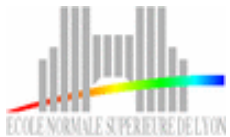
Packet Classification in the NIC for Improved SMP-based Internet Servers

Eric Lemoine[1,2], CongDuc Pham[2], and Laurent Lefèvre[2]

ICN'04

(1) Sun Microsystems Laboratories Europe (Grenoble, France)
(2) UMR CNRS - ENS Lyon - UCB Lyon - INRIA 5668



Context

- Shared-memory multiprocessor (SMP) based Internet servers
  - Multiprocessor machines are well suited to Internet server type applications
    - Internet server applications use multiple threads with independent activities

Problem Statement

- TCP processing must be distributed across CPUs
  - Load-balancing
  - Parallelism (10 Gigabit Ethernet is out there!)
    - Internet servers need simultaneous transfers
- How to distribute TCP processing across CPUs in an efficient and robust manner?

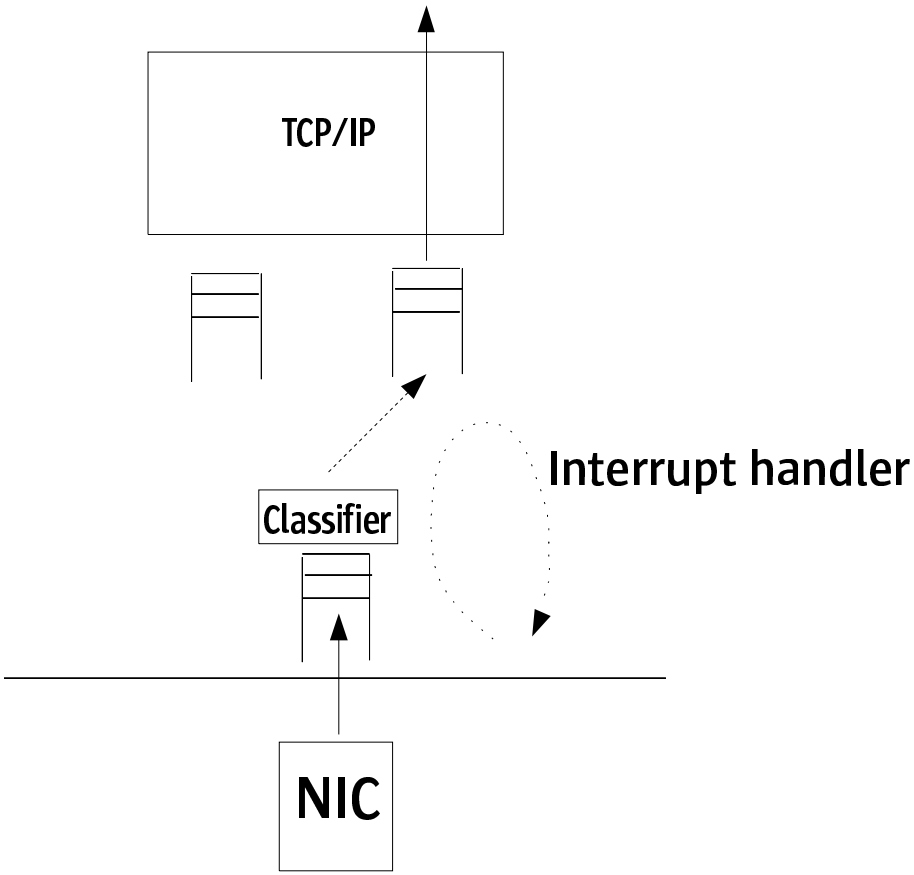
Parallel TCP processing

- If packets are randomly distributed across CPUs:
  - Cache misses on per-connection states
  - Lock contentions on per-connection states
  - Reordering

Connection-level Parallelism
[Nahum, OSDI'94]

