Report Number: WUCS-99-19

1999-01-01

Design Issues for High Performance Active Routers

Tilman Wolf and Jonathan Turner

Active networking is a general approach to incorporating general-purpose computational capabilities within the communications infrastructure of data networks. This paper proposes a design of a scalable, high performance active router. This is used as a vehicle for studying the key design issues that must be resolved to allow active networking to become a mainstream technology.

Design Issues for High Performance Active Routers

Tilman Wolf and Jonathan Turner

WUCS-99-19

August 1999

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
St. Louis MO 63130

Design Issues for High Performance Active Routers

Tilman Wolf, Jonathan Turner
{wolf,jst}@arl.wustl.edu

Abstract

Active networking is a general approach to incorporating general-purpose computational capabilities within the communications infrastructure of data networks. This paper proposes a design of a scalable, high performance active router. This is used as a vehicle for studying the key design issues that must be resolved to allow active networking to become a mainstream technology.

1 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 [8]. By increasing the programmability of network components, it promises to dramatically simplify the deployment of new protocols and new network services.

In active networks employing the *capsule model*, each data packet carries the instruction code that is used to process and forward the packet. Augmenting every packet with processing instructions enables flexible handling of the packet at every node. Active packets can contain code that process th payload in some fashion or which influences the way the network routes or schedules the packet for transmission. The capsule model for active networking raises signficant security questions, which are typically handled using interpretive execution. An alternative model for active networking [3] allows execution of compiled code obtained from trusted code servers, enabling substantially faster processing. Both models for active networking require substantial computational capabilities to keep pace with ever faster networks.

Developments in data transmission and switching technology 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. Routers capable of substantial amounts of active processing will need extensive computational resources and will need to be able to apply those resources flexibly and with high efficiency. To put things in perspective, a 32 bit processor capable of executing 500 million instructions per second, can perform about 100 instructions per word when processing data from a 150 Mb/s link. While processor speeds will continue to improve, it is clear that to support active processing for even one 2.4 or 10 Gb/s link it will take many processors working in parallel. To support large numbers of gigabit links, we must combine many active processing clusters with a scalable, high performance interconnection network.

This paper explores issues associated with the design of high performance active routers, capable of supporting large numbers of gigabit links. 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. This is used as a vehicle for understanding complexity and performance issues, and as a basis for extrapolation into the future.

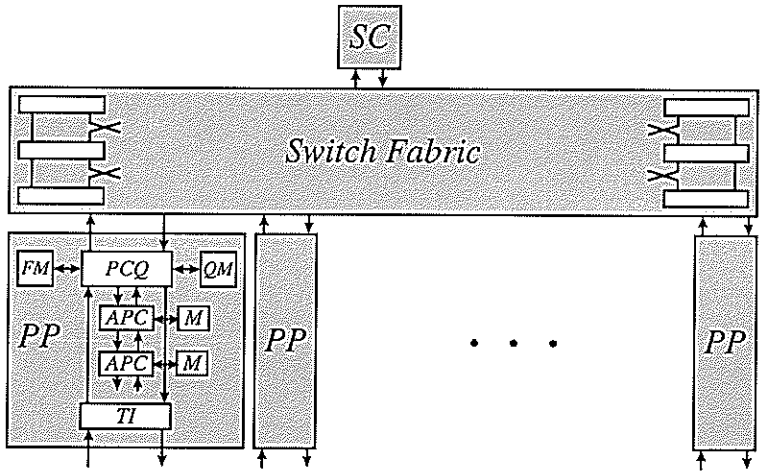

Figure 1: System Organization of Active Router

2 System Organization

A baseline design for a high performance active router is shown in Figure 1. The router is based on a scalable cell switching fabric which connects to external links through active *Port Processors* (PP). The switching fabric can be implemented in a variety of ways. For concreteness, we assume a multistage network such as described in [2]. 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 queueing. The *Switch Controller* (SC) provides a control and management interface to the outside world and implements routing algorithms and other high level operations.

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* will typically be queued for processing at the input port where they arrive, then after processing will be forwarded to the proper output port. Active processing can also be performed at the output, if appropriate. To provide the maximum flexibility, packets may be sent from the input ports where they arrive to another port for active processing, before ultimately being fowarded to the desired outgoing link. This allows system-wide load balancing.

As seen in Figure 1, the Port Processor consists of several components. The *Transmission Interface* (TI) includes the optoelectronic and transmission formatting components. The *Packet Classification and Queueing chip* (PCQ) performs classification of arriving packets, 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. 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 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.

