

# Network Processors for Advanced Services

Jon Turner

Computer Science and Engineering

Jon.Turner@wustl.edu

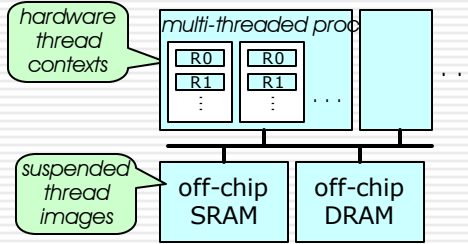
 Washington University in St. Louis

## Motivation

- NPs have great potential for advanced services.
  - » many independent flows  $\Rightarrow$  embarrassingly parallel workload
  - » ideal for scaling characteristics of chip multiprocessors
- But, not there yet.
  - » not well-suited to running many separate programs
  - » requires tight coupling between hardware and software
  - » no isolation of programs
- Extending NP architectures.
  - » thread scheduling for workloads with thousands of threads
    - thread scheduling more flexible than packet flow scheduling
    - fine-grained real-time support (10-100  $\mu$ s time slices, 1-10  $\mu$ s context switch overhead)
    - software scheduler (for flexibility) with hardware acceleration
  - » more complete memory architecture
    - instruction cache
    - low performance overhead memory protection

# Thread Scheduling

- Thread per flow paradigm.
  - » # threads >> # hw contexts
  - » hw contexts switched to hide memory latency
  - » threads can also block on IO or time-slice expiration
- Software scheduler.
  - » maintains list of ready threads (in on-chip SRAM)
    - list nodes point to thread images stored in off-chip SRAM
  - » inserts itself at end of list to trigger scheduling operations
- Hardware implements context switches.
  - » on short-term suspension (memory access)
    - assign next ready hw thread context to processor
  - » on long-term suspension (IO, time-slice expiration)
    - assign next ready hw thread context to processor
    - get next ready thread from scheduling lists and swap thread images



# Memory Protection

- Conventional processors use paged VM for protection.
  - » too much overhead for NPs
    - page tables too big to store on-chip
  - » segmented VM may be better alternative
    - no swapping, limited fragmentation opportunities
    - add 4-8 segment registers per hw thread context
- Pooled segment registers offer more flexible approach.
  - » pool for each processor
  - » segment regs include
    - hw thread context #
    - range of VM addresses
    - base physical addr.
  - » addr mapping involves
    - parallel matching
    - offset computation

