ECE 697J – Advanced Topics in Computer Networks

Software-Based Router Architectures
9/25/03

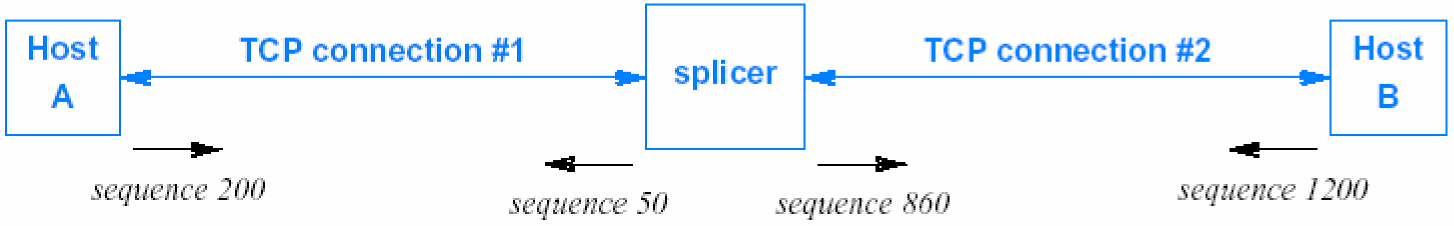
TCP Connection Recognition

- Track currently active connections
- Algorithm:

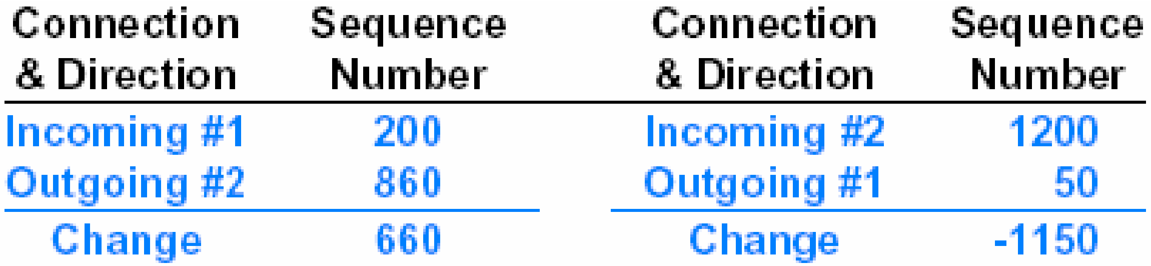
```
For each IP datagram with TCP segment {
    Extract IP source, S, destination, D, address
    Extract source port, $P_1$, and destination port , $P_2$
    Create/update entry in hash table C
          using (S, D, $P_1$, $P_2$) as key
    If segment has RESET bit set, delete entry
    Else if segment has FIN bit set,
          mark connection closed in on direction
    Else if segment has SYN bit set,
          mark connection as established in one direction
    Delete entry if both directions are closed
    Mark completely established if both directions are established
}
```

TCP Splicing

- On TCP connection establishment, node chooses random sequence number
- When two connections are "spliced together", sequence numbers need to be adjusted:



- Need to keep offset between connections and adjust each packet:



Connection & Direction	Sequence Number	Connection & Direction	Sequence Number
Incoming #1	200	Incoming #2	1200
Outgoing #2	860	Outgoing #1	50
Change	660	Change	-1150

