Experiments With Highly Parallel Network Stack Processing

Robert N. M. Watson

17 May 2007

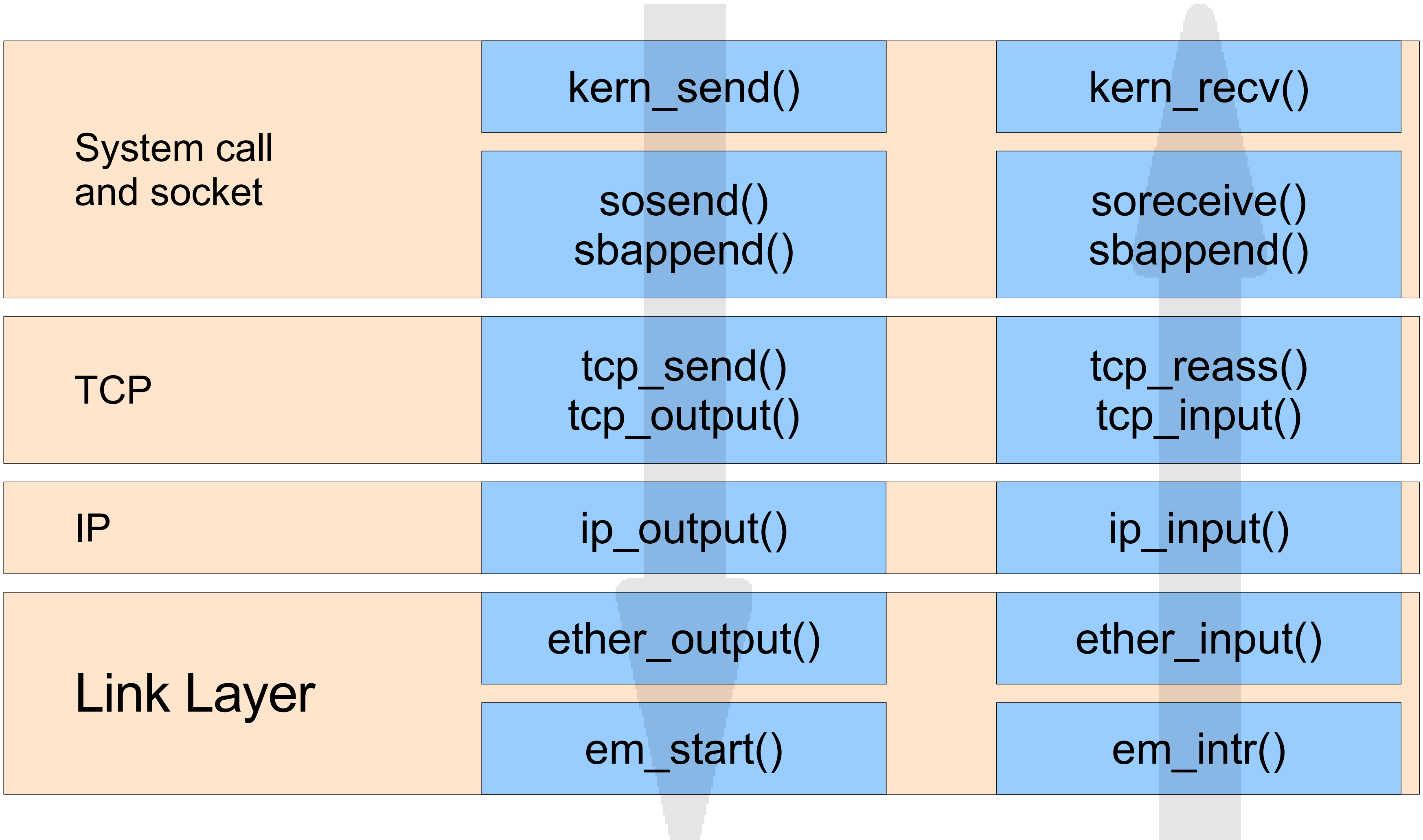
FreeBSD Developer Summit
BSDCan 2007

UNIVERSITY OF CAMBRIDGE

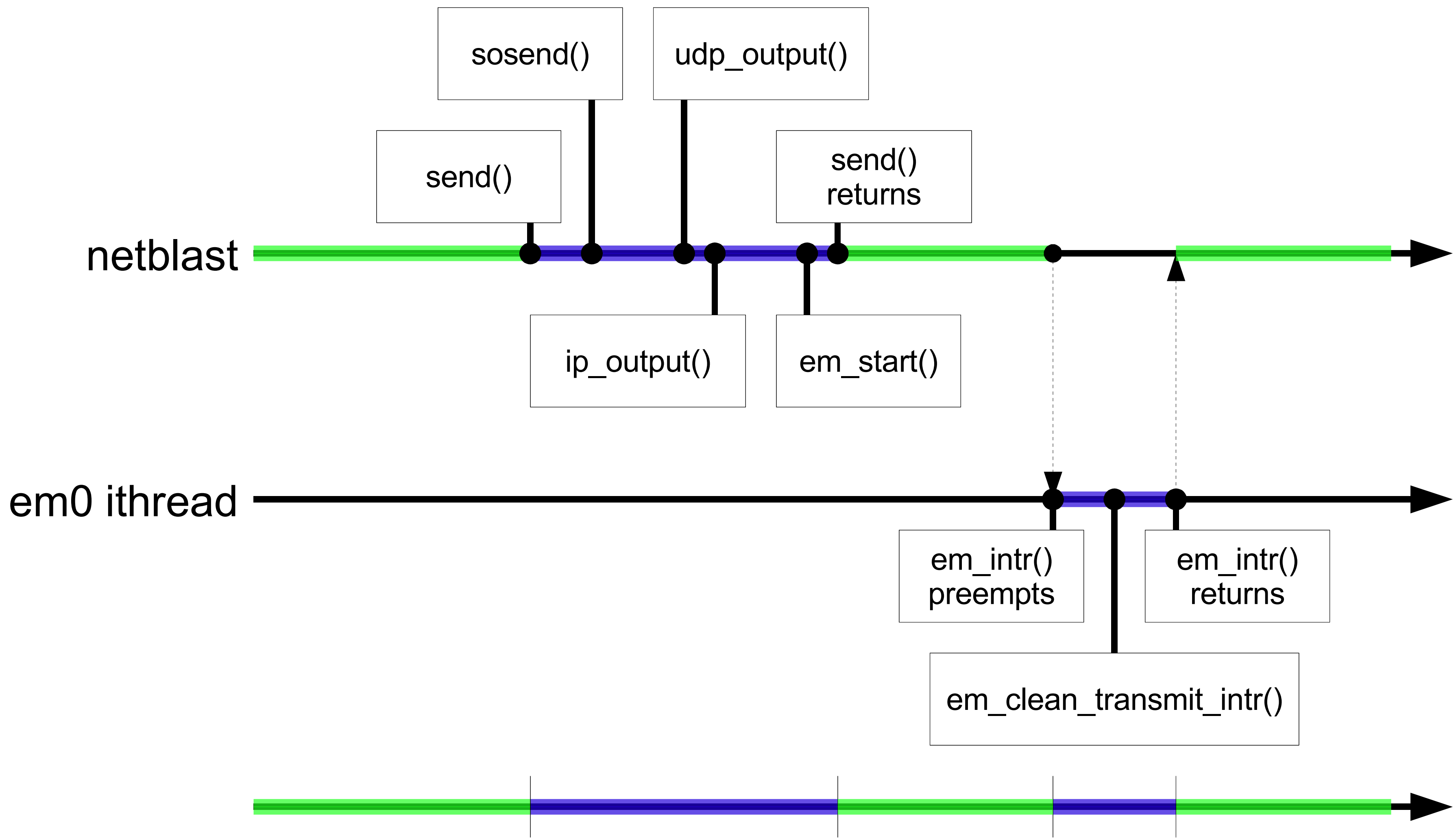
Introduction

- MPSAFE network stack in FreeBSD 5.3
  - November 2004
- Network stack safe to execute concurrently
  - Parallelism – many processors at a time
  - Preemption – low latency context switching
  - Direct dispatch – from interrupt thread context
- Opportunities are limited due to work model
  - Threads represent potential parallelism
  - So **work must occur in multiple threads**