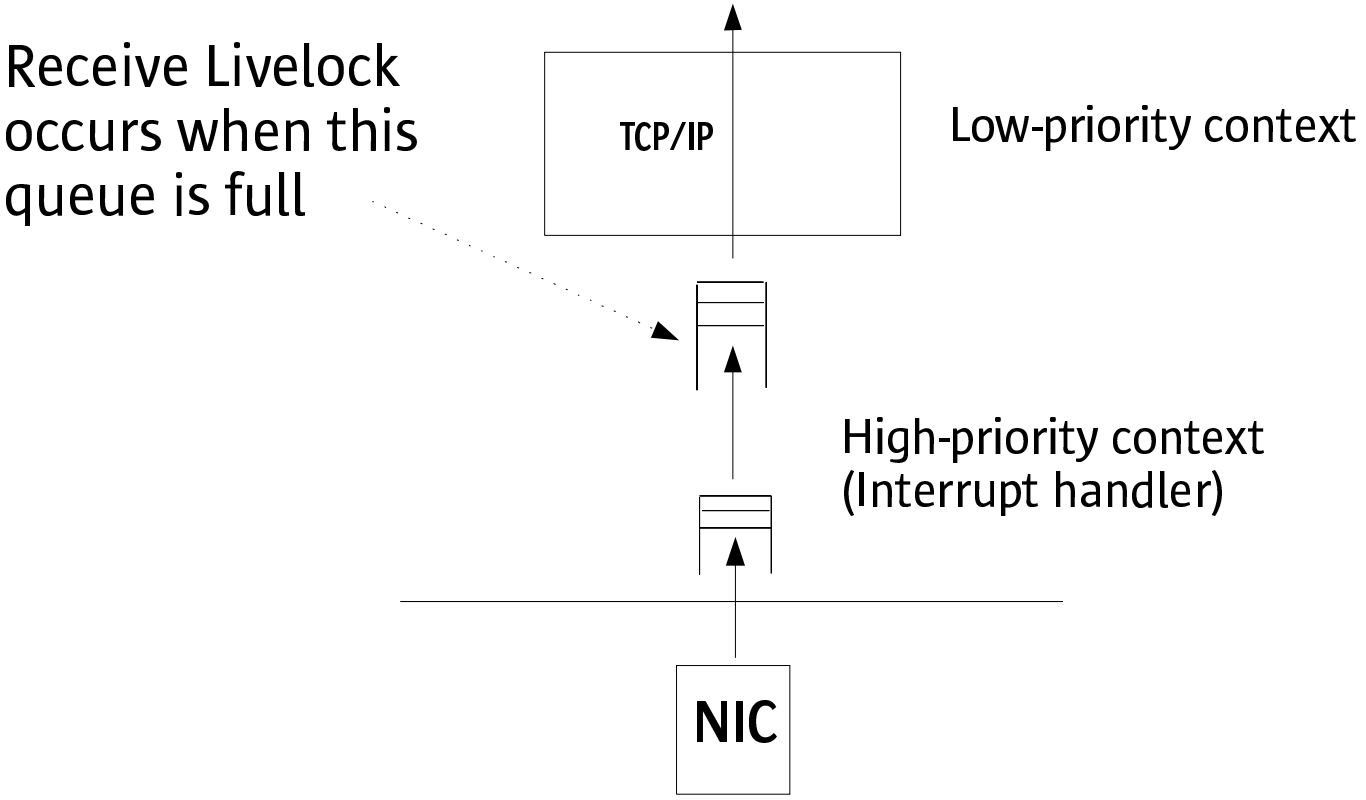
- The connection is the unit of concurrency
- Emulation results in [Nahum, OSDI'94] show good scalability to the number of CPUs
- Appropriate to Internet servers
- Internet servers need parallel transfers
  - Implies parallel receives due to TCP CA algorithm
- Inbound packets must be classified under TCP

In-kernel Classifier



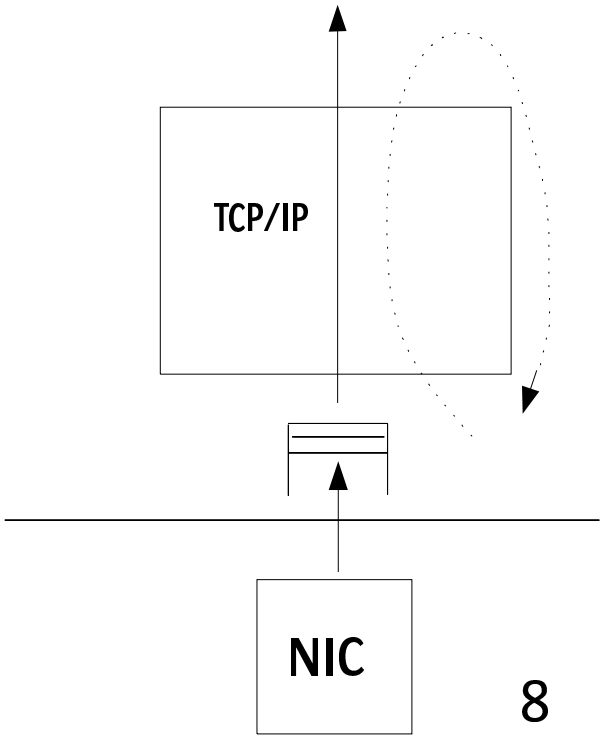
- Lock contentions
- Memory contentions
- Low packets/int ratio
- Prone to Receive Livelock

