

# CS4220 Embedded Systems CS6235 Real-Time Systems

## 3B: Evaluation

**Instructor: Calton Pu**

**calton.pu@cc**

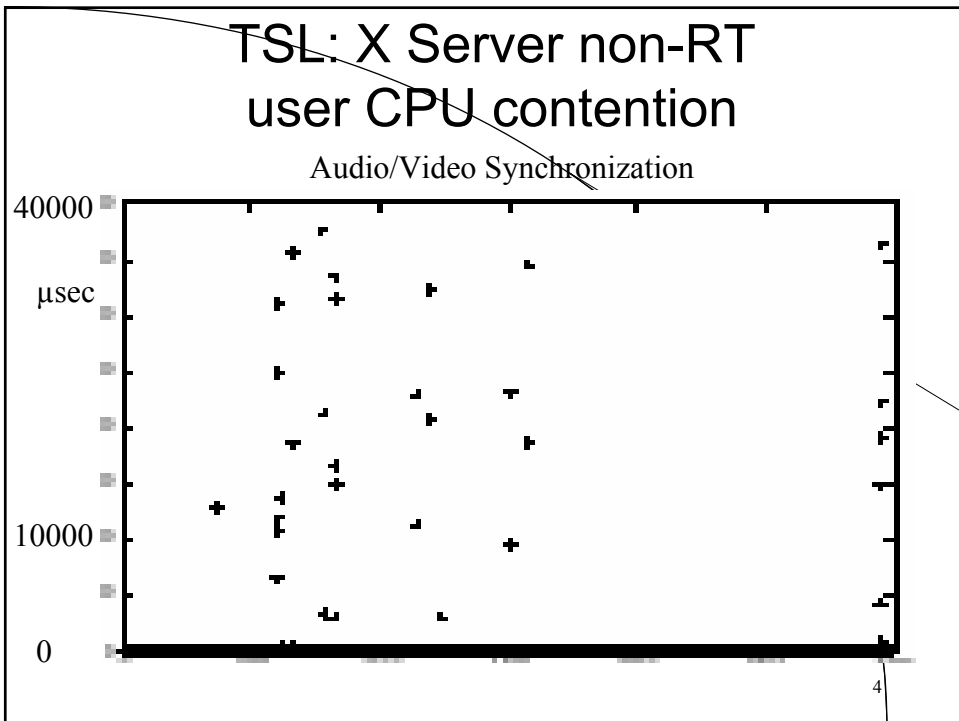
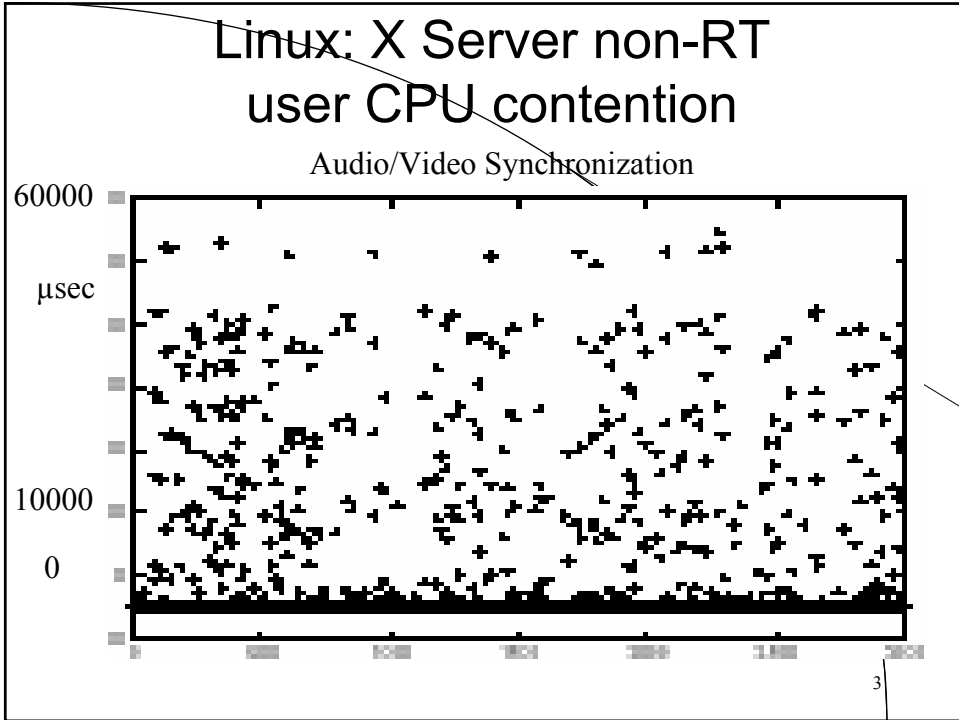
TA: Younggyun Koh (young@cc)

