


# Linux Kernel Issues in End Host Systems



---

Wenji Wu, Matt Crawford

US-LHC End-to-End Networking Meeting  
Fermi National Accelerator Lab, 2006

[wenji@fnal.gov](mailto:wenji@fnal.gov); [crawdad@fnal.gov](mailto:crawdad@fnal.gov)



# Topics

---

- Background
- Linux 2.6 Characteristics
- Kernel Memory Layout vs. Packet Receiving
- Kernel Preemptivity vs. Linux TCP Performance
- Interactivity vs. Fairness in Networked Linux Systems



# Background

---

- **What, Where, and How** are the bottlenecks of Network Applications?
  - Networks?
  - **Network End Systems?**

Linux is widely used in the HEP community



# Linux 2.6 Characteristics

---

- Preemptible Kernel
- $O(1)$  Scheduler
- Improved Interactivity, more responsive
- Improved Fairness
- Improved Scalability

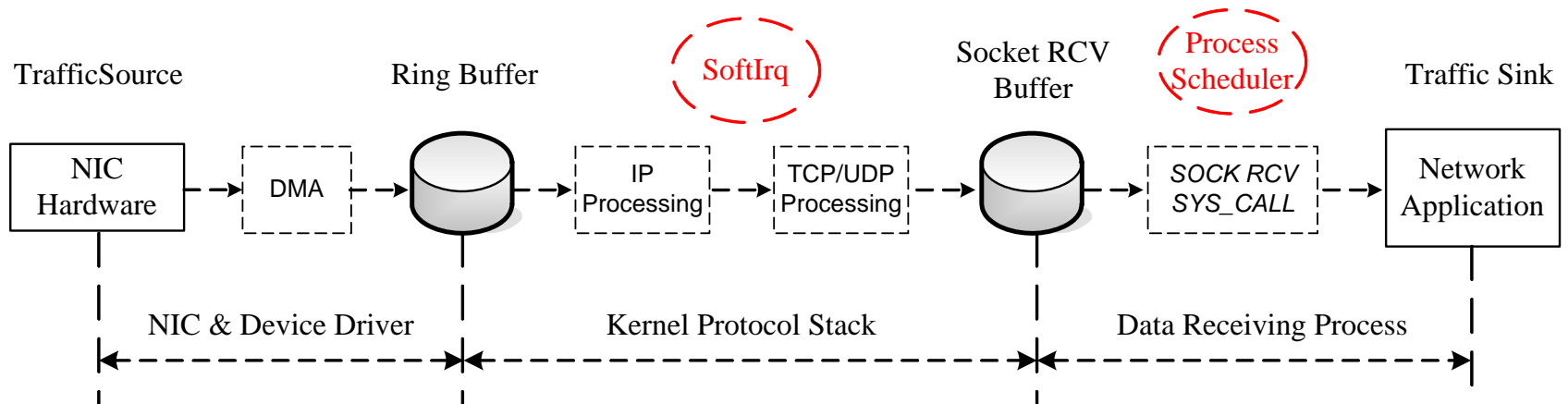


---

# Kernel Memory Layout vs. Packet Receiving

# Linux Networking subsystem: Packet Receiving Process

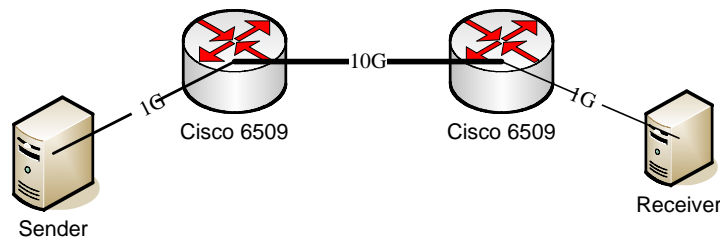
\*



- Stage 1: NIC & Device Driver
  - Packet is transferred from network interface card to ring buffer
- Stage 2: Kernel Protocol Stack
  - Packet is transferred from ring buffer to a socket receive buffer
- Stage 3: Data Receiving Process
  - Packet is copied from the socket receive buffer to the application

