Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom

TELEPHONE: **Cambridge (01223) 515010**
INTERNATIONAL: **+44 1223 515010**
FAX: **+44 1223 359779**
E-MAIL: **apm@ansa.co.uk**

Training

ANSAwise - Real-time Distributed Systems

Chris Mayers

Abstract

Organizations often have a mix of real-time and non-real systems; new applications may need to integrate facilities from both.

This module of the ANSAwise training programme describes typical features of real-time systems, explains how they are supported by real-time Unix, and outlines the issues when extending into distributed real-time systems.

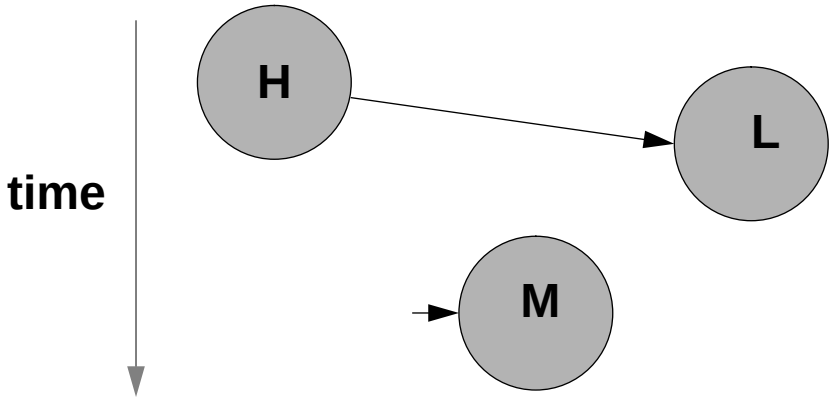
[Note that this module is carefully written to avoid ANSA Phase 3 confidentiality restrictions. A later revision may be able to be more explicit about technical matters.]

APM.1353.01 **Approved** 25th November 1994

Briefing Note

Distribution:
Supersedes:
Superseded by:

Real-time Distributed Systems



In this session

- *Identify the characteristic properties of real-time systems*
- *Show how these properties affect the design of distributed systems*
  - distributed systems with real-time properties
- *Examine the strengths and limitations of programming with priority*

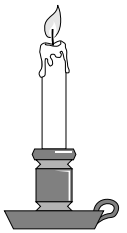
Timeliness

- ***A real-time system is one in which timing is explicitly considered***
  - **in requirements**
  - **in specification**
  - **in design**
  - **in implementation**
- ***Real-time systems require timely responses to events***
  - **even under failure conditions**
  - **even under extreme load conditions**

The nature of real-time systems

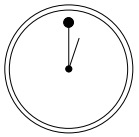
- *Distinctions are usually made between*
  - non-real-time systems
  - soft real-time systems
  - hard real-time systems
- *The distinctions are not rigid, but are realistic*
  - the different trade-offs currently result in very different system designs

Non-real-time systems



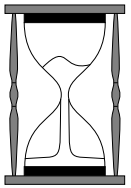
- *Have no timing requirement*
  - best-effort is all that is offered
- *Timings are not monitored*

Soft real-time systems



- *Typically have an "average case" timing requirement*
  - for example, 99% of requests to take less than 1 second
- *Failure to meet the timing requirement is not critical*
  - a late response to a request is bad, but may still be useful
- *Timing may be monitored*
  - consistent failure to meet a timing requirement may require human or automatic intervention
- *Often based on priority schemes*

Hard real-time systems



- *Typically have a “worst case” requirement*
  - for example, all requests must be completed in less than 1 second
- *Failure to meet timing requirement is catastrophic*
  - a late response is useless or even dangerous
- *Timing must be monitored*
- *Failure to meet a deadline requires automated handling*
- *Based on deadline schemes, usually using priority as well*