Receive Livelock



Solution to Receive Livelock
[Mogul, TOCS'97]

- Mixture of interrupt and polling modes
  - Receive interrupts are disabled as long as there are packets in the driver's receive queue

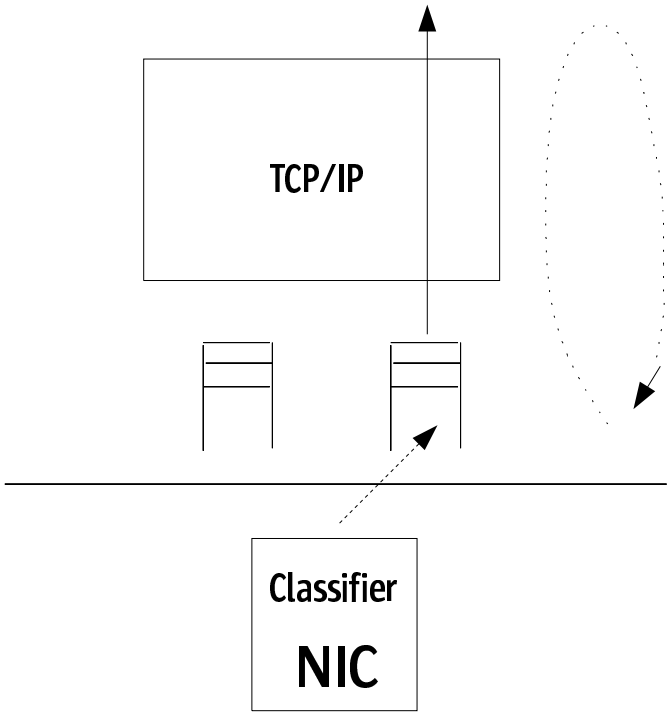


- Implemented in the Linux kernel (as of 2.4.20) [Salim, Usenix'01]

Proposed Architecture

- Classify incoming packets in the NIC
- Make use of per-CPU receive queues in the driver
- Use Mogul's solution to Receive Livelock
- Efficiency:
  - No movements of packets between CPUs
- Robustness:
  - Receive Livelock-free

Proposed Architecture



- No lock contentions on the recv queues
- No memory contentions on packet data structures
- High packets/interrupt ratio
- Receive Livelock-free

Prototyping Environment

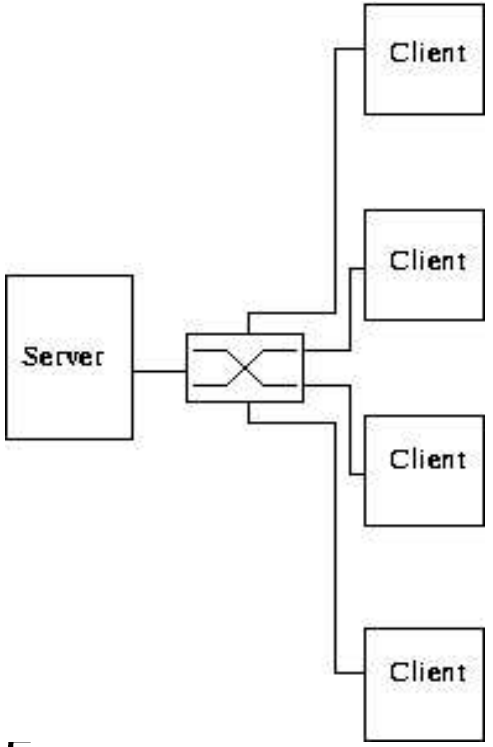
- Myricom's Myrinet
  - Programmable NICs
  - Full-duplex 2Gbps
- Linux kernel

