Summer Institute on Software Architecture

Embedded Systems Architecture 1: Modern Embedded Systems

Instructor: Calton Pu

calton.pu@cc.gatech.edu

Georgia Tech

Overall Structure (Day 1)

- Introduction to modern embedded systems
  - Ubiquitous computing as a vision for integrating future embedded systems
  - From embedded to resource constrained systems
  - Some basic techniques for constructing real-time embedded system software
- Principled embedded software infrastructure
  - Survey of real-time scheduling algorithms: static, dynamic priority, static priority dynamic
  - I/O processing and networking for embedded systems

Georgia Tech

Overall Structure (Day 2)

- Automotive embedded software architecture
  - Component-based software engineering
  - Case study on automotive embedded software
- Sampling of methodical optimization of embedded software
  - Specialization of system software
  - Code generation and translation
  - Aspect-oriented programming

Part 1: Day 1 morning

- Introduction
  - Ubiquitous computing as a vision for integrating future embedded systems
  - From embedded to resource constrained (real-time embedded) systems
  - Survey of principled RTES construction techniques

Embedded and Ubiquitous

- Ubiquitous Computing as vision
  - Proposed by Mark Weiser [CACM 1993]
  - At the time, a new form of computer science
  - Hardware Issues
- Applications
- Where are we now?

UbiComp in 1993

- Very new to the field of CS
  - Xerox PARC has been working on Ubiquitous Computing since early 90's
- Main goal was (and still is) to get the computers "out of the way of everyday activities"
- Technology finally caught up to the proposed ideas for "environmental computing"

UbiComp in 1993 (Cont.)

- Some thought Virtual Reality was the ideal UbiComp solution, but the technology was not advanced enough
- Ruled out GUIs as the complete solution
- Identified several key needs of a successful UbiComp device
- Still struggling with some of the same problems today

Georgia Tech

Phases of Development

- Researchers at Xerox PARC identified the initial set of ubiquitous computing "phases"
  - Construct
  - Deploy
  - Evaluate
- Realized that Phase One would not achieve the "optimal invisibility"

Georgia Tech

Potential Platforms

- Devices of various sizes
- Enough diversity to give some sense of scope
- Must be found in everyday life and used frequently
- Above all, they must be unobtrusive

Large-Size Prototype

- LiveBoard! (Look to the right!)
- Main idea was to simulate an office whiteboard
- Order of 1 per office

Medium-Size Prototype

- XPad
- Main goal was to simulate a personal notebook
- Order of 10+ per person

Small-Size Prototype

- ParcTab
- Main goal was to simulate PostIts
- Order of 100+ per person

New Form of CS

- Valuable lessons learned from the early prototypes
- Development of a new hierarchical abstraction specific to UbiComp framework
- Main goal of this paper is to discuss the motivations behind this new form of CS and the current obstacles

New Hierarchical Abstraction

Hardware

↓

Network Protocols

↓

Interaction Substrates

↓

Applications

Hardware Requirements

- Low Power
  - Speed can be sacrificed
- Wireless
  - One low-speed (64kbps) per person
  - Remember this is 1993
- Pens
  - Wireless (IR beams)
  - Available without touching the screen and up to several feet away

Georgia Tech

Network Protocols

- IP was not the proper protocol because it assumed a static location of the computer
- A "media access" protocol is required
- Some applications require guaranteed bandwidth (voice and video)
- Example – MACA [Karn 90]
  - Uses a handshake algorithm that verifies communication channel and lets others know of upcoming transmission

Georgia Tech

Network Protocols (Cont.)

- Real-Time Protocols
  - Focus on packet-switched networks
  - Attempt to eliminate bottlenecks at basestations
  - Work in progress at the time (no concrete details are provided)
- "Secondary" or "Virtual" IP
  - Adds a level of indirection to account for user mobility

Georgia Tech

Interaction Substrates

- IR Pens
- "No look" touch screens
- Palm size keyboard
  - Found to be only half as fast
- Window migration tools
- "Low Bandwidth X" [Fulton 93]

Georgia Tech

Early Applications

- Active Badge
  - An employee tracker
  - AT&T Labs in Cambridge
- Slate
  - Shared media tool
  - Xerox PARC
- Both widely used even outside of the labs

Georgia Tech

New Theoretical Problems

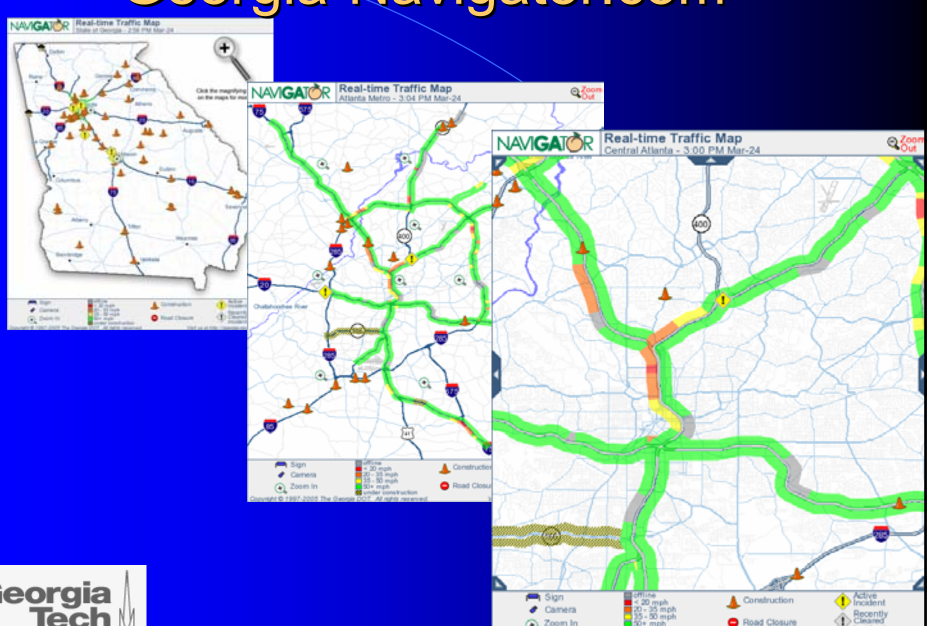
- UbiComp has unveiled several new theoretical problems that need to be solved. For example:
  - Optimal Cache Sharing Problem
    - Optimal strategy for partitioning memory between compressed and uncompressed pages
    - Led to the development of the Lower Bound Theorem for Caches [Bern 93]

Georgia Tech

Where Are We Now?

