IA158 Real Time Systems

Tomáš Brázdil

Organization of This Course

Sources:

- Lectures (slides, notes)
  - based on several sources (hard to obtain)
  - slides are prepared for lectures, lots of stuff on the greenboard
    ($\Rightarrow$ attend the lectures)

Organization of This Course

Sources:

- Lectures (slides, notes)
  - based on several sources (hard to obtain)
  - slides are prepared for lectures, lots of stuff on the greenboard
    ($\Rightarrow$ attend the lectures)

Homework:

- a larger homework project

Organization of This Course

Sources:

- Lectures (slides, notes)
  - based on several sources (hard to obtain)
  - slides are prepared for lectures, lots of stuff on the greenboard
    ($\Rightarrow$ attend the lectures)

Homework:

- a larger homework project

Evaluation:

- Homework project

  (have to do to be allowed to the exam)
- Oral exam

Definition 1 (Time)

Mirriam-Webster: Time is the measured or measurable period during which an action, process, or condition exists or continues.

Real-Time Systems

Definition 1 (Time)

Mirriam-Webster: Time is the measured or measurable period during which an action, process, or condition exists or continues.

Definition 2 (Real-time)

Real-time is a quantitative notion of time measured using a physical clock.

Example: After an event occurs (eg. temperature exceeds 500 degrees) the corresponding action (cooling) must take place within 100ms.

Compare with qualitative notion of time (before, after, eventually, etc.)

Real-Time Systems

Definition 1 (Time)

Mirriam-Webster: Time is the measured or measurable period during which an action, process, or condition exists or continues.

Definition 2 (Real-time)

Real-time is a quantitative notion of time measured using a physical clock.

Example: After an event occurs (eg. temperature exceeds 500 degrees) the corresponding action (cooling) must take place within 100ms.

Compare with qualitative notion of time (before, after, eventually, etc.)

Definition 3 (Real-time system)

A *real-time system* must deliver services in a timely manner.

Not necessarily fast, must satisfy some *quantitative* timing constraints

Real-time Embedded Systems

Definition 4 (Embedded system)

An *embedded system* is a computer system designed for specific control functions within a larger system, usually consisting of electronic as well as mechanical parts.

Real-time Embedded Systems

Definition 4 (Embedded system)

An *embedded system* is a computer system designed for specific control functions within a larger system, usually consisting of electronic as well as mechanical parts.

Most (not all) real-time systems are embedded

Most (not all) embedded systems are real-time

(Few) Examples of Real-time Embedded Systems

- Industrial
  - chemical plant control
  - automated assembly line (e.g. robotic assembly, inspection)

(Few) Examples of Real-time Embedded Systems

- Industrial
  - chemical plant control
  - automated assembly line (e.g. robotic assembly, inspection)
- Medical
  - pacemaker,
  - medical monitoring devices

(Few) Examples of Real-time Embedded Systems

- Industrial
  - chemical plant control
  - automated assembly line (e.g. robotic assembly, inspection)
- Medical
  - pacemaker,
  - medical monitoring devices
- Transportation systems
  - computers in cars (ABS, MPFI, cruise control, airbag ...)
  - aircraft (FMS, fly-by-wire ...)

(Few) Examples of Real-time Embedded Systems

- Industrial
  - chemical plant control
  - automated assembly line (e.g. robotic assembly, inspection)
- Medical
  - pacemaker,
  - medical monitoring devices
- Transportation systems
  - computers in cars (ABS, MPFI, cruise control, airbag ...)
  - aircraft (FMS, fly-by-wire ...)
- Military applications
  - controllers in weapons, missiles, ...
  - radar and sonar tracking

(Few) Examples of Real-time Embedded Systems

- Industrial
  - chemical plant control
  - automated assembly line (e.g. robotic assembly, inspection)
- Medical
  - pacemaker,
  - medical monitoring devices
- Transportation systems
  - computers in cars (ABS, MPFI, cruise control, airbag ...)
  - aircraft (FMS, fly-by-wire ...)
- Military applications
  - controllers in weapons, missiles, ...
  - radar and sonar tracking
- Multimedia – multimedia center, videoconferencing
- ...

(Non-)Real-time (non-)embedded systems

There are real time systems that are not embedded:

- trading systems
- ticket reservation
- multimedia (on PC)
- ...

(Non-)Real-time (non-)embedded systems

There are real time systems that are not embedded:

- trading systems
- ticket reservation
- multimedia (on PC)
- ...

There are embedded systems that are (possibly) not real-time

e.g. a weather station sends data once a day without any deadline – not really real-time system

Caveat: Aren't all systems real-time in a sense?

Characteristics of Real-Time Embedded Systems

Real-time systems often are

- safety critical
  - Serious consequences may result if services are not delivered on timely basis
  - Bugs in embedded real-time systems are often difficult to fix

... need to validate their correctness

Characteristics of Real-Time Embedded Systems

Real-time systems often are

- safety critical
  - Serious consequences may result if services are not delivered on timely basis
  - Bugs in embedded real-time systems are often difficult to fix

  ... need to validate their correctness

- concurrent
  - Real-world devices operate in parallel – better to model this parallelism by concurrent tasks in the program

  ... validation may be difficult, formal methods often needed

Characteristics of Real-Time Embedded Systems

Real-time systems often are

- safety critical
  - Serious consequences may result if services are not delivered on timely basis
  - Bugs in embedded real-time systems are often difficult to fix

  ... need to validate their correctness

- concurrent
  - Real-world devices operate in parallel – better to model this parallelism by concurrent tasks in the program

  ... validation may be difficult, formal methods often needed

- reactive
  - Interact continuously with their environment (as opposed to information processing systems)

  ... “traditional” validation methods do not apply

Validating Time Requirements and Predictability

- Given real-time requirements and an implementation on HW and SW, how to show that the requirements are met?

Validating Time Requirements and Predictability

- Given real-time requirements and an implementation on HW and SW, how to show that the requirements are met?

  ... testing might not suffice:

  Maiden flight of space shuttle, 12 April 1981: 1/67 probability that a transient overload occurs during initialization; and it actually did!

Validating Time Requirements and Predictability

- Given real-time requirements and an implementation on HW and SW, how to show that the requirements are met?

  ... testing might not suffice:

  Maiden flight of space shuttle, 12 April 1981: 1/67 probability that a transient overload occurs during initialization; and it actually did!

- We need a formal model and validation ...

Validating Time Requirements and Predictability

- Given real-time requirements and an implementation on HW and SW, how to show that the requirements are met?

  ... testing might not suffice:

  Maiden flight of space shuttle, 12 April 1981: 1/67 probability that a transient overload occurs during initialization; and it actually did!

- We need a formal model and validation ...

- ... we need **predictable** behavior!
  It is difficult to obtain
  - caches, DMA, unmaskable interrupts
  - memory management
  - scheduling anomalies
  - difficult to compute worst-case execution time
  - ...

Types of Timing Requirements

Time sharing systems: minimize average response time

The goal of scheduling in standard op. systems such as Linux and Windows

Types of Timing Requirements

Time sharing systems: minimize average response time

The goal of scheduling in standard op. systems such as Linux and Windows

Often it is **not** enough to minimize average response time!

(A man drowned crossing a stream with an average depth of 15cm.)

Types of Timing Requirements

Time sharing systems: minimize average response time

The goal of scheduling in standard op. systems such as Linux and Windows

Often it is **not** enough to minimize average response time!

(A man drowned crossing a stream with an average depth of 15cm.)

"hard" real-time tasks must be **always** finished before their deadline!

e.g. airbag in a car: whenever a collision is detected, the airbag must be deployed within 10ms

Types of Timing Requirements

Time sharing systems: minimize average response time

The goal of scheduling in standard op. systems such as Linux and Windows

Often it is **not** enough to minimize average response time!

(A man drowned crossing a stream with an average depth of 15cm.)

"hard" real-time tasks must be **always** finished before their deadline!

e.g. airbag in a car: whenever a collision is detected, the airbag must be deployed within 10ms

Not all tasks in a real-time system are critical, only the quality of service is affected by missing a deadline

Types of Timing Requirements

Time sharing systems: minimize average response time

The goal of scheduling in standard op. systems such as Linux and Windows

Often it is **not** enough to minimize average response time!

(A man drowned crossing a stream with an average depth of 15cm.)

"hard" real-time tasks must be **always** finished before their deadline!

e.g. airbag in a car: whenever a collision is detected, the airbag must be deployed within 10ms

Not all tasks in a real-time system are critical, only the quality of service is affected by missing a deadline

Most "soft" real-time tasks should finish before their deadlines.

e.g. frame rate in a videoconf. should be kept above 15fps most of the time

Types of Timing Requirements

Time sharing systems: minimize average response time

The goal of scheduling in standard op. systems such as Linux and Windows

Often it is **not** enough to minimize average response time!

(A man drowned crossing a stream with an average depth of 15cm.)

"hard" real-time tasks must be **always** finished before their deadline!

e.g. airbag in a car: whenever a collision is detected, the airbag must be deployed within 10ms

Not all tasks in a real-time system are critical, only the quality of service is affected by missing a deadline

Most "soft" real-time tasks should finish before their deadlines.