Packet Processing in Software

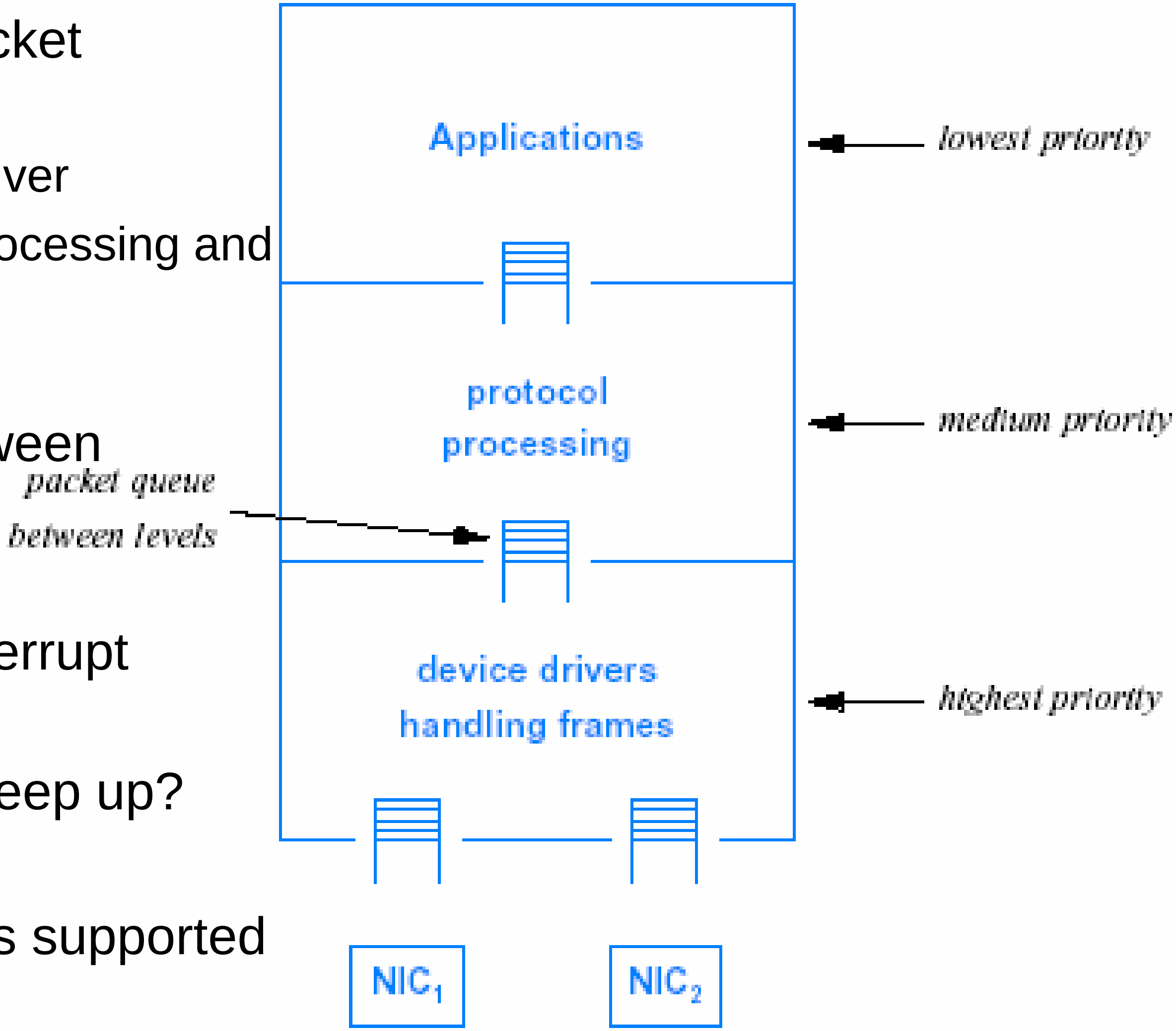
- Can we write a packet processing application?
  - We could, but inefficient
  - OS has better access to hardware resources
  - OS can access packet data without copying
- Embedded System
  - Stand-alone device
  - Software is optimized for particular task
  - No other functionality
  - Possibly difficult to program
- OS Implementation
  - Packet processing part of OS kernel
  - Benefits: OS abstractions, device drivers, reusable for all users

Interrupts and Priorities

- What is an interrupt?
  - Event that signals to operating system
  - Hardware interrupts: raised by device
  - Software interrupts: raised by software
  - “Interrupt handler” is called to process interrupt
- Priorities
  - Interrupts have different priorities
  - Examples?
  - Higher priority interrupts can interrupt lower priority code
  - Kernel software can specify desired interrupt level

Packet Processing and Interrupts

- Interrupt levels for packet processing:
  - Highest to device driver
  - Lower to protocol processing and application
  - Why?
- Requires queues between interrupt levels
  - Why?
- Processing in high interrupt should be kept brief
- What if CPU cannot keep up?
  - Livelock
- Only few priority levels supported

