responds to workload changes quickly, but wary, in that it reduces the frequency of

processor allocation changes. The scheduler minimizes delay variation, and migrates tasks through a save and restore mechanism.

# 5.0 Performance Analysis

Now that we've covered similar systems and benchmarks, we'll cover preliminary performance analysis of the Diversified Router. Keep in mind that the router is a continuously evolving piece of software and hardware, and that by the time you read this the performance numbers may be different. Still, there are benefits to analyzing the factors affecting performance on an unfinished system. The first benefit is the identification of performance bottlenecks; the second is an early base for comparison to other systems.

## 5.1 Simulation Setup

The experiments will run in the Intel Developer Workbench, which includes a cycle-accurate full-architecture simulator. The simulator includes facilities for defining external links and input packets. To match the hardware setup, five input and output ports are defined at 1 Gb/s each. The five input packet streams each run at the full 1 Gb/s rate, and each cycles through a sequence of five packets. The sequence includes varying destination addresses, to achieve full rate.

We'll use the simulator-reported throughput. Since the simulator is cycle-accurate, replications would be pointless. Throughput numbers are reported at cycle 40,000, where human inspection showed that the throughput was reasonably stable. A better analysis would graph the throughput over time, to prove that the simulation has reached a steady state, or simply run for many more cycles. Right now, this is a limitation of the tool environment, and a utility is in the works to log throughput over time.

## 5.2 Experimental Setup

In this experiment, we will use the following factors:

- A: packet size (min-size packets = -1, larger-size packets = +1)
- B: presence of queue managers (QM = -1, no QM = +1)
- C: parse block threading (4 threads = -1, 8 threads = +1)

Min-size packets are 78B at the chip and 90B at the wire, due to Ethernet's 12B inter-frame spacing. These packets are 78B, larger than the Ethernet minimum of 64B, because the current router uses an extra UDP/IP tunnel for compatibility with PlanetLab. Presence of a QM refers to whether the full QM implementation or a simple stub is used. Parse block threading refers to the number of threads parameter passed to the compiler, and the mode in which the code is run; 4-thread mode yields 64 general-purpose registers (GPRs) per hardware thread context, while 8-thread mode yields only 32 GPRs per context.

Each factor was chosen for its potential effect on throughput. Since the header-processing costs per packet are constant, we expect that larger packets will yield greater throughput. The QM is a complicated block and has the potential to be a bottleneck. For parse block threading, it's not clear whether 4 or 8 threads is best. With 4 threads, each thread has more registers available, which may lead to more efficient code. However, more threads means that more latency can be hidden, leading to more efficient processor usage. Note that the pipelined implementation may hide the real effects of some blocks, because the slowest block will cause the majority of the variation. This experiment will assume a linear model for factor effects.

## 5.3 Experimental Results

| | A | B | C | AB | AC | BC | ABC | y |
|---|---|---|---|---|---|---|---|---|
| | -1 | -1 | -1 | 1 | 1 | 1 | -1 | 3190.8 |
| | 1 | -1 | -1 | -1 | -1 | 1 | 1 | 4604.6 |
| | -1 | 1 | -1 | -1 | 1 | -1 | 1 | 3894.3 |
| | 1 | 1 | -1 | 1 | -1 | -1 | -1 | 4402.6 |
| | -1 | -1 | 1 | 1 | -1 | -1 | 1 | 3168.0 |
| | 1 | -1 | 1 | -1 | 1 | -1 | -1 | 4473.1 |
| | -1 | 1 | 1 | -1 | -1 | 1 | -1 | 3917.1 |
| | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 4403.8 |
| Sum | 3713.9 | 1181.2 | -130.2 | -1723.9 | -130.2 | 178.3 | 87.0 | |
| Alpha | **464.24** | **147.65** | **-16.28** | **-215.49** | **-16.28** | **22.28** | **10.87** | |
| Variation | 1724141 | 174407.1 | 2119.98 | 371474.6 | 2120.63 | 3971.63 | 945.91 | |
| Percentage | **75.65%** | **7.65%** | **0.09%** | **16.30%** | **0.09%** | **0.17%** | **0.04%** | |

**Table 1:** Experimental Results

Results are shown in Table 1. The y column on the right represents the observed throughput in Mb/s. To summarize the results:

- Larger packets result in 464 Mb/s greater throughput than average, and account for 75.7% of the variation.
- Removing the queue manager results in 148 Mb/s greater throughput than average, and accounts for 7.7% of the variation.
- The number of threads in the parse block has an inconsequential effect on throughput.
- The interaction between larger packets and the queue manager results in 215 Mb/s lower throughput than average, accounting for 16.3% of the variation.
- Other interactions were minimal.

The large effect of packet size is not surprising, given that header processing costs are dominant for smaller packets. The QM effect is not surprising, given that the code is compute-bound. The lack of a substantial parse effect is a little bit surprising. In isolated tests, parse block threading had a huge effect on overall throughput, however, here, its effect is masked by slower components. Lastly, there may be an explanation for the interaction between the QM and packet sizes. The QM is compute-bound for smaller packets; with larger packets it may become latency-bound, thus having a greater effect than packet size alone.

Unfortunately, none of the configurations achieved full 5 Gb/s rates. This result either implies that the current router is simply incapable of handling full rates for the packet sizes tested, or that unexpected interactions arise at higher rates. The next logical test is to understand the effect of packet sizes in greater detail. If throughput is linear, then packet size is the main contributor to the variation. If not, interactions are throwing off the results and should be investigated more closely. The transmit and receive blocks may also have unexpected interactions at higher speeds, where memory headroom is reduced and memory latencies are higher.