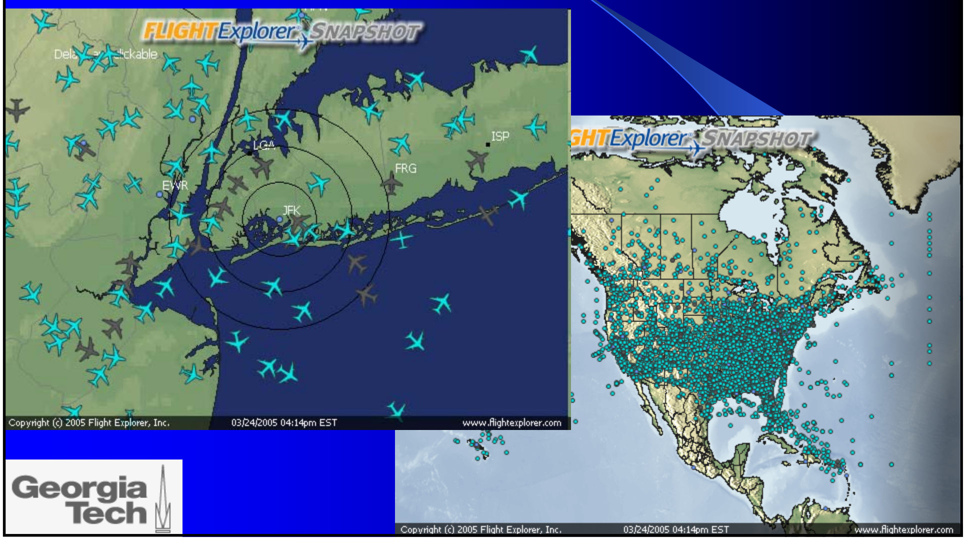
- Still developing new technologies
- Have met the demands for:
  - Wireless Networking (IEEE 802.11)
  - Low Power CPUs (300+ MHz at 1.1v)
  - Real-Time Packet Switching (Numerous algorithms)
  - Applications (Entire OSs have been built)

Georgia-Navigator.com

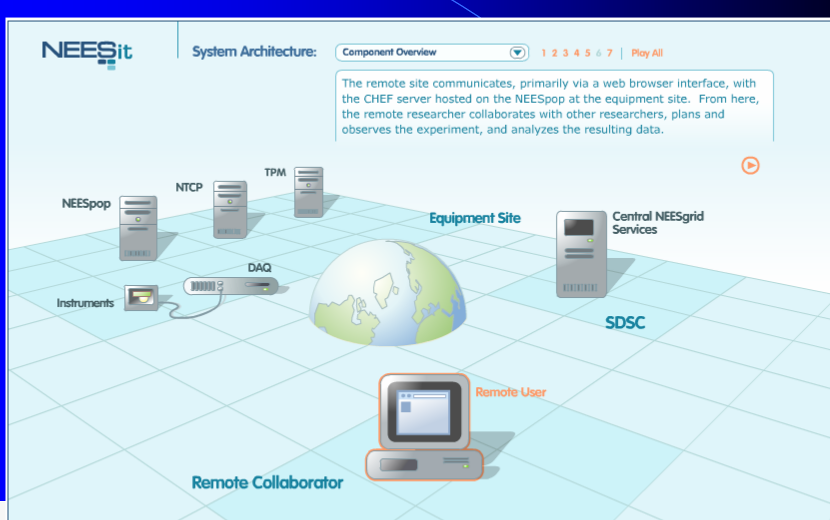


Real-Time Air Traffic

- www.flightexplorer.com

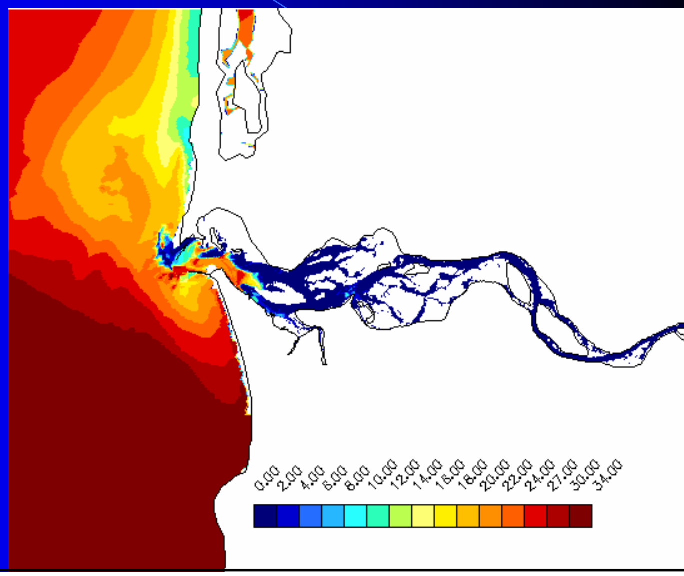


Remote Experiment Control



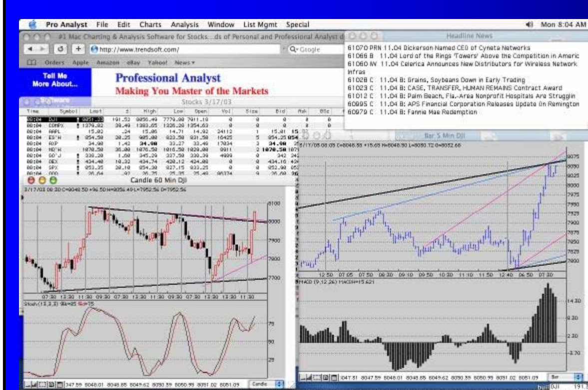
Environmental Forecasting

- Columbia River monitoring and forecasting



Georgia Tech

Financial Application



- Analyze data in real-time
- Respond to market developments as they occur
- Strong visualization
- (Screenshot from TrendSoft ProAnalyst)

