Introduction to Real-Time Systems

Real-Time and Embedded Systems (M)

Lecture 1

UNIVERSITY
of
GLASGOW

Lecture Outline

- Administrivia
  - Aims and objectives
  - Intended learning outcomes
  - Prerequisites
  - Module outline and timetable
  - Reading list
  - Assessment
- Introduction to real-time systems
  - Examples
  - Types of real-time system
  - Implementation considerations

Copyright © 2007 University of Glasgow
All rights reserved.

Lecturer Contact Details

Dr. Colin Perkins
Lilybank Gardens, room S154, but moving to the Sir Alwyn Williams Building, room 405

Email: csp@dcs.gla.ac.uk

Willing to discuss the module and answer questions – *make appointments by email*

http://www.dcs.gla.ac.uk/~csp/teaching/rtes/

Copyright © 2007 University of Glasgow
All rights reserved.

Aims of This Module

- 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.

Copyright © 2007 University of Glasgow
All rights reserved.

Intended Learning Outcomes

- By the end of this module 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.

Copyright © 2007 University of Glasgow
All rights reserved.

Prerequisites

- Students are expected to have done degree-level studies in, and be familiar with, operating systems design and implementation, concurrency and threaded programming, and software analysis and design.

- Some basic familiarity with formal reactive systems modelling techniques and safety critical system design would complement the engineering issues addressed in this module
  - The MRS4 and SCS4 modules cover that material, but are not formal co- or pre-requisites

Copyright © 2007 University of Glasgow
All rights reserved.

Module Outline

- Introduction to Real-Time and Embedded Systems
  - Reference Model
  - Hard versus soft real-time
- Job Scheduling
  - Clock driven scheduling algorithms
  - Priority driven scheduling algorithms
  - Schedulers in commodity and real-time operating systems
- Resource access control
  - Algorithms
  - Implementation
- Real-time communication
  - On best-effort networks
  - Enhanced quality of service
- Other implementation considerations

Copyright © 2007 University of Glasgow
All rights reserved.

Timetable

Week starting:	Tue 14:00-15:00	Wed 10:00-11:00	Thu 10:00-11:00
7 Jan	Lecture 1	Lecture 2	Lecture 3
14 Jan	Tutorial 1	Lecture 4	Lecture 5
21 Jan	Lecture 6	Tutorial 2	Lecture 7
28 Jan	Lecture 8	Tutorial 3	Lecture 9
4 Feb	Lecture 10	Lecture 11	Lecture 12
11 Feb	Tutorial 4	Lecture 13	Lecture 14
18 Feb	Tutorial 5	Lecture 15	Lecture 16
25 Feb	Individual work on programming assignment		
3 Mar	Lecture 17	Tutorial 6	Lecture 18
10 Mar	Lecture 19	Lecture 20	

Copyright © 2007 University of Glasgow
All rights reserved.

Timetable

Week 1	Lecture 1	Introduction to Real-Time Systems
	Lecture 2	A Reference Model for Real-Time Systems
	Lecture 3	Overview of Real-Time Scheduling
Week 2	Tutorial 1	The Basics of Real-Time Systems
	Lecture 4	Clock-Driven Scheduling
	Lecture 5	Priority-driven Scheduling of Periodic Tasks (1)
Week 3	Lecture 6	Priority-driven Scheduling of Periodic Tasks (2)
	Tutorial 2	Scheduling Algorithms (1)
	Lecture 7	Priority-driven Scheduling of Aperiodic and Sporadic Tasks (1)
Week 4	Lecture 8	Priority-driven Scheduling of Aperiodic and Sporadic Tasks (2)
	Tutorial 3	Scheduling Algorithms (2)
	Lecture 9	Implementing Scheduling Algorithms
Week 5	Lecture 10	Real-Time Operating Systems and Languages (1)
	Lecture 11	Real-Time Operating Systems and Languages (2)
	Lecture 12	Real-Time on General Purpose Systems

Copyright © 2007 University of Glasgow
All rights reserved.

