Design Issues for High Performance Active Routers

Tilman Wolf, Jonathan S. Turner

Abstract— **Modern networks require the flexibility to support new protocols and network services without changes in the underlying hardware. Routers with general-purpose processors can perform data path packet processing using software that is dynamically distributed. However, custom processing of packets at link speeds requires immense computational power. This paper proposes a design of a scalable, high performance active router. Multiple network processors with cache and memory on a single application specific integrated circuit are used to overcome the limitations of traditional single processor systems. The proposed design is used as a vehicle for studying the key issues that must be resolved to allow active networking to become a mainstream technology. Benchmark measurements are used to put the design in relation to actual application demands.**

Keywords— **Active networks, Router design, Multiprocessor system-on-a-chip, Programmable network benchmark**

I. Introduction

The growing sophistication of networked applications and the need for more complex network services to support them are creating a growing demand for programmable network components. At the same time, continuing advances in integrated circuit technology are making it possible to implement several complete processor subsystems on a single chip. Active networking is one response to this convergence of application pull and technology push [1], [2]. By increasing the programmability of network components, it promises to dramatically simplify the deployment of new protocols and new network services.

The price of flexible packet handling, though, lies in the inherent lower performance of software processing compared to specialized hardwired logic. Additionally, active networks implement more complex network services than just plain forwarding, which results in even higher requirements for computational power. Another important factor comes from developments in data transmission and switching technology that indicate that link speeds and router capacities will keep growing rapidly. Terabit capacity routers with 2.4 and 10 Gb/s links are becoming commercially available now. Active routers must provide comparable performance levels, in order to keep pace with the rapid growth in bandwidth. As a result, it is important to develop router architectures that can provide enough computational resources to meet these increasing demands.

To put things in perspective, a single processor capable of executing 500 million instructions per second, can perform only about 25 instructions per byte when processing data from a 150 Mb/s link. It is clear that such a single processor system is insufficient to support active processing for even one 2.4 or 10 Gb/s link. This gap can be expected to widen in the future, since link speeds grow faster than processor speeds.

In our approach we combine active processing clusters with a scalable, high performance interconnection network to support large numbers of gigabit links. These processing units can exploit the inherent independence among different traffic flows and process many packets in parallel without the need for complex synchronization mechanisms.

This paper explores issues associated with such a multiprocessor port design for a high performance active router. To enable a concrete examination of these issues, we propose a specific design for an active router that can support hundreds of 2.4 Gb/s links. We use a set of benchmark applications that implement active functionality to determine the computational and IO requirements of such programs. Based on these results, we evaluate the proposed router design.

Section II introduces the overall system design. Sections III and IV explain in more detail the design of the port processor and the active processing engine. Scalability issues are addressed in Section V, and Section VI shows the benchmark results. Section VIII contains a brief summary and conclusions.

II. System Organization

Traditional routers can be augmented to support active processing in different ways. One approach is to add a processing engine at each router port. Another is to provide a shared pool of processing engines that can be used to process traffic from any port. These two baseline system designs are shown in Fig. 1. The routers are based on a scalable cell switching fabric which connects to external links through *Port Processors* (PP). In the first design, all ports are augmented by a *Processing Engine* (PE) that can perform active processing. In the second design, a set of router ports is dedicated to active processing. These ports are equipped with PEs but do not have external interfaces.

The first approach is most appropriate when all ports have comparable requirements for active processing. The second makes sense when ports have widely varying needs. One can also combine these approaches by having both per port processing engines and a shared pool used to augment the processing power of ports with particularly high processing needs.

Packets belonging to *passive flows* (that is, flows that do not require active processing), are passed directly from the input port at which they first arrive to the output port where they are to be forwarded. Such packets encounter no added overhead or delay, compared to a conventional router. Packets belonging to *active flows* are received by the input port and sent to a processing engine (either on the

Tilman Wolf is with the Department of Computer Science, Washington University in St. Louis, MO, USA. E-mail: wolf@arl.wustl.edu
Jonathan Turner is with the Department of Computer Science, Washington University in St. Louis, MO, USA. E-mail: jst@arl.wustl.edu

