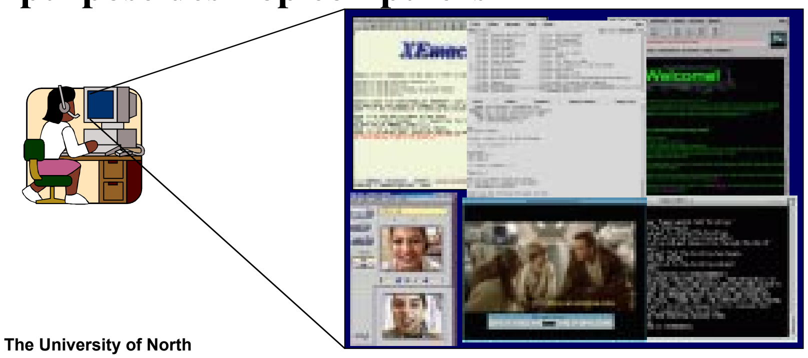
Proportional-Share Scheduling of Operating System Services for Real-Time Applications

Kevin Jeffay
F. Donelson Smith
Arun Moorthy
James H. Anderson

Department of Computer Science
University of North Carolina at Chapel Hill
http://www.cs.unc.edu/Research/Dirt/

Motivating Problem

- **Real-time and non-real-time tasks present in current day workload**
- **Aim: To support this workload on general purpose desktop computers**

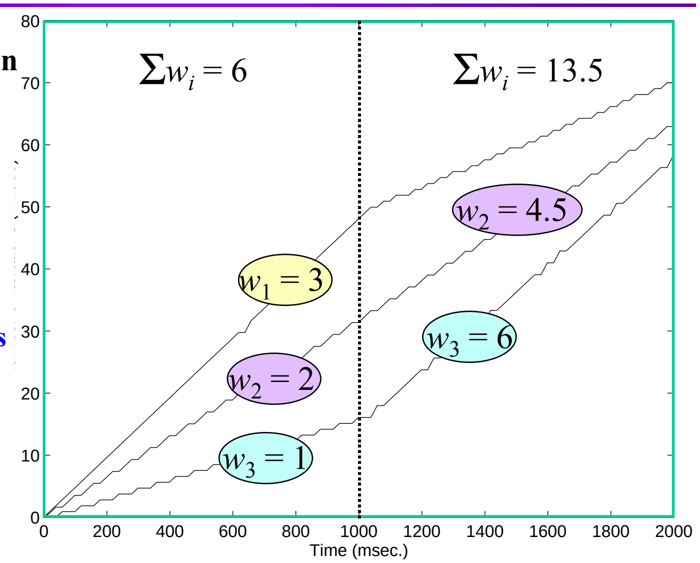


Potential Solutions/Related Work

- **Build a new real-time operating system**
  - » Rialto (Jones et al., 1997)
- **Real-time extensions to existing operating systems**
  - » Real-Time Mach (Tokuda et al., 1990)
  - » SMART Solaris System (Nieh et al., 1997)
- **Virtual Machine Emulation**
  - » Real-Time Linux (Barbarnov & Yodaiken)
  - » Real-Time IBM Microkernel (Bollella & Jeffay, 1995)

Proportional Share Scheduling of User Processes

- **Proportional share ensures *fair* allocation**
- **Processes assigned a weight *w***
  - » weight determines process's *share*
- **Fairness can be used to ensure real-time execution**
  - » Real-time processes have a fixed share
  - » Non-real-time processes have a variable share



Proportional-Share-based Real-Time Extensions

- SFQ SVR4 Unix (Goyal et al., 1996)
- Mach- and FreeBSD-based Lottery Scheduling (Waldspurger & Weihl, 1994)
- FreeBSD: EEVDF version (Stoica et al., 1996)