e.g. frame rate in a videoconf. should be kept above 15fps most of the time

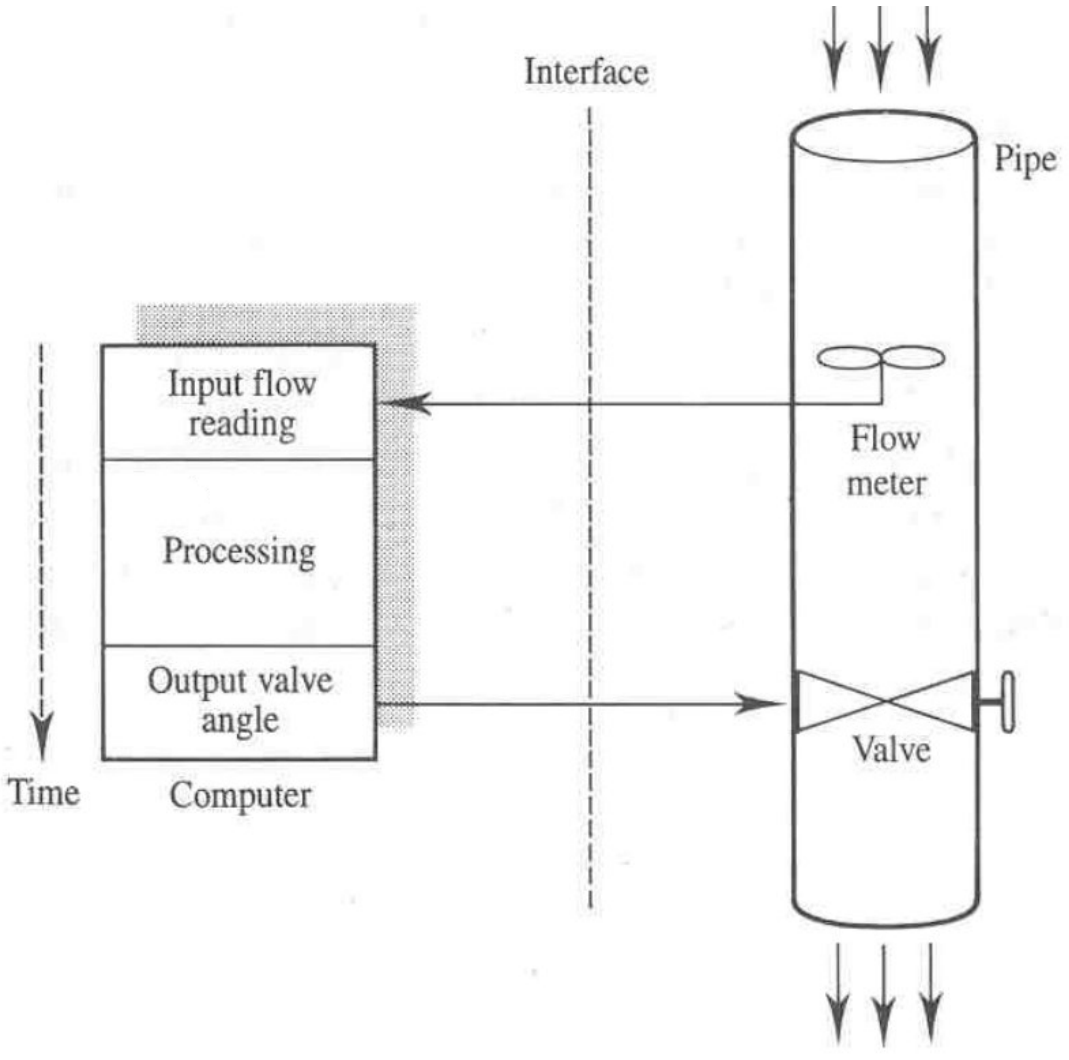
Many real-time systems combine "hard" and "soft" real-time tasks.

i.e. we optimize performance w.r.t. "soft" real-time tasks under the constraint that "hard" real-time tasks are finished before their deadlines

Examples of Real-Time Systems

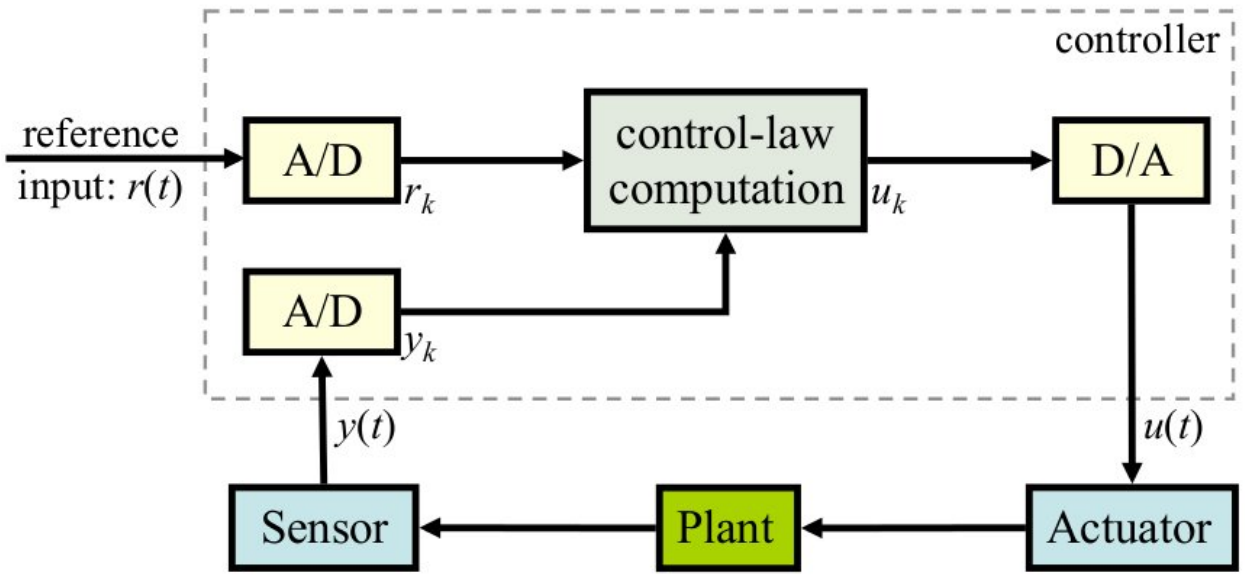
- Digital process control
  - anti-lock braking system
- Higher-level command and control
  - helicopter flight control
- Real-time databases
  - Stock trading systems

Digital Process Control



Computer controls the flow in the pipe in real-time

Digital Process Control



The controller (computer) controls the plant using the actuator (valve) based on sampled data from the sensor (flow meter)

- $y(t)$ – the measured state of the plant
- $r(t)$ – the desired state of the plant
- Calculate control output $u(t)$ as a function of $y(t), r(t)$
  e.g. $u_k = u_{k-2} + \alpha(r_k - y_k) + \beta(r_{k-1} - y_{k-1}) + \gamma(r_{k-2} - y_{k-2})$
  where α, β, γ are suitable constants

Digital Process Control

- Pseudo-code for the controller:

  set timer to interrupt periodically with period T
  foreach timer interrupt **do**
  | analogue-to-digital conversion of $y(t)$ to get y_k
  | compute control output u_k based on r_k and y_k
  | digital-to-analogue conversion of u_k to get $u(t)$
  end

Digital Process Control

- Pseudo-code for the controller:

  set timer to interrupt periodically with period T
  foreach timer interrupt **do**
  | analogue-to-digital conversion of $y(t)$ to get y_k
  | compute control output u_k based on r_k and y_k
  | digital-to-analogue conversion of u_k to get $u(t)$
  end

- Effective control of the plant depends on:
  - The correct reference input and control law computation
  - The accuracy of the sensor measurements
    - Resolution of the sampled data (i.e. bits per sample)
    - Frequency of interrupts (i.e. $1/T$)

Digital Process Control

- Pseudo-code for the controller:

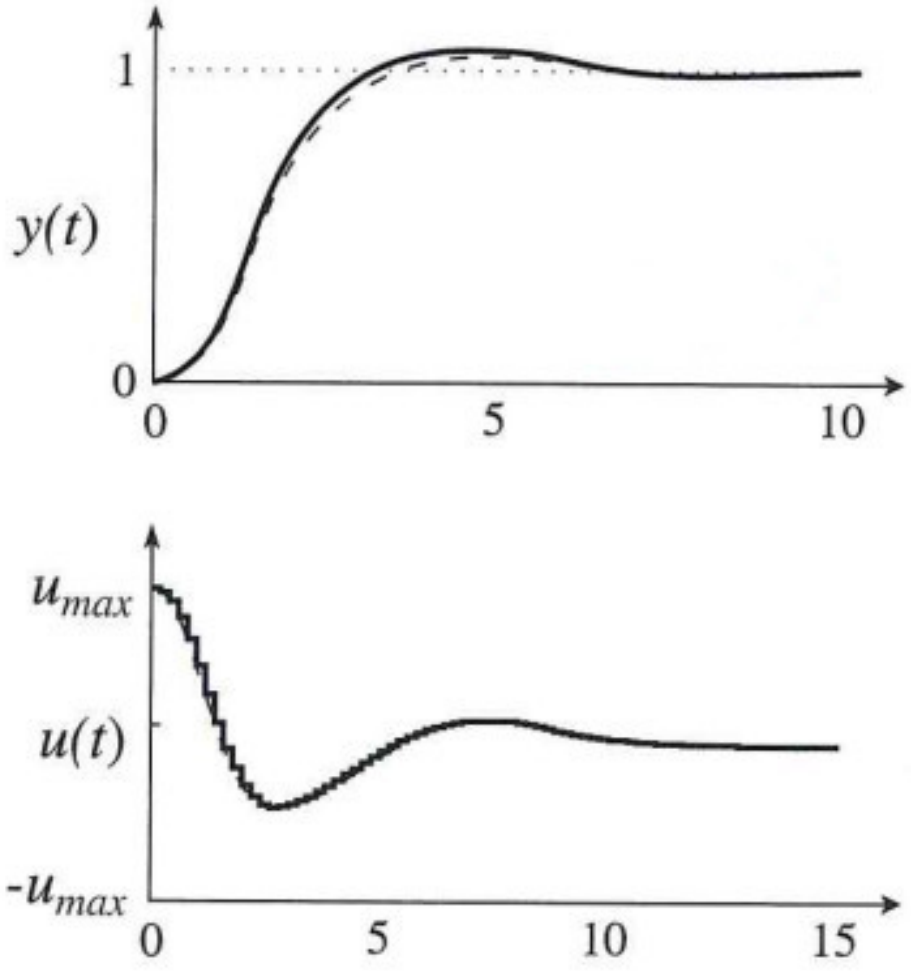
  set timer to interrupt periodically with period T
  foreach timer interrupt **do**
  | analogue-to-digital conversion of $y(t)$ to get y_k
  | compute control output u_k based on r_k and y_k
  | digital-to-analogue conversion of u_k to get $u(t)$
  end

- Effective control of the plant depends on:
  - The correct reference input and control law computation
  - The accuracy of the sensor measurements
    - Resolution of the sampled data (i.e. bits per sample)
    - Frequency of interrupts (i.e. $1/T$)

- T is the *sampling period*
  - Small T better approximates the analogue behavior
  - Large T means less processor-time demand