KNET Prototype Implementation

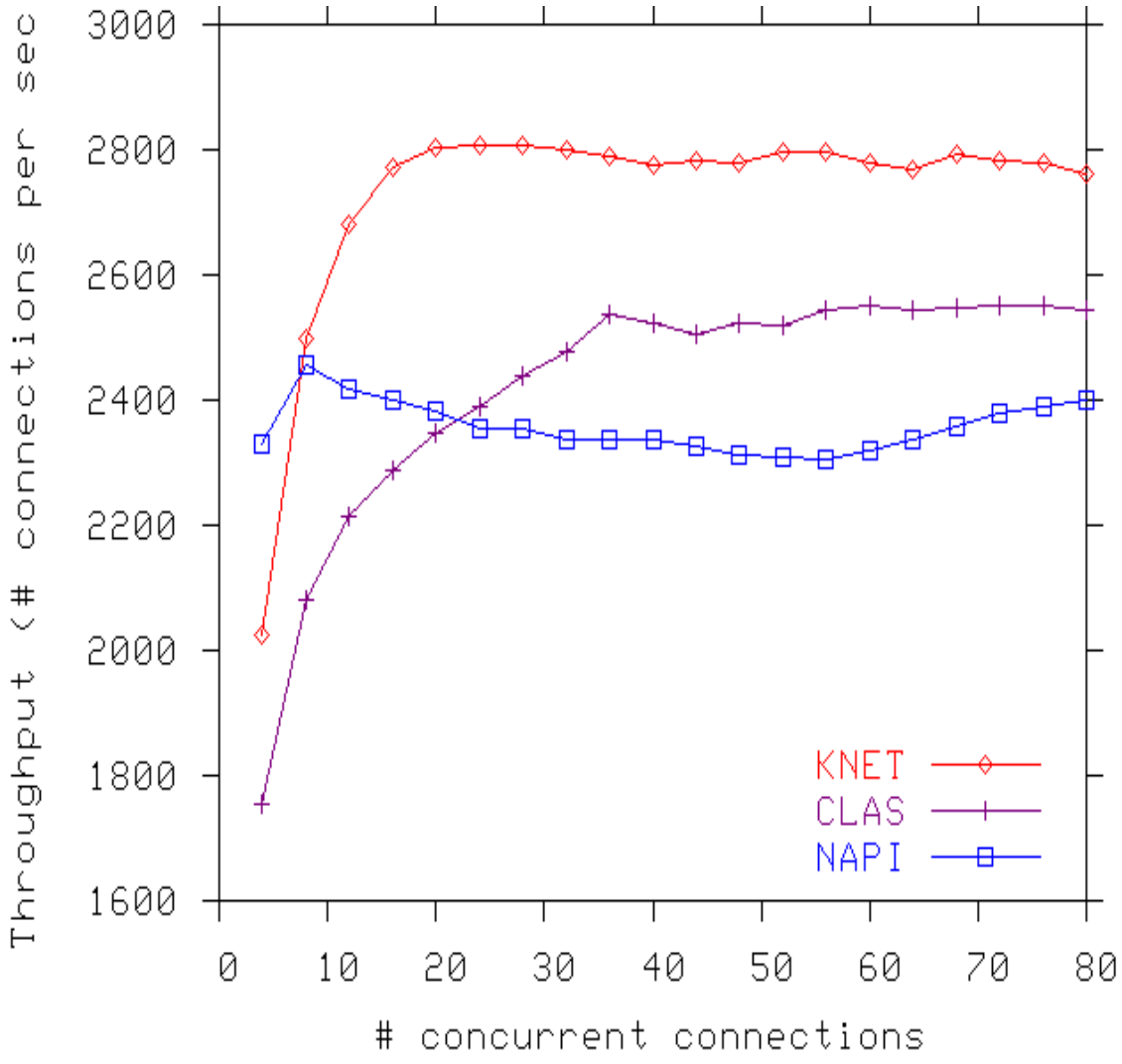
- Linux network subsystem
  - per-CPU kernel-threads dedicated to network processing
- Modifications to the NIC driver
  - per-CPU receive queues
- Modifications to the NIC firmware
  - packet classifier
  - per-CPU receive queues
  - per-queue interrupt activation/deactivation

Experimental platform

- Hardware
  - 4-processor PIII 500Mhz
  - Myrinet2000 (LANai9, 200Mhz)
- Software
  - Linux-2.4
  - Web servers: Webfs, Apache2
  - Traffic generators: Sclient, WebStone2.5



Webfs/Sclient (20K-file)