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.

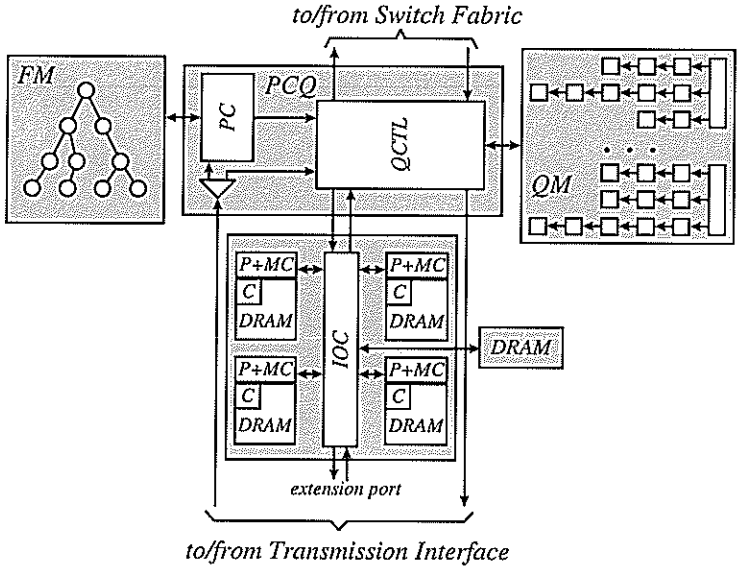

Figure 2: Port Processor

A more detailed picture of the Port Processor is shown in Figure 2. 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. Either per flow or aggregate queues may be used, depending on the QoS requirements. 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 [7].

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 [2], 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 equal 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 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. In the context of large buffers, per flow queueing and sophisticated queueing 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 link on a system-wide basis or can allocate the bandwidth so as to satisfy quality of service guarantees.

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
P+MC area (mm^2)	10	5.2	2.3	1.3	0.6	0.3	0.14
SRAM area/MB	175	90	40	23	10	5.7	2.5
DRAM area/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 word for 2.4 Gb/s link	21	30	90	120	360	470	1,400

Figure 3: APC Technology Scaling

3 Active Processing Chip

The Active Processing Chip provides the general purpose computational resources needed to implement active networking applications. 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 [1, 4]. This represents just 1-2% of a chip with a core area of 200 mm^2, a fairly typical size for high performance ASICs. However, processors require memory for storage of programs and data. In .25 micron technology, dynamic RAM requires about 25 mm^2 per Mbyte, while SRAM requires about 175 mm^2 per Mbyte.

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 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 *Active Processing Unit* (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. This bandwidth will need to increase only based on increases in the external link bandwidth.

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 [5] can provide a bandwidth of 1.6 GB/s using just 30 pins.

Figure 3 shows how continuing technology improvements can be expected to contribute to further increases in density and performance. This assumes that processor clock frequency increases in inverse proportion to feature size. The table shows that by 2005, it should be possible for a single APC to provide

application	description	code (lines)	obj. code (bytes)	instructions/word
DES	data encryption	700	16,500	480
ZIP	data compression	2,000	72,500	640
JPEG	image compression	7,000	100,000	250
Reed-Solomon	forward error correction	250	5,000	1,950
IP	packet forwarding	2,000	25,000	60
Radix-Tree	routing table lookup	1,100	17,500	145

Figure 4: Representative Applications

sufficient processing power to execute hundreds of instructions for every 32 bit word received from the external link, allowing implementation of fairly complex active processing.

Our baseline APC design contains four APUs, as shown in Figure 2. Each APU includes a Processor (P), a Memory Controller (MC), a Cache (C) 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.

The IO Channel is a crucial element of the APC. As discussed above, it should support 1 GB/s data transfers to and from the QCTL chip. This leads natuarally 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.

4 Preliminary Application Benchmark

We have assembled a small set of applications to serve as a benchmark for active processing. They range from fairly complex encryption and compression programs, to simple packet forwarding and IP address lookup. Figure 4 lists the programs in the benchmark set, along with some characteristic data. The column labeled code, gives the number of lines of code after stripping comments and blank lines. The column labeled instructions/word gives the number of instructions executed per 32 bit word of data, when processing a 1 Mbyte file. The instruction counts exclude instructions associated with input and output, to better focus on the core computations. More details on the benchmark can be found in [9].