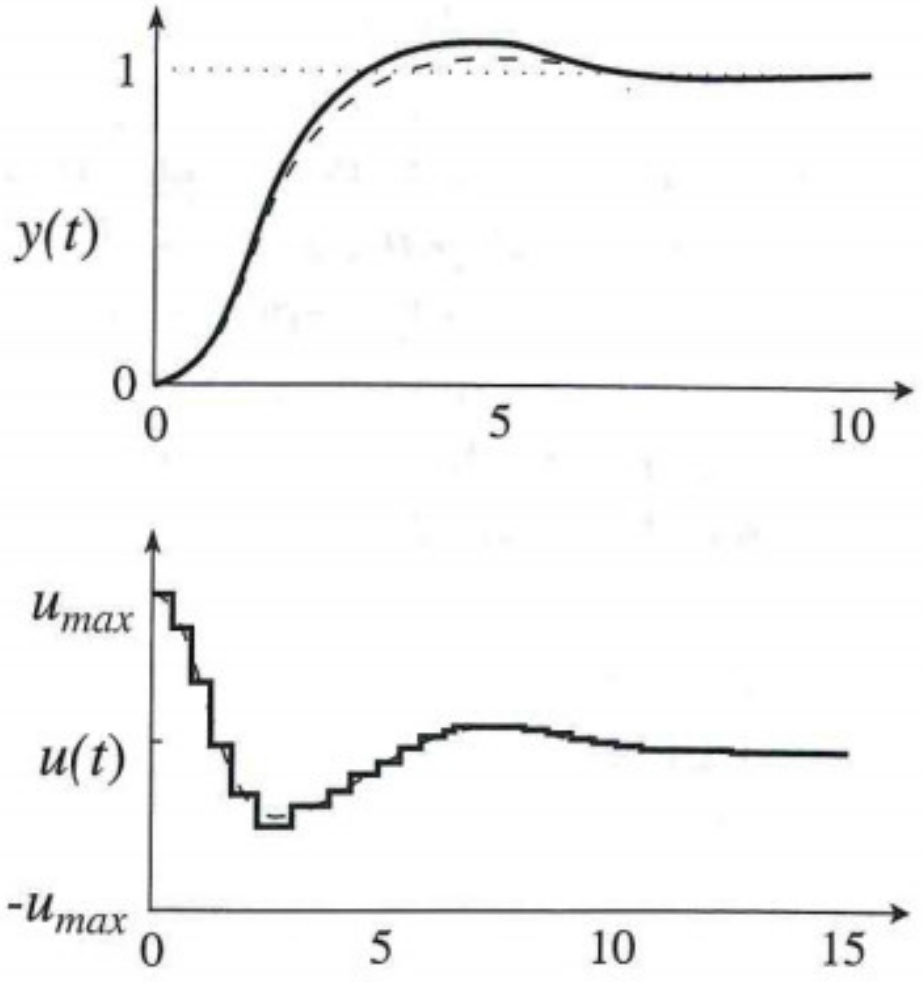
    ... but may result in unstable control

