计算机、通信研究生专业理论课程

Real Time System
实时系统

College of Information Science & Engineering
Qingxu Deng
信息学院：邓庆绪
2009年3月2日
dengqx@mail.neu.edu.cn
Tel: 83690609
Add: Main Building, Room404

Should you take this course?

- You will not learn how to be an embedded software engineer in this course!
- Research-oriented, mostly theory, not programming
- May be useful if you are doing research in these areas:
  - Real-time embedded systems
  - Control theory
  - Systems and networking
    - » Operating systems, wireless sensor networks…
  - Software engineering
  - Ubiquitous/mobile computing

Grading Scheme

- Participation (10%): Participation is based on questions, answers, and active participation in class discussions.
- Class Presentation of papers (30%): Evaluation is based on presentation clarity, technical understanding, identification of contributions and performance in the Q&A.
- Project Proposal (5%)
- Project Presentation (15%)
- Project Report (40%)
- No exams, no homework (except reading and presentation assignments)

The Project

- Proposal due May 20.
- Groups of 1-3 students
- In-class presentation at end of semester
- Cannot be
  - a survey paper
  - SMOP (simple matter of programming) with no research content
- Can be
  - Small, incremental improvement over existing work
  - Negative results OK

During Class (bonus)

- Questions are welcome.
- Insightful questions/answers/discussions will enhance your class participation score.
- Missing class will hurt your participation score
  - Unless you have a good reason and inform me beforehand

Real-Time System Contents

- Chapter 1: Brief Introduction（实时系统的基本概念）
- Chapter 2: The Calculation of WCET（最坏执行时间计算 ）
- Chapter 3: Scheduling Algorithm （常见调度算法 ）
- Chapter 4: Priority scheduling (优先级驱动调度算法)
- Chapter 5: Resource Sharing in Real Time Scheduling(实时调度中的资源共享)
- Chapter 6: Scheduling Algorithm for Multiprocessor(多处理器实时调度算法)
- Chapter 7: Formal Methods Used in Real Time System (实时系统的形式化验证方法)
- Chapter 8: Research on RTOS (实时操作系统的各种研究方法)
- Chapter 9: Case study (案例)

Chapter 1: Brief introduction

- What is a real time System
- Some examples of real time system
- Hard real time vs Soft real time
- Concept and parameters about real time system
- Misconception about real time system
- Challenge about real time system research
- Reference about this chapter

What is a Real-Time System?