Timetable

Week 6	Tutorial 4	Real-Time Operating Systems and Languages
	Lecture 13	Resource Access Control (1)
	Lecture 14	Resource Access Control (2)
Week 7	Tutorial 5	Resource Access Control
	Lecture 15	Introduction to Real-Time Communications
	Lecture 16	Real-Time Communication on IP Networks
Week 8	Individual Work on Programming Assignment	
Week 9	Lecture 17	Quality of Service for Packet Networks
	Tutorial 6	Real-Time Communications/Q&A on Programming Assignment
	Lecture 18	Low-Level and Embedded Programming (1)
Week 10	Lecture 19	Low-Level and Embedded Programming (2)
	Lecture 20	Review of Major Concepts

Copyright © 2007 University of Glasgow
All rights reserved.

Lectures and Tutorials

- Lectures:
  - Highlight relevant material from the book

- Tutorials:
  - Practice problem solving, review material covered in lectures
  - Expect to do worked examples and answer questions!

Copyright © 2007 University of Glasgow
All rights reserved.

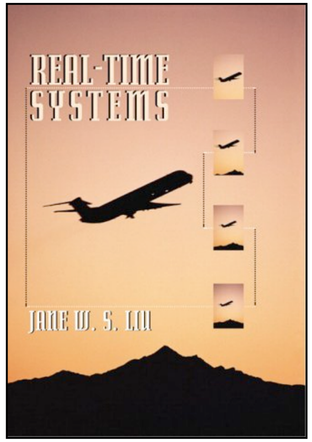
Assessment

- Level M module, worth 10 credits
- 3 Problem sets (each worth 5% of total mark)
  - Problem set 1: available in lecture 3, due 2:00pm on 21st January
  - Problem set 2: available in lecture 6, due 2:00pm on 28th January
  - Problem set 3: available in lecture 8, due 2:00pm on 4th February
    - Hard deadlines: late submissions will receive zero marks unless valid special circumstances form submitted
- Programming assignment (15% of total mark)
  - Available in lecture 15, due at 5pm on 14th March
  - Will involve real-time network programming in C
- Written examination (70% of total mark)
  - All material in the lectures, tutorials and background reading is examinable
  - Aim is to test your understanding of the material, not to test your memory of all the details; explain why – don't recite what

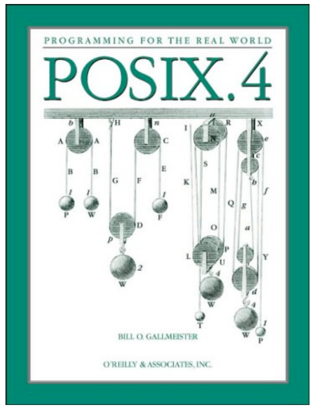
Copyright © 2007 University of Glasgow
All rights reserved.

Reading

- Jane W. S. Liu, “Real-Time Systems”, Prentice-Hall, 2000, ISBN 0130996513
  - This book comprises the lecture notes for the module and is *required reading* for all students
  - *All material in this book is examinable*



- Bill Gallmeister, “POSIX.4: Programming for the Real-World”, O’Reilly and Associates, 1995, ISBN 1565920740
  - *Optional*, but provides further detail on the practical aspects of the module



Copyright © 2007 University of Glasgow
All rights reserved.

Real-Time and Embedded Systems

- A *real-time* system must deliver services in a timely manner
  - *Not* necessarily fast, but must meet some timing deadline
- An *embedded* system is hidden from view within a larger system
- Many real-time and embedded systems exist, often without the awareness of their users
  - Washing machine, photocopier, mobile phone, car, aircraft, industrial plant, microwave oven, toothbrush, CD player, medical devices, etc.

- Must be able to validate real-time systems for correctness
  - Some embedded real-time systems are safety critical – i.e. if they do not complete on a timely basis, serious consequences result
  - Bugs in embedded real-time systems are often difficult or expensive to fix

Copyright © 2007 University of Glasgow
All rights reserved.