1

## Showing Predictability

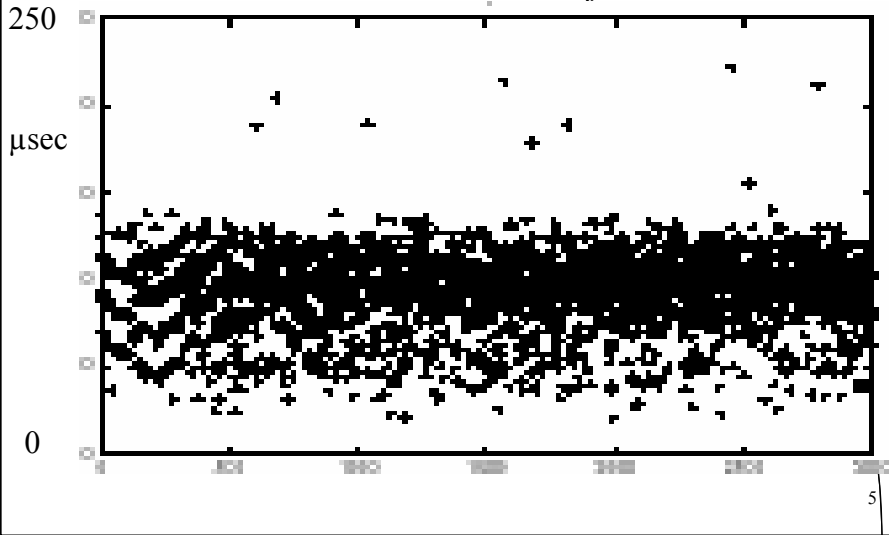
- A representative workload
  - Sufficient coverage and resource usage
  - Easy to understand
- Compare the alternative target systems
  - Show the execution time and variances
  - Vary the environment in a controlled manner
  - Sufficient coverage and consistency