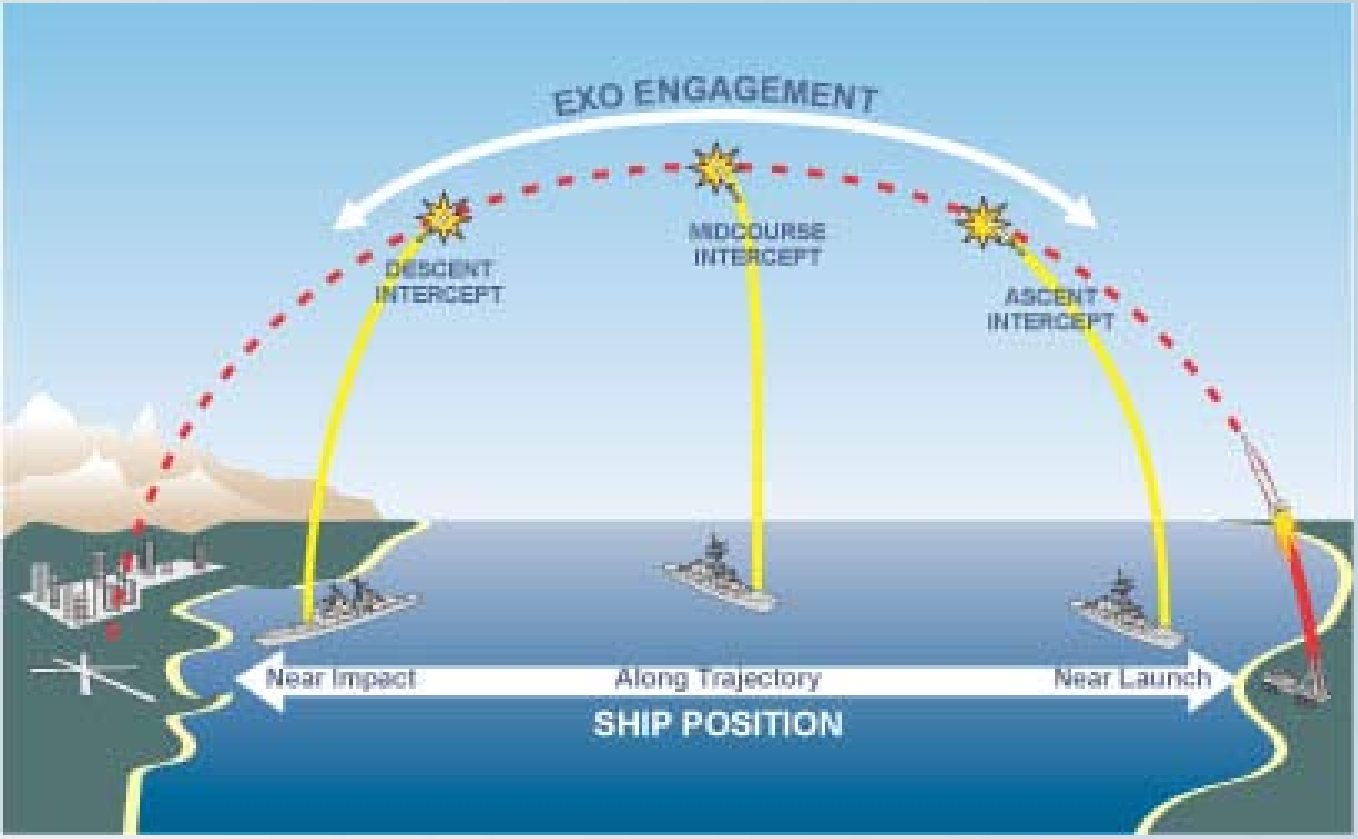
- A real-time system is one in which the correctness of the system depends not only on the logical result of computation, but also on the time at which the results are generated （For more detail see Kang G. Shin reference）
  - J. Stankovic, 1988
- Not necessarily “real-fast”!
  - Predictability is the key
- There was a man who drowned crossing a stream with an **average** depth of six inches
  - J. Stankovic

What is a Real-Time System?

The moment for a bird catching fish
翠鸟潜入水底捕捉小鱼瞬间

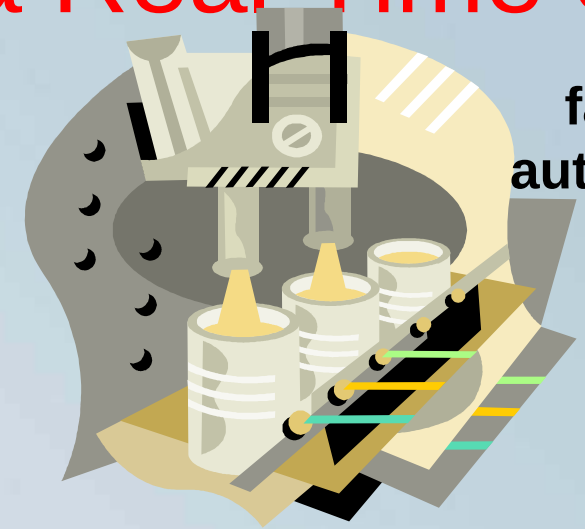
捉到鱼后，翠鸟会快速冲出水面。

What is a Real-Time System?

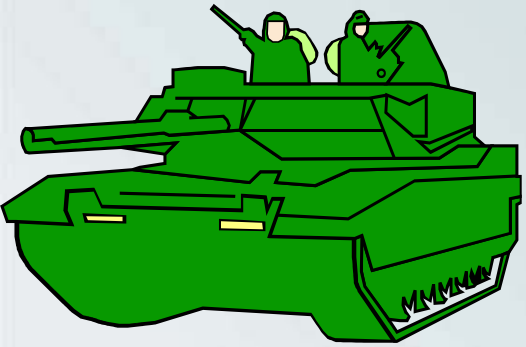


What is a Real-Time System?