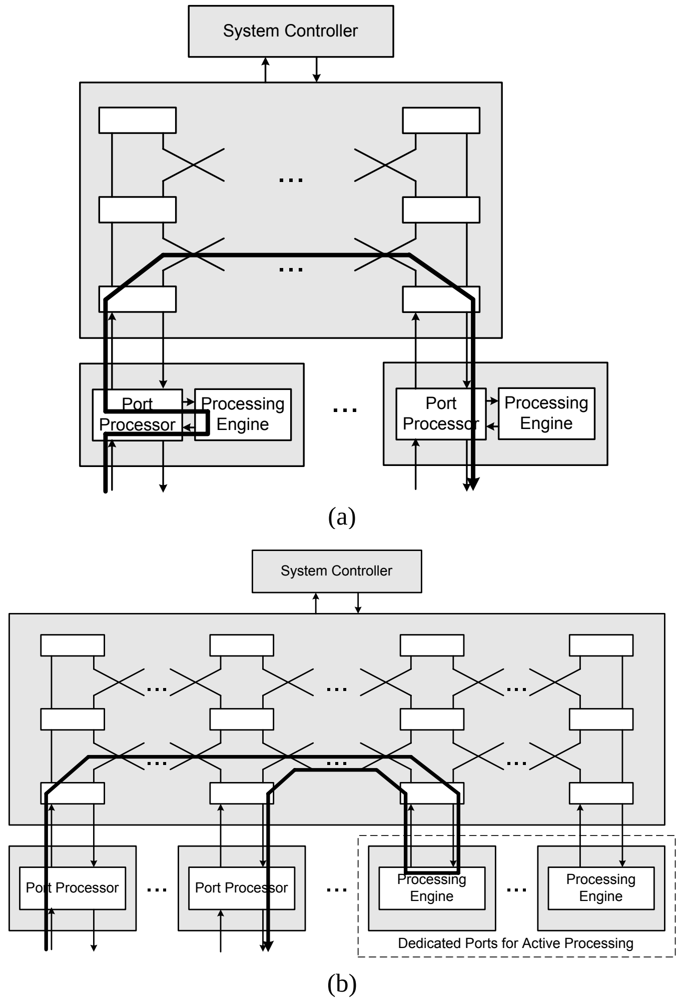

Fig. 1. System Organization of Active Router

same port or on a dedicated processing port) where they are enqueued and eventually processed. After processing, the packets are forwarded to the proper output port. If there are PEs on all ports, processing may also be done at the output port. To provide the maximum flexibility, an input port can distribute packets to various Processing Engines to achieve system-wide load balancing.

The switching fabric can be implemented in a variety of ways. For concreteness, we assume a multistage network such as described in [3]. That system supports external link rates up to 2.4 Gb/s and can be configured to support hundreds or even thousands of such ports. The active router's Port Processors perform packet classification, active processing and fair queuing. The *System Controller* (SC) provides a control and management interface to the outside world and implements routing algorithms and other high level operations.

III. Port Design

The Port Processor and the Processing Engine consist of several components. A detailed picture of a router port equipped with a Processing Engine is shown in Fig. 2.

A. Port Processor

The *Packet Classification and Queuing chip* (PCQ) performs classification of packets arriving from the *Transmission Interface* (TI), to determine how they are to be processed and where they are to be sent. It also manages queues on both the input and output sides of the system. The PCQ has two memory interfaces, one to a *Filter Memory* (FM) used for packet classification and one to a *Queue Memory* (QM) used to store packets awaiting processing or transmission.

As packets are received from the Transmission Interface, the headers are passed to the *Packet Classifier* (PC) which performs flow classification and assigns a tag to the packet. At the same time, the entire packet is passed to the *Queue Controller* (QCTL) which segments the packet into cells and adds it to the appropriate queue. Packets can be assigned to queues in a fully flexible fashion (e.g., per flow or aggregate). The queues can be rate-controlled to provide guaranteed Quality of Service. The filter database determines whether flows are aggregated or handled separately. To provide the required flexibility, a fast general flow classification algorithm is required, such as the one described in [4].

B. Processing Engine

Active processing is provided by one or more *Active Processor Chips* (APC), each containing several on-chip processors with local memory. Each APC also has an external memory interface, providing access to additional memory which is shared by the processors on the chip. The APC processors retrieve active packets from the QM, process them and write them back out to the proper outgoing queue. They are arranged in a daisy-chain configuration to eliminate the need for multiple APC interfaces to the PCQ. Since the bandwidth required between the QCTL and the entire set of APC chips can be bounded by the link bandwidth (assuming each active packet passes from the QCTL chip to an APC once and is returned once), this arrangement does not create a bandwidth bottleneck.

