

Network Processors

Flexibility and Performance for Next-Generation Networks

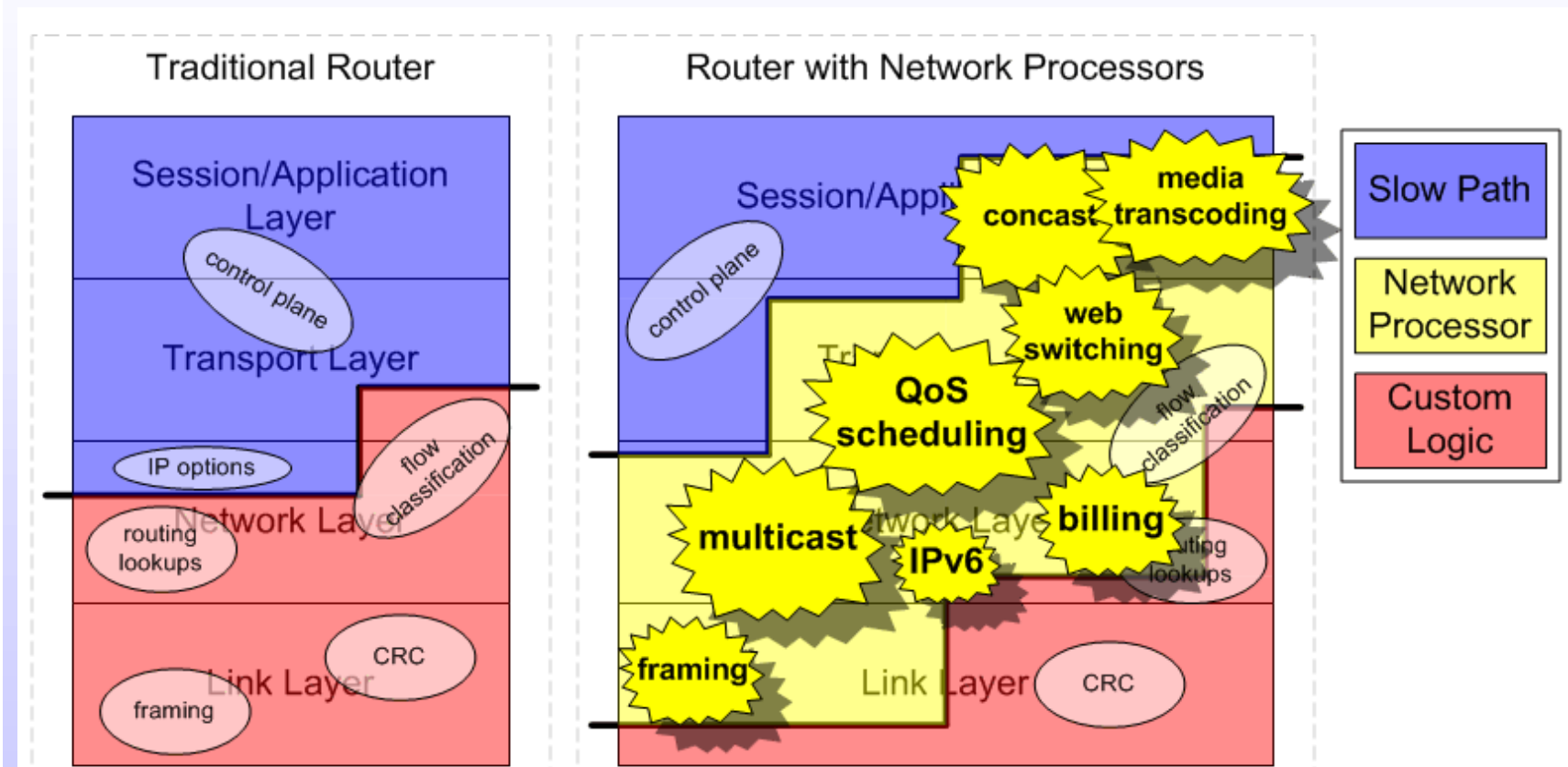
Tilman Wolf

Department of Computer Science, Washington University, St. Louis, MO

wolf@ccrc.wustl.edu

Motivation

- * New protocols and new services in network:



- * Processor in data path:

- ◆ New applications (header and payload processing)
- ◆ Rapid deployment (software changes not hardware)

Problem

"Design a network processor for Gigabit link rates."