spacecraft



factory automation

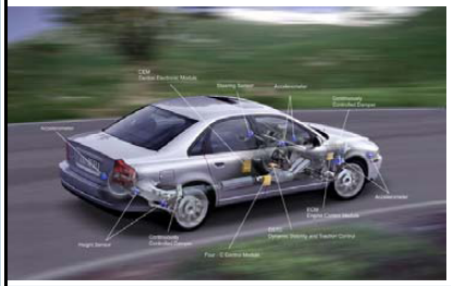


military systems

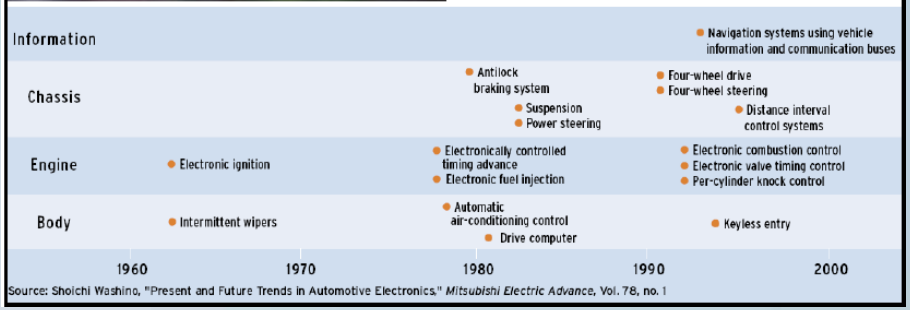
Missile & anti-missile

What is a Real-Time System?

Example Area: Automotive Electronics



- What is "automotive electronics"?
  - Vehicle functions implemented with electronics
    - Body electronics
    - System electronics: chassis, engine
    - Information/entertainment



Source: Shoichi Washino, "Present and Future Trends in Automotive Electronics," *Mitsubishi Electric Advance*, Vol. 78, no. 1

Hard or Soft?

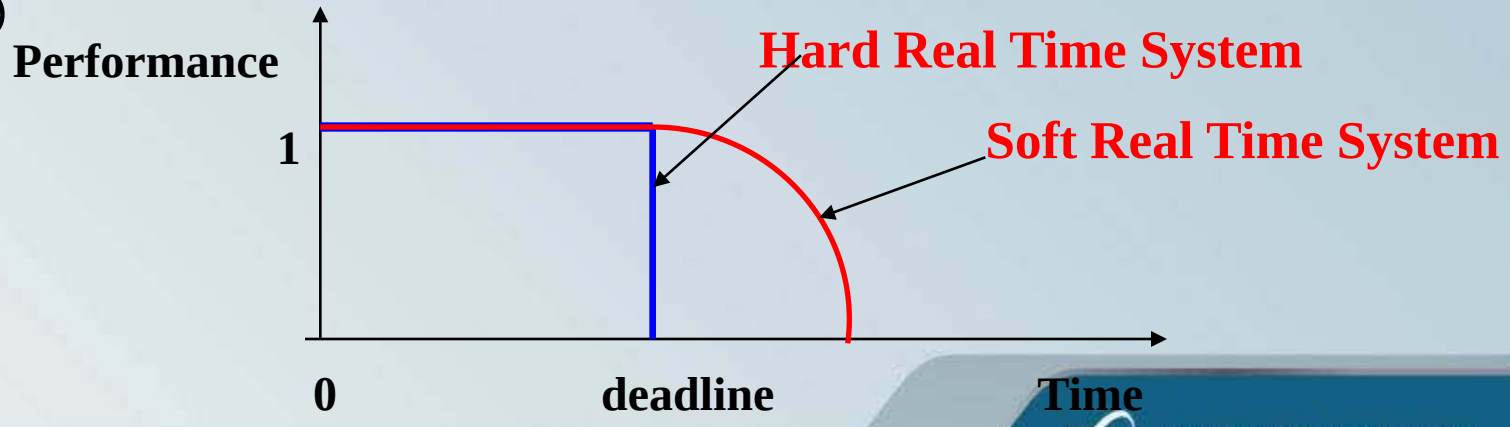
- A real time system can be defined as Hard or Soft real time system

  - ***Hard***: A real-time task is said to be hard, if missing its deadline may cause catastrophic consequences on the environment under control. Examples are sensory data acquisition, detection of critical conditions, actuator servoing.

