Network Processors

A Critical Processing Function for Real-Time Network Security Solution.

An Analysis of leading Architectures *K Masood* *June 2002*

Abstract

The explosive growth of Internet traffic and the increasing complexity of the functions performed by network nodes have given rise to a new breed of programmable micro-processors called network processors. A major hurdle in designing these processors is a lack of understanding of their workload characteristics. Dearth of literature tackling specific architectural issues has also resulted in an unclear understanding of the design space of these processors. I provide a comprehensive survey of ideas and features employed by current network processor designs.

Lastly, as an initial exploration of this large design space, I analyze packet traces from different sources and show that caching can be gainfully employed to enhance performance for routing table lookup functions in network processors.

1 Introduction

As networks start accommodating complex applications like VoIP, firewall services and virtual private networks (VPNs), the amount of packet processing that a network switch needs to do increases. In simple packet forwarding scenarios, a router parses the header to find the destination address and sends the packet in the right direction as rapidly as possible. But when the network starts to provide value added services like the ones described above, the router needs to analyze headers more thoroughly without compromising the speed of the network. This puts a lot of strain on the processing capabilities of the router.

Current routers address this problem by combining a general-purpose processor with application specific integrated circuits (ASICs) that implement proprietary algorithms for providing the additional functionality. But the ASICs are typically large, are expensive ($200--$400) and often take as long as 18 months to develop. A high-end router employing about a dozen of these has a long time to market and is very expensive. An additional challenge is that once deployed it is very difficult to modify or enhance the functionality to keep up with changing needs and evolving protocols. Network processors address these problems by providing the flexibility that ASICs lack while keeping costs down. By building these processors with programmable off-the-shelf components, the time to market can be reduced substantially.

These advantages have contributed to the emergence of the network processor market as the fastest growing segment of the microprocessor industry. While there is a consensus in the industry about the advantages of a network processor, there is substantial variance in the microarchitecture and chip organization employed by each company. I compare the strategies followed by different companies in some important aspects of processor design with the objective of identifying common trends and exploring the design space for these processors.

2 Design space exploration

The tasks performed by a router can be separated into two planes: the control plane and the data plane. The control plane tasks, which are managed by a general purpose processor, perform functions such as executing routing table updates, flow management, traffic shaping, gathering statistics and managing various engines in a distributed architecture. General

purpose processors are well suited for these tasks by providing the flexibility to address changing needs of the protocols that implement these functions.
The data plane is involved in processing the bits within a packet and is often referred to as "wire speed" processing.
Specialized packet engines perform the necessary functions in this plane.

2.1 System Architecture

The operation of a typical network processor is depicted by the block diagram as shown in Figure 1. Packets move in and out of the network processor through the PHY switch interface