- http://www.trendsoft.com/ProAnalyst/main.htm

Georgia Tech

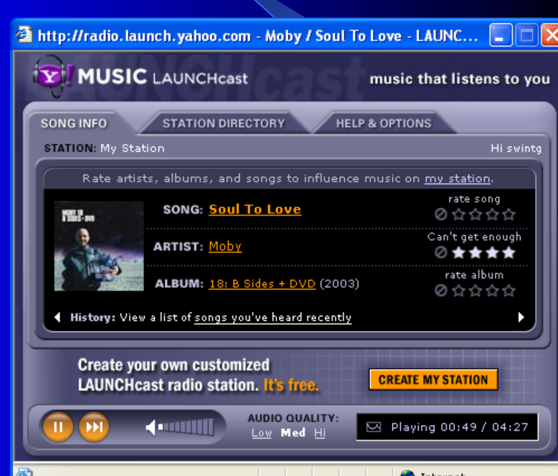
Online Gaming

- Halo2, multi-player

http://www.xbox.com/en-us/halo2/default.htm

Online Entertainment

- Radio, TV, Video
- Virtual reality



Real-Time Tracking

- Car, goods
- People
- etc

Georgia Tech

DARPA Grand Challenge

Discussion

- Evolution of embedded systems from isolated devices to participants in a ubiquitious computing world
- Role of embedded devices in:
  - Computer and communications (e.g., convergence of functionality into hand set)
  - Consumer electronics and appliances (e.g., smart refrigerator and house)
  - Transportation (e.g., cars)

RTES Need *Architecture*

- Traditional embedded systems
  - Isolated, self-contained hardware systems
  - Small, specialized software (e.g., GUI)
  - Insufficient correctness guarantees ($800M software verification costs for Boeing 777)
- Recent and future embedded systems
  - Small form factor, but big capability and capacity (e.g., iPod, cellphones, navig. systems)
  - Short shelf life forces rapid development and expectations of high reliability, security, etc

Need Principled Embedded Software

- Embedded systems becoming complex
  - Convergence of functionality (e.g., evolution of cell phones – PDAs, iPods, …)
  - More than GUI: search thousands of songs
- Integration into Internet
  - VoIP infrastructure integrating data & services
  - Sensor information from the real world
- High confidence systems and components
  - Performance, availability, reliability, security, privacy, trust, scalability, composability, etc

Georgia Tech

Unifying Concept: Constrained Resources

- Embedded systems
  - Constrained CPU, memory, storage, networking bandwidth, battery power, screen real estate, everything
- Real-time systems
  - Guaranteed schedulers under constrained CPU
  - Guaranteed message delivery under constrained network bandwidth
- Embedded systems and real-time systems have same principles and goals

Georgia Tech

Pathfinder Mars Rover

- Landing: July 4, 1997; initial successes
- Intermittent software system resets
  - Delay of mission, serious loss of data
  - Happens when “too much” data are sent over a shared information bus
  - Low priority data collection task locks the bus, gets interrupted by medium priority tasks
  - High priority data distribution task fails to complete: cannot get shared bus
  - Scheduler detects pending high priority task and resets all the hardware and software

Georgia Tech

RTES Techniques

- Problem modeling and abstraction
  - Priority inversion: high priority task delayed in a critical section by low priority tasks
- Solutions proposed
  - Priority inheritance: low priority tasks entering critical section will inherit the highest priority of waiting tasks
- Solved the Pathfinder reset problem

Georgia Tech

Feedback in a Car

- Operating environment: Road conditions and other cars.
- Controlling System
  - Human driver: Sensors - Eyes and Ears of the driver.
  - Computer: Sensors - Cameras, Infrared receiver, and Laser telemeter.
- Controls: Accelerator, Steering wheel, Break-pedal.
- Actuators: Wheels, Engines, and Brakes.

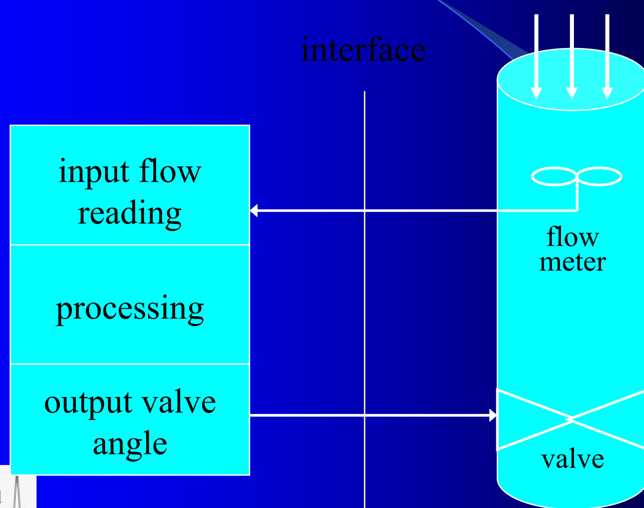
Georgia Tech

Example: cruise control

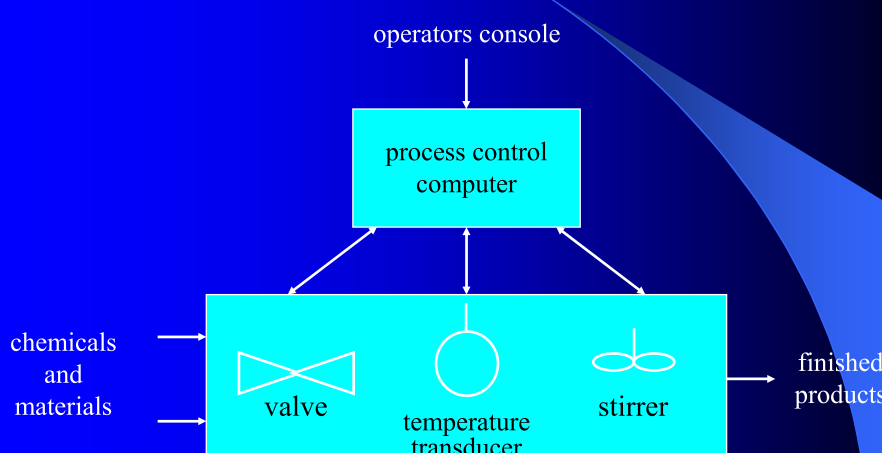
- Regulates speed of car by adjusting the throttle: driver sets a speed and car maintains it.
- Measures speed through device connected to drive shaft.
- Hard real-time: drive shaft revolution events.
- Soft real-time: driver inputs, throttle adjustments.