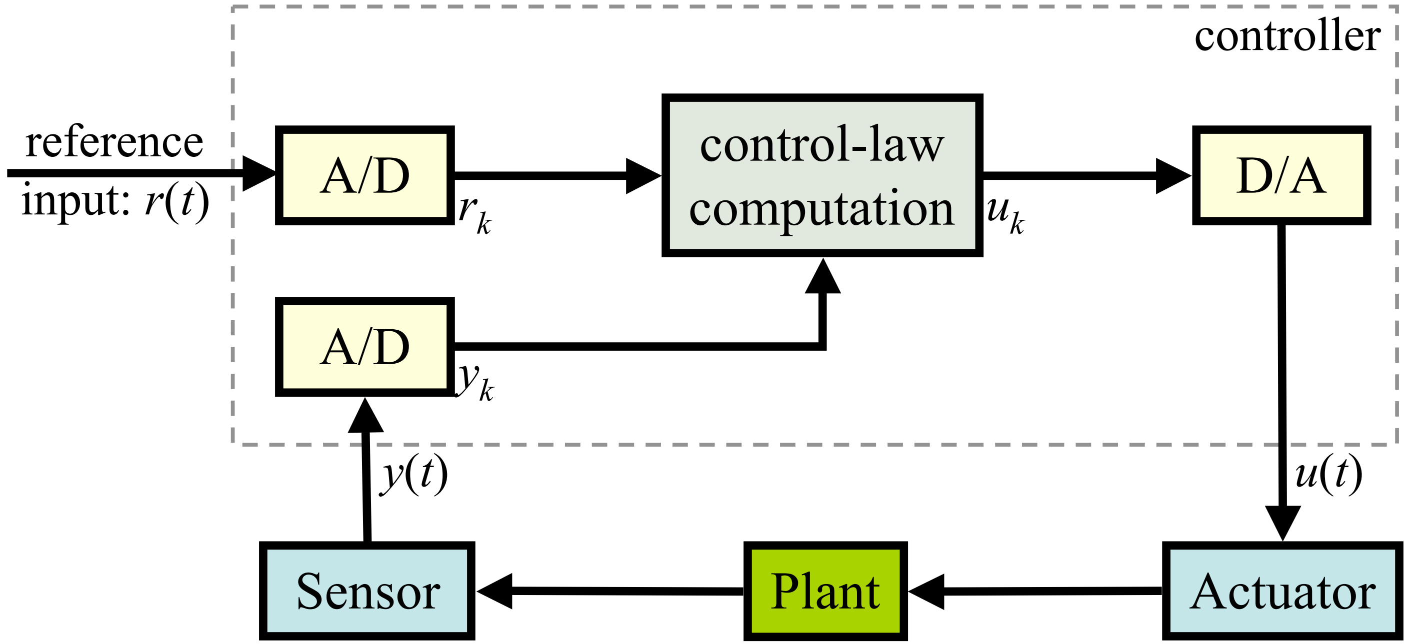
Real-Time and Embedded Systems

- This module will discuss several representative classes of real-time and embedded system:
  - Digital process control
  - Higher-level command and control
  - Tracking and signal processing
  - Real-time databases
  - Telephony and multimedia
- Algorithms for scheduling tasks such that those systems complete in a reliable and timely fashion
- Implementation techniques, operating systems and languages for building such systems

Copyright © 2007 University of Glasgow
All rights reserved.

Digital Process Control

- Controlling some device (the "plant") using an actuator, based on sampled sensor data
  - $y(t)$ is the measured state of the plant
  - $r(t)$ is the desired state of the plant
  - Calculate control output $u(t)$ as a function of $y(t)$, $r(t)$



Copyright © 2007 University of Glasgow
All rights reserved.