The design can be scaled in a couple ways. First, the number of ports can be increased by configuring the multistage interconnection network to have a larger number of stages. For the design in [3], a three stage network can support up to 64 ports and has an aggregate capacity of 154 Gb/s, while a five stage network can support up to 512 ports and has an aggregate capacity of 1.2 Tb/s. One can increase (or decrease) the active processing capacity by incorporating more or fewer APC chips at each port. For systems with only a small amount of active processing, APCs can be omitted from most ports, and packets requiring active processing can be forwarded from the ports at which they arrive to one of the ports containing an APC.

A key design variable for any router is the amount of memory to provide for queues and how to use that memory to best effect. The usual rule of thumb is that the buffer size should be at least equal to the bandwidth of the link times the expected round trip time for packets going through the network. For 2.4 Gb/s links in wide area

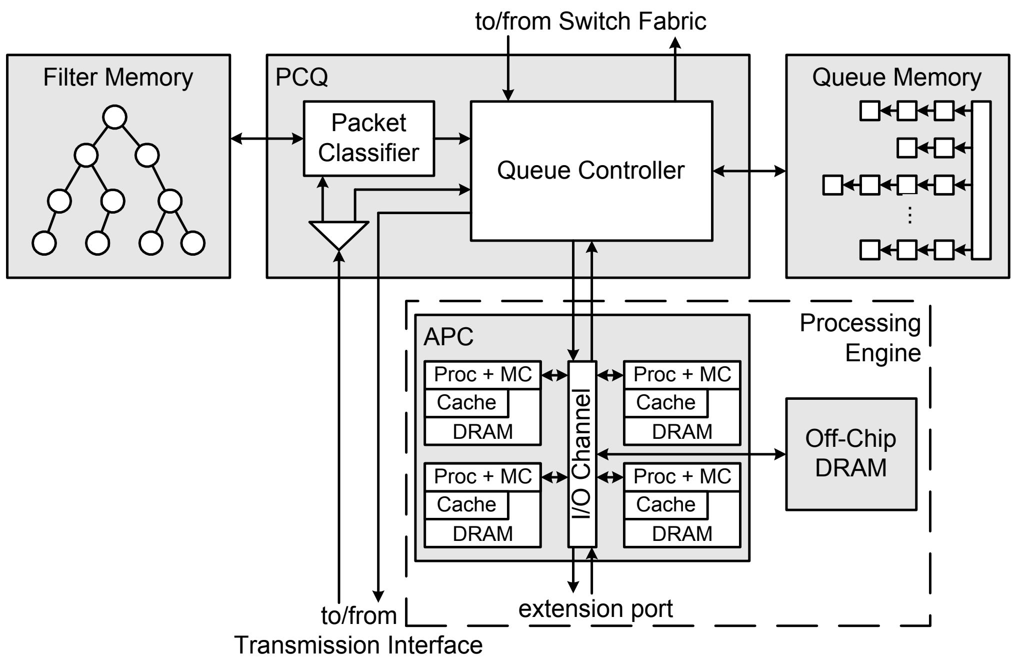

Fig. 2. Port Processor

networks, this leads to buffer dimensions of roughly 100 MB. Such large buffers are needed in IP networks because of the synchronous oscillations in network traffic produced by TCP flow control and the long time constants associated with these oscillations. In the context of large buffers, per flow queuing and sophisticated queuing algorithms are needed to ensure fairness and/or provide quality of service. Flow control is also needed within a router which has hundreds of high speed ports. Without flow control, output links can experience overloads that are severe enough to cause congestion within the switch fabric, interfering with traffic destined for uncongested outputs. Fortunately, the large buffers required by routers make it possible for cross-switch flow control to be implemented with a relatively coarse time granularity (1-10 ms). Using explicit rate control, output PPs can regulate the rate at which different input PPs send them traffic, so as to avoid exceeding the bandwidth of the interface between the switch fabric and the output PP. By adjusting the rates in response to periodic rate adjustment requests from the input PPs, the output PPs can provide fair access to the output links on a system-wide basis or can allocate the bandwidth so as to satisfy quality of service guarantees.

IV. Active Processing Chip

The Active Processing Chip provides the general purpose computational resources needed to implement active networking applications. A set of *Active Processing Units* (APU) which consist of RISC processor cores, cache and DRAM memory are combined on a single application specific integrated circuit (ASIC) to provide a fast and flexible packet processing engine.