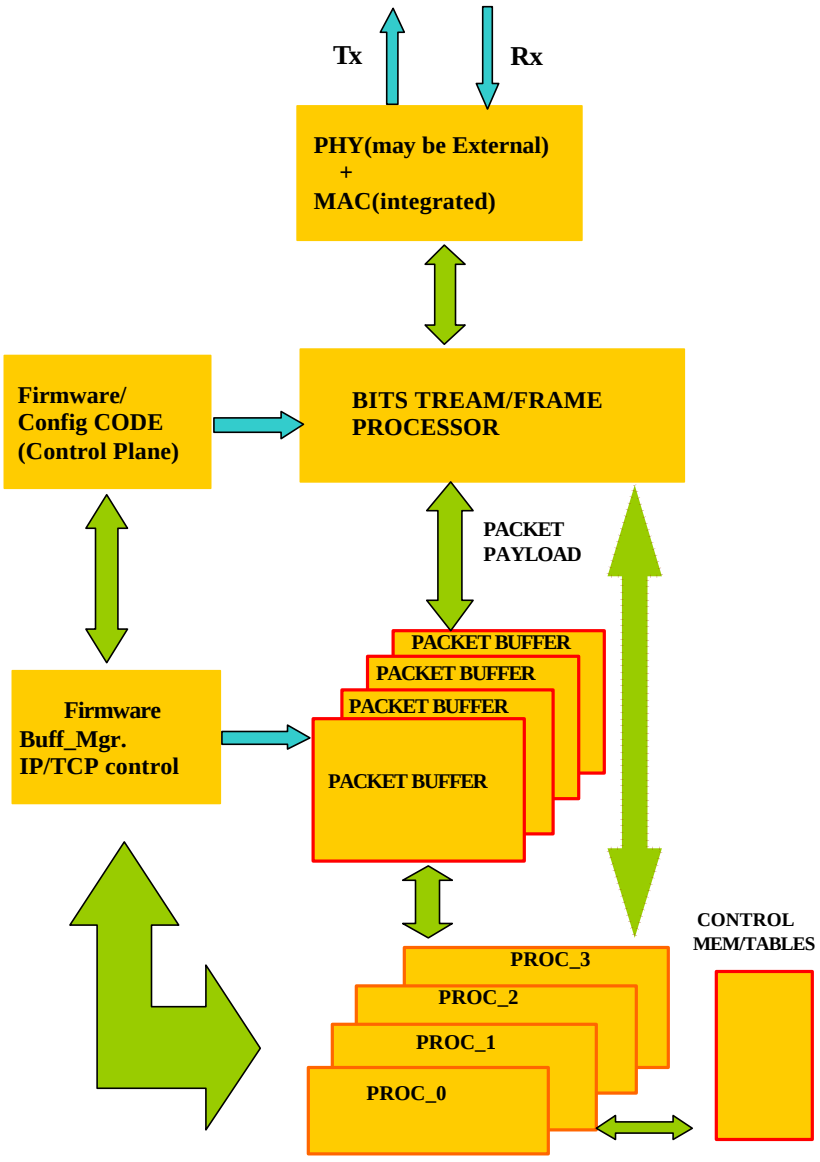

Figure- High Level Block diagram of a typical Network Processor.

The bitstream processors receive the serial stream of packet data and extract the information needed to process the packet, such as the IP source/destination address, type of service bits or

TCP source/destination port numbers. The packet payload is then dispatched to the buffer manager unit which writes the packet into the packet buffer memory. The extracted control information is passed on to the processor complex which constitutes the programmable part of the network processor(NP). Under program control, the processor, if needed, extracts additional information from the packet and submits the relevant part to the search engine, which looks up the next-hop IP address, classifies the packet or does a virtual circuit/path identifier lookup if the packet is recognized as
an asynchronous transfer mode (ATM) cell using the routing and bridging tables and appropriately designed hardware assists. Based on the results returned, the processor instructs the scheduler to determine the appropriate departure time of the packet. The necessary modifications to the packet header are performed when the packet is transmitted to the bitstream processor.

2.2 Processing Power

An OC-48 link operates at a speed of 2.5 Gbps. To sustain this rate, 6 million packets have to be processed per second.
This translates into a throughput requirement of 2.5 billion instructions/second for simple packet forwarding. With OC-768 speeds (40 Gbps) this scales to a whopping 40 billion instructions/second.

3 System Configuration

Network processors are available as standalone products such as the IBM's Rainier and Intel's IXP 1200 and are also available as co-processors to a conventional
RISC microprocessor. An example of the latter is Sitera's IQ2000 which is designed to work with a conventional microprocessor (usually a MIPS or PowerPC chip).
Instruction Functionality FIND BSET find first bit set in any 16 bit register field
HASHx 64 perform 64 bit hash CTX ARB swap contents and wake on event in Intel's IXP ISA additions.
Another example is Agere's PayloadPlus Chipset which is a set of 3 chips assisting a PowerPC processor.

