Router Construction II

Outline

- Network Processors
- Adding Extensions
- Scheduling Cycles

Research Agenda

- Evaluate the use of commodity components to build IP routers.
- Design interfaces and mechanisms needed to support extensibility.

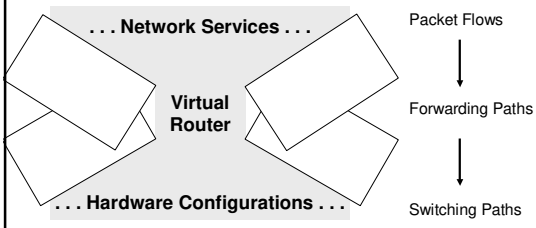
Technology Forces

- Commodity switching fabrics
  - e.g., ATM switches, Infiniband SAN
- Programmable network processors
  - e.g., Intel IXP1200, Sitera PRISM
- Commodity processors
  - e.g., Intel IA32, IA64

Assumption Regions

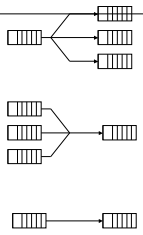
- Original Model
  - Internet as a flat collection of networks
- Current Reality
  - Internet partitioned into *assumption regions*
- Examples
  - trust, reliability, performance, routing, power…
- Discontinuity at assumption boundaries
  - run code on router to mask this discontinuity

Architectural Overview

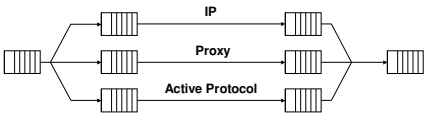


Virtual Router

- Classifiers
- Schedulers
- Forwarders



Simple Example



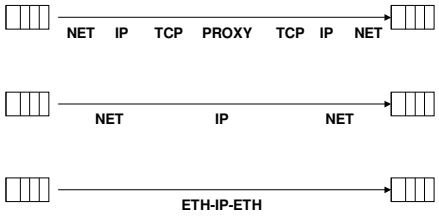
Research Problems

- How to program (the API)
  - forwarders
  - hierarchical classifier
- Map forwarding paths onto switching paths
  - local/static: implement VR on one processor
  - local/dynamic: scheduling
  - global/static: function placement
  - global/dynamic: resource allocation

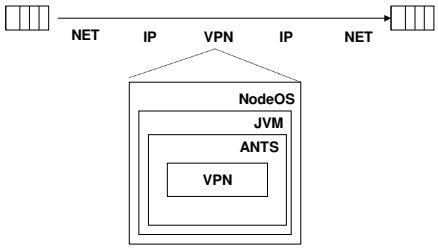
Programming Forwarders

- Component-Based
  - legacy + extension(s)
  - granularity
- What components are trusted
- Structure of extensions
  - control versus data plane components