A. Design

In order to arrive at a suitable design for the APC, it is important to understand the relative complexity of the different design elements that go into the APC. In current production CMOS technology (.25 μm), a single RISC CPU core can be implemented in an area of 2-4 square millimeters [5], [6]. This represents just 1-2% of a chip with a core area of 200 mm^2, a fairly typical size for high performance ASICs (application-specific integrated circuits). However, processors require memory for storage of programs and data. In .25 μm technology, dynamic RAM requires about 25 mm^2 per Mbyte, while SRAM requires about 175 mm^2 per Mbyte.

Our baseline APC design contains four APUs, as shown in Fig. 2. Each APU includes a Processor, a Memory Controller (MC), a Cache and an on-chip Dynamic RAM (DRAM). These are linked to the PCQ and each other through an IO Channel (IOC). The IOC also provides an interface to an external DRAM and an extension interface, used for linking multiple APCs in a daisy-chain configuration. This design can be readily scaled to larger numbers of processors, as technology improvements make this feasible.

For efficient processing of active flows, the processors should have enough memory to store both a small operating system kernel and the code for the active applications being used. In addition, they need to be able to store per flow state information for perhaps a few hundred flows, and the packets currently being processed. Since the packets can be brought in from the Queue Memory as needed, then promptly written back out, not too much on-chip memory is needed for the packets themselves, but the program code and per flow state could easily consume hundreds of kilobytes of memory. This suggests a minimum memory configuration per processor of 1 MB of DRAM. To allow the processor to operate at peak efficiency, this should be augmented by a cache implemented with SRAM. A 1 MB DRAM and a 32 kB cache together consume about 30 mm^2 of area. Adding a processor and memory controller yields an area of 35-40 mm^2 for an entire APU. This allows four to be combined on a single chip in .25 μm technology.

The required IO bandwidth is another key consideration. As noted above, the bandwidth required for the interface to/from the PCQ can be bounded by the link bandwidth. For 2.4 Gb/s links, this implies a bandwidth of 300 MB/s in each direction. To allow for loss of efficiency due to packet fragmentation effects (caused by packets being divided into cells) and to reduce contention at this interface, it is advisable to increase the bandwidth at this interface to 1 GB/s. This can be achieved with a 32 bit interface in each direction, operating at a clock rate of 250 MHz, which is feasible in .25 μm technology.

The IO Channel connecting the APUs to the QCTL and external memory is another crucial element of the APC. As discussed above, it should support 1 GB/s data transfers to and from the QCTL chip. This leads naturally to a design comprising a crossbar with eight inputs and outputs, each of which is 32 bits wide and clocked at 250 MHz. A central arbiter accepts connection requests from the various "clients" of the crossbar and schedules the requested data transfers to deliver optimal performance. To allow clients to quickly send short control messages to one another, the IOC includes a separate *signal dispatcher* that accepts four byte "signals" from clients and transfers them to per client output queues, where they can be read by the destination client as it is ready. When the destination client reads a signal from its queue, the sending client is informed. This allows clients to regulate the flow of signals to avoid exceeding the capacity of a destination client's queue. Signals are used by the QCTL chip to inform an APU that new data has arrived for it. They are also used by APUs to request transfers of data from the QCTL chip. It is also possible to implement the IOC as a ring. This can give a simpler implementation but may yield larger delays.

The bandwidth required between an APC and its external memory is determined by the number of APUs on the chip, the instruction-processing rate of those APUs and the fraction of instructions that generate requests to the external memory. For example, assuming four 32 bit processors operating at a clock rate of 400 MHz with each APU requiring an average of one external memory access for every twenty clock ticks, we get an external memory bandwidth of 320 MB/s. To reduce contention at this interface, this should be increased to say 500 MB/s. Currently, high performance memory interfaces such as RAMBUS [7] can provide a bandwidth of 1.6 GB/s using just 30 pins.

V. Scaling Issues

The need for scalability in the active router is addressed in two ways. For one, the performance of a single Active Processing Element increases as technology advances. Second, the system can be equipped with multiple Processing Chips to handle increasing link bandwidth.

A. Technology Scaling

Continuing ASIC technology improvements can be expected to contribute to further increases in density and performance. This will allow the next generations of APCs to have more processors and more memory. To estimate this development, the following relations are used:

- The number of gates on an ASIC doubles every 18 months (Moore's Law). Thus, the feature size decreases by a factor of two every three years.
- The clock speed of the processor is inversely proportional to the feature size.

Table I shows the resulting APC configurations. With the above assumptions, a total of 16 processors with 256 kB of cache and 8 MB of DRAM each could be implemented on a single APC by 2005. Thus, even memory intensive applications can be executed efficiently without the need to constantly access off-chip memory. With respect to computational complexity, a single APC will provide sufficient processing power to perform almost a hundred instructions for every byte received from the external link, allowing implementation of fairly complex active processing.