- * Challenges:

- ◆ Performance (inherent lower performance of software)
- ◆ System complexity (maintain code and processing state)
- ◆ Technology limits (size of chip)
- ◆ Scheduling (many short tasks)

- * Why is this interesting / novel?

- ◆ Driven by I/O-centric applications
- ◆ Workload, OS, data-flow different from workstations
- ◆ Enables new applications inside the network

Contributions

- * Network Processor Benchmark

- ◆ Workload characterization
- ◆ Estimation of processing requirements

- * Multiprocessor System Design

- ◆ Efficient queuing system
- ◆ Scalable processor design

- * Scheduling Algorithm

- ◆ Fair, efficient scheduling of network processor

- * Optimization of System-On-A-Chip

- ◆ Optimal processor configuration for given workload

- Better understanding of implications from packet processing inside the network

Workload Characterization

- * Processing requirements for processing in software:

- ◆ RISC instructions executed per byte of packet data:

HEADER	64 byte packet	HEADER	Encoding	Decoding
Routing lookup	2.1	Encryption	104	104
IP fragmentation	7.7	Compression	226	35
DRR scheduling	4.1	FEC	603	1052
TCP monitoring	10.3	JPEG coding	81	60

- ◆ Routing at 1.2 Gb/s: $1.2/8 \cdot 2.1 = 315$ MIPS

- ◆ Encryption at 1.2 Gb/s: $1.2/8 \cdot 104 = 15,600$ MIPS

- * Payload processing computationally expensive!

- ◆ Exploit parallelism with multiprocessor

- * Also measured:

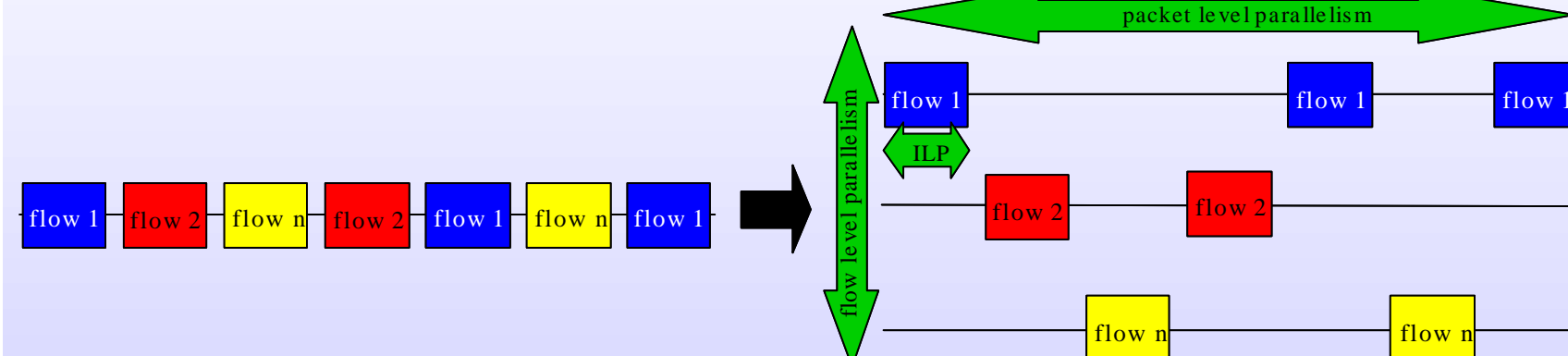
- ◆ Code size, cache performance, instruction mix [1]

Parallelism in Networks

- * Parallelism "everywhere":

- ◆ Flow-level, packet-level, instruction-level

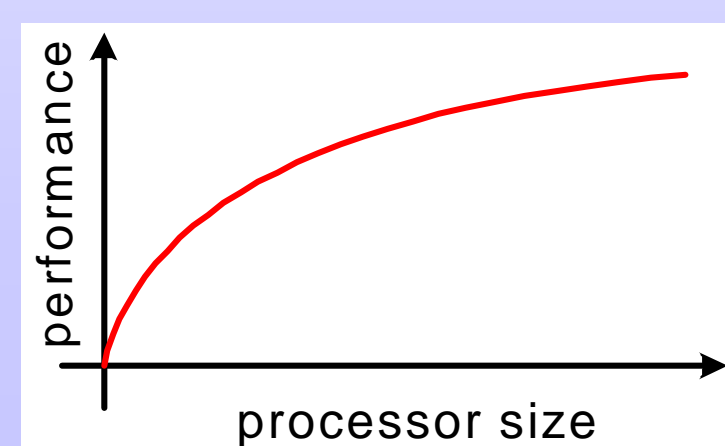
- * Packet stream:



- * Significantly more parallelism than traditional workloads!