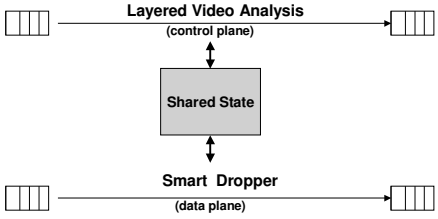
Scout Paths



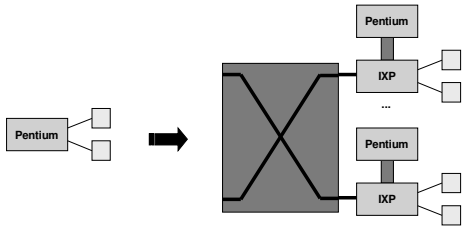
Untrusted Modules



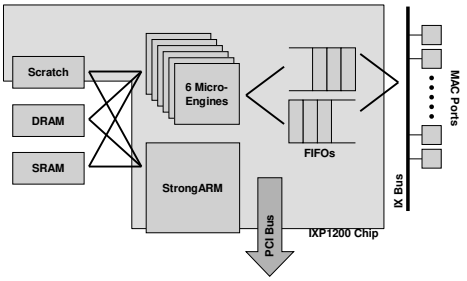
Control and Data Plane



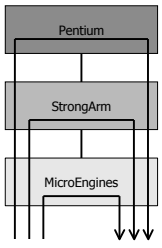
Prototype Hardware



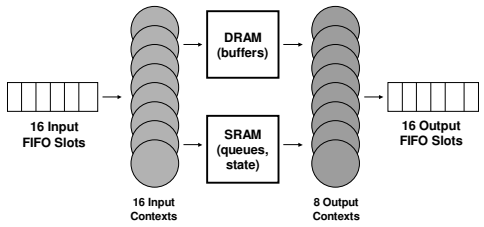
Intel IXP



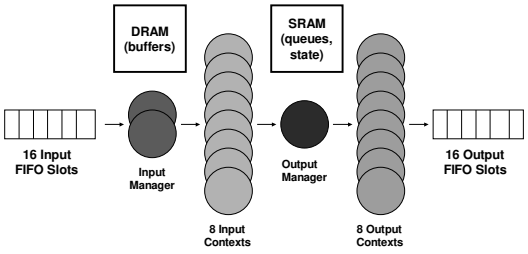
Processor Hierarchy



μE: Design I



μE: Design II



μE: Evaluation

- Desgin I
  - max rate: 3.02Mpps
  - memory bound
  - problem: lock contention
- Design II
  - max rate: 1.48Mpps (1 output mgr cxt)
  - exp rate: 2.96Mpps (2 output mgr cxts)
  - cpu bound

Idea

16 contexts
input loop:
- until_rcv
- move FIFO-to-DRAM
- lookup route
- enqueue

8 contexts
output loop:
- dequeue
- move DRAM-to-FIFO
- init_xmit

Idea

16 contexts
input loop:
- until_rcv
- move FIFO-to-DRAM
- lookup route
- no-op
- no-op
- …
- no-op
- enqueue

8 contexts
output loop:
- dequeue
- move DRAM-to-FIFO
- init_xmit

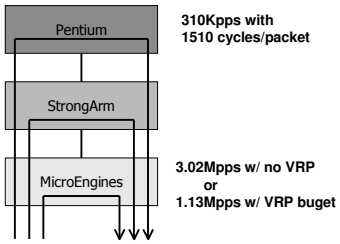
Virtual Router Processor

- Fixed Infrastructure (black code)
  - forward 1.13Mpps for 8 x 100Mbps ports
- Programmable VRP (red code)
  - 4 x 32-byte DRAM transfers (r/w packet payload)
  - 7 x 32-byte SRAM transfers (flow state)
  - 450 register operations
  - per 64-byte chunk
  - total of 650 instructions per MicroEngine

StrongArm

- Handle exceptional packets
- Except…
  - shares DRAM capacity with MicroEngines
  - on the critical path to Pentium
- Solutions
  - limit the role the SA can play
  - migrate code from Pentium to MicroEngines

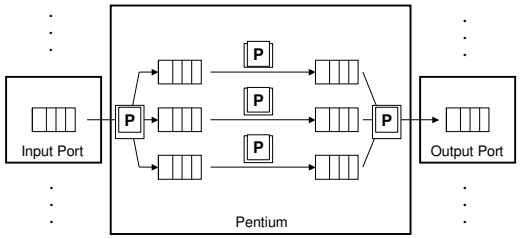
Performance



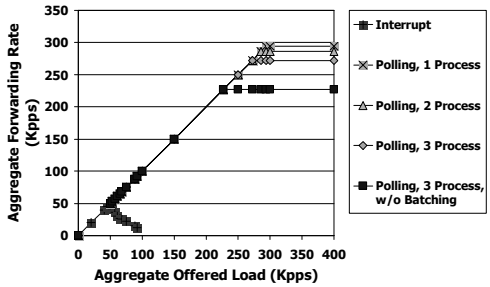
Pentium

- Runs protocols in the control plane
  - e.g., BGP, OSPF, RSVP
- Run other router extensions
  - e.g., proxies, active protocols, overlays
- Implementation
  - runs Scout OS + Linux IXP driver
  - CPU scheduler is key

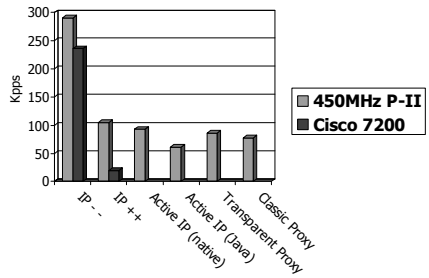
Processes



Performance



Performance (cont)



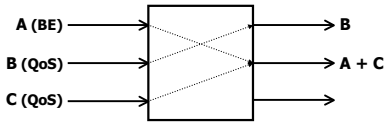
Scheduling Mechanism

- Proportional share forms the base
  - each process reserves a cycle rate
  - provides isolation between processes
  - unused capacity fairly distributed
- Eligibility
  - a process receives its share only when its source queue is not empty and sink queue is not full
- Batching
  - to minimize context switch overhead