The table shows the link speed to remain constant over time to easily compare different APC configurations. Of course, the link speeds will increase significantly over that period of time. Therefore, it is important to also consider scalability of the overall design.

B. Design Scalability

The design of the Active Processing Chip also allows scalability with respect of the number of processing chips in the system. This is important since the scaling of chip performance due to Moore's Law is not enough to close the growing gap between link speeds and processor and memory speeds.

The IO Channels of multiple Active Processing Chips can be connected via the extension port to form a chain of processing chips. Each interface that connects to another processing chips acts as a gateway and routes data to other APCs further down in the chain. This design requires that the IO Channel be able to handle the total bandwidth between the Queue Controller and the Active Processing Chips. It can be assumed that this requirement can be met even for faster link rates. Note, that the total amount of bandwidth between the Queue Controller and the Active Processing Chips is at most twice the external link bandwidth, since each packet is sent at most once to the processing chips and sent at most once back to the Queue Controller. Since each Processing Chip has its own off-chip memory interface, the traffic from off-chip memory accesses is restricted to the individual APC and does not aggregate over multiple chips. Finally, only data traffic that requires active processing needs to be sent to Active Processing Units. While we expect active processing to be an important element of future routers, we expect most packets to be forwarded without active processing for the forseeable future.

Year	1999		2002		2005		2008
Feature size (μm)	0.25	0.18	0.12	0.09	0.06	0.045	0.03
No. of APUs	4	4	8	8	16	16	32
Cache size (kB)	32	64	64	128	128	256	256
DRAM size (MB)	1	2	2	4	4	8	8
Proc + MC area (mm^2)	10	5.2	2.3	1.3	0.6	0.3	0.14
SRAM area per MB	175	90	40	23	10	5.7	2.5
DRAM area per MB	25	13	5.8	3.2	1.4	0.8	0.4
Total APU area (mm^2)	162	148	131	137	122	132	117
Processor clock frequency (MHz)	400	556	833	1,111	1,667	2,222	3,333
External memory bandwidth (MB/s)	500	694	2,083	2,778	8,333	11,111	33,333
Instructions per byte for 2.4 Gb/s link	5.3	7.4	22	30	89	119	356

TABLE I
APC Technology Scaling

VI. Application Requirements

To put the router design in relation to requirements of actual applications, we use an application benchmark to determine computational complexity and IO demands.

A. Benchmark

We have assembled a set of applications to serve as a benchmark for active processing. They range from fairly complex encryption and compression programs to simple IP address lookup and packet scheduling. Table II lists the programs in the benchmark set, along with some characteristic data.

The object code size is the size of the compiled but unlinked code. This size varies significantly over the different applications since it is highly dependent on the implementation. Commercial strength implementations contain large amounts of error handling code, support for various input formats, and other code that is rarely executed. Simple proof-of-concept implementations on the other hand just focus on the basic functionality of the program. To discount this variation, the amount of actually executed code is shown in the next column. This counts all the instruction code that is executed at least once including library code. The remaining "dead code" can be ignored when implementing those applications on an active router. For DRR and FRAG, the library code increases the amount of executed code, but for the other applications this is less than the object code. In general, no applications executes more than 30 kB of code which indicates that small, computationally intense program 'kernels' make up for most of the computations. Still, an amount between 6kB and 30kB per program is fairly large and will consume a significant part of an APU's on-chip cache space. This suggests that it may be necessary to "specialize" the different APUs, by limiting each one to a small set of distinct programs.

The computational complexity shown in the last column gives the number of instructions executed per byte of data in a packet. For applications that process only the packet header (i.e., RTR, FRAG, DRR, and TCP) a 64 byte packet is assumed. For applications that process the payload of a packet, a 1 Mbyte data stream is assumed. The table shows that these programs are computationally demanding. CAST, for example, requires about 100 instructions per byte. Consulting Table I, we see that this implies that a single APC will be able to encrypt all the data on a 2.4 Gb/s link only sometime after 2005. Even this is based on the combined efforts of 16 APUs. Of course, versions of these applications tailored specifically for the active network environment, could well provide significantly better performance. However, it seems clear that to support substantial amounts of active processing, routers will need to configured with enough computational power to execute hundreds of instructions per byte of data processed, a fairly demanding requirement for current technology. More details on the benchmark can be found in [8].