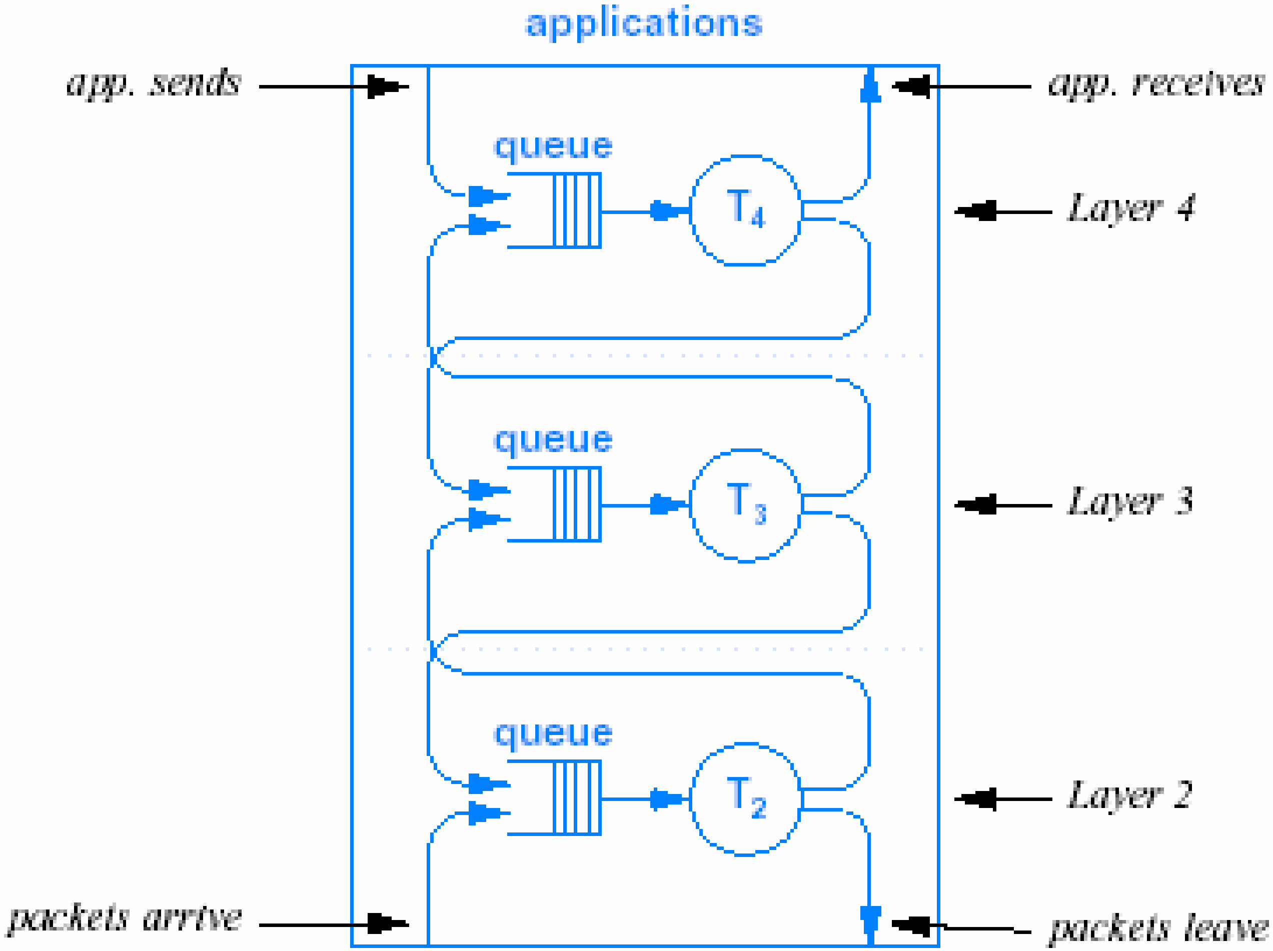


Kernel Threads

- Thread is piece of software that runs in its own context
  - Similar to process, just light-weight
- Different threads can run in different priorities
  - More fine-grained than interrupt levels
  - Scheduling policy allocates CPU to threads
- Thread synchronization handles access to shared data
  - Mutual exclusion: only one thread has access to data structure
  - Notification: thread blocks until event occurs
- How should threads be assigned to protocol software?
  - One thread per layer
  - One thread per protocol
  - Multiple threads per protocol
  - Protocol threads plus timer management
  - One thread per packet