Digital Process Control

- Pseudo-code for the controller:

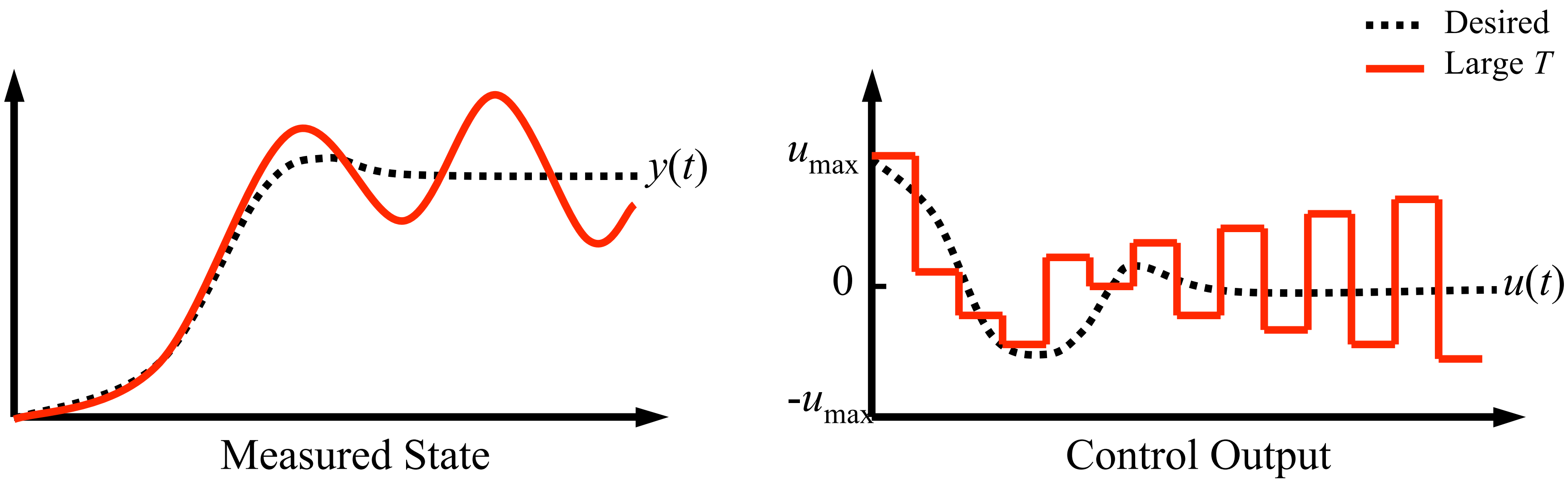
```
set timer to interrupt periodically with period T;
at each timer interrupt, do
    do analogue-to-digital conversion of y(t) to get y_k;
    compute control output u_k based on reference r_k and y_k;
    do digital-to-analogue conversion of u_k to get u(t);
end do;
```

- Effective control of the plant depends on:
  - The correct control law computation and reference input
  - The accuracy of the sensor measurements:
    - Resolution of the sampled data (i.e. bits per sample)
    - Timing of the clock interrupts (i.e. samples per second, $^1/_T$)

Copyright © 2007 University of Glasgow
All rights reserved.

Digital Process Control

- The time T between any two consecutive measurement of $y(t)$, $r(t)$ is the *sampling period*
  - Small T better approximates the analogue behaviour
  - Large T means less processor-time demands
  - Must achieve a compromise
- If T is too large, oscillation will result as the system tries to adapt



Copyright © 2007 University of Glasgow
All rights reserved.

Digital Process Control

- How to choose sampling period?
  - *Rise time* – the amount of time that the plant takes to reach some small neighbourhood around the final state in response to a step change in the reference input
  - If *R* is the rise time, and *T* is the period, a good rule of thumb is that the ratio $10 \leq {}^{R}/_{T} \leq 20$
- Must be chosen correctly, and accurately implemented to ensure stability

- Multi-rate systems – system is composed of multiple sensors and actuators, each of which require different sampling periods
  - Need to run multiple control loops at once, accurately
  - Usually best to have the sampling periods for the different degrees of freedom related in a harmonic way

Copyright © 2007 University of Glasgow
All rights reserved.