- * Diminishing returns on complex processors

- many small processors



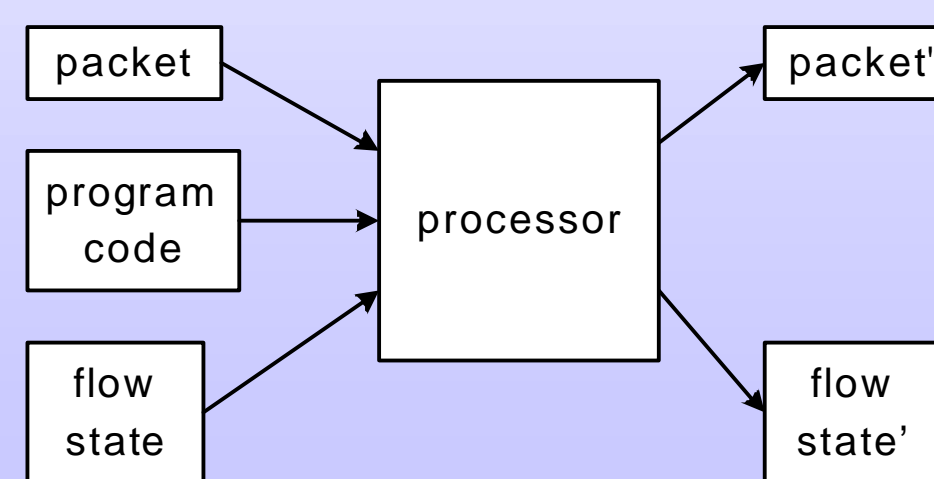
System Issues

- * System characteristics [2]:

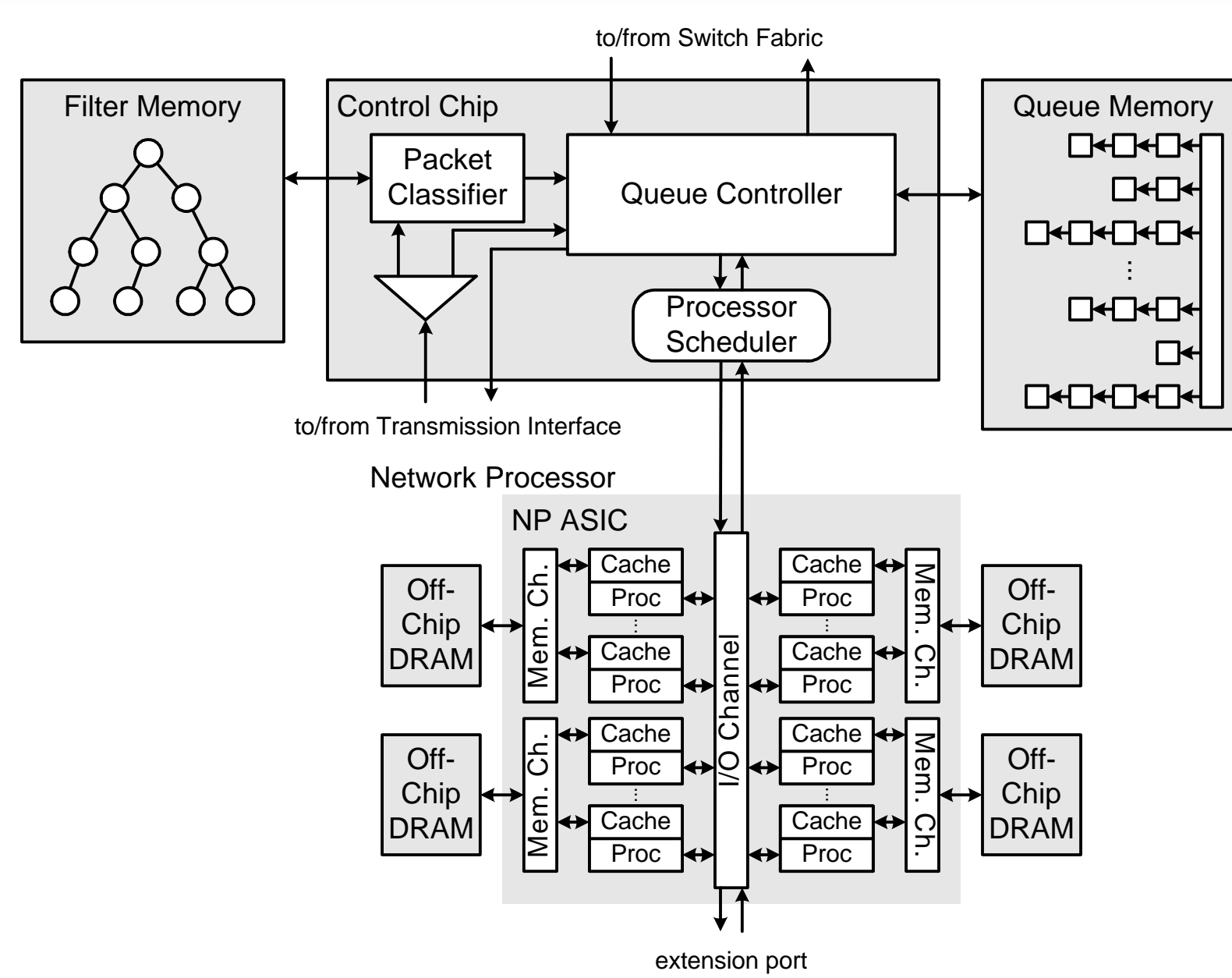
- ◆ Very short tasks (1-100 i sec)
- ◆ Large number of tasks (> 1,000,000 packets/sec)
- ◆ Highly heterogeneous applications (forwarding, MPEG coding)