Example



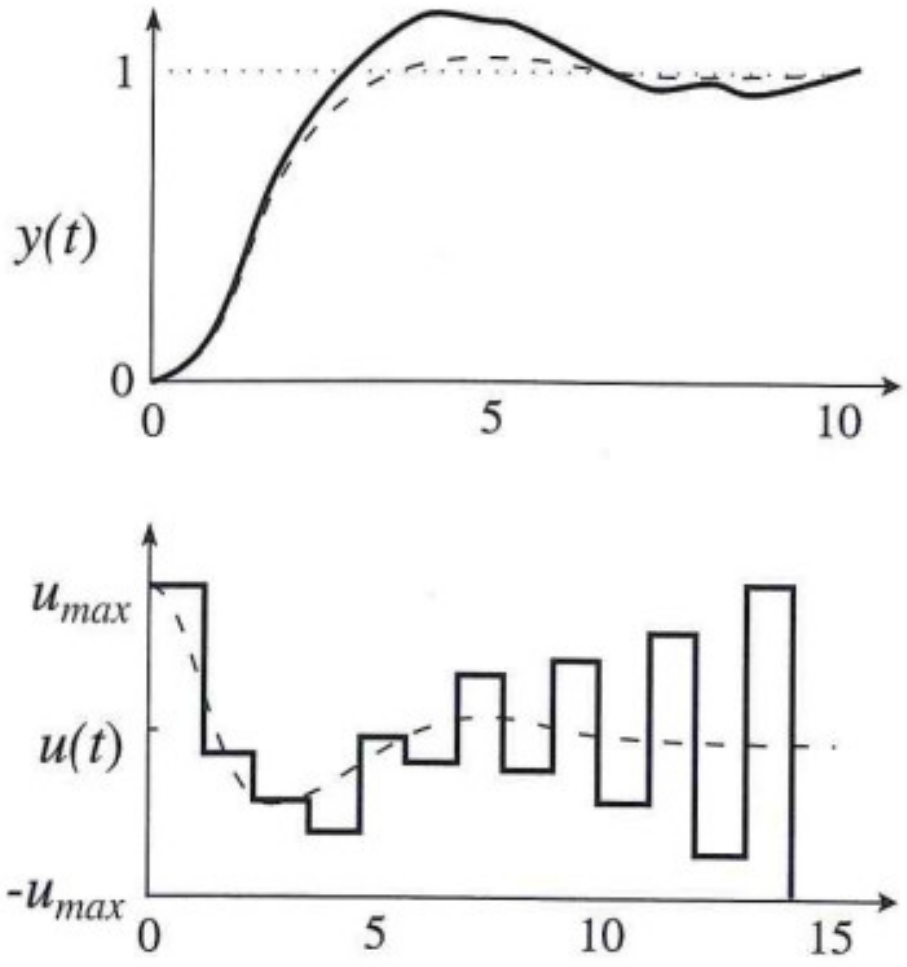
$r(t) = 1$ for $t \geq 0$

Example



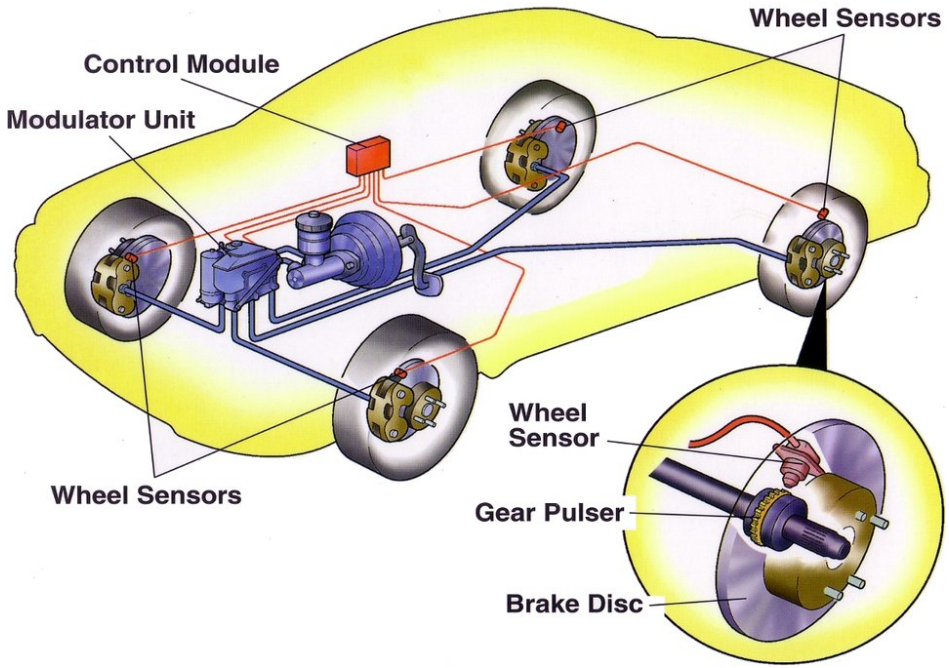
$r(t) = 1$ for $t \geq 0$

Example



$r(t) = 1$ for $t \geq 0$

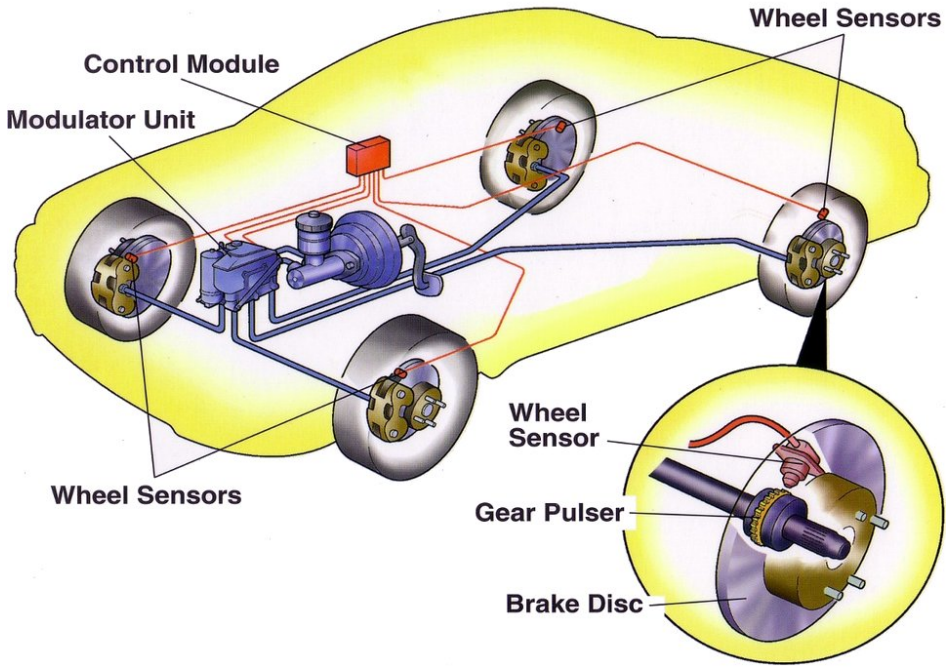
Anti-Lock Braking System



- The controller monitors the speed sensors in wheels

  Right before a wheel locks up, it experiences a rapid deceleration

Anti-Lock Braking System

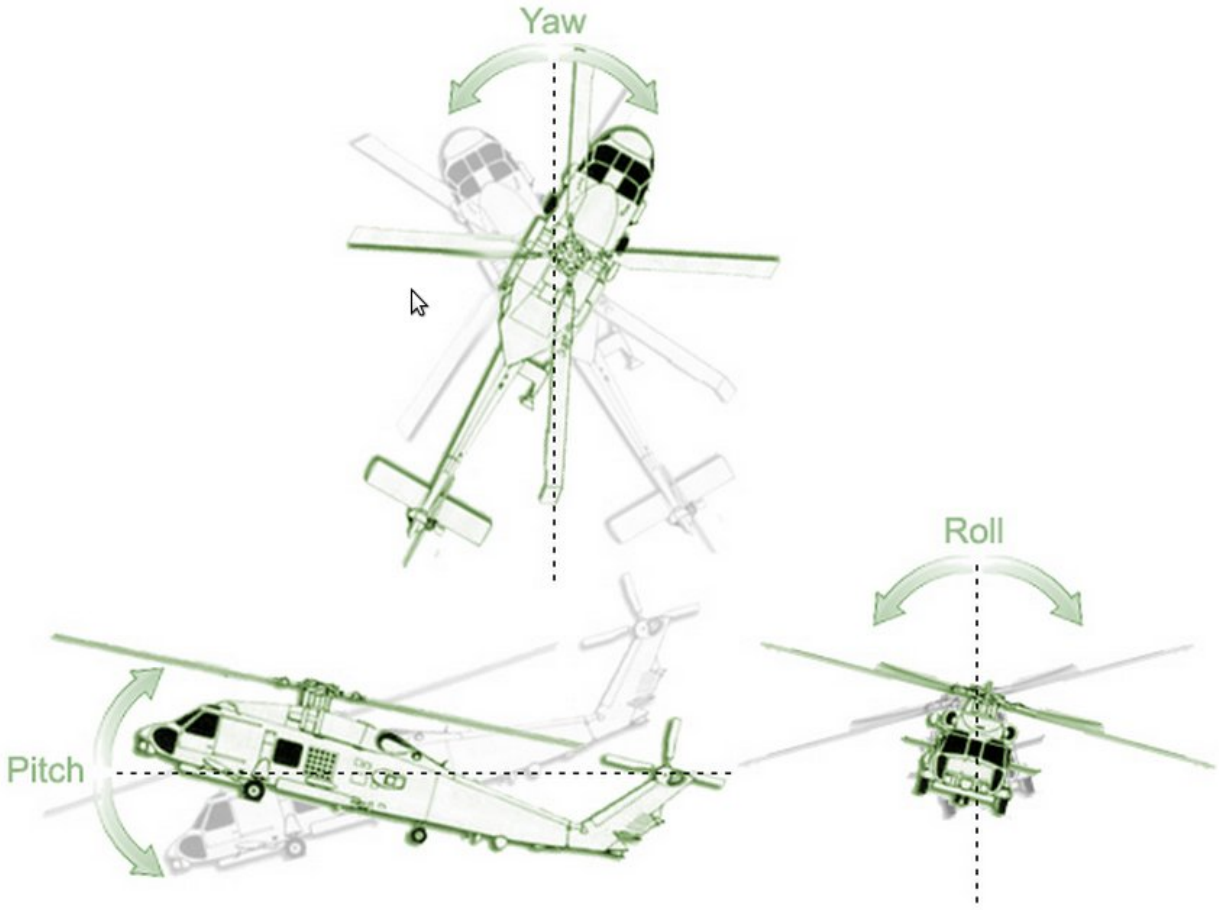


- The controller monitors the speed sensors in wheels

  Right before a wheel locks up, it experiences a rapid deceleration

- If a rapid deceleration of a wheel is observed, the controller alternately
  - reduces pressure on the corresponding brake until acceleration is observed
  - then applies brake until deceleration is observed

Multi-Rate DPC – Helicopter Flight Control



There are also three velocity components

Two control loops: pilot's control (30Hz) and stabilization (90Hz)

Multi-Rate DPC – Helicopter Flight Control

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

- Validate sensor data; in the presence of failures, reconfigure the system

Multi-Rate DPC – Helicopter Flight Control

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

- Validate sensor data; in the presence of failures, reconfigure the system
- Do the following 30-Hz avionics tasks, each one every six cycles:
  - keyboard input and mode selection
  - data normalization and coordinate transformation
  - tracking reference update

Multi-Rate DPC – Helicopter Flight Control

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

- Validate sensor data; in the presence of failures, reconfigure the system
- Do the following 30-Hz avionics tasks, each one every six cycles:
  - keyboard input and mode selection
  - data normalization and coordinate transformation
  - tracking reference update
- Do the following 30-Hz avionics tasks, each one every six 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

Multi-Rate DPC – Helicopter Flight Control

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

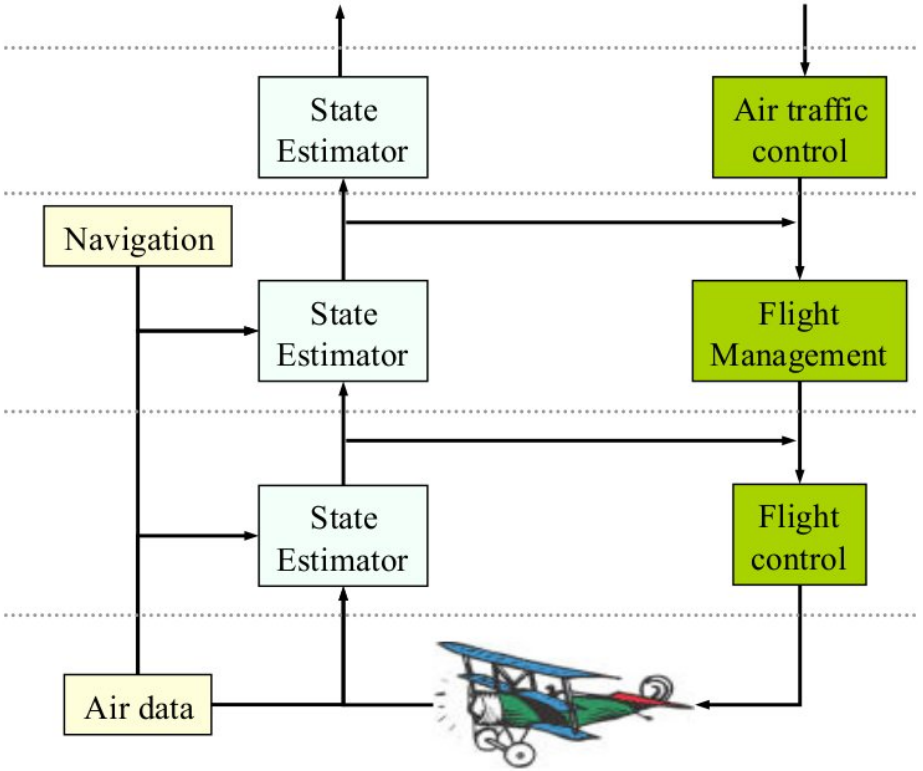
- Validate sensor data; in the presence of failures, reconfigure the system
- Do the following 30-Hz avionics tasks, each one every six cycles:
  - keyboard input and mode selection
  - data normalization and coordinate transformation
  - tracking reference update
- Do the following 30-Hz avionics tasks, each one every six 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 two cycles, using outputs produced by 30-Hz computations and avionics tasks:
  - 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 produced by 90-Hz control-law computations as inputs

Multi-Rate DPC – Helicopter Flight Control

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

- Validate sensor data; in the presence of failures, reconfigure the system
- Do the following 30-Hz avionics tasks, each one every six cycles:
  - keyboard input and mode selection
  - data normalization and coordinate transformation
  - tracking reference update
- Do the following 30-Hz avionics tasks, each one every six 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 two cycles, using outputs produced by 30-Hz computations and avionics tasks:
  - 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 produced by 90-Hz control-law computations as inputs
- Output commands
- Carry out built-in-test
- Wait until the beginning of the next cycle

Higher-Level Command and Control



Controllers organized into a hierarchy

- At the lowest level we place the digital control systems that operate on the physical environment
- Higher level controllers monitor the behavior of lower levels
- Time-scale and complexity of decision making increases as one goes up the hierarchy (from control to planning)

Real-Time Database System

- Databases that contain perishable data, i.e. relevance of data deteriorates with time

  Air traffic control, stock price quotation systems, tracking systems, etc.

Real-Time Database System

- Databases that contain perishable data, i.e. relevance of data deteriorates with time

  Air traffic control, stock price quotation systems, tracking systems, etc.
- The temporal quality of data is quantified by *age of an image object*, i.e. the length of time since last update

Real-Time Database System

- Databases that contain perishable data, i.e. relevance of data deteriorates with time

  Air traffic control, stock price quotation systems, tracking systems, etc.
- The temporal quality of data is quantified by *age of an image object*, i.e. the length of time since last update
- temporal consistency
  - absolute = max. age is bounded by a fixed threshold
  - relative = max. difference in ages is bounded by a threshold

    e.g. planning system correlating traffic density and flow of vehicles

Applications	Size	Ave. Resp. Time	Max Resp. Time	Abs. Cons.	Rel. Cons.
Air traffic control	20,000	0.50 ms	5.00 ms	3.00 sec.	6.00 sec.
Aircraft mission	3,000	0.05 ms	1.00 ms	0.05 sec.	0.20 sec.
Spacecraft control	5,000	0.05 ms	1.00 ms	0.20 sec.	1.00 sec.
Process control		0.80 ms	5.00 sec	1.00 sec.	2.00 sec

- Users of database compete for access – various models for trading consistency with time demands exist.

Stock-Trading System

- A system for selling/buying stock at public prices

Stock-Trading System

- A system for selling/buying stock at public prices
- Prices are volatile in their movement

Stock-Trading System

- A system for selling/buying stock at public prices
- Prices are volatile in their movement
- Stop orders:
  - set upper limit on prices for buying – buy for the best available price once the limit is reached
    e.g. stock currently trading at $30 should be bought when the price rises above $35

Stock-Trading System

- A system for selling/buying stock at public prices
- Prices are volatile in their movement
- Stop orders:
  - set upper limit on prices for buying – buy for the best available price once the limit is reached
    e.g. stock currently trading at $30 should be bought when the price rises above $35
  - set lower limit on prices for selling – sell for the best available price once the limit is reached
    e.g. stock currently trading at $30 should be sold when the price sinks below $25