2.4 Instruction Set Architecture

Keeping in mind the special requirements of packet processing, vendors have augmented the traditional RISC ISA with special instructions. Some samples from Intel's IXP 1200 instructions indicate augmented traditional RISC ISA with special instructions.
If "compute and jump" instruction are combined to replace the conventional

```
subi r1 1h
bnz r1 #l1
```

sequence of instructions with a single

`subbi r1 1h #l1` instruction.

Thus a compute operation followed by a conditional branch occurs frequently enough in the workloads studied to warrant this change. The advantages are compact code size and with single cycle execution, substantial speedups in loop intensive code.
Many other ISA additions have been proposed as well. Authors like Paul Bromley recommend use of special instructions for network-specific computation, field extraction, byte alignment, boolean computations and conditional op-codes (to reduce branches). They

recommend use of MMX-VLIW type of instructions to efficiently implement data streaming functions. SiByte's SB-1 NPU core has a "paired-single" SIMD instruction which operates on a pair of 32-bit values stored in a 64-bit register. Vendors have also included table lookup instructions and customized branch instructions suited to analyzing and modifying packet headers.

2.5 Microarchitecture

The micro-architecture of network processors is marked by emphasis on streaming data throughput and heavy use of architectural parallelism. Designs span all the way from non-pipelined such as IBM's Rainier and CPort's C-5 to aggressively pipelined like Agere's Payload Plus, SiByte's SB-1250, EZChip'sNP-1 and Cisco's Toaster2.

Chip multiprocessing with hardware multithreading seems to be a popular technique to exploit the huge thread-level parallelism available in packet processing workloads. Sitera's IQ2002 has four 32-bit scalar cores with 64-bit memory interfaces, so each core can perform a double-word load or store in a single clock cycle. Each core has 5 identical register files (32 registers, 32 bits wide, triple-ported). This arrangement allows each core to run five concurrent threads of execution with fast context switching, because they don't have to save their register states in memory. SiByte's SB-1250 contains multiple integrated SB-1 cores. Each core is a four-issue in-order superscalar design with 6 functional units. IBM's Rainier integrates 16 pico-processors with 1 PowerPC core in a single chip.

Each pico-processor has support for 2 hardware threads. Two pico-processors are packed into a dyadic protocol processor unit and share a tree search engine and internal memory.

Intel's IXP1200 integrates 6 RISC micro-engines with a Strong Arm core. Each micro-engine has its own physical register file, a 32-bit ALU with a basic 5-stage pipeline, a single cycle shifter, an instruction-control store (4K of SRAM), a micro-engine controller and 4 program counters one for each thread that a micro engine can execute in parallel.

Lexra's LX8000 features a cluster of MIPS-like cores with support for fast context switching and chip multiprocessing Cisco's Toaster 2 contains 16 XMC (express microcontroller) cores. Each core is an enhanced ARM-7 LIW design executing a 64-bit instruction word (that consists of 2 RISC instructions) per cycle. EZChip's NP-1 has 4 different processor cores, each of which is optimized for performing a specific portion of the packet processing task. These 4 cores are arranged in a 4-stage super-pipeline, meaning that each stage has many copies of the appropriate core. The NP-1 has 64 cores altogether

XStream Logic's network processor is based on the dynamic multi-streaming (DMS) technique (also known as simultaneous multithreading). The processor core can support eight threads. Each thread has its own instruction queue and register file. The core is divided into 2 clusters of 4 threads each. Every clock cycle, each cluster can issue up to 16 instructions-four from each threadand four of the 16 are selected and dispatched to one of the four functional units in that core for execution. The DMS core has 9 pipe stages and features a MIPS like ISA. Wolf recommends using VLIW designs.