# Experiment Settings

- Run *iperf* to send data in one direction between two computer systems;
- We have added instrumentation within Linux packet receiving path
- Compiling Linux kernel as background system load by running ***make -nj***
- Receive buffer size is set as 40M bytes



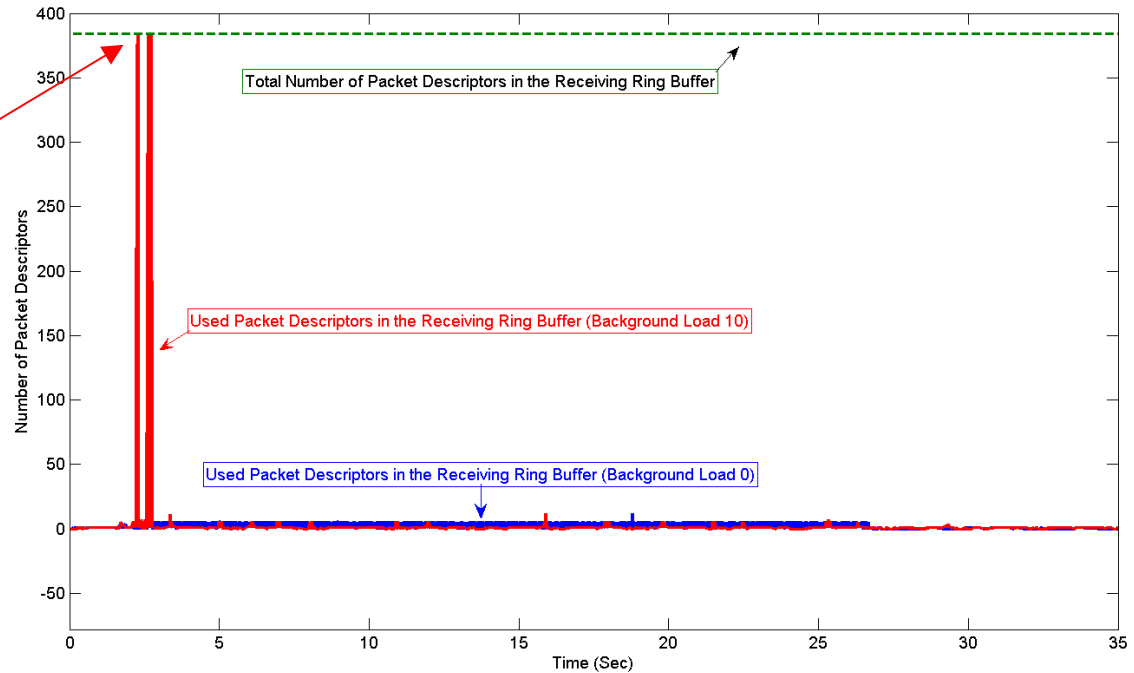
## Fermi Test Network

	Sender	Receiver
CPU	Two Intel Xeon CPUs (3.0 GHz)	One Intel Pentium II CPU (350 MHz)
System Memory	3829 MB	256MB
NIC	Tigon, 64bit-PCI bus slot at 66MHz, 1Gbps/sec, twisted pair	Syskonnect, 32bit-PCI bus slot at 33MHz, 1Gbps/sec, twisted pair