Share Assignment

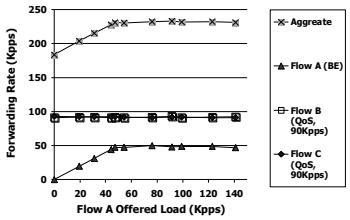
- QoS Flows
  - assume link rate is given, derive cycle rate
  - conservative rate to input process
  - keep batching level low
- Best Effort Flows
  - may be influenced by admin policy
  - use shares to balance system (avoid livelock)
  - keep batching level high

Experiment



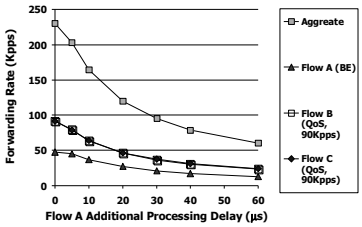
Mixing Best Effort and QoS

- Increase offered load from A

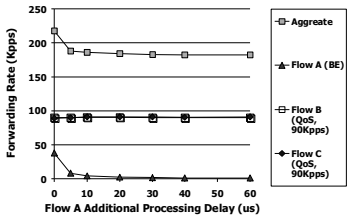


CPU vs Link

- Fix **A** at 50Kpps, increase its processing cost

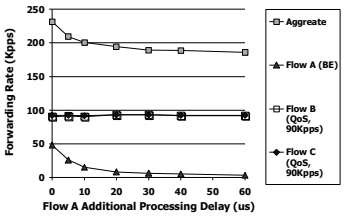


Turn Batching Off



- CPU efficiency: 66.2%

Enforce Time Slice

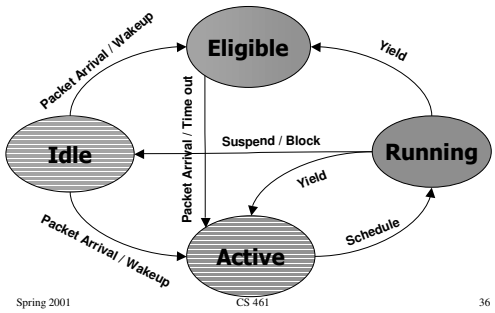


- CPU efficiency: 81.6% (30us quantum)

Batching Throttle

- Scheduler Granularity: G
  - flow processes as many packets as possible w/in G
- Efficiency Index: E, Overhead Threshold: T
  - keep the overhead under T%, then 1 / (1+T) < E
- Batch Threshold: B_i
  - don't consider Flow *i* active until it has accumulated at least B_i packets, where C_{sw} / (B_i x C_i) < T
- Delay Threshold: D_i
  - consider a flow that has waited D_i active

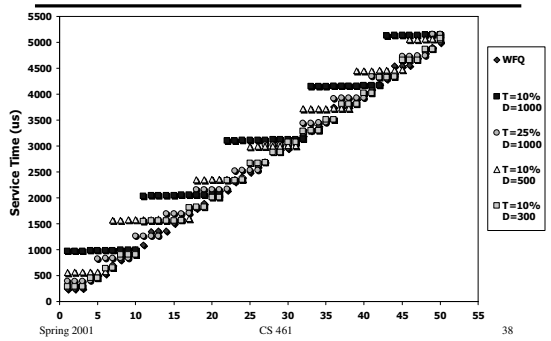
State Machine



Dynamic Control

- Flow specifies delay requirement D
- Measure context switch overhead offline
- Record average flow runtime
- Set E based on workload
- Calculate batch-level B for flow

Packet Trace



Acknowledgements

- Graduate Students
  - Andy Bavier
  - Yitzchak Gottlieb
  - Scott Karlin
  - Tammo Spalink
  - Xiaohu Qie
- More Information
  - www.cs.princeton.edu/nsg/

Simulation

- Setup
  - CPU: 1MHz
  - Context-Switch: 3μs
  - 16 QoS Flows: 10Kpps, 3μs/pkt
  - 1 BE Flow: 100Kpps, 3μs/pkt

  In order to forward all packets, E must be at least
  $$10\times10^{3}\times3\times10^{-6}\times16+100\times10^{3}\times3\times10^{-6}=78\%$$
- Parameters:
  - G=100μs, T=10% / 25%
  - QoS Flows: D_i =1ms / 0.5ms / 0.3ms