Georgia Tech

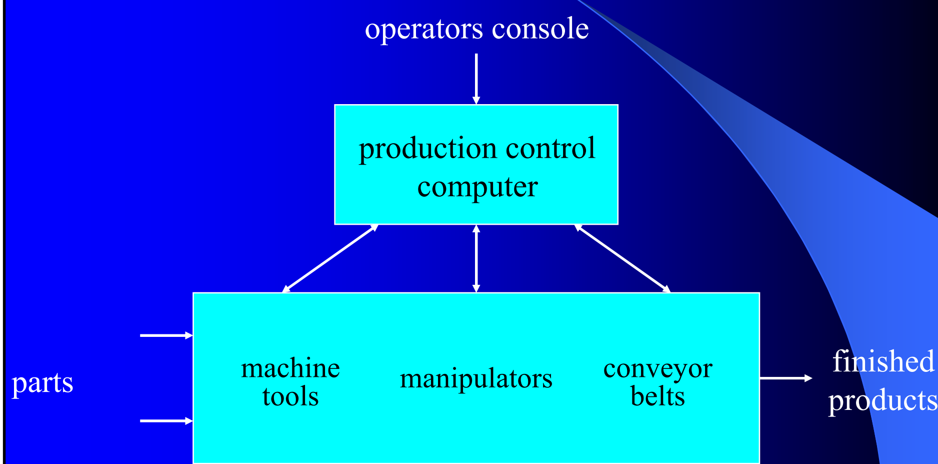
Simple Valve Control



Process Control

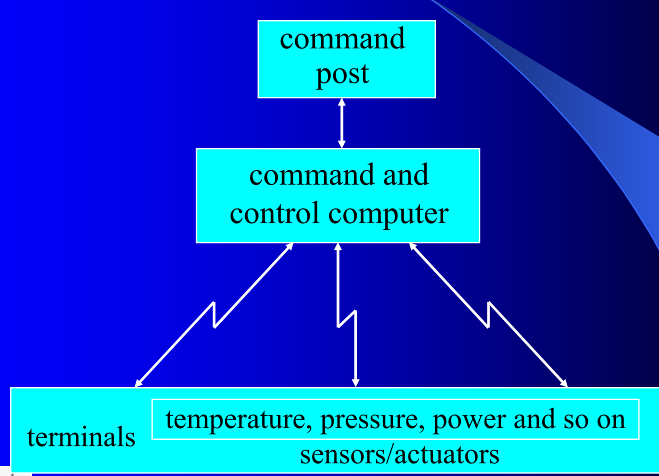


Manufacturing

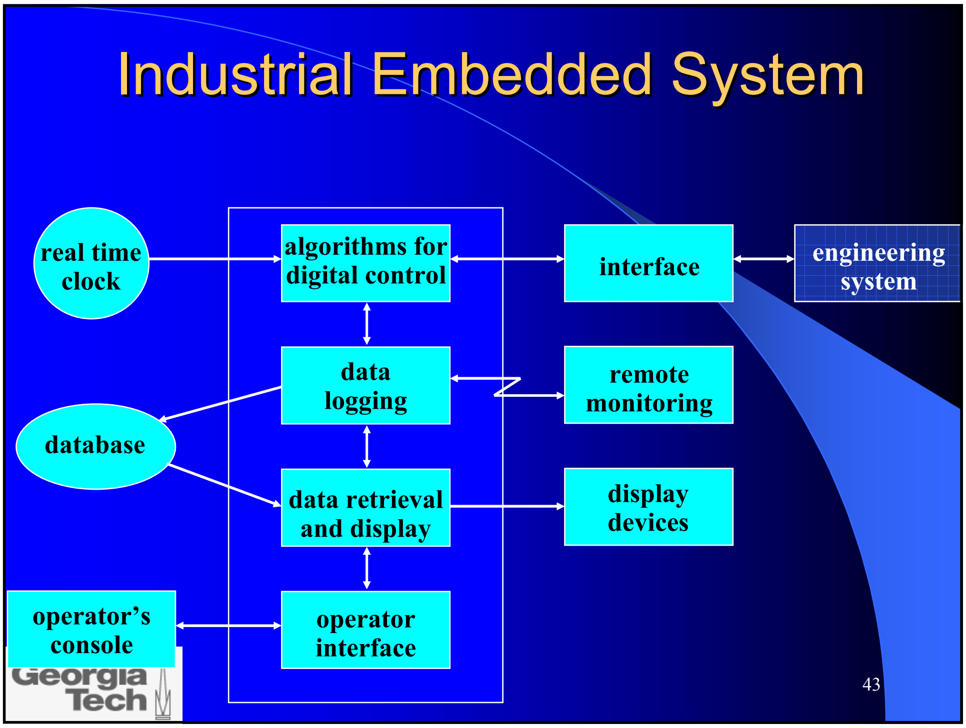


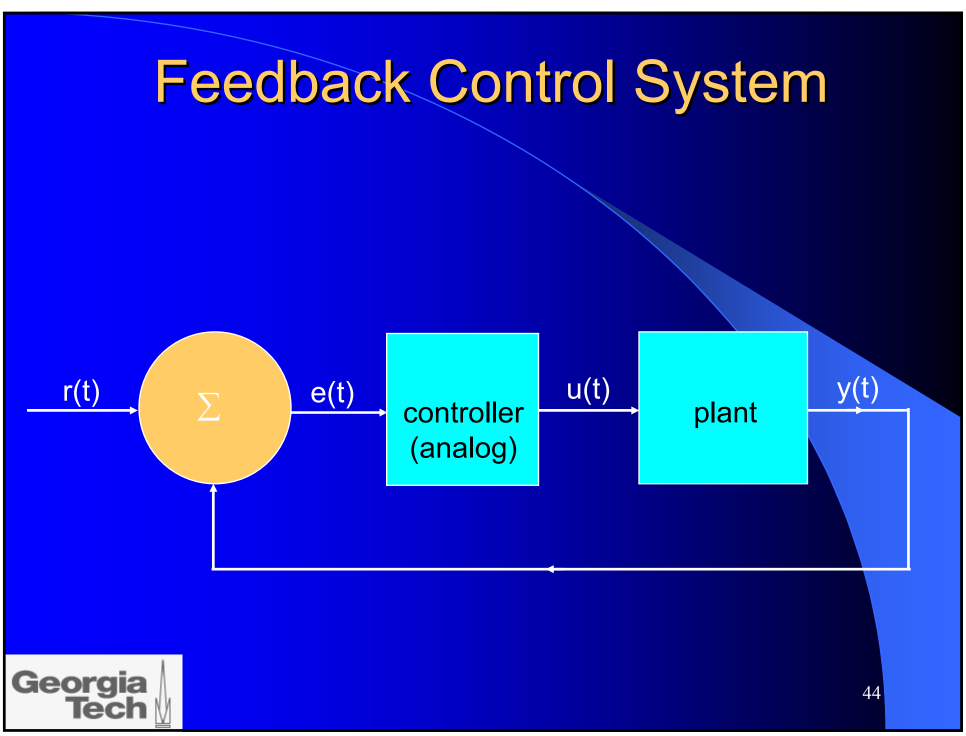
a production control system

Command, Control, Communications

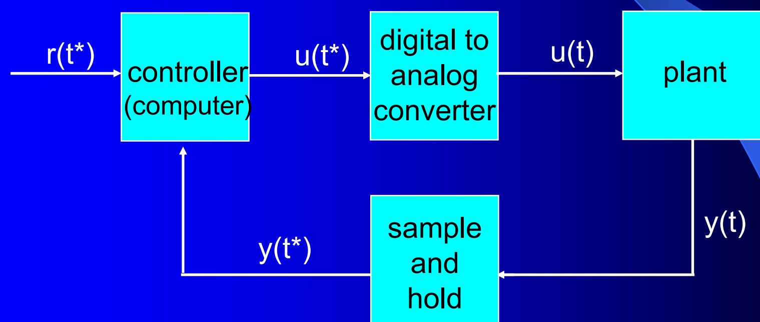


a command and control system


Industrial Embedded System
real time clock
algorithms for digital control
interface
engineering system
data logging
remote monitoring
database
data retrieval and display
display devices
operator's console
operator interface
Georgia Tech
43


Feedback Control System
r(t)
Σ
e(t)
controller (analog)
u(t)
plant
y(t)
Georgia Tech
44

Digital Feedback Control



More Examples of RTES

- Cars: engine control, ABS, drive-by-wire
- Planes: stability, jet engine, fly-by-wire
- Computers: peripherals, applications
- Military: weapons, satellites
- Small appliances: microwave, thermostat, dishwasher
- Medical: pacemaker, medical monitoring
- Security: intruder alarm, smoke/gas detection

RTES Terminology

- System: black box with n inputs and m outputs
- Response time: time between presentation of a set of inputs and the appearance of the corresponding outputs
- Utilization: measure of 'useful' work a system performs
- Events: Change of state causing a change of flow-of-control of a computer program

Georgia Tech

Classification of RTES Systems

- synchronous: events occur at predictable times in the flow-of-control.
- asynchronous: interrupts.
- state-based vs. event-based:
  - plane wing is at an angle of 32º (state)
  - plane wing moved up 4º (event)