All perform Proportional Share Scheduling at the User-Level

Scheduling of OS Services
Integrated Resource Allocation

Process Lifetime

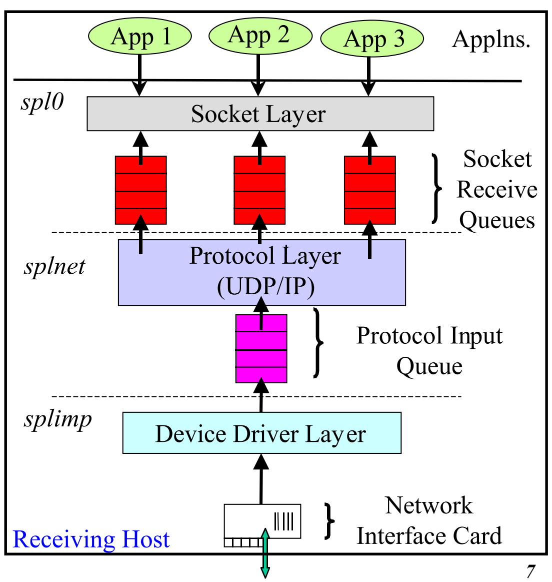
User Context	Kernel Context
• Extensive research • Sophisticated process scheduling	• Relatively less attention • Process-independent scheduling

Undesirable Effect: Improper Allocation of Resources within the Kernel might adversely affect Real-Time performance

Solution: Integrate Scheduling of Operating System Activities and Application Scheduling

Example: Protocol Processing in BSD Unix

+ Advantages
  - » Fast response
  - » High throughput
– Disadvantages
  - » Static priority network processing
  - » Receive livelock
  - » No packet distinction



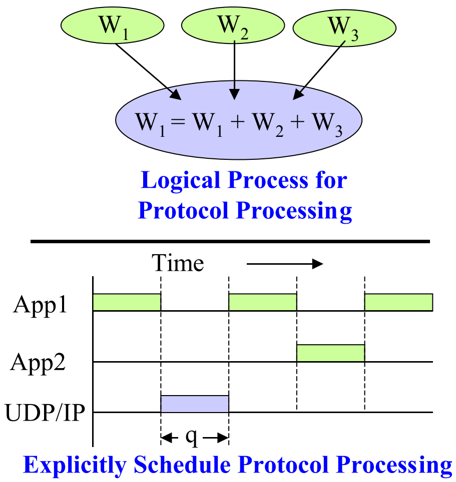
Real-Time Network & Protocol Processing
Principles

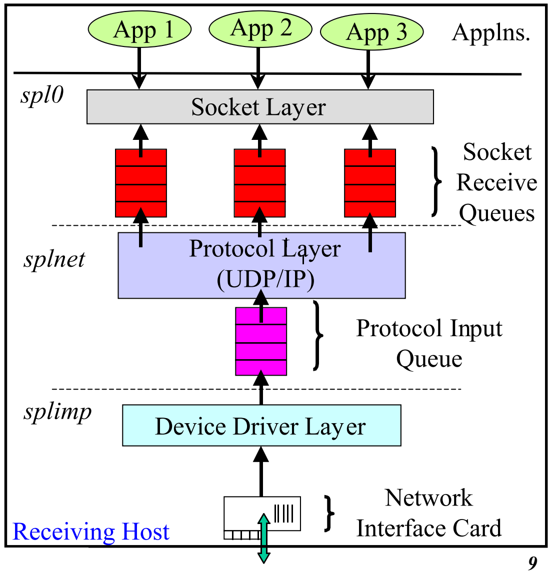
Schedule Protocol Processing exactly like any other activity

Example: Real-Time Mach (Lee et al., 1996)

- Protocol stack is a library
- Protocol processing is
  - » schedulable
  - » fully preemptible

Proportional-Share Network & Protocol Processing





Real-Time Network & Protocol Processing

Principles (Contd.)

Early Packet Demultiplexing

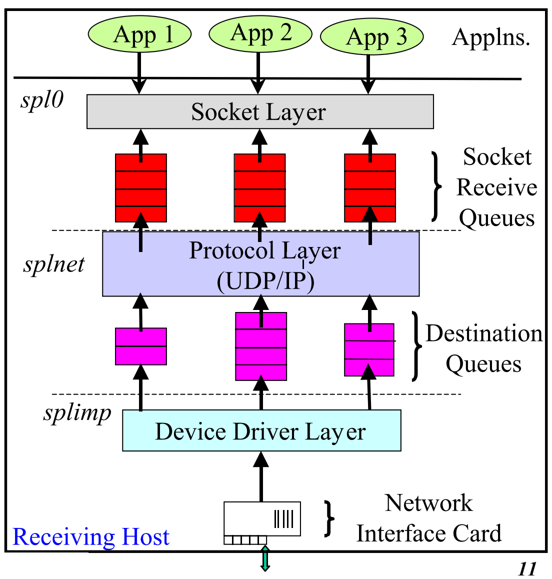
Example: Lazy Receiver Processing (Druschel & Banga, 1996)