2

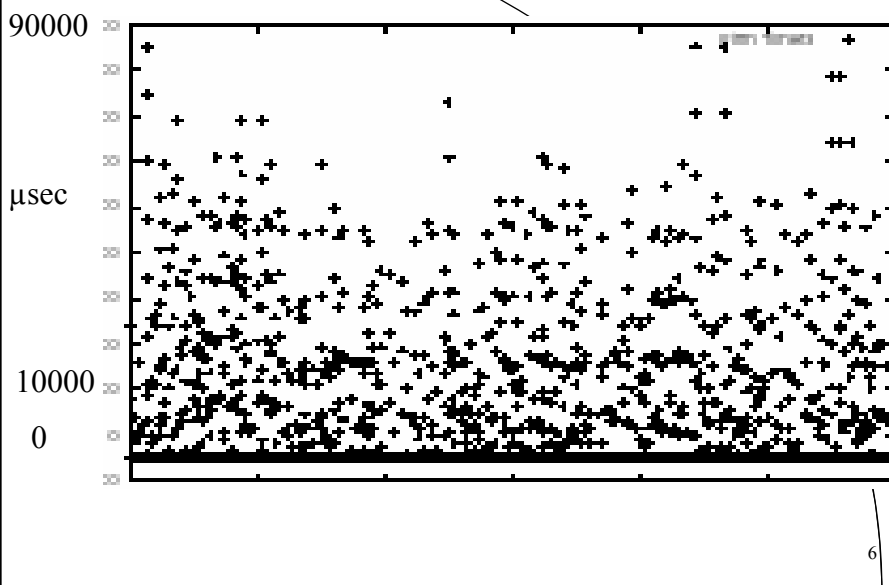


# TSL: X Server RT user CPU contention

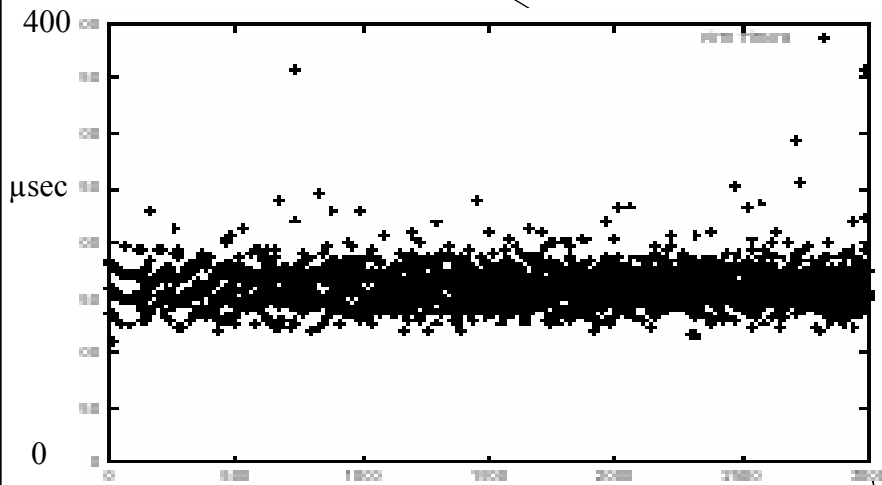
Audio/Video Synchronization



# Linux, kernel CPU contention

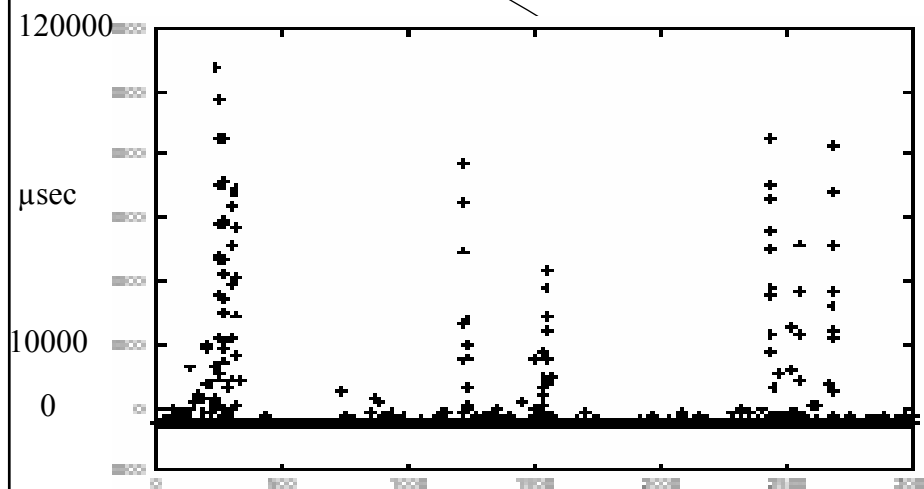


## TSL, kernel CPU contention



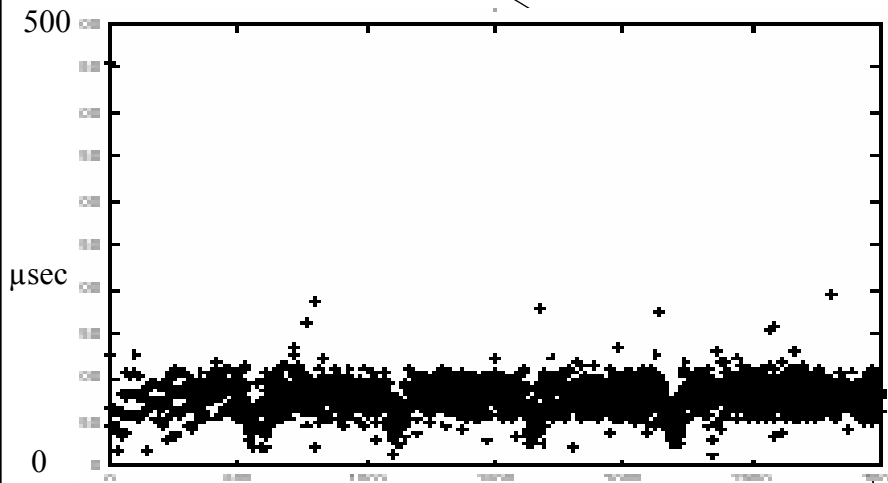
7

## Linux, file system contention



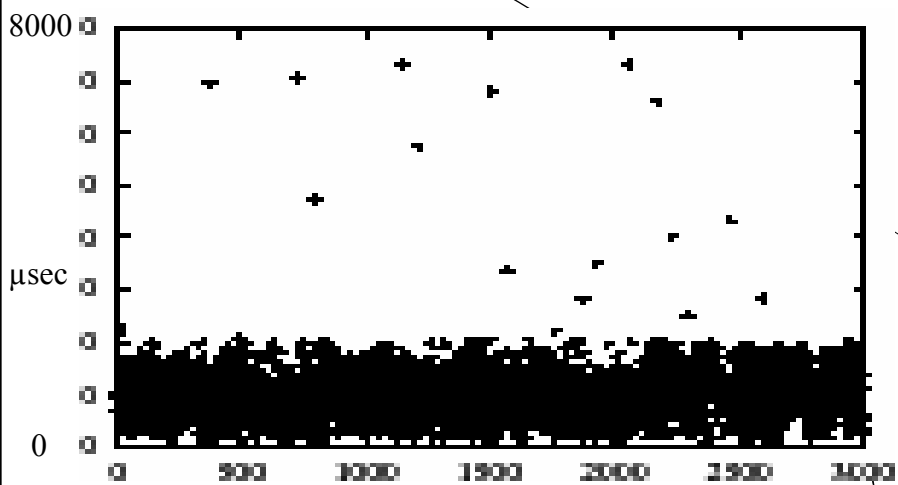
8

## TSL, file system contention



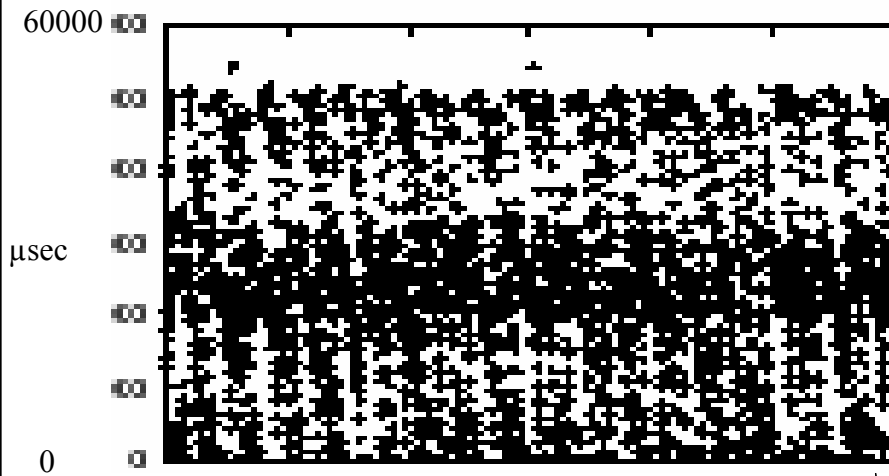
9

## Linux-SRT, user CPU contention

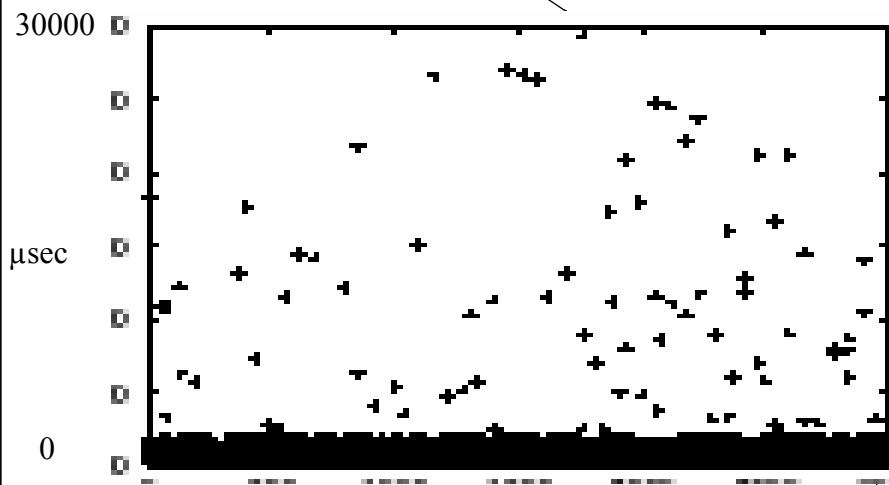


10

# Linux-SRT, kernel CPU contention



# Linux-SRT, file system contention

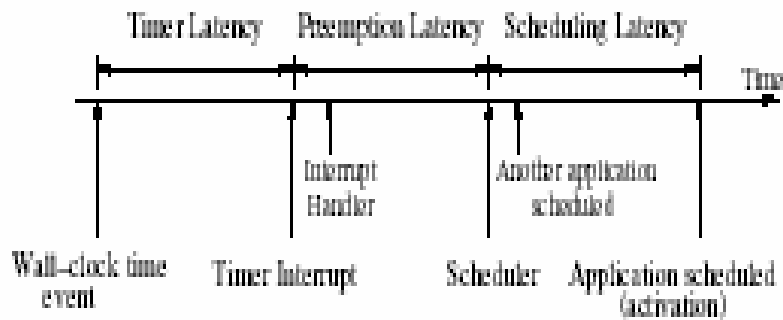


# Paper's Contribution

- Situation: Linux (and many other OS's) cannot handle real-time applications
- Hypothesis: a few techniques can overcome the limitations of Linux