## Sender & Receiver Features

# Receive ring buffer

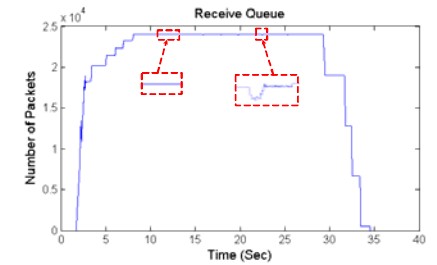
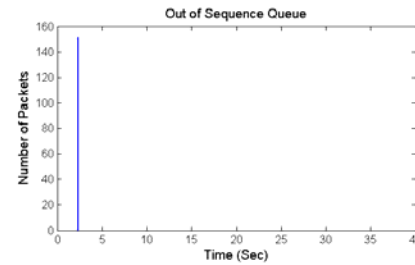
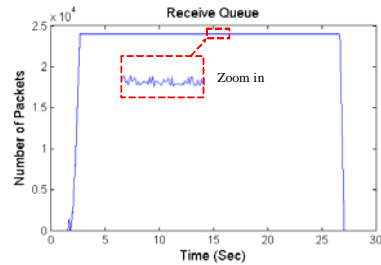
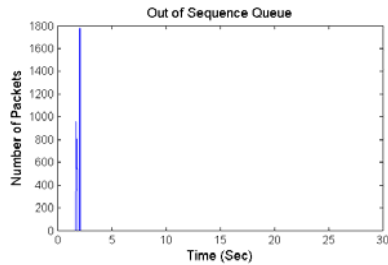
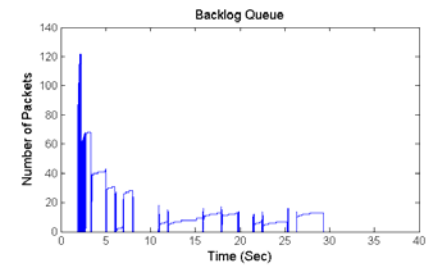
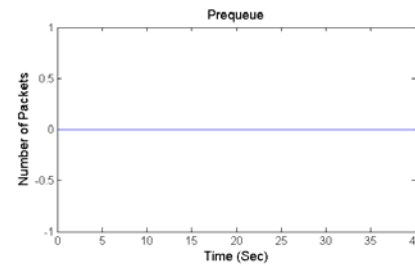
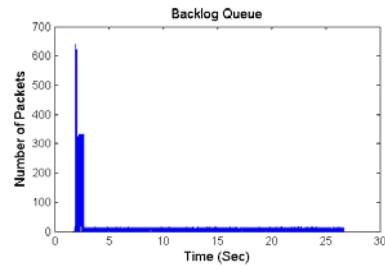
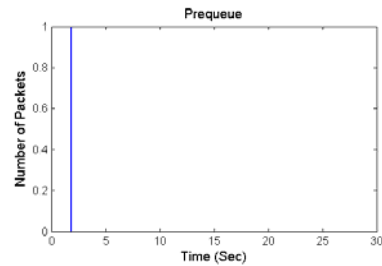
Running out  
packet descriptors



Total number of packet descriptors in the reception ring buffer of the NIC is 384

Receive ring buffer could run out of its packet descriptors: Performance Bottleneck!

# Various TCP Receive Buffer Queues



Background Load 0

Background Load 10

Receive buffer size is set as 40M bytes

What do the results mean?

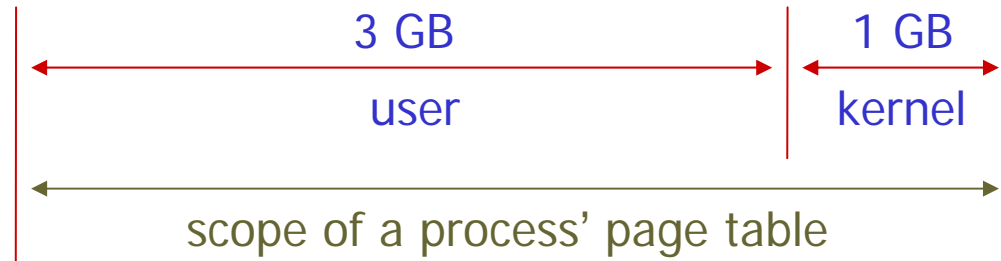
# How to configure socket receive buffer size?

%\$

- We usually configure the socket receive buffer to the BDP.
- In real world, system administrators often configure `/proc/net/ipv4/tcp_rmem` high to accommodate high BDP connections.

What could be wrong?