- One queue per socket on receiver
- *Lazy* protocol processing

Protocol Processing

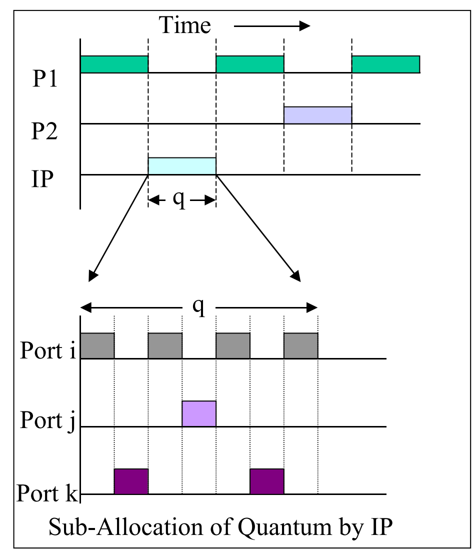
Destination Queues

- One queue per socket
  - » as in LRP
- Varying queue lengths
  - » Queue length is a function of process weights



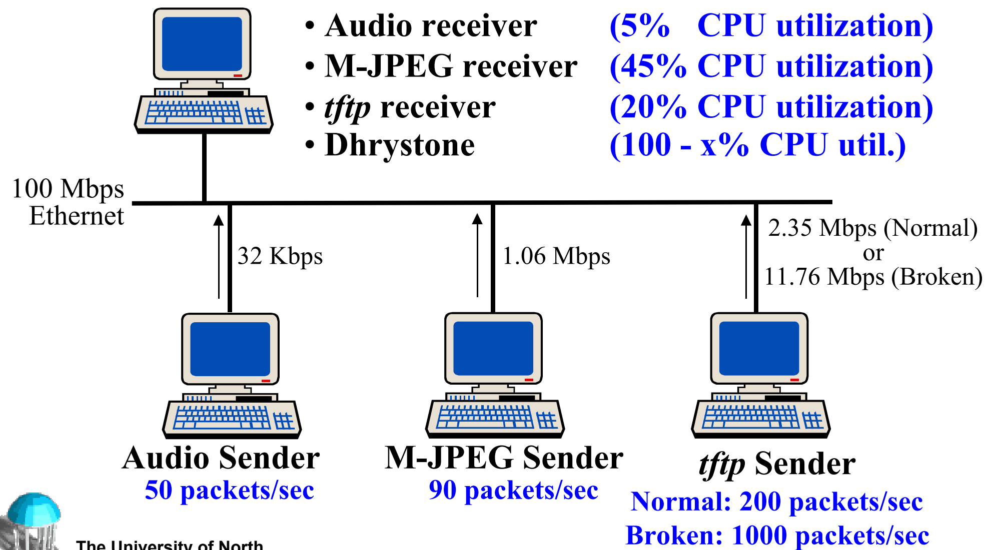
Hierarchical Packet Scheduler

- Assign weight/cost to each packet
- Proportional-share sub-allocation of quantum



Experimental Setup
(FreeBSD 2.2.2-Release)

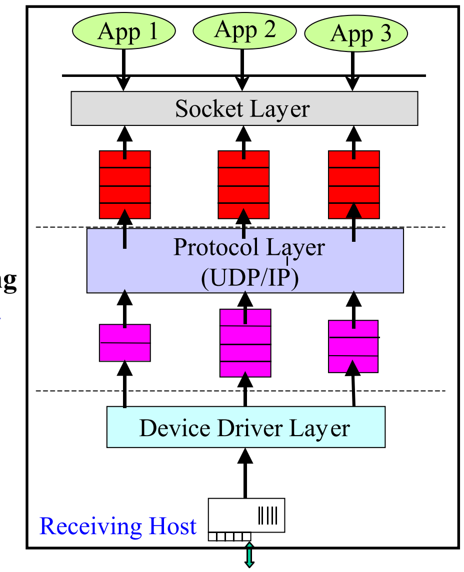
- Audio receiver (5% CPU utilization)
- M-JPEG receiver (45% CPU utilization)
- *tftp* receiver (20% CPU utilization)
- Dhrystone (100 - x% CPU util.)



