

Review of Major Concepts

Real-Time and Embedded Systems (M)

Lecture 20

Review of Module Aims

- To introduce and explore the programming language and operating systems facilities essential to the implementation of real-time, reactive, embedded and networked systems.
- To provide the participants with an understanding of the practical engineering issues raised by the design and programming of real-time, reactive, embedded and networked systems.

Review of Intended Learning Outcomes

- Participants should be able to:
 - Clearly differentiate the different issues that arise in designing soft and hard real-time, concurrent, reactive, safety-critical and embedded systems.
 - Explain the various concepts of time that arise in real-time systems.
 - Analyse and apply a variety of static and dynamic scheduling mechanisms suitable for soft and hard real-time systems. Conduct simple performance and schedulability analysis to demonstrate that a system can successfully meet real-time constraints.
 - Explain the additional problems that arise in developing distributed and networked real-time systems.
 - Describe the design and implementation of systems that support real-time applications. Justify and critique facilities provided by real-time operating systems and networks.
 - Design, construct and analyse a small, concurrent, reactive, real-time system. Select and use appropriate engineering techniques, and explain the effect of your design decisions on the behaviour of the system.

Review of the Module

- Reference model for real-time systems
- Scheduling theory
 - Clock-driven scheduling
 - Priority-driven scheduling
 - Periodic, aperiodic and sporadic tasks
- Real-time support in operating systems
 - Implementing task schedulers; two-level schedulers
 - Flexible applications to run on general purpose systems
- Resource access control
- Real-time communication
 - Model of the network
 - Real-time on IP networks, need for QoS
- Low-level embedded programming
 - Current approach and possible future directions

A Reference Model for Real-time Systems

- Jobs and tasks
- Processors and resources
- Time and timing constraints
 - Hard and soft real-time
 - Requirements for a system to be hard real-time; validation of the system
- Periodic, aperiodic and sporadic tasks
 - Parameters of a periodic task: φ, p, e, D
 - Aperiodic and sporadic tasks have unpredictable release time; sporadic tasks also have deadlines
- Precedence constraints and dependencies
- Scheduling

Clock-Driven Scheduling

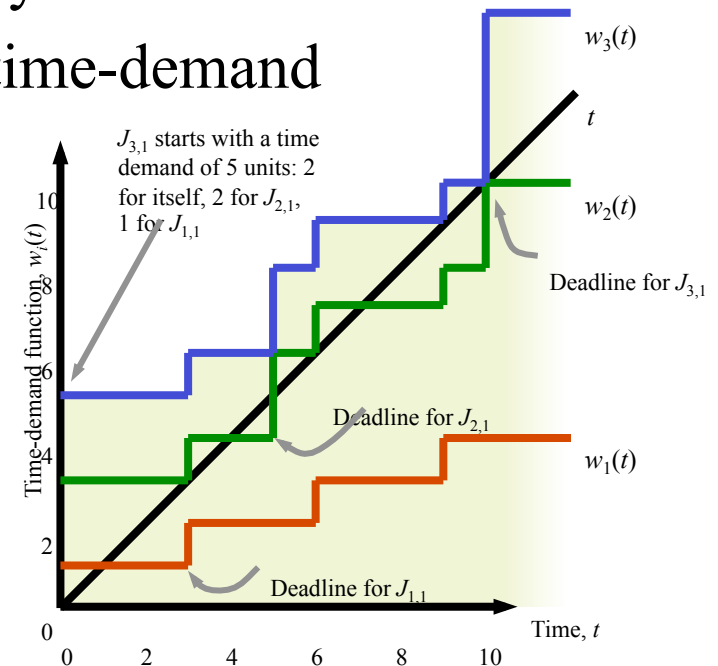
- Static, clock-driven schedules and the cyclic executive
 - Cyclic schedules and scheduler tables
 - Frame-based static schedules; frame size constraints; splitting jobs
- Handling aperiodic and sporadic jobs
 - Slack stealing
- Advantages and disadvantages of clock driven scheduling
 - Simple; applicable to static systems with a small number of aperiodic jobs
 - Doesn't handle dynamic systems; varying sets of tasks