Example: Helicopter Flight Control

Do the following in each 1/180-second cycle:

- Validate sensor data and select data source; on failure reconfigure the system
- Do the following 30-Hz avionics tasks, each once every 6 cycles:
  - Keyboard input and mode selection
  - Data normalization and coordinate transformation
  - Tracking reference update
- Do the following 30-Hz computations, each once every 6 cycles
  - Control laws of the outer pitch-control loop
  - Control laws of the outer roll-control loop
  - Control laws of the outer yaw- and collective-control loop
- Do each of the following 90-Hz computations once every 2 cycles, using outputs produced by the 30-Hz computations
  - Control laws of the inner pitch-control loop
  - Control laws of the inner roll- and collective-control loop
- Compute the control laws of the inner yaw-control loop, using outputs from the 90-Hz computations
- Output commands to control surfaces
- Carry out built-in-test

Copyright © 2007 University of Glasgow
All rights reserved.

Digital Process Control

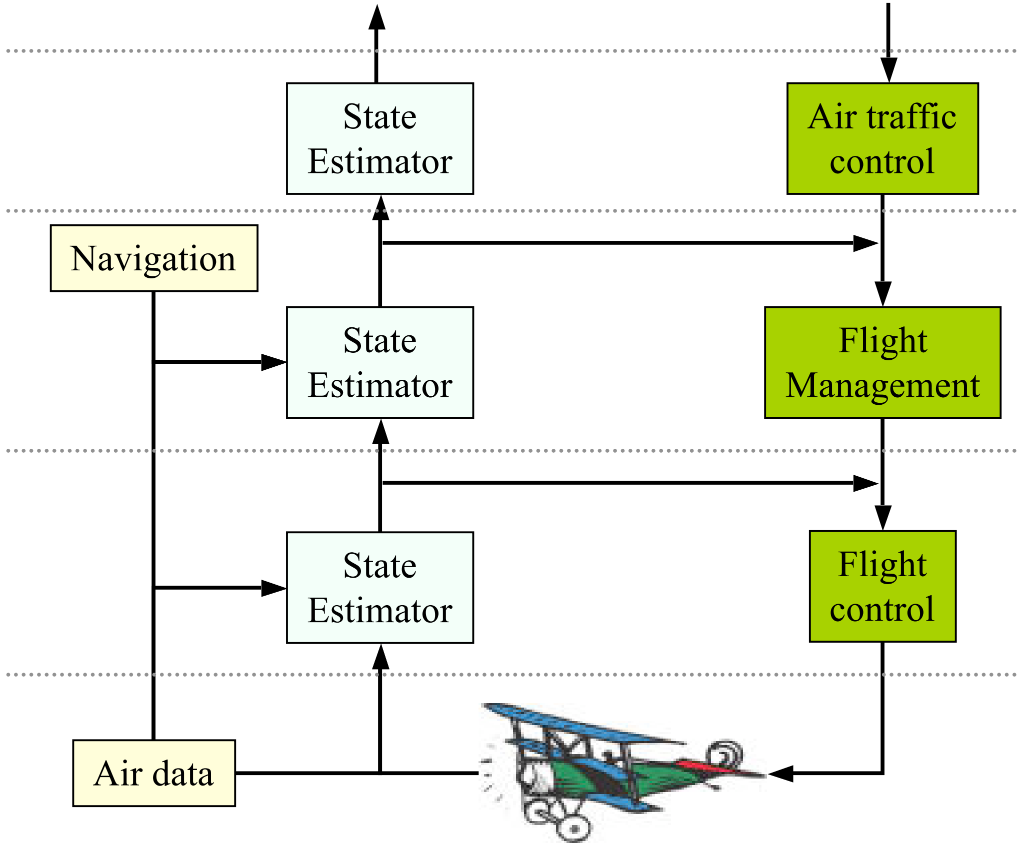
- Digital controllers make three assumptions:
  - Sensor data give accurate estimates of the state-variables being monitored and controlled - noiseless
  - The sensor data gives the state of the plant – usually must compute plant state from measured values
  - All parameters representing the dynamics of the plant are known
- If any of these assumptions are not valid, a digital controller must include a model of the correct system behaviour
  - Estimate actual state based on noisy measurement each iteration of the control loop
  - Use estimated plant state instead of measured state to derive control output
  - Often requires complex calculation, modelling

- We'll cover scheduling dynamics; the system model is domain-specific

Copyright © 2007 University of Glasgow
All rights reserved.