Socket to Interface Code Flow



Transmit

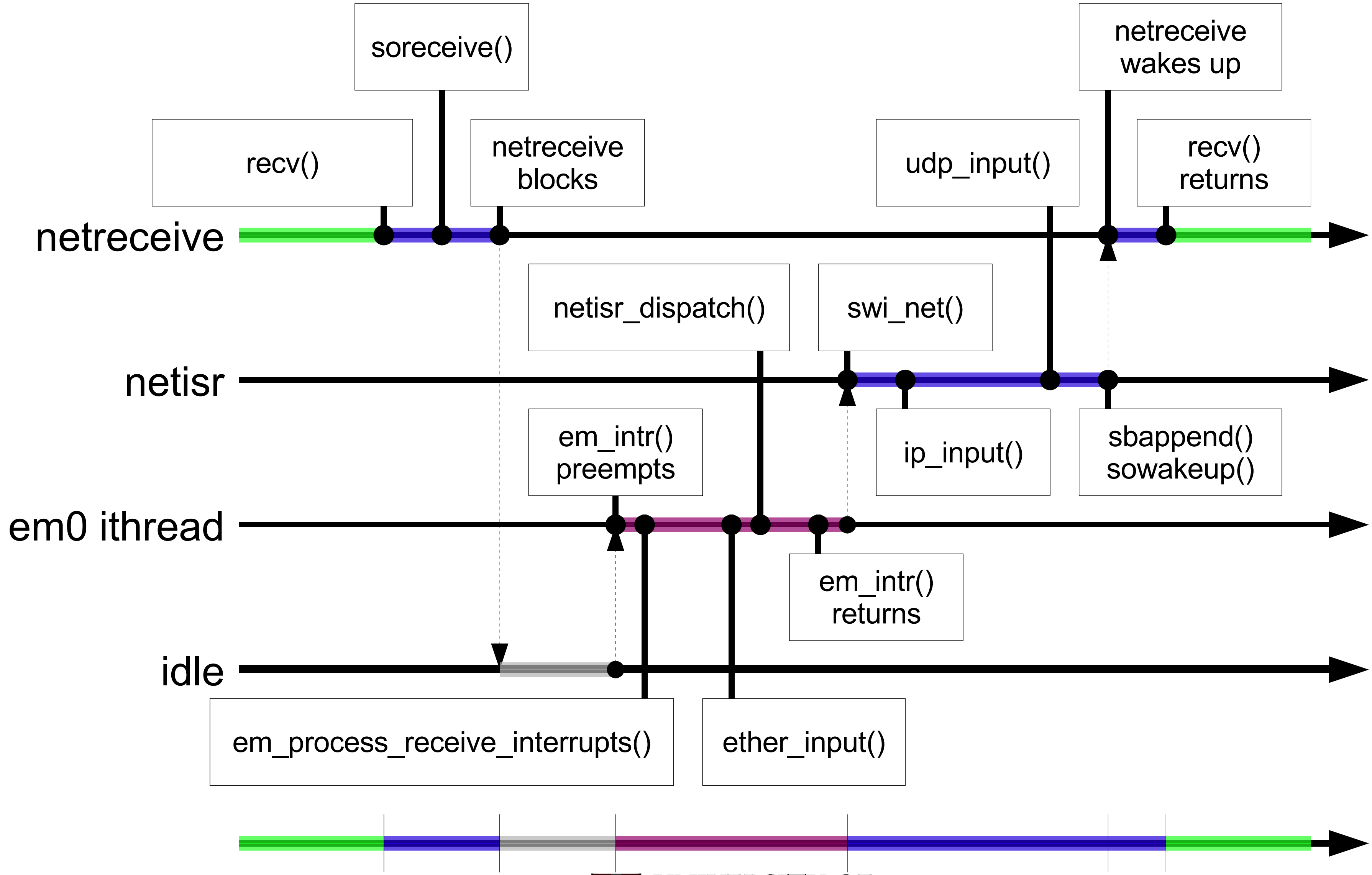


UNIVERSITY OF CAMBRIDGE

Input Path Parallelism

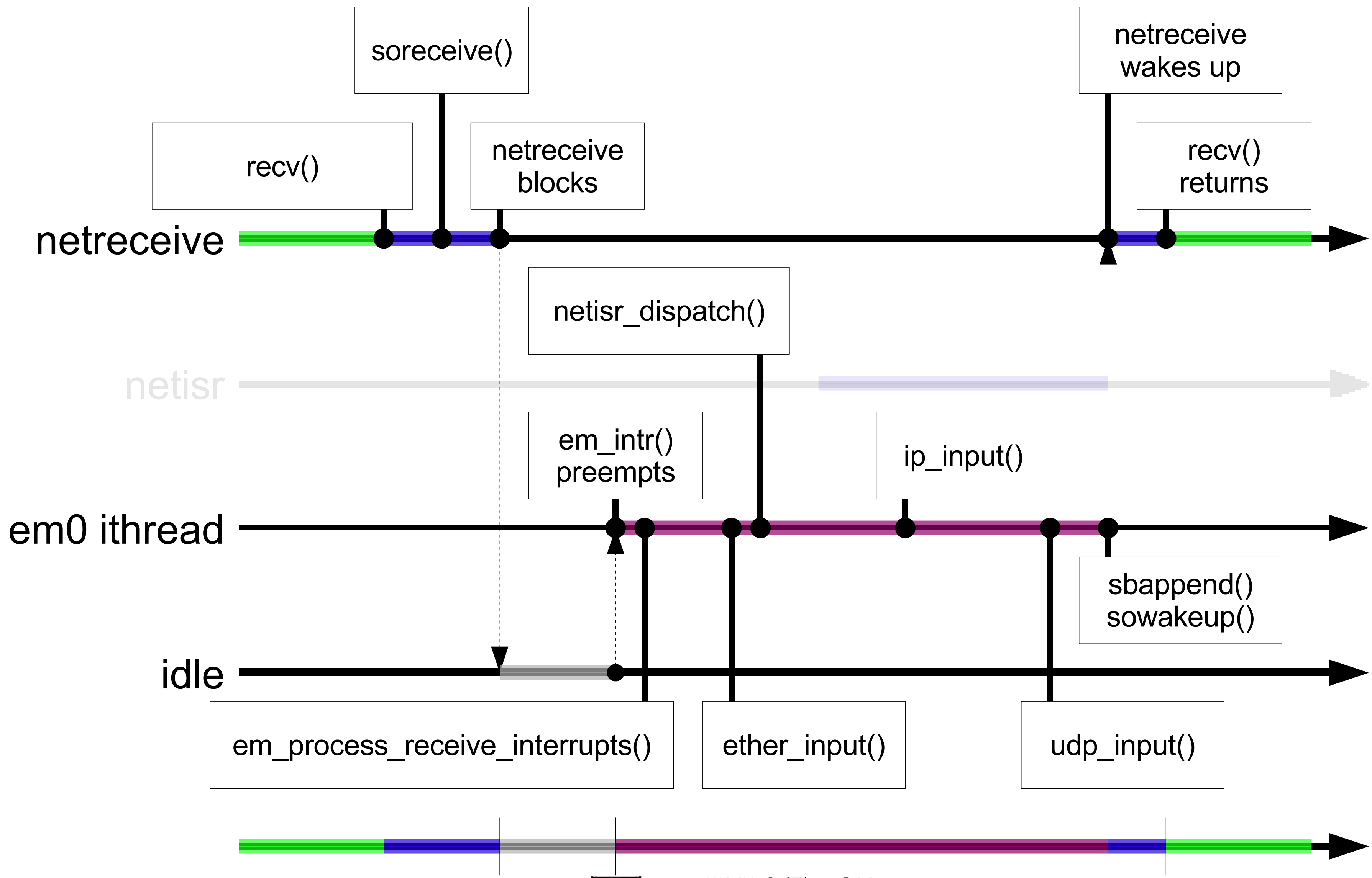
- Does have native concurrency
  - Work processed in ithread, optionally netisr, and user thread
- Cost of context switching measurable
  - Default in 7.x is direct dispatch as frequently faster
- Ordered with respect to source
  - Lack of parallelism for high bandwidth sources

Deferred netisr



UNIVERSITY OF CAMBRIDGE

Direct netisr



netisr2

- Function of netisr after direct dispatch
  - Encapsulation/decapsulation, loopback traffic
- Netisr2 re-implement netisr infrastructure
  - Per-cpu kernel worker threads
  - Maintain dispatch/queue distinction
- How to balance ordering and parallelism?
  - For direct dispatch, source ordering
  - For remote source input, per-protocol affinity lookup
  - For locally source, affinity passed down stack

TCP Input Parallelism

- Serious bottleneck for TCP input processing
- Two IP-layer locks per protocol
  - Global pcbinfo lock protecting inpcb lists
    - Insert, lookup, removal
  - Per-inpcb lock protects per-connection state
    - Global list manipulations involving connection
    - Per-connection state
- Input path acquires and holds tcbinfo lock
  - inpcb state may change requiring list changes

Addressing Locking Granularity

- Big issue is tcbinfo lock
  - Held over input paths that may reset connection
  - In 7.x, no longer over common case output paths
- Option 1: True reference counting on inpcbs
  - tcbinfo can be dropped and then safely re-acquired without race when inpcb lock is dropped
- Option 2: Decompose tcbinfo lock
  - Connection groups (ConnP-L per Willman, et al)
  - Less disruptive of current code

TODO

- Identify key code paths that would benefit from increased parallelism
  - IP forwarding path, netisr/loopback
- Develop new work management models to allow feedback from scheduler
  - Is it cheaper to direct dispatch or are we overloading the current thread/CPU?
  - Are there CPUs available to do additional work?
  - How should the network stack tell the scheduler about data/connection affinity?