Misconceptions about real-time systems

- ***Real-time Means Fast***
  - ***NO*: real-time means predictable timing/*'fast enough'*...**
  - **... acceptable response times in seconds are not unusual**
- ***Real-time Means Assembler***
  - ***NO*: selective programming in assembler may be necessary for performance**
  - **... but it doesn't guarantee predictable timing**
- ***Faster Hardware Solves All Real-time Problems***
  - ***NO*: faster hardware may make new problems solvable**
  - **... but it doesn't guarantee predictable timing, either**

More misconceptions

- ***Real-time is Art, not Science***
  - **It may have been in the past...**
  - **... this is no longer so**
- ***Real-time Problems Are Solved By Conventional Techniques***
  - **The nature of timing constraints poses special challenges for systems designers...**
  - **... challenges we can handle**

Resource reservation is a key facility

- ***Reservation of resources (pre-allocation) is vital to provide guarantees...***
  - **just as in real life generally**
- ***...but computers often try to share resources***
  - **by multiplexing**
  - **by pooling**
  - **by time-sharing**
- ***It must be possible to selectively reserve resources***

Resources are needed immediately

- ***Real-time means that resources must be delivered immediately they are required***
  - **Waiting Is Evil!**
- ***Resources include***
  - **CPU**
  - **Memory**
  - **I/O bandwidth**

Resource reservation in distributed systems

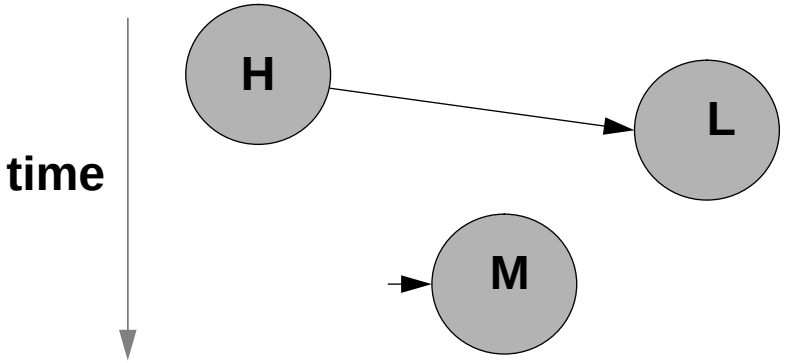
- *In distributed systems, the resources must be reserved end-to-end*
  - in the end systems
  - throughout the network
- *This means end-to-end quality-of-service (QoS) guarantees*

Programming with priorities

- *Every multi-tasking operating system offers 'task priorities'...*
  - ... but exactly what do they mean?
- *Many such systems evolved from time-sharing environments*
  - offering round-robin scheduling (task quanta) for tasks of equal priority
  - ... often with automatic adjustment of task priority
- *Real-time systems require strict priority*
  - highest priority first
  - no round-robin scheduling
- *Some systems offering 'real-time' priorities give you only this*
  - but strict priority is not enough

The priority inversion phenomenon

- *Suppose we have a system with three priority levels*
  - high, medium, and low



- *High-priority process synchronizes with low-priority process*
  - ... meanwhile, it can be pre-empted by a medium-priority process
  - ... this is not what is wanted

Solutions to priority inversion

- ***The scheduler's protocol must prevent priority inversions, typically using***
  - **PIP (Priority Inheritance Protocol)**
  - **PCP (Priority Ceiling Protocol)**
- ***The POSIX 1003.4 standard requires PIP***
  - **with PCP as an option**
- ***You'll need a scheduler that prevents priority inversions***
  - **you can buy such systems now**

Priority is not a panacea

- ***Priority is only a measure of importance***
  - **it is not a measure of urgency**
  - **... it does not give you deadline guarantees**
  - **... which are necessary for hard real-time systems**
- ***Separate mechanisms are needed to support importance and urgency***
- ***Priority does not directly help meet the average-case guarantee for soft real-time systems either***
  - **it's not clear how to allocate priorities at design time**
  - **... it's often done by tuning the priorities by hand in the final system**
  - **... this does not assure confidence in the system**