Stock-Trading System

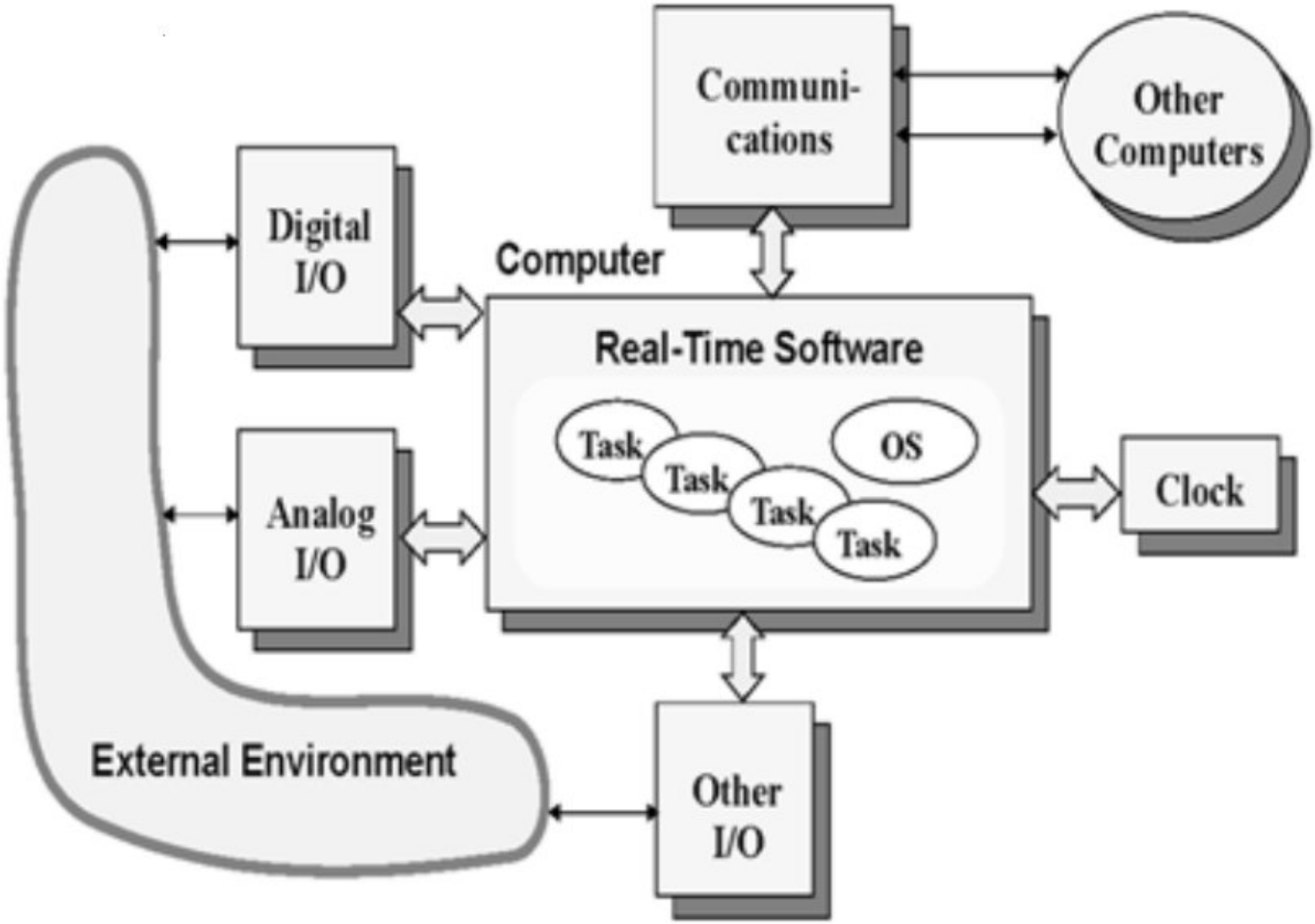
- A system for selling/buying stock at public prices
- Prices are volatile in their movement
- Stop orders:
  - set upper limit on prices for buying – buy for the best available price once the limit is reached

    e.g. stock currently trading at $30 should be bought when the price rises above $35
  - set lower limit on prices for selling – sell for the best available price once the limit is reached

    e.g. stock currently trading at $30 should be sold when the price sinks below $25
- Depending on the delay, the available price may be different from the limit

  successful stop orders depend on the timely delivery of stock trade data and the ability to trade on the changing prices in a timely manner

Structure of Real-Time (Embedded) Applications



Types of Real-Time Systems

- Purely cyclic
  - every task executes periodically; I/O operations are polled; demands in resources do not vary

e.g. digital controllers

Types of Real-Time Systems

- Purely cyclic
  - every task executes periodically; I/O operations are polled; demands in resources do not vary

  e.g. digital controllers
- Mostly cyclic
  - most tasks execute periodically; system also responds to external events (fault recovery and external commands) asynchronously

  e.g. avionics

Types of Real-Time Systems

- Purely cyclic
  - every task executes periodically; I/O operations are polled; demands in resources do not vary

  e.g. digital controllers
- Mostly cyclic
  - most tasks execute periodically; system also responds to external events (fault recovery and external commands) asynchronously

  e.g. avionics
- Asynchronous and somewhat predictable
  - durations between consecutive executions of a task as well as demands in resources may vary considerably. These variations have either bounded range, or known statistics.

  e.g. radar signal processing, tracking

Types of Real-Time Systems

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

Real-Time Systems Failures

- AT&T *long* distance calls
- Therac-25 medical accelerator disaster
- Patriot missile mistiming

AT&T Long Distance Calls

114 computer-operated electronic switches scattered across USA

Handling up to 700,000 calls an hour

The problem:

AT&T Long Distance Calls

114 computer-operated electronic switches scattered across USA

Handling up to 700,000 calls an hour

The problem:

- the switch in New York City neared its load limit

AT&T Long Distance Calls

114 computer-operated electronic switches scattered across USA

Handling up to 700,000 calls an hour

The problem:

- the switch in New York City neared its load limit
- entered a four-second maintenance reset

AT&T Long Distance Calls

114 computer-operated electronic switches scattered across USA

Handling up to 700,000 calls an hour

The problem:

- the switch in New York City neared its load limit
- entered a four-second maintenance reset
- sent “do not disturb” to neighbors

AT&T Long Distance Calls

114 computer-operated electronic switches scattered across USA

Handling up to 700,000 calls an hour

The problem:

- the switch in New York City neared its load limit
- entered a four-second maintenance reset
- sent “do not disturb” to neighbors
- after the reset, the switch began to distribute calls (quickly)

AT&T Long Distance Calls

114 computer-operated electronic switches scattered across USA

Handling up to 700,000 calls an hour

The problem:

- the switch in New York City neared its load limit
- entered a four-second maintenance reset
- sent “do not disturb” to neighbors
- after the reset, the switch began to distribute calls (quickly)
- then another switch received one of these calls from New York

AT&T Long Distance Calls

114 computer-operated electronic switches scattered across USA

Handling up to 700,000 calls an hour

The problem:

- the switch in New York City neared its load limit
- entered a four-second maintenance reset
- sent “do not disturb” to neighbors
- after the reset, the switch began to distribute calls (quickly)
- then another switch received one of these calls from New York
- began to update its records that New York was back on line

AT&T Long Distance Calls

114 computer-operated electronic switches scattered across USA
Handling up to 700,000 calls an hour

The problem:

- the switch in New York City neared its load limit
- entered a four-second maintenance reset
- sent “do not disturb” to neighbors
- after the reset, the switch began to distribute calls (quickly)
- then another switch received one of these calls from New York
- began to update its records that New York was back on line
- a second call from New York arrived less than 10 milliseconds after the first, i.e. while the first hadn’t yet been handled;
  this together with a SW bug caused maintenance reset

AT&T Long Distance Calls

114 computer-operated electronic switches scattered across USA
Handling up to 700,000 calls an hour

The problem:

- the switch in New York City neared its load limit
- entered a four-second maintenance reset
- sent “do not disturb” to neighbors
- after the reset, the switch began to distribute calls (quickly)
- then another switch received one of these calls from New York
- began to update its records that New York was back on line
- a second call from New York arrived less than 10 milliseconds after the first, i.e. while the first hadn’t yet been handled;
  this together with a SW bug caused maintenance reset
- the error was propagated further ....

AT&T Long Distance Calls

114 computer-operated electronic switches scattered across USA
Handling up to 700,000 calls an hour

The problem:

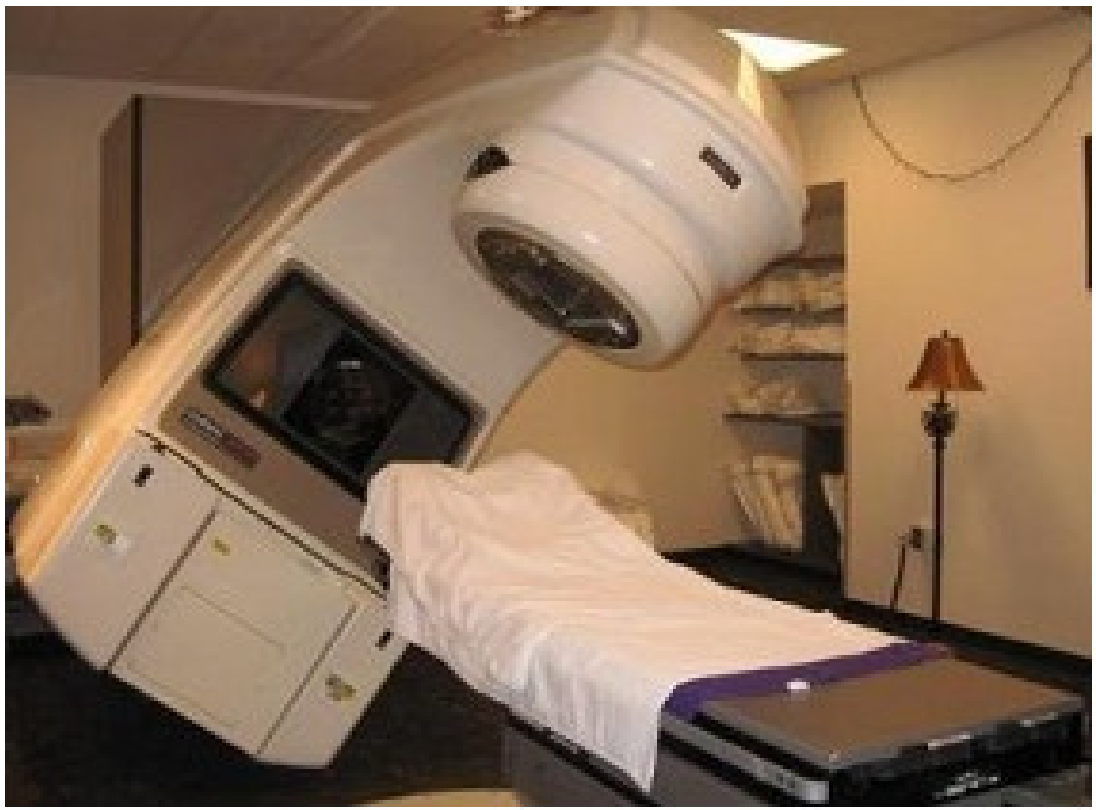
- the switch in New York City neared its load limit
- entered a four-second maintenance reset
- sent “do not disturb” to neighbors
- after the reset, the switch began to distribute calls (quickly)
- then another switch received one of these calls from New York
- began to update its records that New York was back on line
- a second call from New York arrived less than 10 milliseconds after the first, i.e. while the first hadn’t yet been handled;
  this together with a SW bug caused maintenance reset
- the error was propagated further ....