A few companies have taken this approach. Cisco's Toaster 2 as discussed earlier is a LIW design. Motorola's NetDSP has a SC-140 core jointly developed by Motorola and Lucent. The core is a six-issue VLIW DSP with 16 function units, including four multiply accumulate (MAC) units that can execute 1.2 billion operations at the chip's target frequency of 300 Mhz

2.6 Branch Prediction

Branch prediction for network processors doesn't appear to have received as much attention in the literature as other areas. While some processors, like SiByte's SB-1 core have a sophisticated branch prediction mechanism. other designs, such as Sitera's IQ2000, feature only static branch prediction with the branch being predicted always not taken. As of now, the effect of branch prediction on network processor performance seems to be unclear.

2.7 Memory Bandwidth

Estimates of required memory bandwidth to offchip DRAMs depends very much on the amount of memory available on-chip and the way packets are processed. Thus while IBM estimates that it requires 40Gb/s bandwidth for a 10Gbps throughput, EZChip says that it requires 500 Gb/s for the same throughput. But either way, current AM solutions are far from delivering the expected andwidth required for these processors. Techniques like pipelined acccess to memory and very highly interleaved memory structures can be used to alleviate this memory bottleneck. In the future, multiple parallel DRAMs and on-chip ultrawide DRAMs may come to be employed.

2.8 Benchmarks

Benchmarking network processors is a difficult process because of the wide variety of applications that these processors handle and the wide categories into which these products fit in. Vendors quote number of packets processed per second but this doesn't take into the account the amount of processing done per packet. As a consequence, this metric is not very useful. As of now, there is no standard benchmark available. But a Network Processor Benchmark Forum has been formed with the goal of developing benchmark suites in all relevant application domains within the NP space. Parallel efforts in academia have resulted in the publication of a paper titled "Commbench" which describes a suite of applications intended to serve as a benchmark for network processors. The suite is divided into a set of 4 header processing applications and 4 packet processing applications. The authors present results of simulations made on an Ultra-Sparc processor. Their conclusion is that the benchmarks have better instruction cache miss rates compared to SPEC (3.8% vs 8.3% for a 1 KB cache, 0.1% vs 0.5% for a 32KB cache). They also find that a 16 KB data cache is sufficient to limit miss rates to less than 1%. The same group has also published a paper describing a benchmark titled "NetBench". They compare the instruction mix of Netbench with SPEC and show that Netbench executes a lot more arithmetic instructions and fewer load/store instructions compared to SPEC as shown in Figure 2. They claim that only 67 instructions out of the 242 available in the ISA were used by the benchmarks and just 30 instructions were responsible for over 95% of all executed instructions. They argue in favor of a relatively simple RISC core with a fairly limited instruction set. Their stand seems to be vindicated as judged by the design of network processors in the industry.

3 From the general to the specific

As I have described it, the design space of network processors is shown to be very large indeed. As such, it is beyond the scope of this paper to thoroughly examine all areas of this design space. Rather I hope to encourage others to join us in this effort of both exploring the general design space and examining each of its areas in some

depth. As a first step in this direction, I have choosen to focus on the routing table lookup functionality of the network processor. Since routing decisions must be made for every packet moving through the system, I consider this a very important area in which to focus our efforts.

4 Methodology

To examine the design space of the routing table lookup in a network processor, I have collected and analyzed real network packet traces. I am aware of the boot-strapping problem that I face in this endeavor. Predicting the future from the past is an NP-complete problem. Computer designers perhaps face this problem more directly than designers of other systems given the fast moving pace of the computer industry. For example, no one could have predicted twenty years ago the current popularity of the internet.
Even more difficult than predicting growth is attempting to predict the innovations by which enterprising users can use new technologies in novel and unexpected ways. This is the dilemma I have faced in attempting to explore the design space of network processors by examining contemporary packet traces. However, in the absence of packet traces from the future, this is an unsolvable problem. I can only hope to design systems that answer contemporary concerns and attempt to make them sufficiently robust and flexible enough to adapt as users take advantage of the improved technologies.