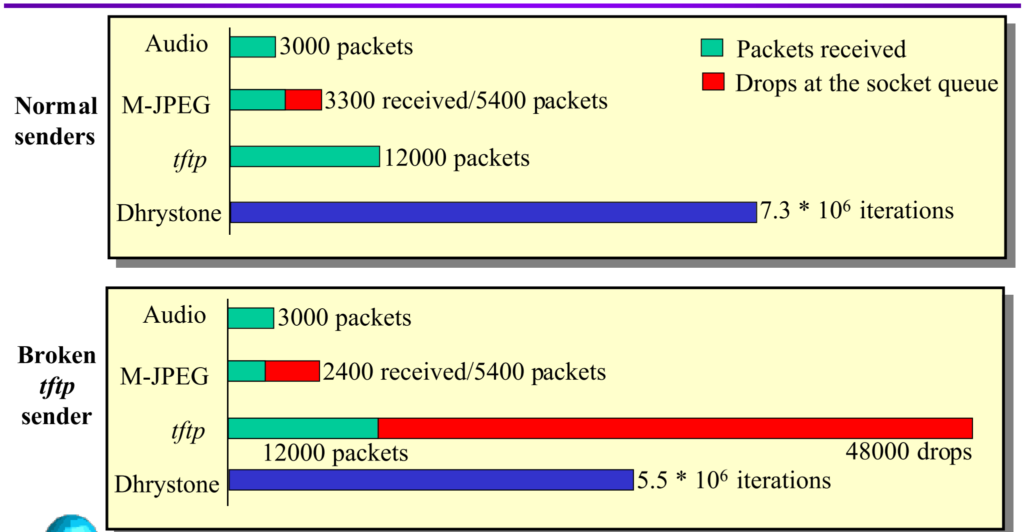
Outline of Experiments

1. Baseline: Unmodified FreeBSD
2. Prop-share at user-level
3. Prop-share at user-level & IP
4. Prop-share at user-level & IP with destination queues
5. Prop-share at user-level & IP with destination queues & packet scheduling

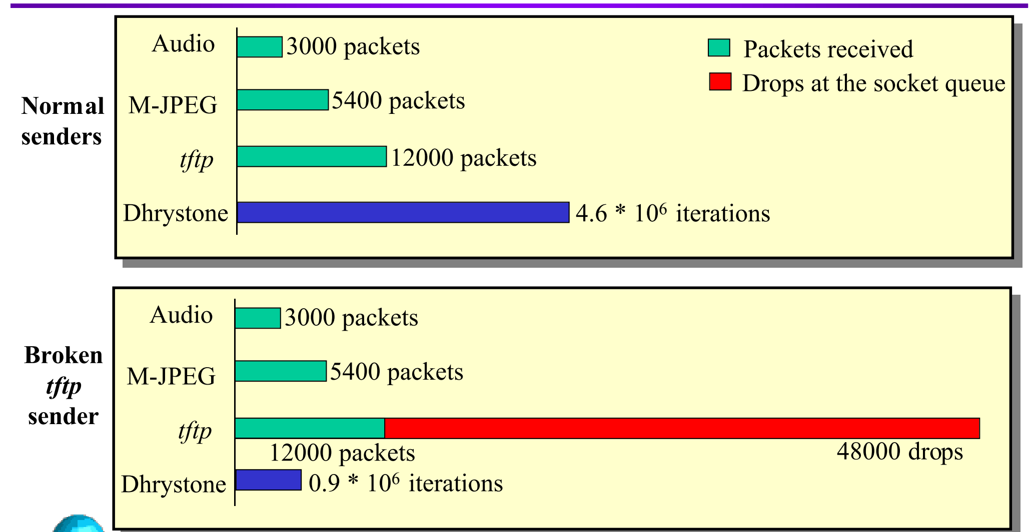
In each trial:
- Regular senders
- Bursty senders
- Broken tftp sender