The reason for failure: The system was unable to react to closely timed messages

Therac-25 medical accelerator disaster

Therac-25 = a machine for radiotheratpy

- between 1985 and 1987 (at least) six accidents involving enormous radiation overdoses to patients
- Half of these patients died due to the overdoses



Therac-25 – the modes

1. electron mode
   - electron beam (low current)
   - various levels of energy (5 to 25-MeV)
   - scanning magnets used to spread the beam to a safe concentration

Therac-25 – the modes

1. electron mode
   - electron beam (low current)
   - various levels of energy (5 to 25-MeV)
   - scanning magnets used to spread the beam to a safe concentration
2. photon mode
   - only one level of energy (25-MeV), much larger electron-beam current
   - electron beam strikes a metal foil to produce X-rays (photons)
   - the X-ray beam is "flattened" by a device below the foil

Therac-25 – the modes

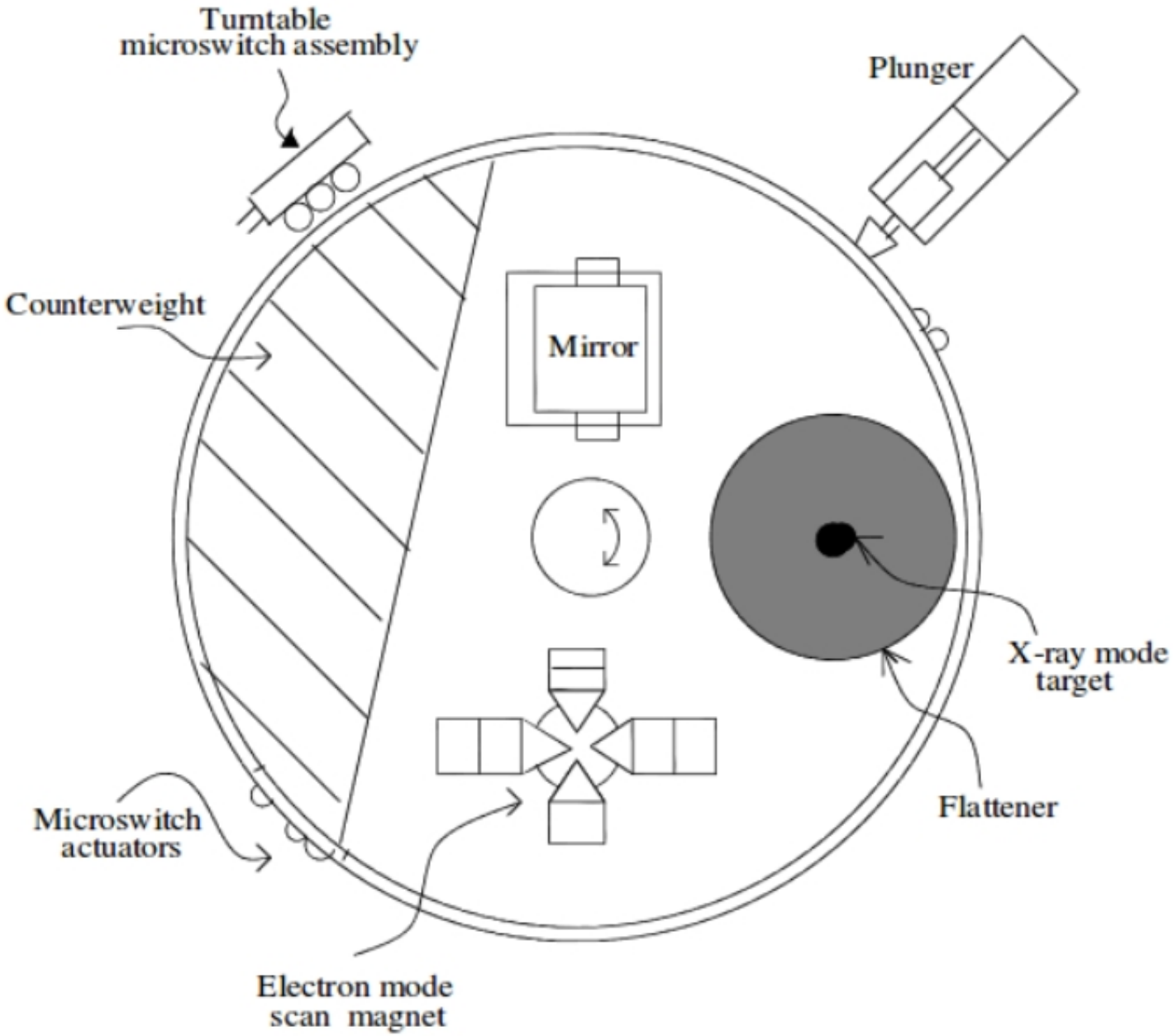
1. electron mode
   - electron beam (low current)
   - various levels of energy (5 to 25-MeV)
   - scanning magnets used to spread the beam to a safe concentration
2. photon mode
   - only one level of energy (25-MeV), much larger electron-beam current
   - electron beam strikes a metal foil to produce X-rays (photons)
   - the X-ray beam is "flattened" by a device below the foil
3. light mode – just light beam used to illuminate the field on the surface of the patient's body that will be treated

Therac-25 – the modes

1. electron mode
   - electron beam (low current)
   - various levels of energy (5 to 25-MeV)
   - scanning magnets used to spread the beam to a safe concentration
2. photon mode
   - only one level of energy (25-MeV), much larger electron-beam current
   - electron beam strikes a metal foil to produce X-rays (photons)
   - the X-ray beam is "flattened" by a device below the foil
3. light mode – just light beam used to illuminate the field on the surface of the patient's body that will be treated

All devices placed on a turntable, supposed to be rotated to the correct position before the beam is started up

Therac-25 – turntable



The Software

The software responsible for

- Operator
  - Monitoring input and editing changes from an operator
  - Updating the screen to show current status of machine
  - Printing in response to an operator commands

The Software

The software responsible for

- Operator
  - Monitoring input and editing changes from an operator
  - Updating the screen to show current status of machine
  - Printing in response to an operator commands
- Machine
  - monitoring the machine status
  - placement of turntable
  - strength and shape of beam
  - operation of bending and scanning magnets
  - setting the machine up for the specified treatment
  - turning the beam on
  - turning the beam off (after treatment, on operator command, or if a malfunction is detected)

The Software

The software responsible for

- Operator
  - Monitoring input and editing changes from an operator
  - Updating the screen to show current status of machine
  - Printing in response to an operator commands
- Machine
  - monitoring the machine status
  - placement of turntable
  - strength and shape of beam
  - operation of bending and scanning magnets
  - setting the machine up for the specified treatment
  - turning the beam on
  - turning the beam off (after treatment, on operator command, or if a malfunction is detected)

Software running several safety critical tasks in parallel!

Insufficient hardware protection (as opposed to previous models)!!

Therac-25 – software

- The Therac-25 runs on a real-time operating system

Therac-25 – software

- The Therac-25 runs on a real-time operating system
- Four major components of software: stored data, a scheduler, a set of tasks, and interrupt services (e.g. the computer clock and handling of computer-hardware-generated errors)

Therac-25 – software

- The Therac-25 runs on a real-time operating system
- Four major components of software: stored data, a scheduler, a set of tasks, and interrupt services (e.g. the computer clock and handling of computer-hardware-generated errors)
- The software segregated the tasks above into
  - critical tasks: e.g. setup and operation of the beam
  - non-critical tasks: e.g. monitoring the keyboard

Therac-25 – software

- The Therac-25 runs on a real-time operating system
- Four major components of software: stored data, a scheduler, a set of tasks, and interrupt services (e.g. the computer clock and handling of computer-hardware-generated errors)
- The software segregated the tasks above into
  - critical tasks: e.g. setup and operation of the beam
  - non-critical tasks: e.g. monitoring the keyboard
- The scheduler directs all non-interrupt events and orders simultaneous events

Therac-25 – software

- The Therac-25 runs on a real-time operating system
- Four major components of software: stored data, a scheduler, a set of tasks, and interrupt services (e.g. the computer clock and handling of computer-hardware-generated errors)
- The software segregated the tasks above into
  - critical tasks: e.g. setup and operation of the beam
  - non-critical tasks: e.g. monitoring the keyboard
- The scheduler directs all non-interrupt events and orders simultaneous events
- Every 0.1 seconds tasks are initiated and critical tasks are executed first, with non-critical tasks taking up any remaining time

Therac-25 – software

- The Therac-25 runs on a real-time operating system
- Four major components of software: stored data, a scheduler, a set of tasks, and interrupt services (e.g. the computer clock and handling of computer-hardware-generated errors)
- The software segregated the tasks above into
  - critical tasks: e.g. setup and operation of the beam
  - non-critical tasks: e.g. monitoring the keyboard
- The scheduler directs all non-interrupt events and orders simultaneous events
- Every 0.1 seconds tasks are initiated and critical tasks are executed first, with non-critical tasks taking up any remaining time

Communication between tasks based on shared variables (without proper atomic test-and-set instructions)

What happened?

There were several accidents due to various bugs in software

What happened?

There were several accidents due to various bugs in software

One of them proceeded as follows (much simplified):

- the operator entered parameters for X-rays treatment

What happened?

There were several accidents due to various bugs in software

One of them proceeded as follows (much simplified):

- the operator entered parameters for X-rays treatment
- the machine started to set up for the treatment

What happened?

There were several accidents due to various bugs in software

One of them proceeded as follows (much simplified):

- the operator entered parameters for X-rays treatment
- the machine started to set up for the treatment
- the operator changed the mode from X-rays to electron (within the interval from 1s to 8s from the end of the original editing)

What happened?

There were several accidents due to various bugs in software

One of them proceeded as follows (much simplified):

- the operator entered parameters for X-rays treatment
- the machine started to set up for the treatment
- the operator changed the mode from X-rays to electron (within the interval from 1s to 8s from the end of the original editing)
- the patient received X-ray “treatment” with turntable in the electron position (i.e. unshielded)

What happened?

There were several accidents due to various bugs in software

One of them proceeded as follows (much simplified):

- the operator entered parameters for X-rays treatment
- the machine started to set up for the treatment
- the operator changed the mode from X-rays to electron (within the interval from 1s to 8s from the end of the original editing)
- the patient received X-ray “treatment” with turntable in the electron position (i.e. unshielded)

The cause:

- The turntable and treatment parameters were set by *different* concurrent procedures HAND and DATENT, respectively.

What happened?

There were several accidents due to various bugs in software

One of them proceeded as follows (much simplified):

- the operator entered parameters for X-rays treatment
- the machine started to set up for the treatment
- the operator changed the mode from X-rays to electron (within the interval from 1s to 8s from the end of the original editing)
- the patient received X-ray “treatment” with turntable in the electron position (i.e. unshielded)

The cause:

- The turntable and treatment parameters were set by *different* concurrent procedures Hand and Datent, respectively.
- If the change in parameters came in the “right” time, only Hand reacted to the change.

Patriot missile mistiming

vs

Patriot missile mistiming

- Patriot – Air defense missile system

Patriot missile mistiming

- Patriot – Air defense missile system
- Failed to intercept a scud missile on February 25, 1991 at Dhahran, Saudi Arabia

  (missile hit US army barracks, 28 persons killed)

Patriot missile mistiming

- Patriot – Air defense missile system
- Failed to intercept a scud missile on February 25, 1991 at Dhahran, Saudi Arabia

  (missile hit US army barracks, 28 persons killed)
- The problem was caused by incorrect measurement of time

Patriot missile mistiming

- Patriot – Air defense missile system
- Failed to intercept a scud missile on February 25, 1991 at Dhahran, Saudi Arabia

  (missile hit US army barracks, 28 persons killed)
- The problem was caused by incorrect measurement of time

Simplified principle of function:

Patriot missile mistiming

- Patriot – Air defense missile system
- Failed to intercept a scud missile on February 25, 1991 at Dhahran, Saudi Arabia

  (missile hit US army barracks, 28 persons killed)
- The problem was caused by incorrect measurement of time

Simplified principle of function:

- Patriot's radar detects an airborne object

Patriot missile mistiming

- Patriot – Air defense missile system
- Failed to intercept a scud missile on February 25, 1991 at Dhahran, Saudi Arabia

  (missile hit US army barracks, 28 persons killed)
- The problem was caused by incorrect measurement of time

Simplified principle of function:

- Patriot's radar detects an airborne object
- the object is identified as a scud missile (according to speed, size, etc.)