Priority-Driven Scheduling: Periodic Tasks

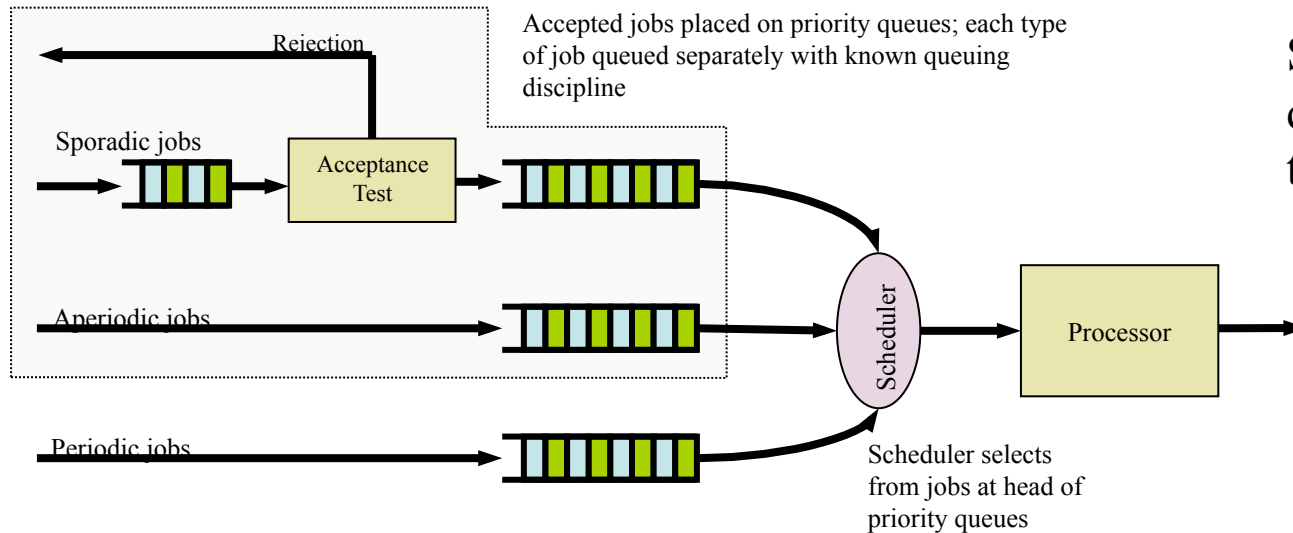
- More complex, but applicable to dynamic systems
- Fixed-priority algorithms
 - Rate monotonic
 - Deadline monotonic
- Dynamic-priority algorithms
 - Earliest deadline first
 - Least slack time
- Relative merits of fixed- and dynamic-priority scheduling
- Demonstration of correctness through simulation
- Use of maximum schedulable utilization as proof of schedulability
 - $U_{EDF} = 1$
 - $U_{RM}(n) = n \cdot (2^{1/n} - 1)$

Priority-Driven Scheduling: Periodic Tasks

- Optimality of EDF
- Optimality of RM for simply periodic systems
- Demonstration of correctness through time-demand analysis
 - Critical instants
- Effects of practical factors:
 - Non-preemptable regions
 - Jobs that self-suspend
 - Jobs with non-distinct priority
 - Blocking and priority inversion



Scheduling Aperiodic and Sporadic Jobs



- Problems with background and interrupt based scheduling of aperiodic jobs
- Use of a periodic server to schedule aperiodic/sporadic jobs:
 - Polling server
 - Bandwidth preserving servers; consumption and replenishment rules
 - Deferrable; sporadic; constant utilization and total bandwidth
 - Using EDF scheduling within the server for sporadic jobs
 - Acceptance tests