# Linux Virtual Address Layout

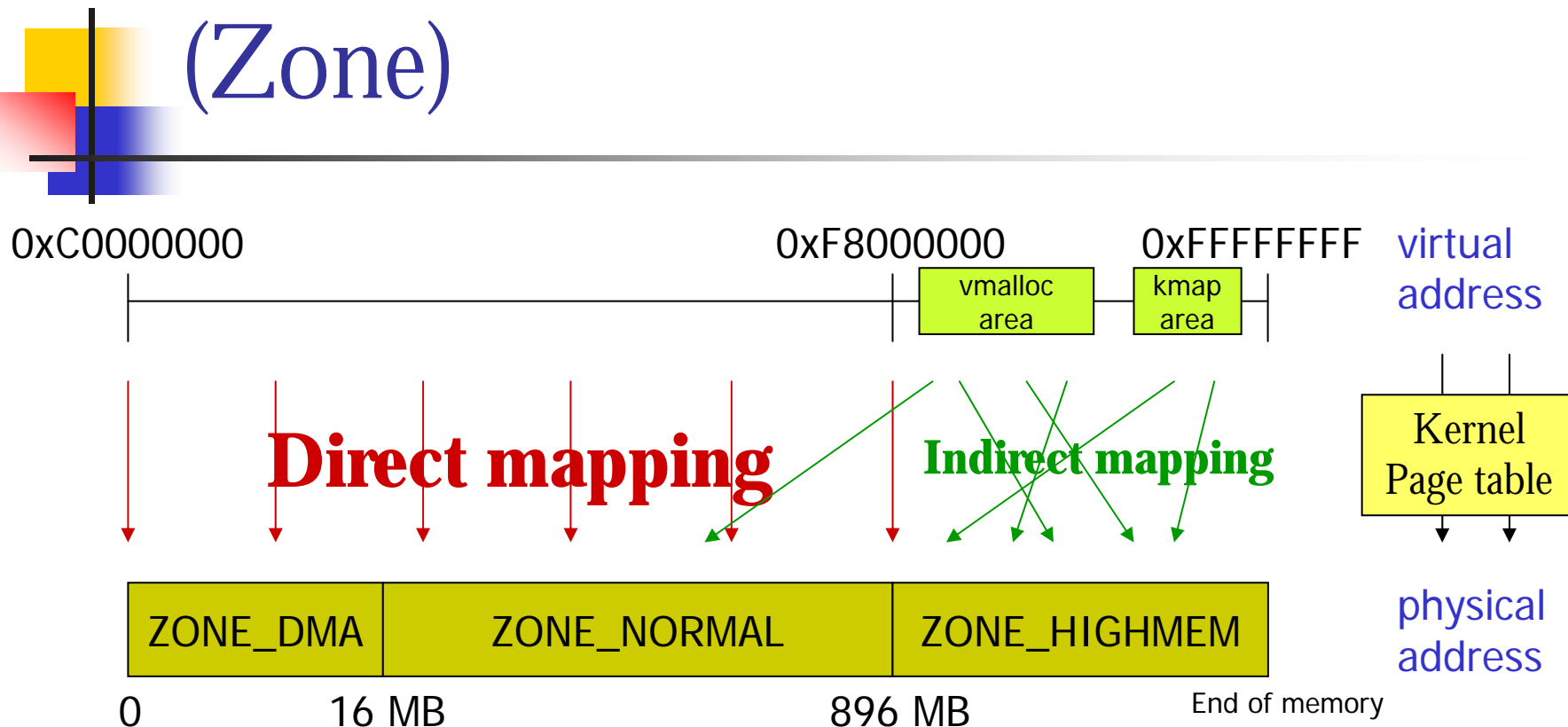


## ■ 3G/1G partition

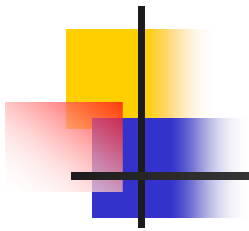
- The way Linux partition a 32-bit address space
- Cover user and kernel address space at the same time
- Advantage
  - Incurs no extra overhead (no TLB flushing) for system calls
- Disadvantage
  - With 64 GB RAM, mem\_map alone takes up 512 MB memory from lowmem (ZONE\_NORMAL).