Patriot missile mistiming

- Patriot – Air defense missile system
- Failed to intercept a scud missile on February 25, 1991 at Dhahran, Saudi Arabia

  (missile hit US army barracks, 28 persons killed)
- The problem was caused by incorrect measurement of time

Simplified principle of function:

- Patriot's radar detects an airborne object
- the object is identified as a scud missile (according to speed, size, etc.)
- the range gate computes an area in the air space where the system should next look for it

Patriot missile mistiming

- Patriot – Air defense missile system
- Failed to intercept a scud missile on February 25, 1991 at Dhahran, Saudi Arabia

  (missile hit US army barracks, 28 persons killed)
- The problem was caused by incorrect measurement of time

Simplified principle of function:

- Patriot's radar detects an airborne object
- the object is identified as a scud missile (according to speed, size, etc.)
- the range gate computes an area in the air space where the system should next look for it
- finding the object in the calculated area confirms that it is a scud

Patriot missile mistiming

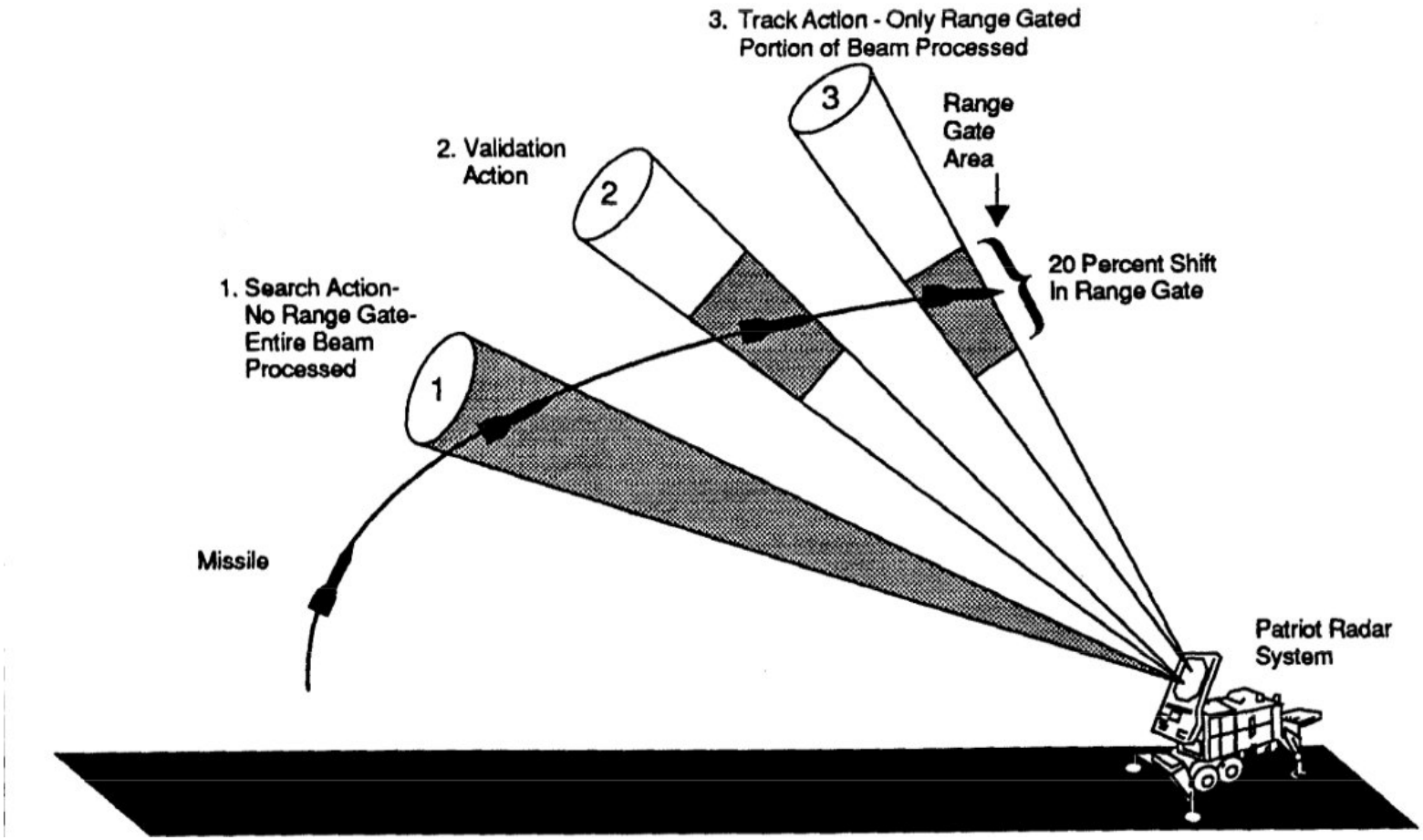
- Patriot – Air defense missile system
- Failed to intercept a scud missile on February 25, 1991 at Dhahran, Saudi Arabia

  (missile hit US army barracks, 28 persons killed)
- The problem was caused by incorrect measurement of time

Simplified principle of function:

- Patriot's radar detects an airborne object
- the object is identified as a scud missile (according to speed, size, etc.)
- the range gate computes an area in the air space where the system should next look for it
- finding the object in the calculated area confirms that it is a scud
- then the scud is intercepted

Patriot Missile Mistiming



Patriot Missile Mistiming

Prediction of the new area:

Patriot Missile Mistiming

Prediction of the new area:

- a function of *velocity* and *time* of the last radar detection

Patriot Missile Mistiming

Prediction of the new area:

- a function of *velocity* and *time* of the last radar detection
- velocity represented as a real number

Patriot Missile Mistiming

Prediction of the new area:

- a function of *velocity* and *time* of the last radar detection
- velocity represented as a real number
- **the current time kept by incrementing whole number counter counting tenths of seconds**

Patriot Missile Mistiming

Prediction of the new area:

- a function of *velocity* and *time* of the last radar detection
- velocity represented as a real number
- **the current time kept by incrementing whole number counter counting tenths of seconds**
- computation in 24bit fixed floating point numbers

Patriot Missile Mistiming

Prediction of the new area:

- a function of *velocity* and *time* of the last radar detection
- velocity represented as a real number
- **the current time kept by incrementing whole number counter counting tenths of seconds**
- computation in 24bit fixed floating point numbers

The time converted to 24bit real number and multiplied with 1/10 represented in 24bit (i.e. the real value of 1/10 was 0.099999905)

Patriot Missile Mistiming

Prediction of the new area:

- a function of *velocity* and *time* of the last radar detection
- velocity represented as a real number
- **the current time kept by incrementing whole number counter counting tenths of seconds**
- computation in 24bit fixed floating point numbers

The time converted to 24bit real number and multiplied with 1/10 represented in 24bit (i.e. the real value of 1/10 was 0.099999905)

- the system was already running for 100 hours, i.e. the counter value was 360000, i.e. $360000 \cdot 0.099999905 = 35999.6568$

Patriot Missile Mistiming

Prediction of the new area:

- a function of *velocity* and *time* of the last radar detection
- velocity represented as a real number
- **the current time kept by incrementing whole number counter counting tenths of seconds**
- computation in 24bit fixed floating point numbers

The time converted to 24bit real number and multiplied with 1/10 represented in 24bit (i.e. the real value of 1/10 was 0.099999905)

- the system was already running for 100 hours, i.e. the counter value was 360000, i.e. $360000 \cdot 0.099999905 = 35999.6568$
- the error was 0.3432 seconds, which means 687 m off MACH 5 scud missile

Patriot Missile Mistiming

Prediction of the new area:

- a function of *velocity* and *time* of the last radar detection
- velocity represented as a real number
- **the current time kept by incrementing whole number counter counting tenths of seconds**
- computation in 24bit fixed floating point numbers

The time converted to 24bit real number and multiplied with 1/10 represented in 24bit (i.e. the real value of 1/10 was 0.099999905)

- the system was already running for 100 hours, i.e. the counter value was 360000, i.e. $360000 \cdot 0.099999905 = 35999.6568$
- the error was 0.3432 seconds, which means 687 m off MACH 5 scud missile
- the problem was not only in wrong conversion but in the fact that at some points correct conversion was used (after incomplete bug fix), so the errors did not cancel out

Patriot Missile Mistiming

Prediction of the new area:

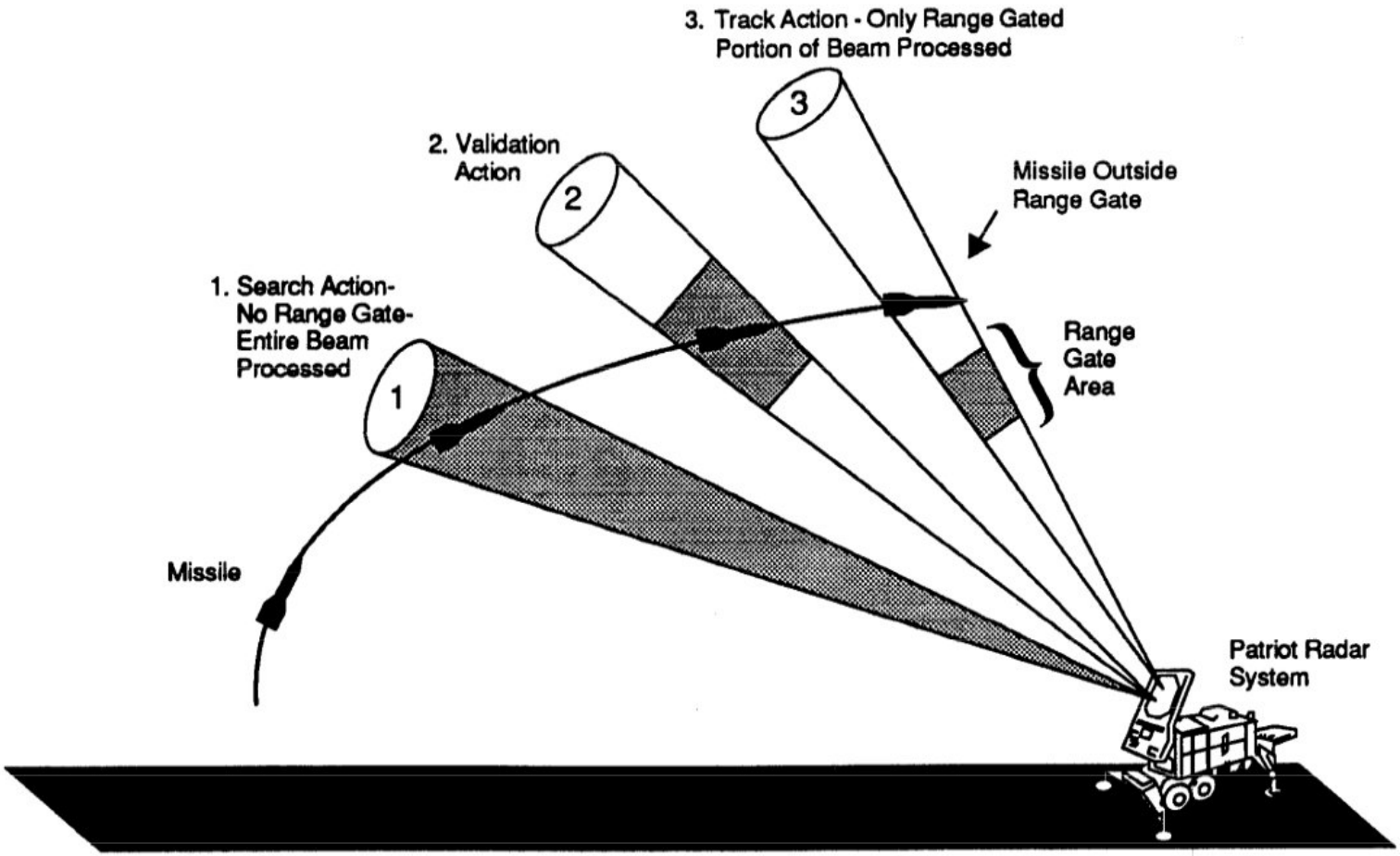
- a function of *velocity* and *time* of the last radar detection
- velocity represented as a real number
- **the current time kept by incrementing whole number counter counting tenths of seconds**
- computation in 24bit fixed floating point numbers

The time converted to 24bit real number and multiplied with 1/10 represented in 24bit (i.e. the real value of 1/10 was 0.099999905)

- the system was already running for 100 hours, i.e. the counter value was 360000, i.e. $360000 \cdot 0.099999905 = 35999.6568$
- the error was 0.3432 seconds, which means 687 m off MACH 5 scud missile
- the problem was not only in wrong conversion but in the fact that at some points correct conversion was used (after incomplete bug fix), so the errors did not cancel out

As a result, the tracking gate looked into wrong area

Patriot Missile Mistiming



Starliner

- Developed by Boeing & NASA
- Seven passengers, or a mix of crew and cargo, for missions to low-Earth orbit

Starliner

- Developed by Boeing & NASA
- Seven passengers, or a mix of crew and cargo, for missions to low-Earth orbit

- A timing issue occured on the last Orbital Flight Test on December 20, 2019

Starliner

- Developed by Boeing & NASA
- Seven passengers, or a mix of crew and cargo, for missions to low-Earth orbit

- A timing issue occured on the last Orbital Flight Test on December 20, 2019
- What is supposed to happen:
  - Atlas V leaves Starliner on a suborbital trajectory.
  - Starliner's own propulsion system takes the spacecraft into orbit and to ISS.

Starliner

- Developed by Boeing & NASA
- Seven passengers, or a mix of crew and cargo, for missions to low-Earth orbit

- A timing issue occured on the last Orbital Flight Test on December 20, 2019
- What is supposed to happen:
  - Atlas V leaves Starliner on a suborbital trajectory.
  - Starliner's own propulsion system takes the spacecraft into orbit and to ISS.
- What happened:
  - Mission Elapsed Timer (MET), or clock, on Starliner was set to the wrong time and did not trigger the engines to fire correctly.
  - Other onboard systems compensated and it reached orbit, but had depleted so much fuel there was not enough to continue the journey.

(Rough) Course Outline

- Real-time scheduling
  - Time and priority driven
  - Resource control
  - Multi-processor (a bit)

(Rough) Course Outline

- Real-time scheduling
  - Time and priority driven
  - Resource control
  - Multi-processor (a bit)
- A little bit on programming real-time systems
  - Real-time operating systems

Outline – Scheduling

The Scheduling problem:

Input:

- available processors, resources
- set of tasks/jobs

  with their requirements, deadlines, etc.

Outline – Scheduling

The Scheduling problem:

Input:

- available processors, resources
- set of tasks/jobs

  with their requirements, deadlines, etc.

Question: How to assign processors and resources to tasks/jobs so that all requirements are met?

Outline – Scheduling

The Scheduling problem:

Input:

- available processors, resources
- set of tasks/jobs

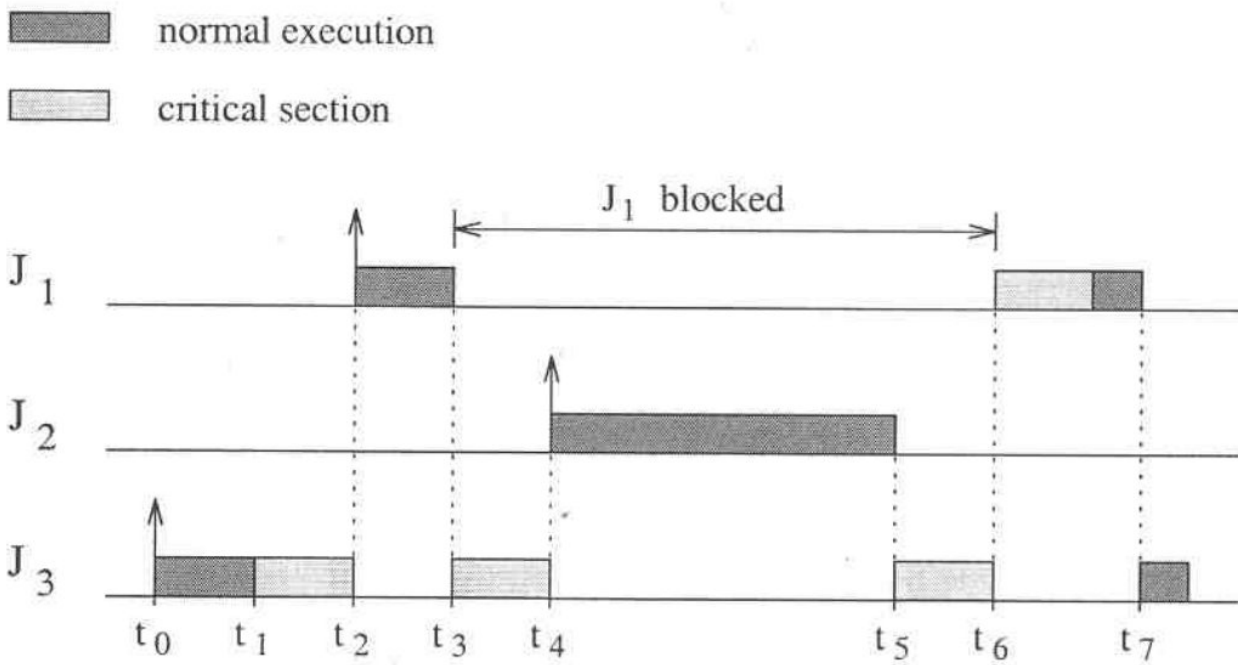
  with their requirements, deadlines, etc.

Question: How to assign processors and resources to tasks/jobs so that all requirements are met?

Example:

- 1 processor, one critical section shared by job 1 and job 3
- job 1: release time 1, computation time 4, deadline 8
- job 2: release time 1, computation time 2, deadline 5
- job 3: release time 0, computation time 3, deadline 4
- ...

Outline – Scheduling

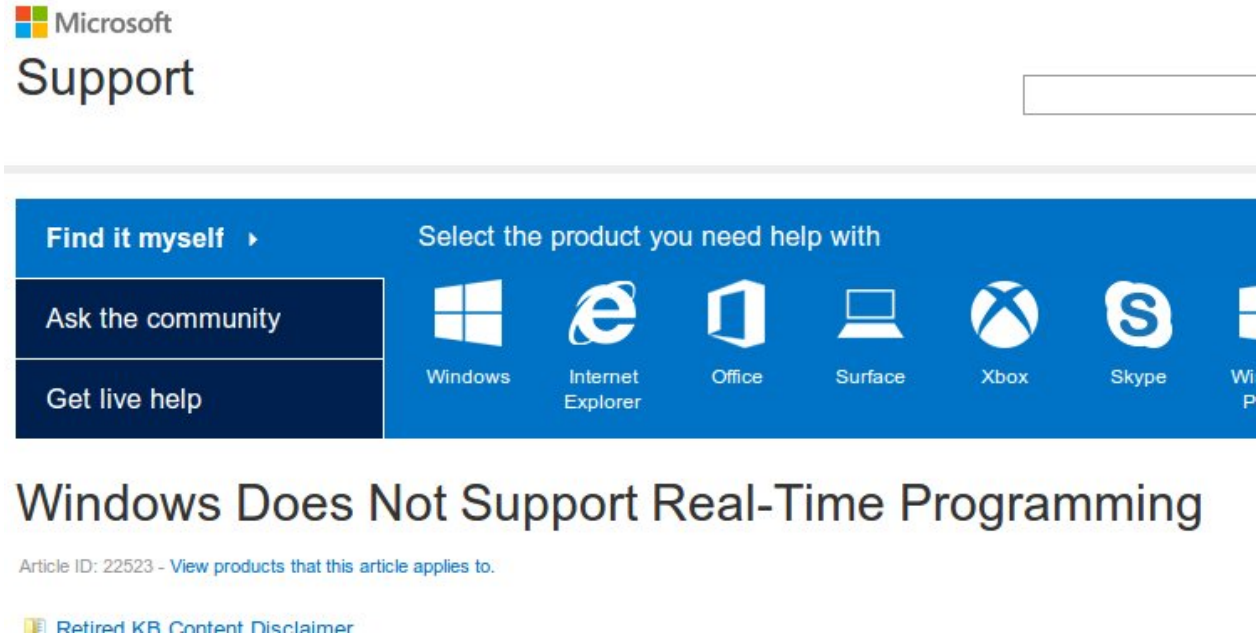


- We consider a formal model of systems with parallel jobs that possibly contend for shared resources

  consider periodic as well as aperiodic jobs
- Consider various algorithms that schedule jobs to meet their timing constraints

  offline and online algorithms, RM, EDF, etc.

Outline – Programming



Basic information about RTOS and RT programming languages

- RTOS – overview
  - real-time in non-real-time operating systems
  - **implementation of theoretical concepts in freeRTOS**
- RT in programming languages – short overview