4.1 Packet trace acquisition

Many organizations, like the ones listed in Table 2, make packet traces publicly available. These traces are collected by the National Laboratory for Applied Network Research (NLANR) and made available at their website. However, due to privacy considerations both the destination and source IP addresses of the packets traces have been anonymized. Although some analyses are still possible with these anonymized traces, evaluating routing table lookups is more difficult for two reasons. The first is that the trace files cannot be run against a routing table because anonymized routing tables are not available. The second is that it is impossible to know which machine addresses exist within the same physical network.

4.1.1 Historical background

However, consistently anonymizing network routing tables and packet traces requires complicated software algorithms. Historically, 32 bit addressed networks have been divided into three categories, Class A, B and C networks. This distinction refers to the length of the network mask that allows routing tables to aggregate collections of machines into logical networks. Without this aggregation, routing tables would need entries for every assigned IP address on the internet. These classes however have led to an inefficient allocation of IP addresses that threaten to exhaust the available 32 bit address space. The reason for this is the lack of flexibility in the system. For example, the smallest possible assignable network, Class C has 256 IP addresses; when a class C network was assigned to a network of only two machines, 253 IP addresses were wasted. This problem gets exponentially worse for each larger network class: the smallest Class B network wastes more than sixty-five thousand addresses and the smallest Class A network wastes almost seventeen million addresses.

In response to this inefficient allocation, Classless Inter domain Routing was introduced in 1993. CIDR allows networks to be of arbitrarily sized between one and seventeen million as long as they are multiples of two.

With the old Class A, B and C networks, it would be trivial to anonymize routing tables and packet traces consistently. The reason for this is that the familiar ”dot” notation of IP addresses (w.x.y.z) corresponds perfectly with the network mask boundaries. Each of the four segments of the address can be viewed as a unique 8 bit identifier. All networks have masks of 8, 16 or 24 bits and therefore do not violate any of the ”dot” boundaries.
The simplest algorithm to anonymize pre-CIDR network routing tables and packets would be to assign each integer between 0 and 255 a different unique value also between 0 and 255. The routing tables and packet traces could be read and each segment of each IP address could be replaced with the random mappings. Privacy would be protected and network identities would be preserved.
Entries in the routing table are sorted descendingly by network mask length. Networks of the same network size are then randomly shuffled. Each network is then assigned the next available and properly aligned addressed starting from any 32 bit offset to the human user. Network mask lengths can fall anywhere within the 32 bit address. CIDR tables need much more complicated algorithms and data structures to be anonymized. Additionally, the encryption created by the anonymization scheme must be strong enough to prevent any decodings.
A naive implementation would merely start at a random offset within the 32 bit address space and assign the next available, properly aligned 1 network to each routing table entry. But this scheme can exhaust the IP address space before all routing table entries can be assigned. This means that any anonymizing scheme must employ some sort of first-fit or best-fit allocation strategy to prevent exhausting the IP space.
Each network must be assigned a starting IP address such that the entire network shares the common netmask prefix. Although this sounds complicated, in practice it can be enforced simply by assigning each network a 32 bit starting address that is evenly divisible by the size of: 5.Router AIX MEM
-Trace File Size 360 MB 180 KB
-Number of Packets 8,200,000 4250
-Number of Distinct IP’s 126,000 85
-Throughput 40 MB/sec 15 KB/sec
-Sample Trace information from two different routers

4.1.3 A modest proposal

What I have done in order to most efficiently allocate anonymized networks to routing table entries is read the entries into a STL multi-map container sorted descendingly
by length of the network mask. Encryption is provided by randomly swapping networks of the same size. The networks are anonymized by iterating through the sorted, randomized networks and assigning each the next available, aligned network. The starting point is irrelevant in this case; for the sake of simplicity, I am currently starting from zero. An example of this anonymization is shown in Table 3. With this encoding scheme, CIDR routing tables and packet traces can now be consistently anonymized. By using patricia trees, longest prefix matching quickly finds the appropriate routing table entry for every IP address

in the packet trace. Our algorithm stores each routing table entry in a patricia tree structure along with its anonymized network value and a hash table. The hash table is used to provide each seen IP address with the next available address within the anonymized network. Each packet trace file can then be read, filtered through the anonymization data structures and written back to disk. In this way, I have developed a novel scheme to allow researchers the ability to run valid trace files against valid routing tables without violating the privacy of the owners of the trace files.