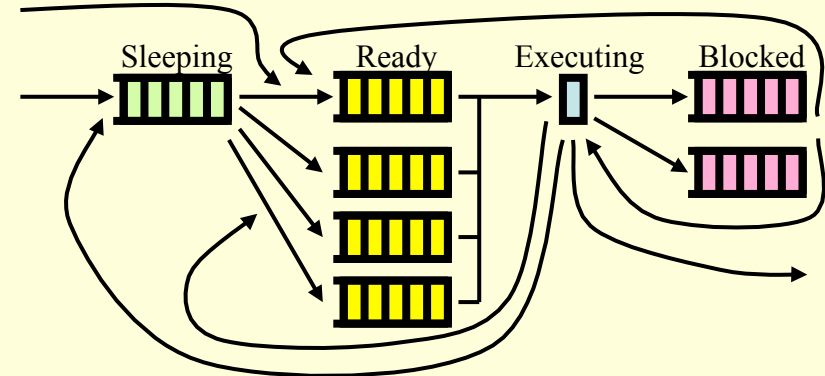
Real-time Support in Operating Systems

- Real time vs. general purpose operating systems
 - Real time constraints; requirements for predictability
 - Implications on real time operating system design
- Real time operating system concepts
 - Overall system architecture; flexible microkernel
 - Time services and scheduling
 - Interrupts, hardware, and system calls
- Example systems
 - C and POSIX real-time scheduling standards
 - Rate monotonic scheduling
 - Sporadic scheduling
 - Real-time Java
 - Advantages and disadvantages compared to POSIX/C
 - RTLinux

Implementation of Task Schedulers

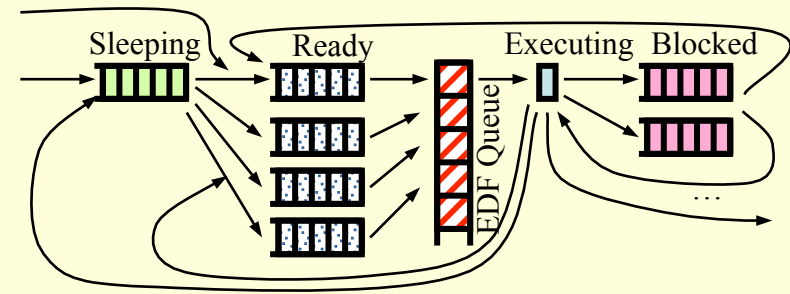
- Implementing priority scheduling

- Tasks, threads and queues
- Building a priority scheduler
 - RM/DM
 - EDF/LST
- Sporadic and aperiodic tasks



- Priority scheduling standards

- POSIX 1003.1b (a.k.a. POSIX.4)
- POSIX 1003.1c (a.k.a. pthreads)
- Implementation details

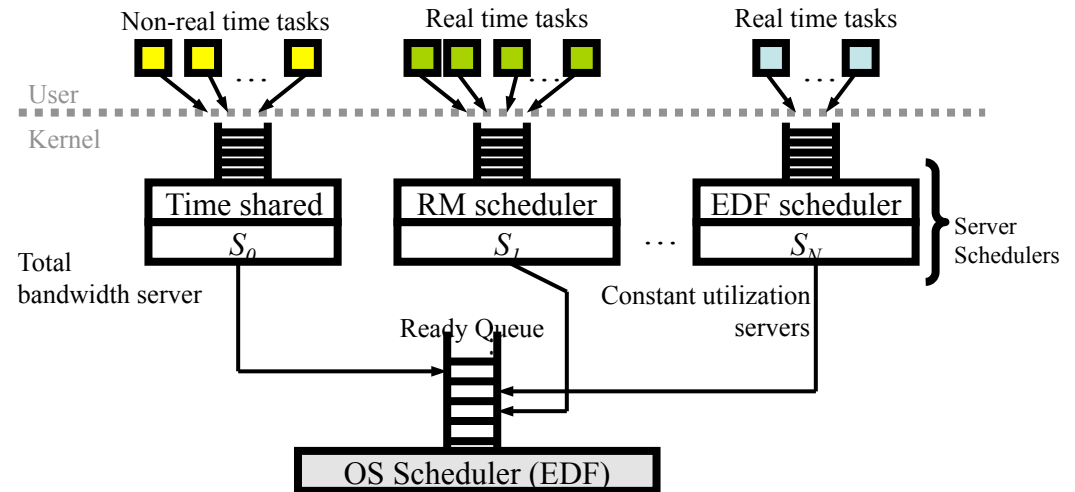


- Use of priority scheduling standards

- Rate monotonic and deadline monotonic scheduling
- User level servers to support aperiodic and sporadic tasks

Two-Level Schedulers

- Real-time application as part of a larger system
 - Open system architecture
- The concept of the two-level scheduler to share system resources
- Isolation of applications
 - Independent design choice
 - Independent validation
 - Independent admission and timing guarantees
- Implementation considerations; schedulability tests
- Case study: RTLinux