The table shows that some of the programs are fairly large and will consume a significant part of an APU's on-chip memory 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 table also shows that these programs are computationally demanding. DES, for example, requires nearly 500 instructions per word. Consulting Figure 3 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 excute hundreds of instructions per word of data processed, a fairly demanding requirement for current technology.

The benchmark has also been used to make a preliminary assessment of the mix of machine instructions that might be expected for active applications. Figure 5 compares the instruction mix for our benchmark against the SPECint92 benchmark, commonly used for comparing processor performance.

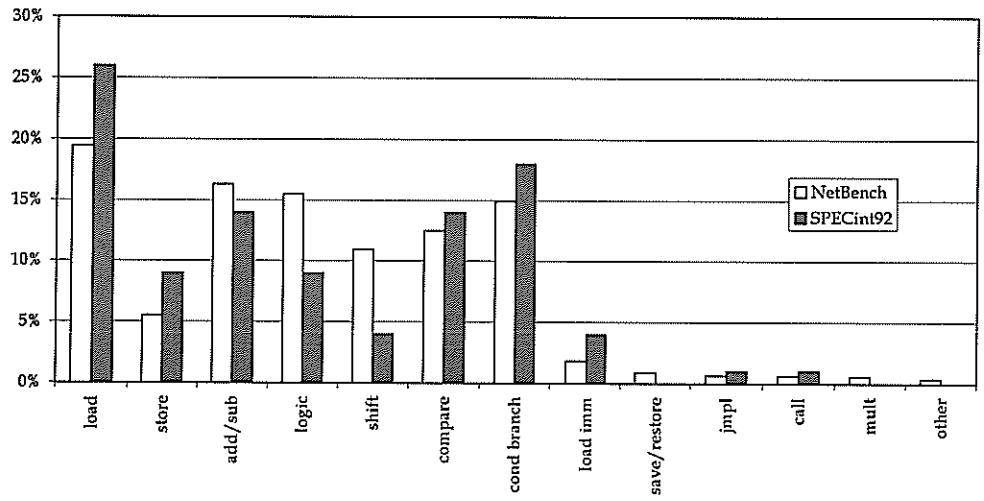

Figure 5: Comparison of Networking Applications to SPECint92 Benchmark

Compared to the SPEC benchmark, our "NetBench" benchmark has a somewhat lower fraction of load and store operations and a somewhat higher fraction of arithmetic and logic operations. A more detailed examination of the results shows that the execution is dominated by a fairly small number of instructions. During the entire NetBench measurement run, only 67 instruction out of the 242 available were ever executed (the experiments were run on a SPARC Ultra). Just 30 instructions were responsible for 95% of all executed instructions. Floating point instructions played a negligible role. These results support the use of a relatively simple RISC processor core, with a fairly limited instruction set. However, further study is needed to determine if certain specialized instructions could be beneficial in this environment. In particular, one expects that support for short vector processing (e.g. Intel's MMX extensions) might be beneficial for applications like encryption and multimedia.

5 Closing Remarks

Active networking is an important new direction in networking research and potentially for commercial networks. This paper is a first attempt at determining how a practical high performance active router might be built. 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] ARC Inc. [1999]. "ARC Architecture," http://www.arccores.com/products/.

[2] Chaney, T., Fingerhut, A., Flucke, M., Turner, J. [1997]. "Design of a Gigabit ATM Switch," *Proc. of INFOCOM 97*, IEEE, Kobe, Japan.

[3] Decasper, D., Plattner, B. [1998]. "DAN - Distributed Code Caching for Active Networks," *Proc. of INFOCOM 98*, IEEE, San Francisco.

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

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

[6] Shreedhar, M., Varghese, G. [1995]. "Efficient Fair Queueing using Deficit Round Robin," *Proc. of SIGCOMM 95*, ACM, Cambridge, Mass.

[7] Srinivasan, V., Suri, S., Varghese, G. [1999]. "Packet Classification using Tuple Space Search," to appear in *Proc. of SIGCOMM 99*, ACM, Cambridge, Mass.

[8] Tennenhouse, D., Smith, J., Sincoskie, W., Wetherall, D., Minden, G. [1997]. "A Survey of Active Network Research," *IEEE Communications*, 35:1, 80-86.

[9] Wolf, T. [1999]. "NetBench - A Benchmark for Active Networks," to be submitted to *Allerton Conference on Communications, Control, and Computing*, Illinois.