- * Queuing system / OS must maintain:

- ◆ Packets
- ◆ Flow state
- ◆ Instruction code



Router Port Design



- * System-On-A-Chip: processors, cache, I/O on ASIC

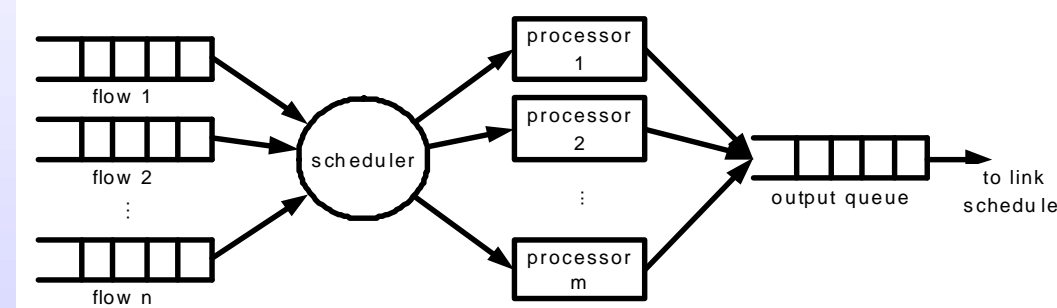
Scheduling

- * Which packet to process on which processor?

- ◆ Processing time unknown!

- ◆ Enforce fairness / reservation

- ◆ Preserve packet dependencies



- * Scheduling strategy:

- ◆ Use processing time prediction [3]

- ◆ Exploit instruction-cache locality [4]

- ◆ Feedback actual processing time

- * Fast context-switching: context prefetch in background

- * Load balancing: just move flow-state

System-On-A-Chip Optimization

- * On ASIC: How many processors? How much memory?