Another important measure that can be obtained from the benchmark is the amount of IO traffic generated on the different interfaces of the APUs and APCs. The results are based on the application instruction mix and the cache miss rates shown in Table III. Table IV shows the IO rates for a configuration with a 400 MHz processor, a 32 kB cache that is logically split into a 16 kB instruction cache and a 16 kB data cache, and on-chip DRAM of 1 MB. The table shows the amount of traffic generated on the interface between cache and on-chip DRAM as well as between the on-chip memory and external memory. For these simulations it was assumed that cache misses do not introduce stall cycles. This is a worst case assumption, since IO rates only decrease when stall cycles are introduced. It can clearly be seen that an external memory interface of 500 MB/s, as suggested in Table I, is sufficient for any type of application, even if the interface is shared among four APUs. As on-chip cache and memory sizes and IO bandwidth increase, the contention on this interface becomes even less. This measurement assumes that one APU processes only one application. It has to be investigated how much the IO rate increases when on-chip cache and memory is shared among different applications due to additional capacity cache misses. Still, it is important to provide significantly more bandwidth than just the average IO rate since memory access patterns are often bursty and

Application	Description	Object Code (bytes)	Executed Code (bytes)	Complexity (instr. / byte)
RTR	routing table lookup	16,000	15,220	2.1
DRR	packet scheduling	2,500	5,412	4.1
FRAG	packet fragmentation	2,400	5,032	7.7
TCP	traffic monitoring	352,000	29,028	10
JPEG	image compression	260,000	24,620	81
CAST	data encryption	19,500	10,116	104
ZIP	data compression	117,000	14,152	226
REED	forward error correction	6,900	6,040	603

TABLE II
Size and Computational Complexity of Benchmark Applications

Application	% Load & Store Instr.	% Arithm. & Logic Instr.	I-Cache Miss Rate (16kB cache)	D-Cache Miss Rate (16kB cache)
RTR	42.9%	11.4%	0.18%	0.29%
DRR	47.6%	14.7%	< 0.01%	0.88%
FRAG	19.6%	36.0%	< 0.01%	3.98%
TCP	24.0%	33.0%	< 0.01%	< 0.01%
JPEG	25.6%	51.0%	0.01%	0.97%
CAST	27.1%	54.6%	< 0.01%	0.39%
ZIP	25.8%	39.6%	< 0.01%	16.28%
REED	19.6%	42.3%	< 0.01%	0.02%

TABLE III
Instruction Mix and Cache Performance of Benchmark Applications

Application	Cache→DRAM (MB/s)	DRAM→Cache (MB/s)	DRAM→Off-chip DRAM (MB/s)	Off-chip DRAM→DRAM (MB/s)
RTR	37.24	4.55	4.87	2.94
DRR	53.69	37.08	0.06	0.03
FRAG	38.46	16.54	0.31	0.17
TCP	0.53	0.15	0.45	0.12
JPEG	33.99	15.82	9.04	5.23
CAST	15.00	5.26	0.14	0.03
ZIP	541.42	74.40	0.56	0.47
REED	0.41	0.20	0.02	0.01
Average	90.09	19.25	1.93	1.13

TABLE IV
IO Transfer Rates of Benchmark Applications with 16 kB Instruction Cache, 16 kB data Cache, and 1 MB On-Chip DRAM on a 400 MHz Processor.

thus lead to peaks in IO usage.

These measurements of computational and IO requirements of our benchmark applications show that the proposed router design will be able to accommodate such programs and process them efficiently.

VII. Related Work

Programmable packet processing engines for routers have recently become available commercially. So called "network processors" perform processing from the data link layer to the application layer. The following list of products gives a brief overview over available and publicly announced systems and some of their basic characteristics:

- IBM PowerNP [9]: 16 processing units, one control processor, 133 MHz clock rate, 1.6 GB/s DRAM bandwidth, 8 Gb/s line speed, 2 threads per processor.
- Intel IPX1200 [10]: 6 processing engines, one control processor, 200 MHz clock rate, 0.8 GB/s DRAM bandwidth, 2.6 Gb/s line speed, 4 threads per processor.
- Lexra NetVortex [11]: 16 processing units, 450 MHz clock

rate, 4 threads per processor.

• Lucent Fast Pattern Processor [12]: 3 VLIW processing units, one control processor, 133 MHz clock rate, 1.1 GB/s DRAM bandwidth, 5 Gb/s link rate, 64 threads per processor.