It should be noted that the randomization of the machine id's is therefore based on the order by which they were read from the trace files. Although this is weaker encryption than that used to anonymize the network addresses, it is sufficient given the strong encryption of the network addresses. Additionally, this scheme is designed to operate on multiple packet traces using the same anonymization mapping. Therefore, the order in which the traces files are read is randomized and reverse engineering of the machine addresses remains difficult even assuming that the network address could be compromised.

4.1.4 Physical requirements

Using network statistics collected by the DoIT organization, I have formed an idea about the hardware required to snoop network routers and collect packet traces.

These expectations have been further validated by an analysis of the publicly available traces from NLANR as discussed in Section 5 and shown in Table 4. In our opinion, the speed of the trace traffic is easily handled by current hardware. However, the sheer magnitude of the traces can be somewhat overwhelming. For example, the DoIT routers transfer approximately 200 megabits per second of network traffic. This is handled easily with a gigabit NIC. This traffic consists of an average of 40,000 packets per second. Depending on the packet header information being collected, up to 50 bytes may be written to disk for each packet. This amounts to a bandwidth of 2 MB/s which is easily matched by today's commodity disks. However, these 2 MB/s add up very quickly, filling 1 GB every eight minutes, a commodity disk with 80 GB capacity in 10 hours and a terabyte in less than six days.

4.1.5 Current status

Although complicated, the anonymization algorithms have been implemented. And although the physical requirements of packet collection are easily manageable, we have still found the acquisition of packet traces to be surprisingly difficult. Initially, political considerations of privacy were a stumbling block, especially due to our somewhat unique demand for accompanying routing tables.

However, our novel anonymization software seems to have dissipated these political concerns. Both the network administrator in the CSL and the network administrator in DoIT have agreed to work with us and allow us to collect anonymized routing tables and packet traces. Currently, however, neither of these efforts is proceeding as quickly as I would like. The CSL administrator is yet to reply to our most recent set of emails asking him how and when I should give him the machine that he requested for the CSL snooping. As per his request, I have added the necessary disks and installed RedHat 7.1 on this machine. I now await his availability to receive the machine and attach it to the router. Meanwhile, the DoIT administrator is reading over our anonymization software to check whether he can find any possibilities for reverse engineering the randomization algorithm.

Once validated, he should be able to start collecting anonymized traces as well. However, there has been some discussion about integrating the anonymization algorithms into his patricia tree perl module.

Spatial locality at a single router

A histogram shows the spatial locality of packet references as recorded at SDSC on April 31, 2001. Note that fully 80% of all packets were destined for an IP address which had been previously seen within the last ten packets. it is not entirely clear to us, it seems as if the packet acquisition may be contingent upon this integration. Whether he wants us to do it or whether he wants to do it himself is also not immediately obvious. However, what is clear is that he is very excited about the research ramifications created by this novel ability to publicly provide consistently anonymized routing tables
and packet traces. His enthusiasm for this project and knowledge about the subject have proven very helpful to us in designing the anonymization algorithm and in using
the patricia trie data structure.