  - Firm timer (high resolution)
  - Fine-grain kernel preemption
  - Priority and reservation-based scheduling
- Good performance and low overhead

13

# Event Handling Structure



14

## Problems and Solutions

- Firm Timer reduces timer latency
  - Low overhead important
- Fine-grain kernel preemption reduces preemption latency
  - Responsive kernel
- Good schedulers reduce scheduling latency
  - Proportion-period scheduler

15

## Firm Timers

- Periodic timer in Linux (10ms)
  - Trade-off between latency and cost
- One-shot timers
  - Reprogrammed for next timer interrupt
- Soft timers reduce interrupt overhead
  - Polling at convenient times (e.g., syscall return)
- Timer overshoot
  - Reduce variance due to soft timer uncertainty

16

## Fine-Grain Kernel Preemption

- Linux (and many OS's)
  - Long sections of disabled interrupts
- Limiting the length of disabled interrupts
  - Only disable interrupts when accessing shared data structures (protected by spin locks)
  - Voluntarily check for interrupts when long sections of code being locked

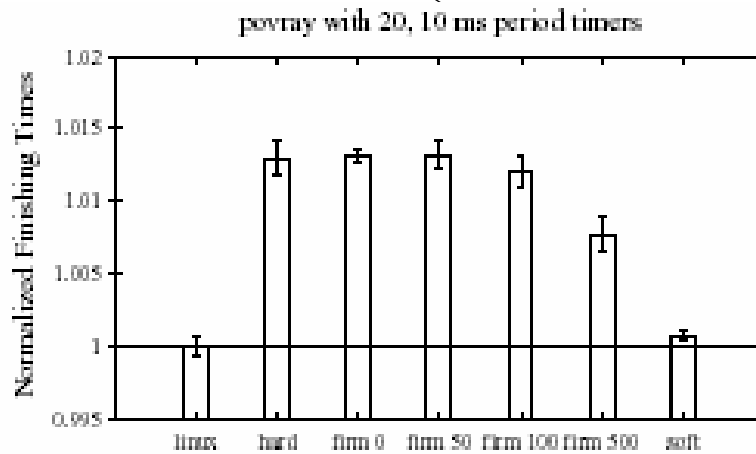
17

## Proportion-Period Scheduler

- For each period, a proportion of CPU is allocated
  - A periodic scheduler, proportional share
  - Highest locking priority (HLP) protocol
  - Bounded execution time for shared code that inherited high priority

18

# Overhead of Firm Timers



19

## Conclusions

- Good graphs are the only thing
- Luck helps: they only needed three things, and most of work has been done
- When luck strikes repeatedly, it's called intuition

20