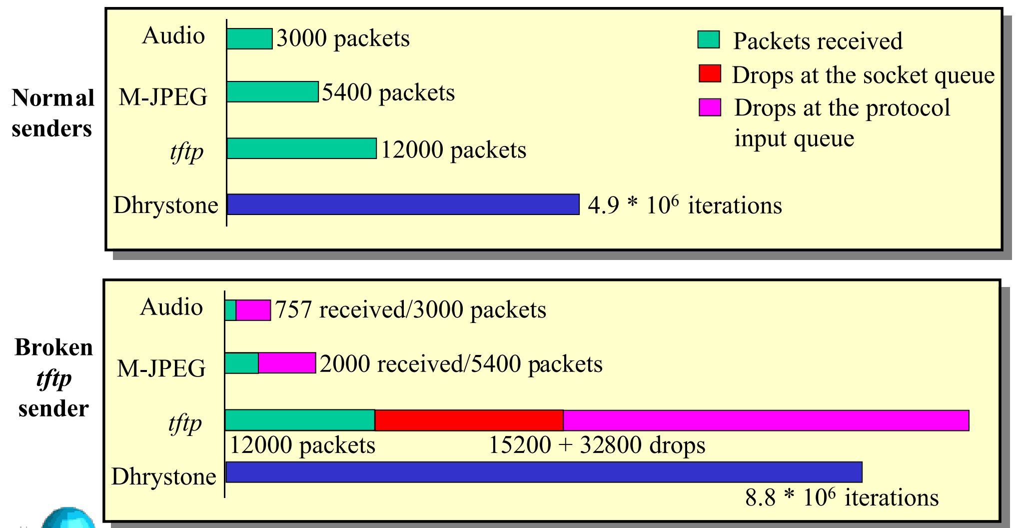
Experimental Results
Unmodified FreeBSD: 1ms quantum



Proportional-Share Scheduling at the User-Level

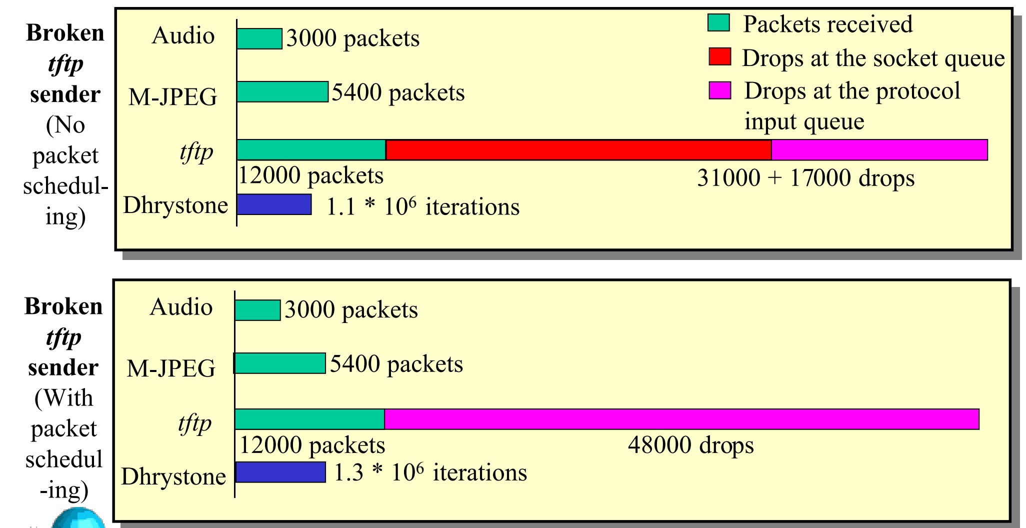


Proportional-Share at the User-Level and IP



Proportional-Share at the User-Level and IP

Destination Queues (With and Without Packet Scheduling)



Proportional-Share Scheduling of Operating System Services for Real-Time Applications

Conclusions

- **Operating system activities need to be scheduled as well as user processes**
- **Proportional-share is effective in both domains**
- **Developed a limited proportional-share version of FreeBSD**
  - » **Network subsystem in kernel is implemented in a prop-share manner**
  - » **User processes are scheduled in a prop-share fashion**
  - » **Solution to the *receive livelock* problem**