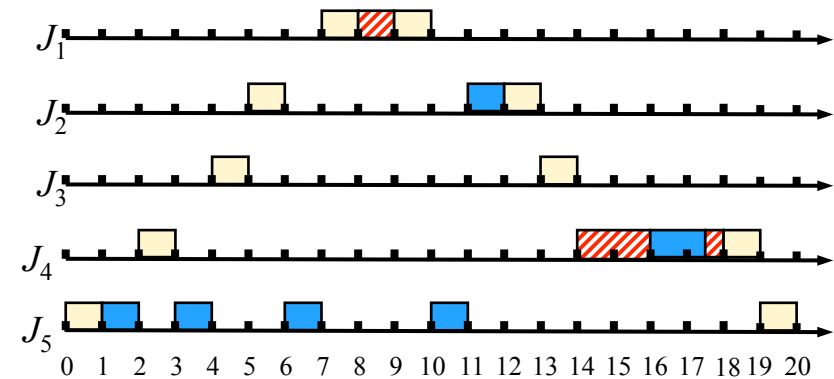


Real-time on General Purpose Systems

- Constraints of running real-time on general purpose platforms
- Desire for flexible applications if system can be overloaded
 - Trade quality for timeliness
 - Given knowledge of current time/deadline, application sheds work
 - Sieve, incremental with milestones, alternate algorithm
 - Very much heuristic driven, rather than explicitly scheduled
 - Inherently imprecise, and difficult to reason about

Resource Access Control

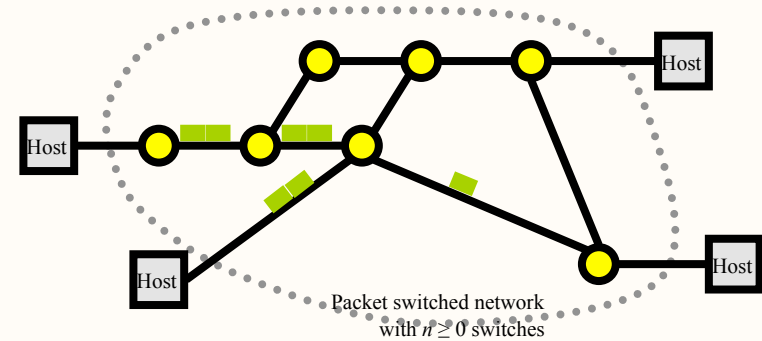
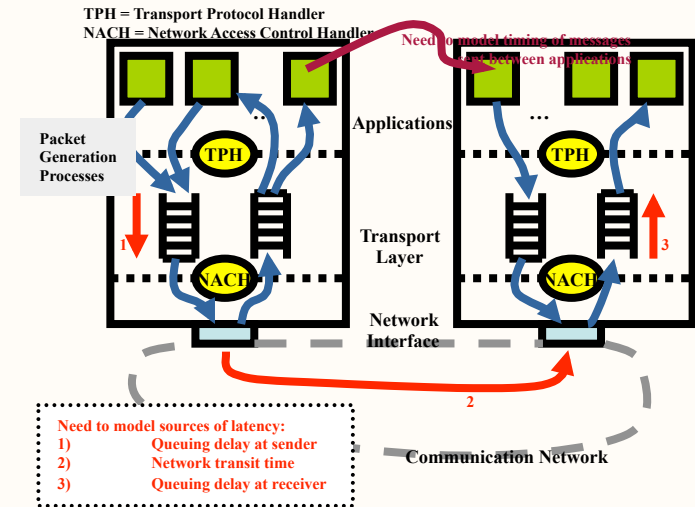
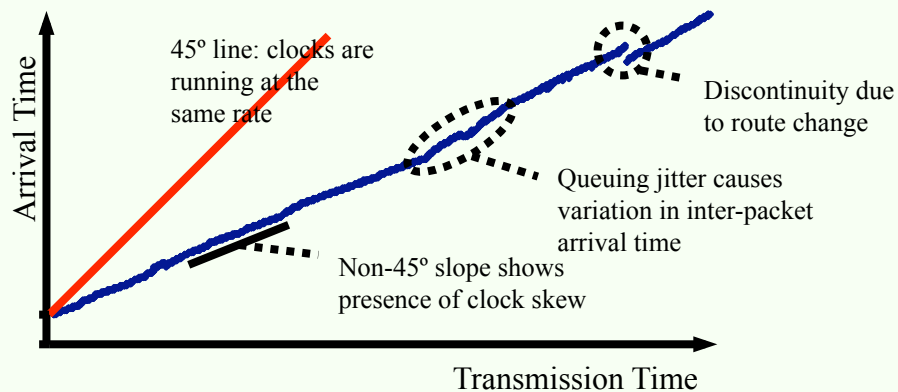
- Definition of system resources
- Contention and conflict for resources
 - Timing anomalies and priority inversion
 - Need for resource access control