- deterministic system: for each possible state and each set of inputs, a unique set of outputs and next state of the system can be determined.

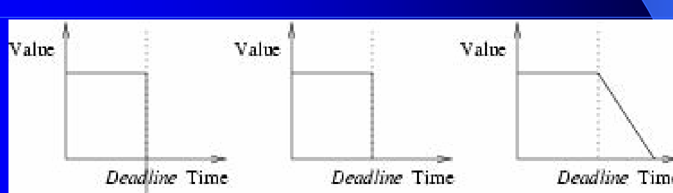
Georgia Tech

More RTES Terminology

- RTS: Correctness depends on results PLUS the time of delivery! Failure can have severe consequences.
- What are real-time systems? Planes, cars, washer, video player, thermostat, video games, weapons,...
- Related: QoS management, resource management, adaptive systems, embedded systems, pervasive and ubiquitous computing, ...

RTES Systems Classification (2)

- **HARD:** miss a deadline and you're in trouble! (planes, trains, factory control, nuclear facilities, ...)
- **SOFT:** try to meet deadlines, but if not, system still works, although with degraded performance (multimedia, thermostat, ...)
- **FIRM:** late results are worthless, but you are not in trouble



(a) Hard deadline (b) Firm deadline (c) Soft deadline

Characteristics of RTES Systems

- size: small assembler code or large C++, Ada, ... code (example: 20 million lines of Ada for Intl. Space Station).
- concurrent control of separate components (model this parallelism with parallelism in your program).
- use of special purpose hardware and tools to program devices for this hardware in a reliable manner.

Common Misconceptions

- "real fast" is real-time: a computer system may satisfy an application's requirement, but no predictability (no real-time resource management).
- hardware over-capacity is enough: again, without real-time resource management no appropriate balance of resource distribution.