4.2 Packet trace analysis software

We have developed software to analyze the packet traces and measure the spatial and temporal locality of references, average packet lengths, and router throughputs.
The analysis software basically consists of three parts:
parsing the input files, interpretation of the acquired data and drawing the corresponding graphs. The NLANR trace files come in many different binary file formats. Although there are some publicly available scripts which allow conversions from the different file formats into simple ASCII representations, I have chosen to do our own parsing of the binary files. Our first reason to do this was the fact that there are storage and time overheads involved in doing conversions as a preprocess stage to the analysis. Secondly, the perl scripts are not very consistent and produce different ascii formats for the different binary representations, which would require us to either modify the perl scripts or add functions to parse each of their different ASCII output formats. Finally, the perl scripts do not always read all the fields out of the packet spacing
(# of packets between two hits to the same IP) Spacial Locality (finer granularity)
These factors motivated us to do our own binary parsing of the trace files and hide its complexity from the rest of our analysis code. This is abstracted in the code by using virtual, object-oriented `read` methods. I used a fast hash-table based implementation to store and search the parsed data in memory. This implementation has improved the speed of our analysis tool by more than an order of magnitude. I choose the simple linked list initially due to its ease of implementation. While debugging our program I read only small trace files; it was only when our code was finished that I attempted to read a large trace file and found that our implementation was prohibitively slow. At this point, I swapped in the faster data structure and are now analyzing packet traces quickly enough so that it is hidden by the I/O latency of moving the trace from disk. Generation of the charts is done by our analysis tool at runtime by redirecting the output of the analysis to gnu plot. Again, this was done out of consideration for both storage and time.

5 Experimental results

The analysis results showed that the traces from different routers have extensive differences in basic characteristics such as the router throughput and the number of distinct IP addresses to which packets passing through that router are forwarded. Although the traces were collected for the same intervals of time from all of the routers, the trace file sizes differ by more than three orders of magnitude in the most extreme cases.

Scatter plot of spatial locality.

The scatter plot shows that spatial locality measurements are consistent across different traces collected from different routers addresses hit and shows these extremes. Despite the differences in the loads and usages of different routers, analysis of the traces showed that all network traffic behaves similarly in terms of spatial and temporal locality and average packet lengths. For most of the routers, about 80% of the packets are forwarded to an IP address which had been previously hit within the last 10
packets. And more interestingly, about 40% of the packets are forwarded to the same IP address as the very previous packet. I believe that these results stress the importance of caching in routers, and show that even minimal caching of the routing table could result in a hit ratio of more than 80%. It shows the spatial locality information for the router.
Analysis on temporal locality is very parallel to the spatial locality and also confirms the importance of caching of the routing table. More than 80% of the packets are forwarded to an IP address which had been previously hit within the last ten milliseconds. Figure 6 shows the tem poral locality information for the SDSC router on April 31, 2001. Finally, the last metric which our analyses measured was the average packet length. Figure 7 shows that more than 60% of the packets from one trace are smaller than 100 bytes in size. The scatter plot in Figure 8 shows that while this is mostly consistent across different routers that there is in some cases a fair amount of large packets as well. This is particularly salient given recent proposals from Juniper that are stretching the definition of the routing RFC's. Amid loud cries of protest from rival Cisco, Juniper has begun optimizing throughput by reordering its internal flow of packets. This strategy may result in larger numbers of packets being buffered at the router and this is even more viable given our observations that most network packets are very small sized.

6 Future work

I can visualize our research proceeding in several different directions. The IP addresses from the anonymized packet traces can be looked up against routing tables from several different network access points to see if the reference patterns change substantially from one point in the network to another. This can have important implications for the design of routing table caches. However packet forwarding is only one of many functions performed by a network processor. The collected packet traces can conceivably be used as input data sets to a simulator that can generate code depending on the type of function that needs to be performed on a packet. The instruction traces thus obtained can then be used to study network processors more effectively.

7 Conclusion

Network processors are a new and exciting offshoot of conventional microprocessors. With the increasing complexity of network applications, they are rapidly gaining importance and

have acquired a niche of their own in the microprocessor market. However due to their recent appearance, the workloads for these processors and the architectural requirements are not understood clearly and are being debated fiercely. I have taken a first step towards understanding the design space of these processors by doing an analysis of current design strategies.