- Operation of several resource access control protocols
 - Non-preemptable critical sections
 - Priority inheritance and priority ceiling protocols
 - Performance characteristics: deadlock, blocking times, etc.
 - Performance of different variants of the protocols
 - Resource access control for dynamic priority systems
- Practical methods to implement resource access control
 - Use of POSIX real-time extensions and mutexes for locking, to directly implement the ideas described
 - Other mechanisms: semaphores, message queues, signals, etc.

Introduction to Real-time Communication

- What is real time communication
 - Reference model for hosts and network
- Factors that affect real time communication
 - Throughput, delay and jitter
 - Clock skew
 - Congestion and loss
- Networks and their timing properties
 - Controller area networks vs. Ethernet
 - Concept that some networks provide timing guarantees, others do not

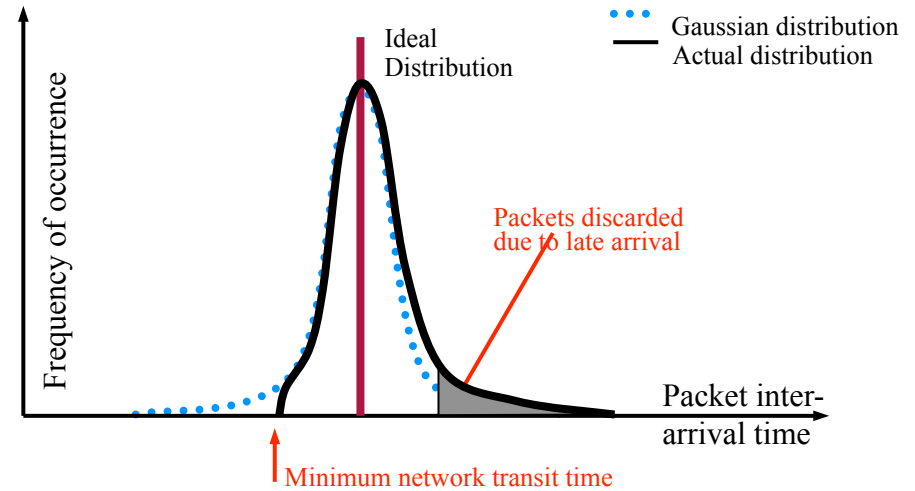


Quality of Service for Packet Networks

- Why enhanced service is needed
- What is needed to support enhanced services
 - Queue discipline
 - Acceptance test
 - Signalling protocol
- Two approaches to implementing priority queuing
 - Weighted Fair Queuing
 - Structure of packet queues; concept of finish number; algorithm operation
 - Control latency and jitter; isolate traffic flows
 - Bounds on per-hop and end-to-end latency for traffic
 - Guaranteed network capacity
 - Weighted Round Robin
 - Structure of packet queues; operation of the algorithm
 - Throughput guarantees and delay bounds
- Concepts of signalling protocols: RSVP

Real-time Communication on IP Networks

- Network timing properties
 - Delay and jitter; clock skew
- Using TCP/IP and UDP/IP for real-time traffic
 - TCP congestion control and its effects on real-time traffic
 - Timing vs. reliability trade-off



- Overview of RTP
 - End-to-end argument; application level framing
 - Buffering for timing recovery
 - Inter-media synchronisation
- Understanding that real-time on IP networks is limited to soft real-time, with flexible applications

Low-Level Embedded Programming

- Hardware influences on embedded systems performance:
 - Interrupt and timer latency
 - Memory issues
 - Memory protection and virtual memory performance
 - Memory allocation, locking, leaks and garbage collection
 - Effects of caches
 - Power, size and performance constraints
 - System longevity
 - Development and debugging
- Consider system issues; features that improve general purpose system may hinder real time work
- Consider constraints on embedded systems, differences in how they are engineered
- Future directions

The End...

- Exam format:
 - 2 hours
 - Answer any three questions out of four
 - All material in book, lectures notes and handouts examinable
 - Sample and past papers on Moodle

- Any questions?