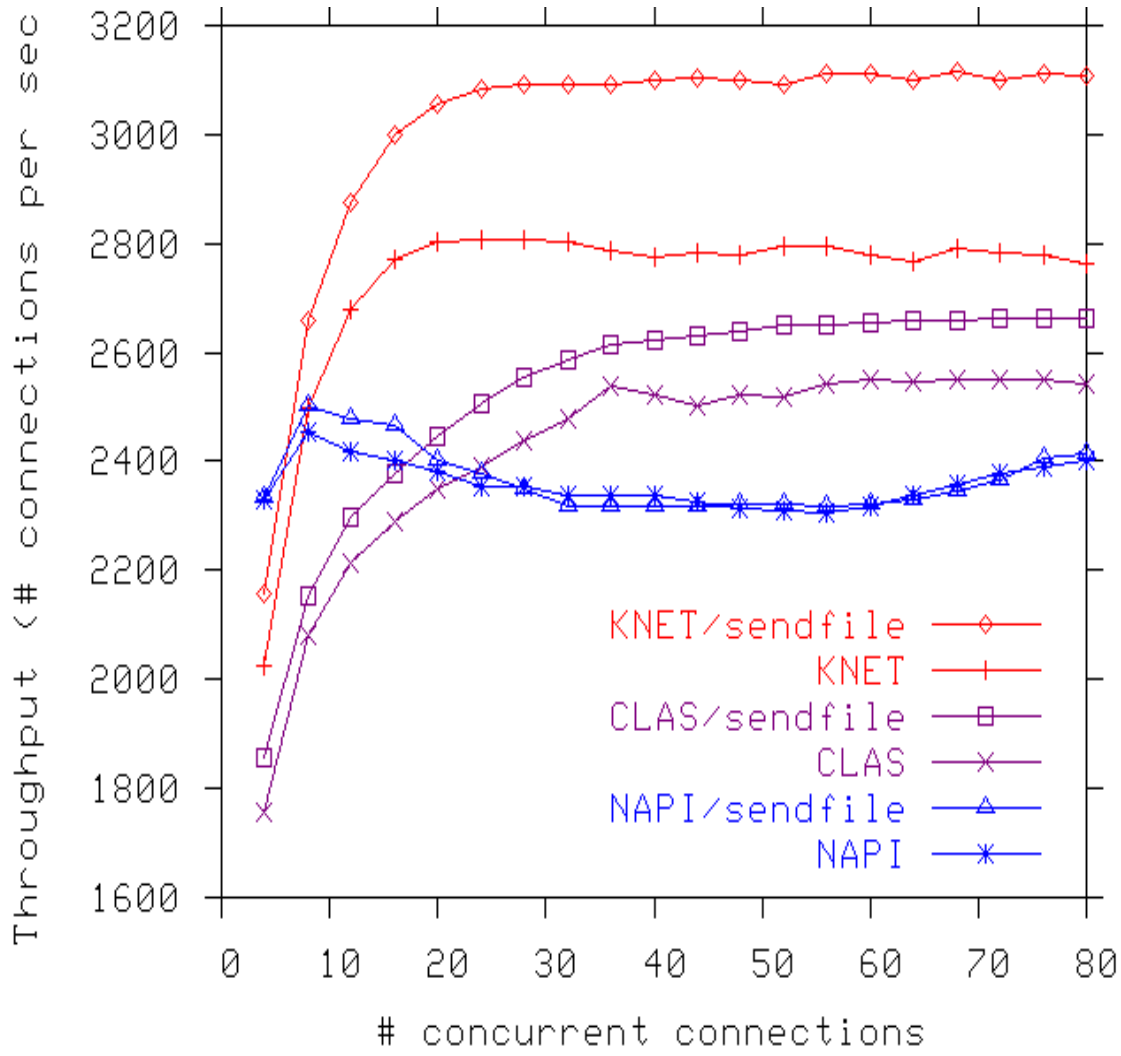
KNET: 12% gain over CLAS
17% gain over NAPI

