Network Processors - Flexibility and Performance for Next-Generation Networks

Tilman Wolf
Department of Computer Science
Washington University
St. Louis, MO
wolf@ccrc.wustl.edu

Significant efforts in networking research have been aimed at developing new protocols and services. However, the ability to deploy these in the current Internet is greatly inhibited by their need for changes in the forwarding loops of routers, which for performance considerations are usually implemented in custom logic. To overcome this obstacle, it has been proposed to place general-purpose processing engines into the data path of routers. Such network processors extend the traditional store-and-forward paradigm to store-process-and-forward, which opens vast possibilities for applications from simple QoS forwarding to complex payload transcoding for wireless clients. Our research addresses the system issues of network processors and points out the basic challenges associated with processing packets inside the network.

We have developed a network processor benchmark, CommBench, which can be used to characterize processing requirements and system behavior in the networking domain [3]. Results from the benchmark analysis show that typical processing kernels are small and achieve good performance using simple RISC cores with small caches of a few kilobytes. However, the computational complexity of applications that not only process the header of a packet, but also the payload (e.g., encryption), require in the order of 100 RISC instruction per byte of packet data. For high link speeds, these processing requirements cannot be managed by a single processor. Furthermore, this gap will grow with time as applications become more complex and link speeds increase faster than processor and memory speeds. However, the parallelism found in the network environment between flows, between packets within a flow, and on the application instruction level lends itself to a highly parallel system with many simple processing engines.

The system issues of such network multiprocessors are quite unique. While packet processing tasks are very short-lived, there is a large number of them that have to be maintained by the operating system. Furthermore, it has to be assured that each packet gets associated with the correct program code and has access to flow state that was possibly created by previous packets of the flow. We have proposed a router design that addresses these issues in [5]. The system stores packets in per-flow queues and a scheduler assigns them to processors. The scheduling of packets for processing is a challenging topic because it differs significantly from bandwidth scheduling. The processing time for a packet cannot be known in advance, which limits the ability to give hard guarantees on fairness and delay bounds. Our measurements have shown, though, that processing times are highly regular and typically linearly dependent on the packet size. Thus, the scheduler can use a processing time prediction to determine a good schedule. After the processing, a feed-back mechanism accounts for the actual processing time to ensure overall fairness [1]. We have also shown that exploiting instruction cache locality in the scheduler can improve the overall system performance [4].

Network processors are designed to be implemented on a single chip. Such a system-on-a-chip contains processors, memory caches, and memory and I/O interfaces. It is important to know the number of processors and the cache sizes that result in optimal performance for a given workload. A large number of processors provides much processing power, but limits the cache size, which leads to many cache misses and slow off-chip memory accesses. We have proposed an analytic performance model of such a system that can be used to configure a network processor optimally [2].

Our work addresses some key issues in network processor design and enhances the understanding of this environment. Future work will extend the scheduling to include context switching, an investigation on the benefit of hardware accelerators, and a system simulation. These results will help making it feasible to implement most router functions in software, which is key in providing the necessary flexibility for future networks.

REFERENCES

[1] P. Pappu and T. Wolf. Scheduling processing resources in programmable routers. Technical Report WUCS-01-32, Department of Computer Science, Washington University in St. Louis, July 2001.

[2] T. Wolf and M. A. Franklin. Design tradeoffs for embedded network processors. unpublished, 2001.

[3] T. Wolf and M. A. Franklin. CommBench - a telecommunications benchmark for network processors. In *Proc. of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 154–162, Austin, TX, Apr. 2000.

[4] T. Wolf and M. A. Franklin. Locality-aware predictive scheduling for network processors. In *Proc. of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Tucson, AZ, Nov. 2001.

[5] T. Wolf and J. S. Turner. Design issues for high performance active routers. *IEEE Journal on Selected Areas of Communication*, 19(3):404–409, Mar. 2001.