## 5.4 Other Planned Experiments

Since packet size has such an effect on performance, examining it further would be logical. The next planned test is to run packet streams with payload sizes between 0 and 1480B through the router. We'll run the test without a QM first, to better isolate the effects of packet size. If we assume fixed per-packet costs, which is

the case for everything but the receive and transmit blocks, and the receive and transmit blocks are not the bottlenecks, then one would expect to see a linear increase in traffic until the line rate total of 5 Gb/s is reached, at which point the line should be flat. The next test will see if adding the QM results in similar trends over packet sizes. Unexpected memory interaction might cause non-linear performance.

The next set of experiments will investigate the effects of the number of processors on throughput. The current Diversified Router cannot run this experiment, because changing the number of processors (pipeline stages) would require repartitioning the processing, a painful task that may yield no speedup because of communication costs. A *parallel* organization, where each processor runs all the blocks (parse, lookup, and header format), may scale linearly with the number of processors. In fact, the original motivation for this case study was to analyze the effect of parallel and pipelined processor organizations for the stages. This code took too long to develop, but is nearing completion.

The parallel approach is easier to scale to higher speeds than the pipelined approach, which requires repartitioning the application or low-level optimizations to improve speed. The next planned test will look at throughput from one to eight processors. I expect linear scaling up to 4 processors, with throughput saturation around 5 Gb/s for minimum-size packets after that ?unless the QM is the bottleneck. The highest numbers of processors may induce bus saturation, thus reducing the throughput.

# 6.0 Summary

The preliminary data suggests that packet size effects dominate the variation in router throughput in an NP-based Diversified Router. More analysis is necessary to better quantize the sources of variance, demonstrate that the test data truly represents throughput at steady-state, and verify statistical significance of the results. Other systems come nowhere near the Diversified Router's performance; the closest competitor, PL-VINI, is ~500x slower for minimum-size packets. Other tests in the planning will lead to a greater understanding of factors affecting its performance, and enable a direct comparison to existing systems, for additional metrics. When the system is running at speed, the next steps will include implementing and testing network processing benchmarks.

# References

[Bavier06] Andy Bavier, Nick Feamster, Mark Huang, Larry Peterson, Jen Rexford. In VINI Veritas: Realistic and Controlled Network Experimentation: SIGCOMM 06.

[Kohler00] Kohler, E., Morris, R., Chen, B., Jannotti, J., and Kaashoek, M. F, The click modular router. ACM Trans. Comput. Syst. 18, 3, Aug. 2000.

[Shah04] Niraj Shah, William Plishker, Kaushik Ravindran, Kurt Keutzer, NP-Click: A Productive Software Development Approach for Network Processors, IEEE Micro, vol. 24, no. 5, pp. 45-54, Sept/Oct, 2004.

[Kokku05] Ravindranath Kokku, ShaRE: Run-time System for High-performance Virtualized Routers, PhD Thesis, Dec 2005

[Wolf00] Wolf, T, Franklin, M. CommBench-a telecommunications benchmark for network processors. IEEE International Symposium on Performance Analysis of Systems and Software, 2000.

[Memik01] G Memik, WH Mangione-Smith, W Hu, NetBench: A Benchmarking Suite for Network Processors, ICCAD 2001.

[Lee03] Lee, B.K. John, L.K. NpBench: a benchmark suite for control plane and data plane applications for

network processors. Proceedings of 21st International Conference on Computer Design, Oct., 2003.

[Ramaswamy03] Ramaswamy and T. Wolf, ﷺPacketBench: A tool for workload characterization of network processing,?in Proc. of IEEE 6th Annual Workshop on Workload Characterization (WWC-6), Austin, TX, Oct. 2003, pp. 42?0.

[Peterson06] Larry Peterson and Timothy Roscoe, The Design Principles of PlanetLab, June 2004 (updated January 2006). Appears in Operating Systems Review, 40(1):11-16, January 2006.

[RFC1812] F. Baker, RFC1812: Requirements for IP Version 4 Routers, June 1995.

[Shah04] Shah, N., et al. NP-Click: A Productive Software Development Approach for Network Processors, IEEE Micro, 24, 5, (Sept.-Oct 2004), 45-54.

[Kokku05] Kokku, R. ShaRE: Run-time System for High-performance Virtualized Routers. Ph.D. Thesis, University of Texas, Austin, TX, 2005.

Back to Table of Contents

# List of Acronyms

**ASIC** Application-Specific Integrated Circuit
**ATCA** Advanced Telecommunications Architecture
**CB** CommBench
**CMP** Chip Multiprocessor
**DRAM** Dynamic Random Access Memory
**Gb** Gigibit
**GENI** Global Environment for Network Innovation
**GP** General-Purpose Processor
**IP** Internet Protocol
**IPC** Instructions Per Cycle
**IXP** Internet eXchange Processor
**ME** Microengine
**NB** NetBench
**NP** Network Processors
**NPB** NpBench
**NSF** National Science Foundation
**PPS** Packet Processing Stages
**QM** Queue Manager
**RISC** Reduced Instruction Set Computing
**Rx** Receive
**ShaRE** Shangri-La Runtime Environment
**SPEC** Standard Performance Evaluation Council
**SRAM** Static Random Access Memory
**Tx** Transmit
**VINI** Virtual Internet Infrastructure
**VLIW** Very Long Instruction Word
**VPN** Virtual Private Network
**XORP** eXtensible Open-source Router Platform

Back to Table of Contents

---

This report is available on-line at http://www.cse.wustl.edu/~jain/cse567-06/router.htm
List of other reports in this series
Back to Raj Jain's home page