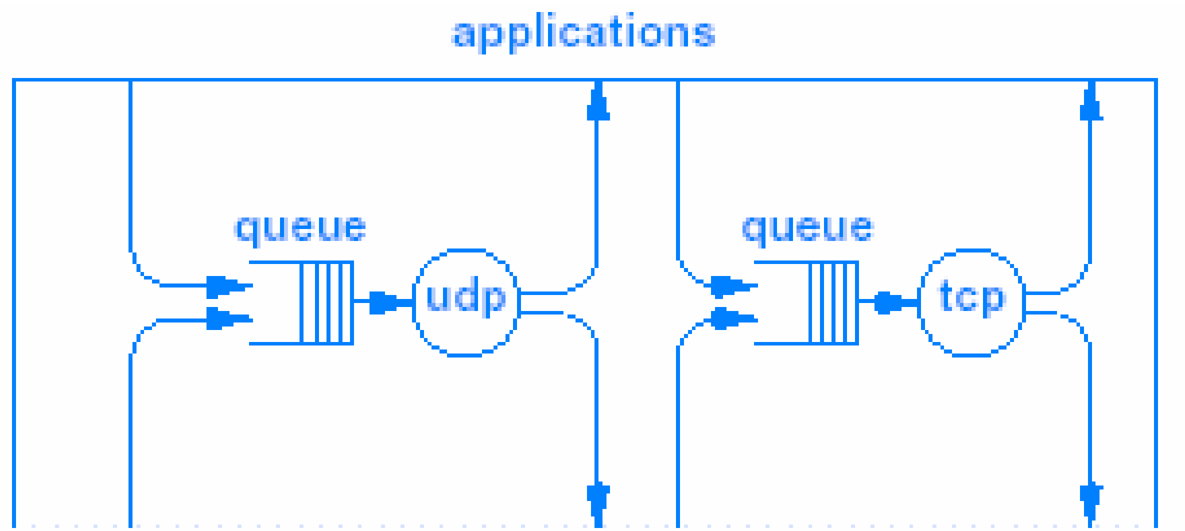
One Thread per Layer

- Different layers have different priorities
- Requires queues between layers



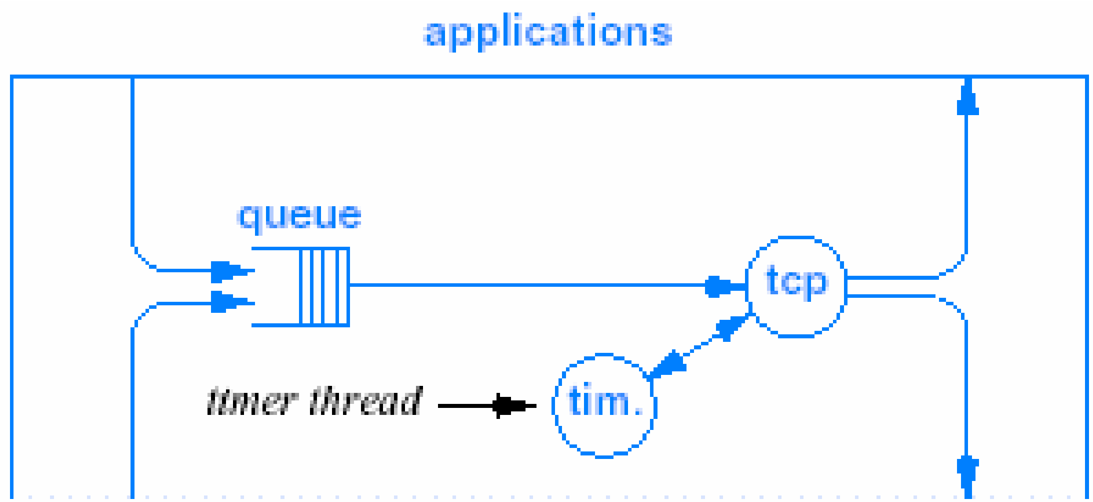
Per Protocol Threads

- Example: UDP and TCP
- Each protocol has queue
- Multiple threads per protocol:
  - Handle incoming and outgoing packets separately



Timer Management Threads

- Timer management is fundamental functionality for all protocols
- Timer thread handles all timers in system
- Problem: timer can range from mircoseconds to minutes
  - Shorter timers need higher priority
  - Why?



One Thread per Packet

- Layers introduce overhead
  - Queuing
  - Context switching
- Requires many threads
- Packet is processed entirely by one thread
- “Run to completion” programming model

Asynchronous vs. Synchronous

- Asynchronous programming
  - Program is structured around interrupts
- Synchronous programming
  - Program is structured around threads
  - “packet centric”
- Synchronous is easier to understand
  - Thread abstraction handles notification and mutual exclusion
  - Possibly less efficient
  - Probably less complex to understand

Next Class

- Hardware-based router architectures
  - Read chapter 8
- Router design paper
  - Read paper