Static Predictability

- RTES: satisfying the time constraints
  - Certain assumptions about workload and sufficient resource availability
  - Certify at "design time" that all the timing constraints of the application will be met
- For static systems, 100% guarantees can be given at design time
  - Immutable workload and system resources
  - System must be re-certified if anything changes

Georgia Tech

Dynamic Predictability

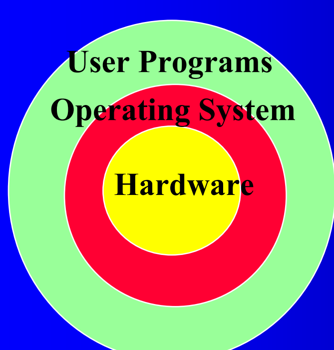
- Dynamic systems: not statically defined
  - Changeable system configuration
  - Changeable workload
- Dynamic predictability
  - Under appropriate assumptions (sufficient resources)
  - Tasks will satisfy time constraints

Georgia Tech

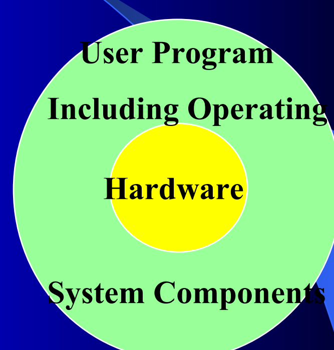
Reliability

- Reliability
  – Randell et al (1978)
  "a measure of the success with which the system conforms to some authoritative specification of its behavior"
- Safety and reliability often interchangeable
  – Usually expressed in probabilities
- Other frequently used term: dependability.

Role of Operating Systems



Typical OS Configuration



Typical Embedded Configuration

Real-Time OSs

- Real-Time OS: VxWorks, QNX, LynxOS, eCos, DeltaOS, PSX, embOS, ...
- GPOS: no support for real-time applications, focus on ‘fairness’.
- BUT, people love GPOSs, e.g., Linux:
  - RTLinux (FSMLabs)
  - KURT (Kansas U.)
  - Linux/RT (TimeSys)

Georgia Tech

RT OSs

- Determinism / Predictability
  - Ability to meet deadlines
  - Traditional operating systems non-deterministic
- Standards: Real-Time POSIX 1003.1
  - Pre-emptive fixed-priority scheduling
  - Synchronization methods
  - Task scheduling options

Georgia Tech

Lynx OS

- Lynx OS
  - Microkernel Architecture
  - Provides scheduling, interrupt, and synchronization support
  - Real-Time POSIX support
  - Easy transition from Linux

VxWorks

- Monolithic Kernel
  - Reduced run-time overhead, but increased kernel size compared to Microkernel designs
- Supports Real-Time POSIX standards
- Common in industry
  - Mars missions
  - Honda ASIMO robot
  - Switches
  - MRI scanners
  - Car engine control systems

RT Linux

- "Workaround" on top of a generic O/S
  - Generic O/S – optimizes average case scenario
  - RTOS – need to consider WORST CASE scenarios to ensure deadlines are met
- Dual-kernel approach
  - Makes Linux a low-priority pre-emptable thread running on a separate RTLinux kernel
  - Tradeoff between determinism of pure real-time O/S and flexibility of conventional O/S
- Periodic tasks only

Georgia Tech

RT Concepts

- Concurrency
- Scheduling: priorities, time driven, event driven, task scheduling (RMS).
- Processes, threads.
- Synchronization: test-and-set instructions, semaphores, deadlocks (circular waits), ...

Georgia Tech

RT Scheduling

- static: all scheduling decisions are determined before execution.
- dynamic: run-time decisions are used.
- periodic: processes that repeatedly execute
- aperiodic: processes that are triggered by asynchronous events from the physical world.
- sporadic: aperiodic processes w/ known minimum inter-arrival jitter between any two aperiodic events.

Georgia Tech

Preemptive vs. Non-preemptive

- Preemptive Scheduling
  - Task execution is preempted and resumed later.
  - Preemption takes place to execute a higher priority task.
  - Offers higher schedulability.
  - Involves higher scheduling overhead due to context switching.
- Non-preemptive Scheduling
  - Once a task is started executing, it completes its execution.
  - Offers lower schedulability.
  - Has less scheduling overhead because of less context switching.

Georgia Tech

Rate Monotonic Priority Assignment

- each process has a unique priority based on its period; the shorter the period, the higher the priority.
- Rate Monotonic proven optimal in the sense that if any process set can be scheduled (using preemptive priority-based scheduling) with a fixed priority-based assignment scheme, then RMA can also schedule the process set.

Rate Monotonic Analysis

- Each task has a period T and run-time C.
- System utilization $U=\Sigma(Ci/Ti)$. Measure for computational load on the CPU due to the task set.
- There exists a maximum value of U, below which a task set is schedulable and above which it is not schedulable.
- Liu and Layland 1973

$$\Sigma(Ci/Ti) <= n(21/n-1)$$

Real-Time Languages

- Support for the management of time
  - Language constructs for expressing timing constraint, keeping track of resource utilization.
- Schedulability analysis
  - Aid compile-time schedulability check.
- Reusable real-time software modules
  - Object-oriented methodology.
- Support for distributed programming and fault-tolerance

Georgia Tech

Real-Time Databases

- Most conventional database systems are disk-based.
- They use transaction logging and two-phase locking protocols to ensure transaction atomicity and serializability.
- These characteristics preserve data integrity, but they also result in relatively slow and unpredictable response times.

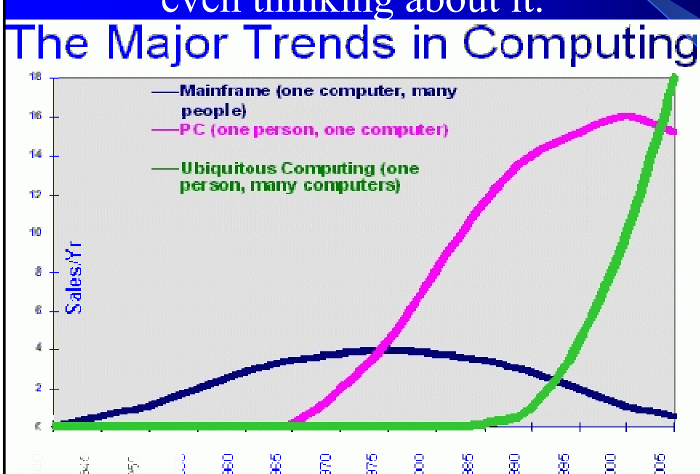
Georgia Tech

Real-Time Databases (2)

- In a real-time database system, important issues include:
  - transaction scheduling to meet deadlines.
  - explicit semantics for specifying timing and other constraints.
  - checking the database system's ability of meeting transaction deadlines during application initialization.