- * Analytic model for optimal configuration, given workload [5]

- * Too many processors on chip:

- Smaller caches due to chip are limit
- Smaller caches increase cache misses
- Higher cache misses cause more memory stalls
- Lower performance

optimal configuration for 400mm² ASIC:

workload	procs	instr. cache (per proc)	data cache (per proc)
HEADER	60	8kB	8kB
PAYLOAD	60	4kB	8kB

workload	memory channels	I/O pins	MIPS
HEADER	3	365	19,700
PAYLOAD	6	389	12,600

Status and Future Work

- * Current status:

- ◆ Completed NP benchmark analysis
- ◆ Completed system design
- ◆ Analyzed locality-aware and predictive scheduling
- ◆ Completed system-on-a-chip optimization
- ◆ Applications (sensor aggregation, web content transcoding)

- * Future work:

- ◆ Analysis of scheduling with context switching (with and without context prefetching)
- ◆ Investigation of hardware accelerators and programmable logic
- ◆ System simulator

Related Work

- * Active Networks / Programmable Networks:

- ◆ Instruction code distribution
- ◆ Safe program execution
- ◆ Resource sharing

- * Computer architecture

- ◆ Processor components (RISC cores, accelerators, ...)
- ◆ Faster memory, I/O

- * Commercial Network Processors:

- ◆ IBM PowerNP, Intel IXP1200, Motorola C-5, ...
- ◆ Few parallel processors, some hardware accelerators
- ◆ Mainly for header processing

Further Reading

- [1] Wolf, T., Franklin, M.: "CommBench - A Telecommunication Benchmark for Network Processors," *Proceedings of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 154-162, Austin, TX, April 2000.
- [2] Wolf, T., Turner, J.: "Design Issues for High Performance Active Routers," *IEEE Journal on Selected Areas of Communications - Special Issue on Active and Programmable Networks*, vol. 19, no. 3, pp. 404-409, March 2001.
- [3] Pappu, P., Wolf, T.: "Scheduling Processing Resources in Programmable Routers," submitted to *IEEE INFOCOM 2002*.
- [4] Wolf, T., Franklin, M.: "Locality-aware Predictive Scheduling of Network Processors," to appear in *Proceedings of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Tucson, AZ, November 2001.
- [5] Wolf, T., Franklin, M., Spitznagel, E.: "Design Tradeoffs for Embedded Network Processors," *Tech. Report WUCS-00-24*, Dept. of Computer Science, Washington University in St. Louis, MO, July 2000.

Network Processors - Flexibility and Performance for Next-Generation Networks

Tilman Wolf

Department on Computer Science
Washington University, St. Louis, MO
wolf@ccrc.wustl.edu

There is significant effort in networking research to develop new protocols and services. However, the ability to deploy these in the current Internet is greatly inhibited by the 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. The research presented in this poster addresses the system issues of network processors and points out the basic challenges associated with processing packets inside the network.

Results from a benchmark analysis show that the complexity of applications that not only process the header of a packet, but also make modifications to the payload (e.g., encryption) require in the order of 100 RISC instruction per byte of packet data [1]. 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. The tasks of processing packets are very short and there is a very large number of them. The queuing system in the router has to assure that each packet gets associated with the correct program code and has access to flow state that was created by previous packets of the flow. We have proposed a router design that addresses these issues in [2]. The system stores packets in per-flow queues and a scheduler assigns them to processors.

The scheduling of packets for processing is interesting because -unlike bandwidth scheduling- the processing time cannot be known in advance. Thus, the scheduler needs to use a processing time prediction and account for the actual time used after processing has completed [3]. We have shown that processing times are often predictable and that when exploiting instruction cache locality, higher throughput can be achieved [4].

The network processor is 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 [5].

This 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.

Further Reading:

- [1] Wolf, T., Franklin, M.: "CommBench - A Telecommunication Benchmark for Network Processors," Proc. of IEEE ISPASS 2000, pp. 154-162, Austin, TX, April 2000.
- [2] Wolf, T., Turner, J.: "Design Issues for High Performance Active Routers," IEEE JSAC, vol. 19, no. 3, pp. 404-409, March 2001.
- [3] Pappu, P., Wolf, T.: "Scheduling Processing Resources in Programmable Routers," submitted to IEEE INFOCOM 2002.
- [4] Wolf, T., Franklin, M.: "Locality-aware Predictive Scheduling of Network Processors," to appear in Proc. of IEEE ISPASS, Tucson, AZ, November 2001.
- [5] Wolf, T., Franklin, M., Spitznagel, W.: "Design Tradeoffs for Embedded Network Processors," Tech. Report WUCS-00-24, Department of Computer Science, Washington Univ., St. Louis, MO, July 2000.