Higher-Level Control

- Controllers often organized in a hierarchy
  - Multiple control loops, higher level controllers monitoring the behaviour of low-level controllers
  - Time-scale, complexity of decision making, increases as go up hierarchy; Move from control to planning
  - Higher level planning must still be done in real-time, although deadlines are less tight



Copyright © 2007 University of Glasgow
All rights reserved.

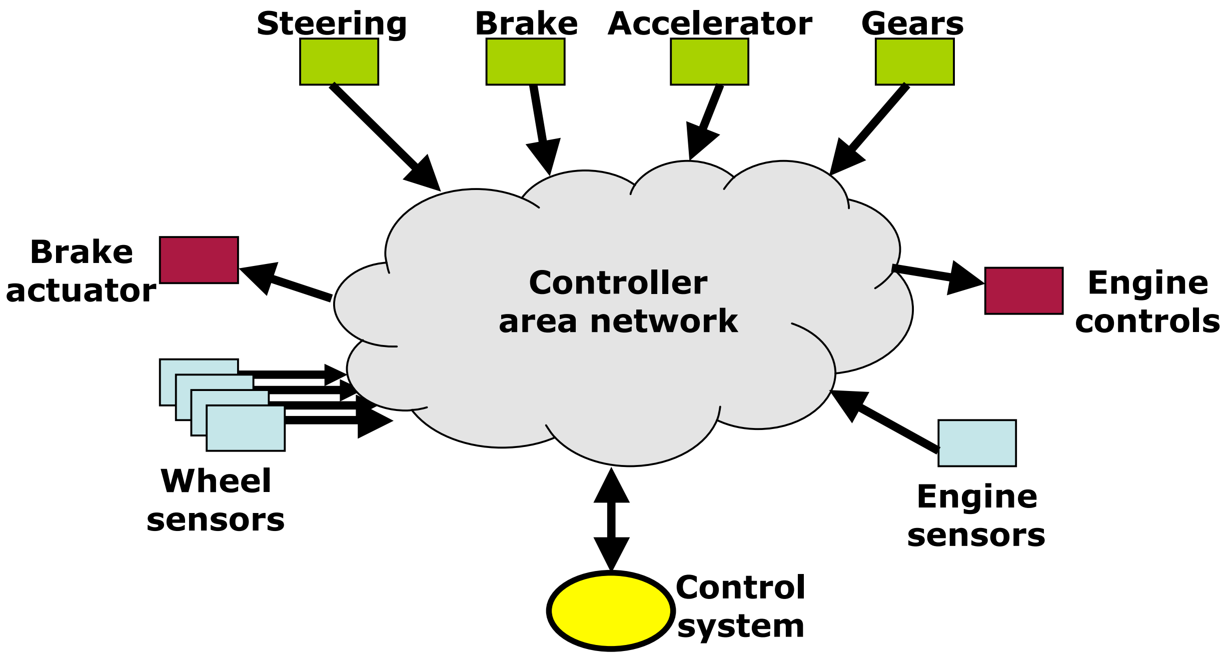
Real-Time Communications

- Real-time systems are increasingly distributed, including communication networks
  - Control loop may include a communication step
  - System may depend on network stimuli

- Not only does a system need to run a control law with time constraints, it must also schedule communications, sending and receiving messages according to deadlines

Copyright © 2007 University of Glasgow
All rights reserved.