Priority is useful

- ***Well supported***
- ***Well accepted***
- ***Well understood***

Priority in real-time distributed systems

- *The same issues arise in a remote procedure call*
  - strict priority
  - prevention of priority inversion
- *This means that*
  - both end systems must support this
  - the RPC must convey the priority information between the systems
- *This is done with a modification to the RPC mechanism*
  - making it a Timed RPC

Market pressures on real-time operating systems

- ***Real-time has been a profitable market, but a niche market***
  - **it used to be occupied solely by proprietary solutions**
- ***Like other computing markets it is now being driven by***
  - **time-to-market**
  - **software rather than hardware solutions**
  - **value-added integration of standard components**
- ***...The market is converging on Real-time Unix with distribution***

Real-time Unix convergence

- ***Proprietary real-time executives***
  - **bottom-up**
- ***'Open' Unix***
  - **top-down**
- ***Distributed systems***
  - **integrated distributed communications**

Problems with traditional Unix for real-time

- ***Process scheduling***
  - no guarantee that the highest-priority process is scheduled first
  - processes cannot change the priorities of other processes
- ***Coarse timing***
  - as a rule of thumb, timer resolution should be 100 times greater than event periods
  - ... to avoid quantization error
  - ... but Unix offers only 1/100ths of a second

More problems with Unix

- *Memory management*
  - cannot fix a process into real memory
  - ... also need the flexibility to fix part of a process into real memory
- *No user-level synchronization*
- *Priority in inter-process communications*
  - priority inversions can’t be avoided
- *Scheduling latency*
  - non-preemptible kernel
  - interrupts disabled
  - queuing delays (need associated process priority)

Yet more Unix problems

- ***Intolerable overhead for some functions***
  - **bulk data copying between user and kernel address spaces**
- ***Incomplete implementations***
  - **functions simply not provided**

Re-engineered Unix

- ***Based on micro-kernel technology***
  - **more modular and flexible**
- ***Preemptible kernel***
  - **low interrupt latency**
  - **global internal locks removed**

Real-time Unix solves many problems

- *The POSIX 1003.4/4a Real-time extensions meet many of the needs*
  - and convergence is beginning to deliver usable products
- *There are still some gaps*
  - no support for deadlines for hard real-time systems
  - no generalized support for resource reservation

The wider picture - contributory technologies

Open Systems

Generic IPC
RPC
Client / Server
Transparency tools
ANSA
ODP
CORBA

Real-time OS
Microkernel
Real-time scheduling
Real-time comms
Mission Orientation
ATM (B-ISDN)
Multimedia

Real-time Systems

Object Models
OO programming
OO data bases
Object Management
OMA

Object-Oriented Systems

Real-time Reaches the Desktop

- ***Current and future applications will have real-time requirements***
  - **extending to the desktop**
  - **via broad-band WANs**
- ***It must be possible to have real-time and non-real-time objects in the same system***

Related needs and issues

- *Typical related needs are*
  - fault tolerance
  - automated recovery
  - load balancing
- *No programming language provides adequate support for real-time*
  - this will have to be provided by auxiliary tools

General principles

- *Design in real-time capability from the start*
- *Do not change the problem to fit the hardware*
- *Partition functions between objects with real-time aims in mind*
- *Specify testability interfaces*
- *Construct from general components into special configurations*

Summary

- ***Decide whether your needs can be met by specific systems***
  - **do not be fooled by features simply labelled as ‘real-time’**
  - **beware of systems that claim to offer simply ‘real-time’ priorities**
- ***Distributed real-time is a topic of the ANSA Phase 3 programme***
  - **ANSAware/RT supported over OSF/1, LynxOS, HP-UX/RT**
- ***For more on this topic***
  - **see *Misconceptions about Real-Time Computing* by John A. Stankovic (IEEE Computer Oct 1988)**