• MMC nP3400 [13]: 2 processing units, 220 MHz clock rate, 0.5 GB/s DRAM bandwidth, 5 Gb/s aggregate throughput, 8 threads per processor.

• Motorola C-5 [14]: 16 processing units, one control processor, 200 MHz clock rate, 1.6 GB/s DRAM bandwidth, 5 Gb/s line speed, 4 threads per processor.

• Tsqware TS704 [15]: 4 processing units, 90 MHz clock rate, 0.3 Gb/s DRAM bandwidth.

• Vitesse Prism IQ2000 [16]: 4 processing units, 200 MHz clock rate, 1.6 GB/s DRAM bandwidth, 6.4 Gb/s aggregate throughput, 5 threads per processor.

All network processors are system-on-a-chip designs that combine processors, memory, and IO on a single ASIC. The processing engines in these network processors are typically RISC cores, like, MIPS [17], ARM [18], or PowerPC [6], which are augmented by specialized instruction, multithreading, and zero-overhead context switching mechanisms. The on-chip memory of these processors is in the range of 100 to 500 kB.

Another approach is to use SIMD (single instruction, multiple data) processors, which are mainly used for media processing and scientific applications, and adapt them to the network environment. PixelFusion uses their Fuzion 150 [19] for this purpose. It has 1,500 simple processing units clocked at 200 MHz with 24 MB on-chip DRAM memory and 6.4 Gb/s external bandwidth. It has yet to be shown, though, that network processing can be parallelized in a fashion that allows parallel processing with a single instruction stream.

Today's commercial network processors are mainly optimized for packet header processing applications. The active router that we propose is insofar different that it is not geared towards processing of independent packets, but towards true general purpose processing of data streams that span many packets. This requires the ability to store per-flow state information (e.g. encryption keys or partial video frames) and make the accessible to all packets of one flow.

Software environments for packet processing on programmable routers have received much attention in recent years and many such systems have been developed [20], [21], [22], [23], [24]. Most prototypes are implemented on workstations that act as routers, but the basic concepts apply to multi-port, multi-processor routers, too. In principle, any such system could be used to handle operating system, code distribution, and control operations in a network processor.

VIII. Conclusions

Active networking is an important new direction in networking research and potentially for commercial networks. This paper is an attempt to determine how a practical high performance active router might be built. We use multiple processing clusters that operate in parallel to provide in computational power that is required for active applications. Developments in chip technology will allow configurations that can handle even complex applications that touch every byte of payload at link rates of 2.4 Gb/s. Our proposed design provides a concrete reference point that can help focus more detailed studies of specific design issues. It also provides a useful basis for extrapolation, as underlying IC technologies continue their inexorable progress to ever smaller geometries and higher performance levels.

References

[1] David L. Tennenhouse, Jonathan M. Smith, W. David Sincoskie, David J. Wetherall, and Gary J. Minden, "A survey of active network research," *IEEE Communications Magazine*, vol. 35, no. 1, pp. 80–86, Jan. 1997.

[2] Andrew T. Campbell, Herman G. De Meer, Michael E. Kounavis, Kazuho Miki, John B. Vincente, and Daniel Villela, "A survey of programmable networks," *Computer Communication Review*, vol. 29, no. 2, pp. 7–23, Apr. 1999.

[3] Tom Chaney, Andy Fingerhut, Margaret Flucke, and Jonathan Turner, "Design of a gigabit ATM switch," in *Proc. of IEEE INFOCOM 97*, Kobe, Japan, Apr. 1997.

[4] Venkatachary Srinivasan, Subhash Suri, and George Varghese, "Packet classification using tuple space search," in *Proc. of ACM SIGCOMM 99*, Cambridge, MA, Sept. 1999.

[5] ARC Inc, *ARC Architecture*, 1999, http://www.arc-cores.com/product/features.htm.

[6] IBM Microelectronics Division, http://www.chips.ibm.com/products/powerpc/cores/405cr_wp.pdf, *The PowerPC 405^{TM} Core*, 1998.

[7] Rambus Inc., *Rambus(R) Technology Overview*, Feb. 1999, http://www.rambus.com/docs/techover.pdf.

[8] Tilman Wolf and Mark A. Franklin, "CommBench - a telecommunications benchmark for network processors," in *Proc. of IEEE International Symposium on Performance Analysis of Systems and Software*, Austin, TX, Apr. 2000, pp. 154–162.

[9] IBM Corp., *IBM Power Network Processors*, 2000, http://www.chips.ibm.com/products/wired/communications/network_processors.html.