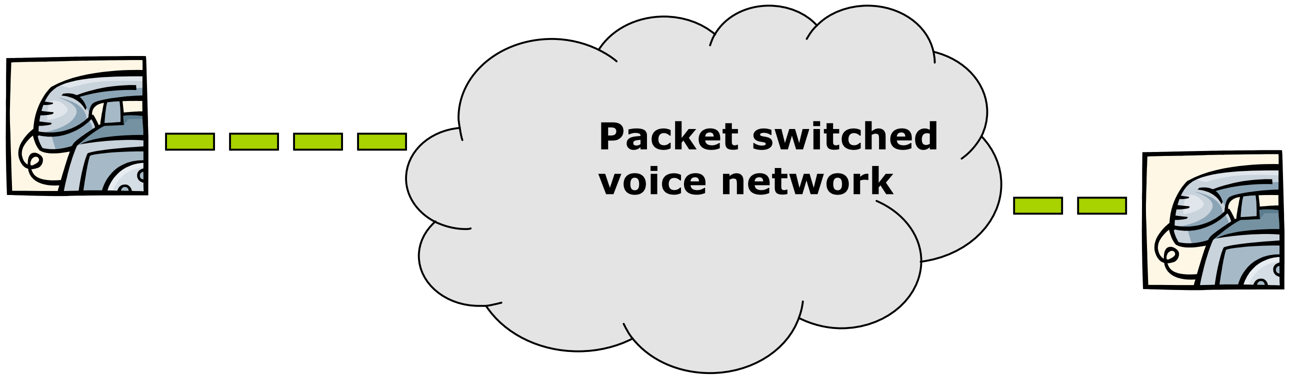
Example: Drive by Wire



- All data must be delivered reliably
  - Bad if you turn the steering wheel, and nothing happens
- Commands from control system have highest priority, then sensors and actuators, then control inputs
  - Anti-lock brakes have a faster response time than the driver, so prioritise to ensure the car doesn't skid
- Network must schedule and prioritise communications

Copyright © 2007 University of Glasgow
All rights reserved.

Example: Packet Voice



- Voice is digitised and sent as a sequence of packets
  - Constant spacing, every 10-30ms depending on codec
- Strict timeliness requirement
  - Mouth to ear delay needs to be less than approximately 150ms
  - Packets must be played out with equal spacing
- Relaxed reliability requirement
  - Some small fraction of packets can be lost, and just sound like crackles on the wire; most need to arrive
- Emergency calls may have priority

Copyright © 2007 University of Glasgow
All rights reserved.

Types of Real-Time Application

- Purely cyclic
  - Every task executes periodically
  - Demands in (computing, communication, and storage) resources do not vary significantly from period to period
  - Example: most digital controllers and real-time monitors
- Mostly cyclic
  - Most tasks execute periodically
  - The system must also respond to some external events (fault recovery and external commands) asynchronously
  - Example: modern avionics and process control systems
- Asynchronous: mostly predictable
  - Most tasks are not periodic
  - The time between consecutive executions of a task may vary considerably, or the variations in resource utilization in different periods may be large
  - These variations have either bounded ranges or known statistics
- Asynchronous: unpredictable
  - Applications that react to asynchronous events and have tasks with high run-time complexity
  - Example: intelligent real-time control systems

Copyright © 2007 University of Glasgow
All rights reserved.

Types of Real-Time Application

- As we will see later, the type of application affects how we schedule tasks, prove correctness
- It is easier to reason about applications that are more cyclic, synchronous and predictable
  - Many real-time systems designed in this manner
  - Safe, conservative, design approach, if it works for your application

Copyright © 2007 University of Glasgow
All rights reserved.

Implementation Considerations

- Some real-time embedded systems are complex, implemented on high-performance hardware
  - Industrial plant control
  - Civilian flight control
- Many must be implemented on hardware chosen to be low cost, low power, light-weight and robust; with performance a distant concern
  - Military flight control, space craft control
  - Consumer goods
- Often-times implemented in C or assembler, fitting within a few kilobytes of memory
  - Correctness a primary concern, efficiency a close second

Copyright © 2007 University of Glasgow
All rights reserved.

Summary

- Outline of the module structure, assessment, etc.
- Introduction to real-time and embedded systems
  - Examples of digital control, higher-level control, communication
- Types of real-time system
  - Cyclic synchronous vs. asynchronous and unpredictable
- Implementation considerations

[Liu chapter 1]

Copyright © 2007 University of Glasgow
All rights reserved.