KNET: 45 rx pkts/int
5% CPU in driver rx

CLAS (In-kernel classific):
5.6 rx pkts/int
12% CPU in driver rx
6% CPU in locks

NAPI (New Linux net sys):
40 rx pkts/int

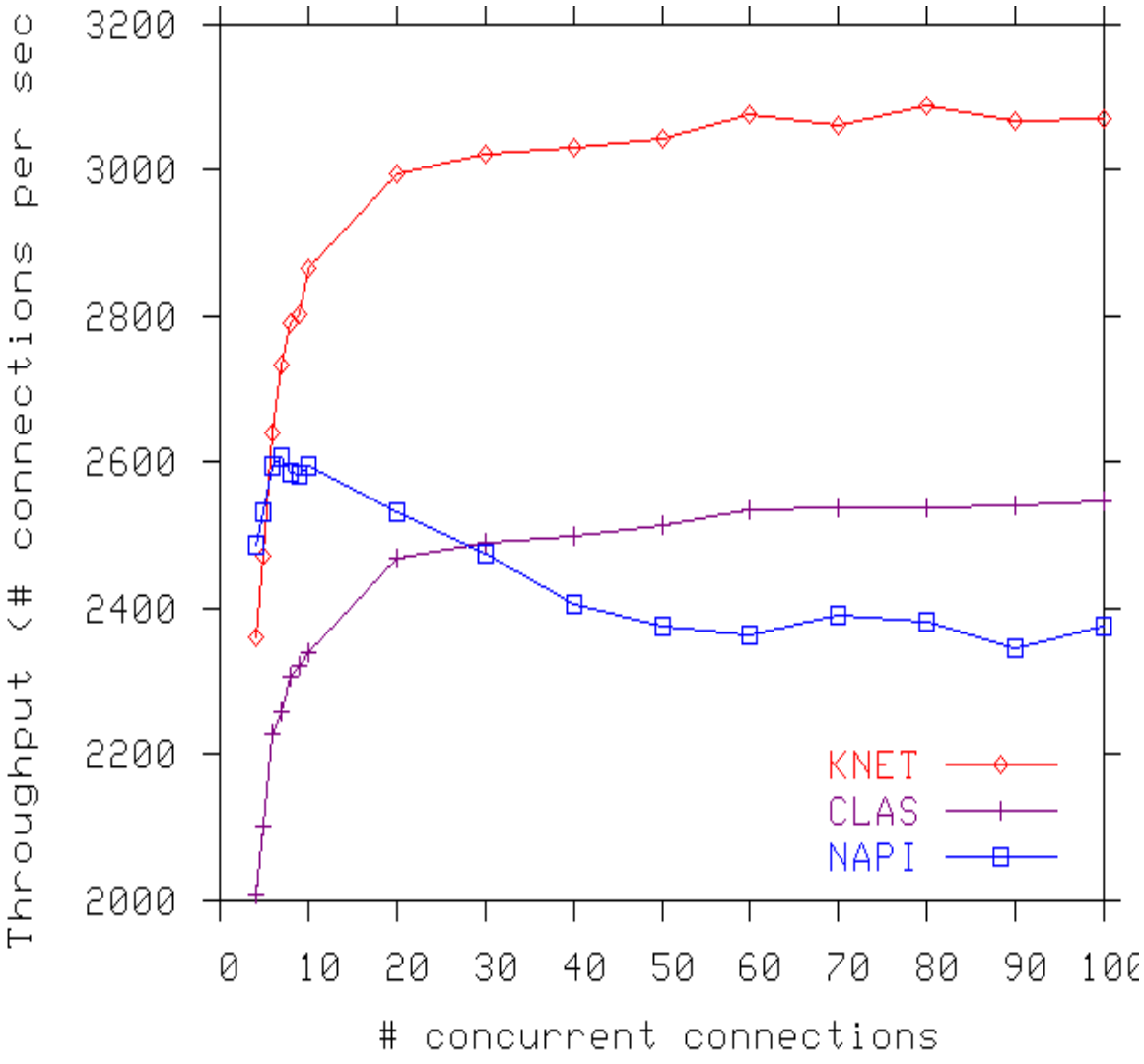
Webfs/Sclient (20K-file) using sendfile()



KNET: 17% gain over CLAS
30% gain over NAPI

KNET benefits more from sendfile

Apache2/WebStone



KNET: 20% gain over CLAS
27% gain over NAPI

Conclusion

- Packet classification in the NIC enables
  - Robustness (by design)
  - Efficiency
    - Our implementation leads to 20+% gains

Future Work

- MSI (Message Signalled Interrupt)
  - Device can interrupt any CPU
    - Eliminate scheduling latency
    - Eliminate lock contentions
- Chip Multi-Threading (CMT)
  - L2$ can be shared among threads

Thank you!