# Partition of Physical Memory (Zone)

%8



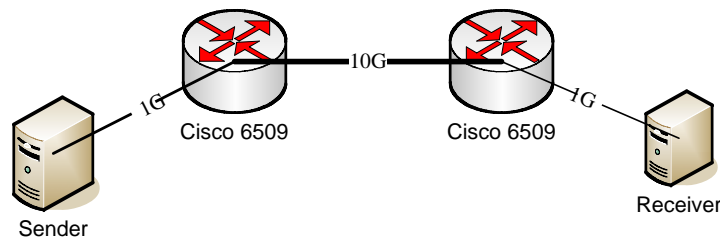
This figure shows the partition of physical memory its mapping to virtual address in 3G/1G layout



# Kernel Preemptivity vs. Linux TCP Performance

# Preemptivity vs. Linux TCP Performance Experiment Settings

- Run *iperf* to send data in one direction between two computer systems
- We have added instrumentation within Linux packet receiving path
- Compiling Linux kernel as background system load by running *make -nj*
- Receive buffer size is set as 40M bytes



Fermi Test Network

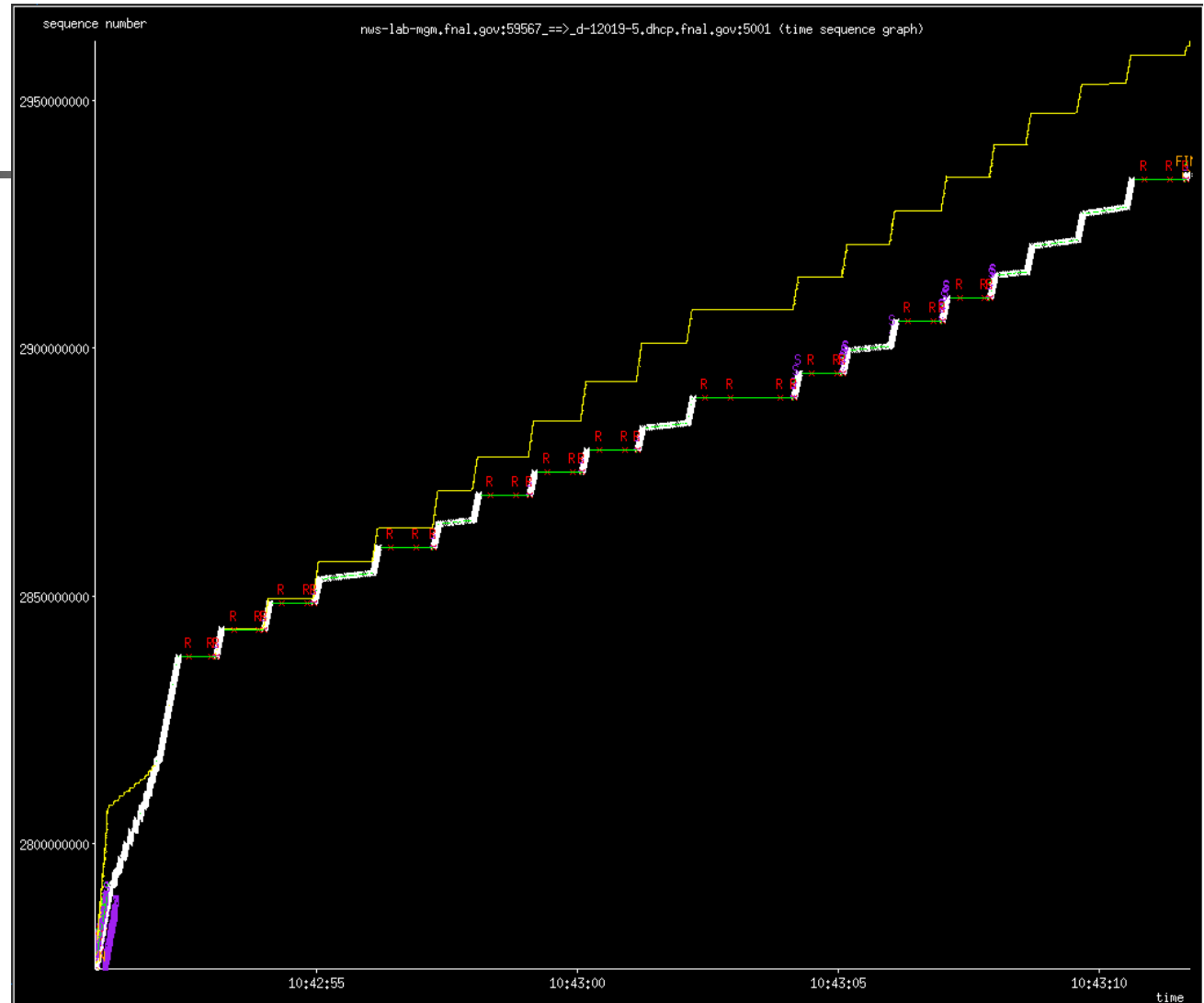
	Sender	Receiver
CPU	Two Intel Xeon CPUs (3.0 GHz)	One Intel Pentium II CPU (350 MHz)
System Memory	3829 MB	256MB
NIC	Tigon, 64bit-PCI bus slot at 66MHz, 1Gbps/sec, twisted pair	Sysconnect, 32bit-PCI bus slot at 33MHz, 1Gbps/sec, twisted pair