Ubiquitous Computing

- Make computers invisible, so embedded , so fitting, so natural, that we use it without even thinking about it.



Another Vision of Future

- Autonomous Computing:
  - self-configurable
  - self-adapting
  - optimizing
  - self-healing
- Building real-time systems:
  - toolkits, validation tools, program composition
  - Boeing 777: $4Billion, >50% system integration & validation!

Georgia Tech

New Constraints

- Soft real-time applications:
  - mainstream applications
  - notion of QoS
- Multi-dimensional requirements:
  - real-time, power, size, cost, security, fault tolerance
  - conflicting resource requirements and system architecture
- Unpredictable environments:
  - Internet (servers), real-time databases, ...

Georgia Tech

RTES Systems

- An “engineering approach” to RTES
  - Model of RTE systems (e.g., tasks with time constraints)
  - Techniques that satisfy the constraints (e.g., scheduling algorithms such as RMA)
  - Implementation of these techniques
- Uncertainties of today’s environments
  - Ubiquitous/pervasive/environmental computing

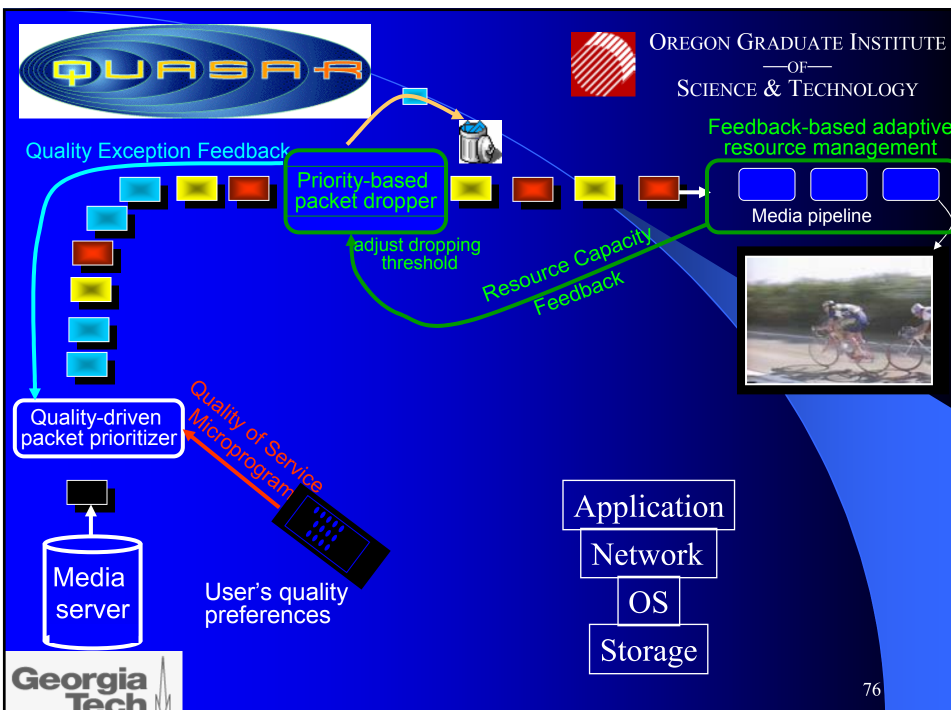
Discussion

- Technical components of RTES
  - Same as “normal” computer systems?
- Extra-functional requirements of RTES
  - System performance, availability, reliability, security, trust
  - Information security, privacy, trust
  - Adaptiveness, renewability

End-to-end Properties

- Real-time requirements
  - Need to achieve some goals (e.g., enough CPU to finish a job) before deadline
- End-to-end properties (limited by the weakest link)
  - Performance bottlenecks (e.g., bandwidth and latency in networks)
  - Reliability, availability, security
- Soft real-time multimedia application

Georgia Tech



Multimedia Requirements

- Table 1

Media Specifications	Bandwidth requirements
Voice audio	0.008 MBps
CD quality audio (2x16 at 44.1 kHz)	0.18 MBps
NTSC video (640x480x8 bits)	8.7 MBps
HDTV video (1024x2000x24)	351 MBps

Buffer Requirements

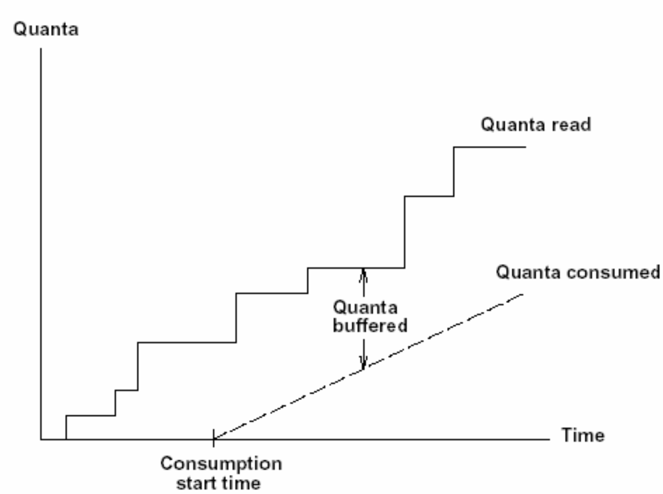


Figure 1 : Ensuring continuous retrieval of media stream from disk

Storage Server Requirements

- Achieve simultaneous serving
  - Processing in rounds (each round per stream)
- Production keeps up with consumption
  - Buffer-conserving (no decrease in the amount of buffered data)
- Duration of a round
  - Retrieving media blocks from storage
  - Regular playback according to media spec

Media Block Retrieval

- Need enough blocks for each round
  - Transmission + rotational delay + seek time
- Seek time dominates (tens of milliseconds)
  - Disk scheduling algorithms (e.g., Grouped Sweeping Scheme)
- Prevention of saturation
  - Admission control