[10] Intel Corp., *Intel IXP1200 Network Processor*, 2000, http://developer.intel.com/design/network/ixp1200.htm.

[11] Lexra Inc., *NetVortex Network Communications System Multiprocessor NPU*, 2000, http://www.lexra.com/products.html.

[12] Lucent Technologies Inc., *PayloadPlus^{TM} Fast Pattern Processor*, Apr. 2000, http://www.agere.com/support/non-nda/docs/FPPProductBrief.pdf.

[13] MMC Networks, Inc., *nP3400*, 2000, http://www.mmcnet.com/.

[14] C-Port Corporation, *C-5^{TM} Digital Communications Processor*, 1999, http://www.cportcorp.com/solutions/docs/c5-brief.pdf.

[15] T.sqware Inc., *TS704 Edge Processor Product Brief*, 1999, http://www.tsqware.com/.

[16] Sitera Inc., *Prism IQ2000 Network Processor Family*, 2000, http://www.sitera.com/products/iq2000.pdf.

[17] MIPS Technologies, Inc., *JADE - Embedded MIPS Processor Core*, 1998, http://www.mips.com/products/Jade1030.pdf.

[18] ARM Ltd., *ARM9E-S - Technical Reference Manual*, Dec. 1999, http://www.arm.com.

[19] PixelFusion, Ltd., *Fuzion 150 Product Overview*, 2000, http://www.pixelfusion.com/products/FUZION150A4_Product_Overview.pdf.

[20] Dan Decasper, Guru Parulkar, Sumi Choi, John DeHart, Tilman Wolf, and Bernhard Plattner, "A scalable, high performance active network node," *IEEE Network Magazine*, vol. 31, no. 1, pp. 8–19, Jan. 1999.

[21] D. Scott Alexander, William A. Arbaugh, Michael W. Hicks, Pankaj Kakkar, Angelos D. Keromytis, Jonathan T. Moore, Carl A. Gunter, Scott M. Nettles, and Jonathan M. Smith, "The SwitchWare active network architecture," *IEEE Network Special Issue on Active and Controllable Networks*, vol. 12, no. 3, pp. 29–36, Aug. 1998.

[22] Shashi Merugu, Samrat Bhattacharjee, Ellen W. Zegura, and James Sterbenz, "Bowman: A node OS for active networks," in *Proc. of IEEE INFOCOM 2000*, Tel Aviv, Israel, Mar. 2000.
[23] David J. Wetherall, John Guttag, and David L. Tennenhouse, "ANTS: A toolkit for building and dynamically deploying network protocols," in *Proc. of IEEE OPENARCH 98*, San Francisco, CA, Apr. 1998.
[24] John J. Hartman, Peter A. Bigot, Patrick Bridges, Brady Montz, Rob Piltz, Oliver Spatscheck, Todd A. Proebsting, Larry L. Peterson, and Andy Bavier, "Joust: A platform for liquid software," *IEEE Computer Magazine*, vol. 32, no. 4, pp. 50–56, Apr. 1999.

Tilman Wolf received his Diplom in Computer Science from the Universität Stuttgart, Germany in 1998. From Washington University in St. Louis, he obtained a M.S. in Computer Science in 1998 and a M.S. in Computer Engineering in 2000. Currently, he is pursuing a doctoral degree at the Department of Computer Science at Washington University. His research interests are high-performance programmable routers, network processor design, active networking, and benchmarking.

Jonathan S. Turner received the MS and PhD degrees in computer science from Northwestern University in 1979 and 1981. He holds the Henry Edwin Sever Chair of Engineering at Washington University, and is Director of the Applied Research Laboratory. He served as Chief Scientist for Growth Networks, a startup company that developed scalable switching components for Internet routers and ATM switches. The company was bought by Cisco Systems for $355 million in early 2000. Professor Turner's primary research interest is the design and analysis of switching systems, with special interest in systems supporting multicast communication. The Applied Research Laboratory is currently engaged in a variety of projects ranging from Active Networking, to Network Management and Visualization, to WDM Burst Switching. He received the Koji Kobayashi Computers and Communications Award from the IEEE in 1994 and the IEEE Millenium Medal in 2000. He has been awarded more than 20 patents for his work on switching systems and has many widely cited publications. His research interests also include the study of algorithms and computational complexity, with particular interest in the probable performance of heuristic algorithms for NP-complete problems. He is a member of the the ACM, SIAM and a fellow of the IEEE.