Sender & Receiver Features

# What, Why, and How?

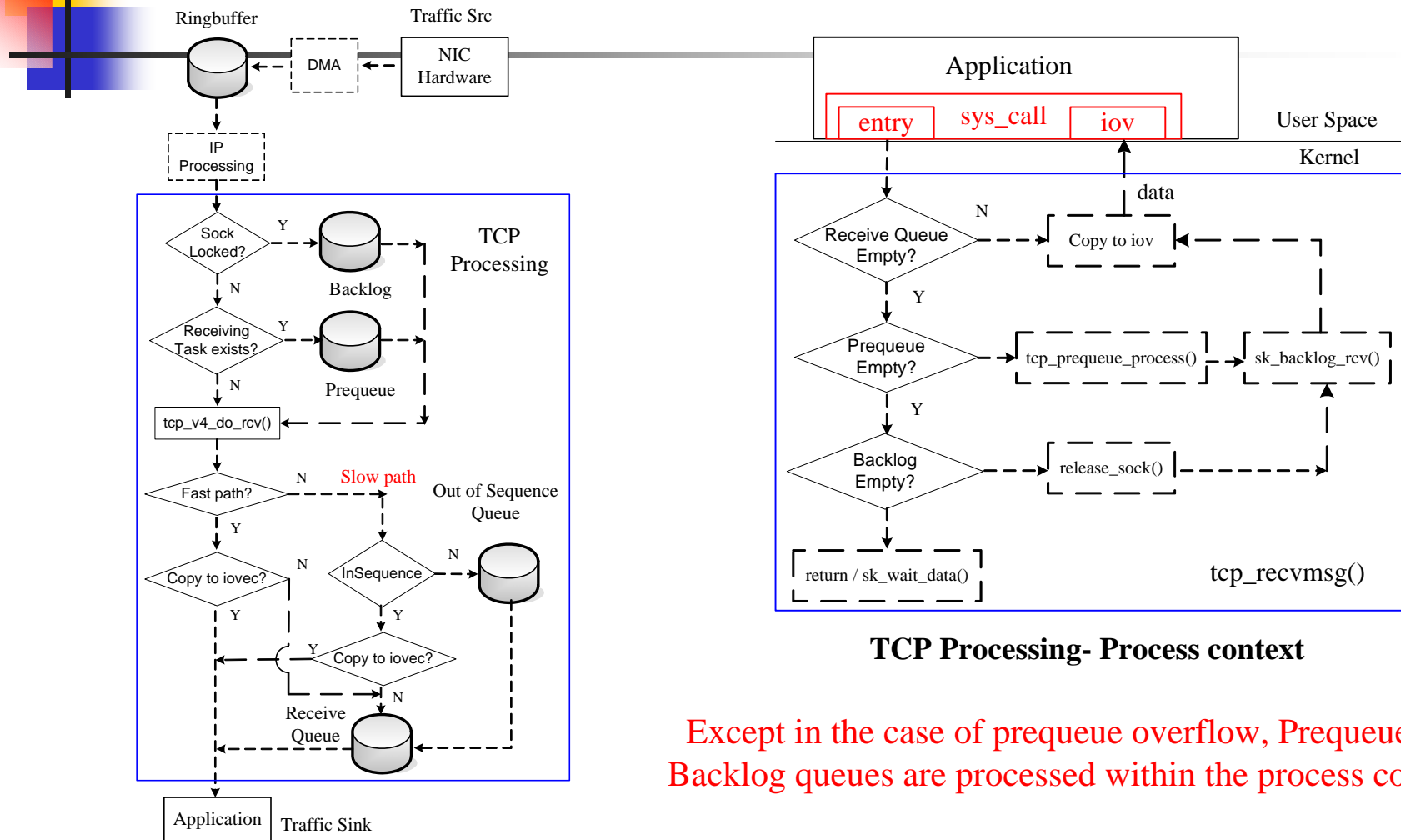
90

Background Load 10



Tcptrace time-sequence diagram from the sender side

# Kernel Protocol Stack – TCP

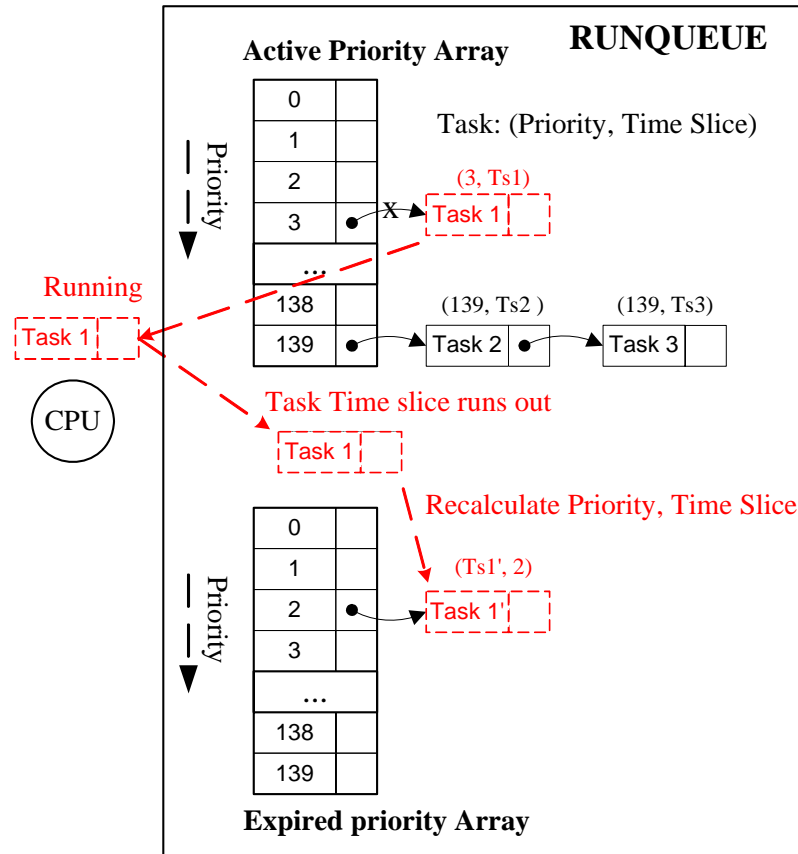


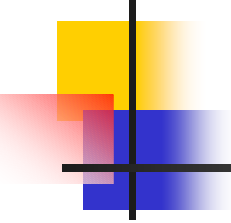
**TCP Processing- Interrupt context**

**TCP Processing- Process context**

Except in the case of prequeue overflow, Prequeue and Backlog queues are processed within the process context!

# Linux Scheduling Mechanism





---

# Interactivity vs. Fairness in Networked Linux Systems