Block Placement Optimization

- Disk storage and retrieval times
  - Contiguous (easiest retrieval)
  - Scattered (scheduled retrieval)
- Multiple disks
  - RAID data striping (low level synchronization)
  - Data interleaving (high level synchronization)
  - Combination of striping/interleaving

Disk/Memory Hierarchies

- Single disks attached to processor bus
- Multiple disks networked to server
  - Attached to some bus (RAID)
  - Networked through SAN
  - Networked through new protocols, e.g., IPSCSI
- Clustered server machines
  - Very fast processors
  - Caches, main memory, silicon memory, etc

Adaptive Suspension Vehicle

Control Software Components

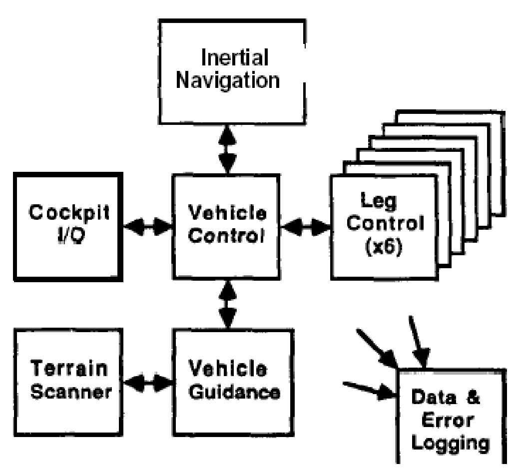


Fig. 1.
ware-su

GEM Robotics OS

- CPU management
  - Processes and Microprocesses
- Memory management
  - GEM process = single address space
- Inter-task communications
  - Asynchronous execution with data loss (under saturation
  - Synchronous execution without data loss
  - Hybrid combining async and sync

Communications Example

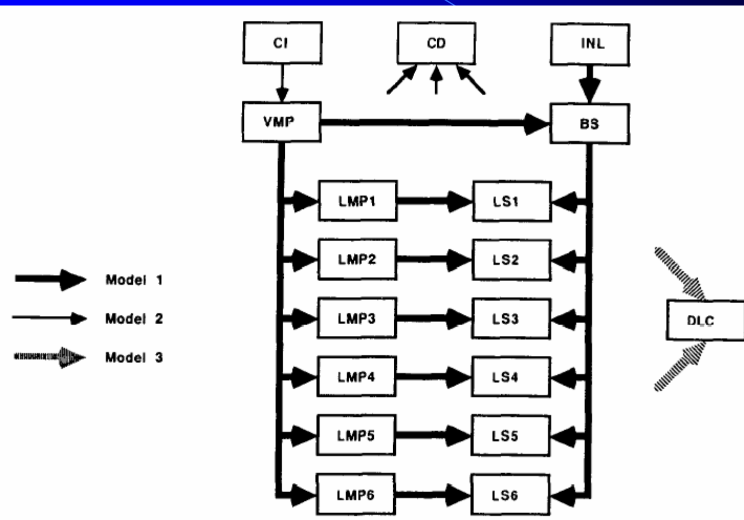
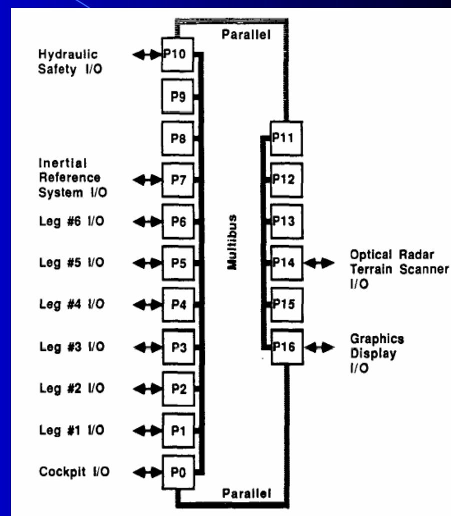


Fig. 2. The ASV Robot's operating software—interaction examples.

GEM Hardware Platform

- Parallel processors
  - 2 clusters (11 + 6)
  - 8086 (8 MHz, 750ns basic cycle)
  - 8087 co-processor
  - 128Kb-256Kb memory (750ns)
- GEM configuration
  - Kernel: 20Kb



Process Switch Time

Table I. Cost of Process Switching

Operation	Local time
EventTable/ReadyQueue	305 μs
Restore State	105 μs
Asleep-W → Running	410 μs
Save State	140 μs
ReadyQueue dequeue	260 μs
Running → Asleep-W	400 μs

Microprocess Savings

Table II. Process versus Microprocess Overheads

	Process	Microprocess
Scheduling and Descheduling		
Respond to WakeUp\Poke	580 μs (810 μs)	140 μs (950 μs)
Execution	—	—
Output		
Data	—	—
Control	WakeUp: 180 μs	Poke: 245 μs (425 μs)
Total	760 μs (975 μs)	395 μs (1375 μs)

Microprocess overhead includes accessing parent process

Georgia Tech

Communications Overhead

Table III. Trade-offs in Mailbox Location—Intracluster

Operation	Local	Intracluster
GetEnvelope	155 μs	160 μs
Transfer of 87 bytes	260 μs	335 μs
SendLetter	180 μs	190 μs
Other processing	95 μs	95 μs
SendLetterCopy (87 bytes)	690 μs	780 μs
GetLetter	185 μs	200 μs
Transfer of 87 bytes	260 μs	335 μs
DiscardEnvelope	160 μs	165 μs
Other processing	80 μs	80 μs
GetLetterCopy (87 bytes)	685 μs	780 μs

Tech

Inter-cluster Overhead

Table IV. Trade-offs in Mailbox Location—Intercluster

Operation	Intracluster	Intercluster
WakeUp	180 μs	1500 μs
Link transfer (87 bytes)	335 μs	6500–7000 μs
SendLetterCopy (87 bytes)	780 μs	8150 μs
GetLetterCopy (87 bytes)	780 μs	10550 μs

Georgia Tech

Discussion

- Impact of Moore's Law on RTES
  - CPU (power, multicore)
  - Storage (memory, disk)
  - Network (wireless bandwidth)
- What are the (new) requirements of modern RTES?
- What are the principles that guarantee RTES will work as designed?

Georgia Tech