  - ***Soft***: A real-time task is called soft, if meeting its deadline is desirable for performance reasons, but missing its deadline does not cause serious damage to the environment and does not jeopardize correct system behavior. Examples are command interpreter of the user interface, displaying messages on the screen.

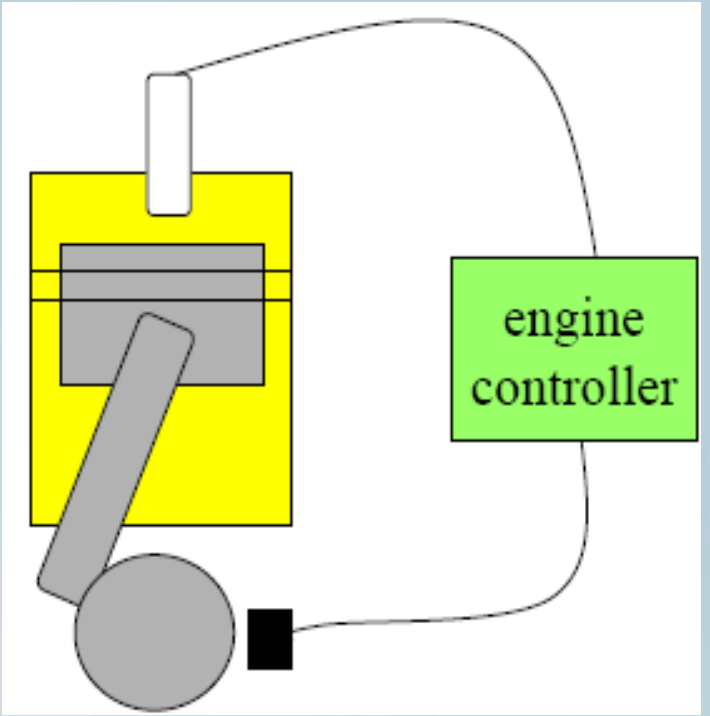
Hard or Soft?

- 硬实时系统是指其时限必须要满足的系统,如不满足将会引起灾难性的后果.譬如发电厂中的汽轮机进汽阀门的控制，核电站控制系统，必须在规定时间内正确控制,否则将会引起灾难性的后果
- 而“软实时系统”在截止期限被错过的情况下，只造成系统性能下降而不会带来严重恶果（网络浏览，媒体播放）



Concept and parameters about real time system

- ***Task***
  - spark control
  - crankshaft sensing
  - fuel/air mixture
  - oxygen sensor
  - Kalman filter – control algorithm



Engine Control System

Task, Job and Process

- ***Task:*** *the term mostly used in real time system (Scheduling system) theory, also commonly used in practice. A task can have many instances for repeated ones, each call its Job. While its implementation through a process .*
- ***Job:*** *an instance of a task*
- ***Process:*** *A process is a unique execution of a program*.

Process

- The concept of ***concurrent processes*** reflects the intuition about the functionality of real time systems.
- Processes help us ***manage timing complexity***:
  - multiple rates
    - multimedia
    - automotive
  - asynchronous input
    - user interfaces
    - communication systems

Task classes

Task classes

- *Period task:*
- *Aperiod task:*
- *Sporadic task:*

classified by the predictability

- *Critical*
- *Noncritical*

by the consequences of not being executed on time

Task classes

- ***Periodic task:*** ***There are many tasks in real time systems that are done repetitively.*** ***For example, the aircraft control system should monitor the speed , altitude, and attitude every 100ms. This sensor information will be used by periodic tasks. Called*** ***Periodic task***
- ***Aperiodic task:*** ***in contrast, many tasks that occur only occasionally.*** ***For instance, when the pilot wishes to execute a turn, a large number of subtasks associated with that action are set off. By the very nature, aperiodic tasks cannot be predicted. Called*** ***aperiodic task***

Task classes

- ***Sporadic task:*** ***For some aperiodic tasks, even it is not predictable, but sufficient computing power must be held in reserve to execute them in a timely fashion. Aperiodic task with a bounded inter-arrival time are called sporadic tasks.***

Task classes

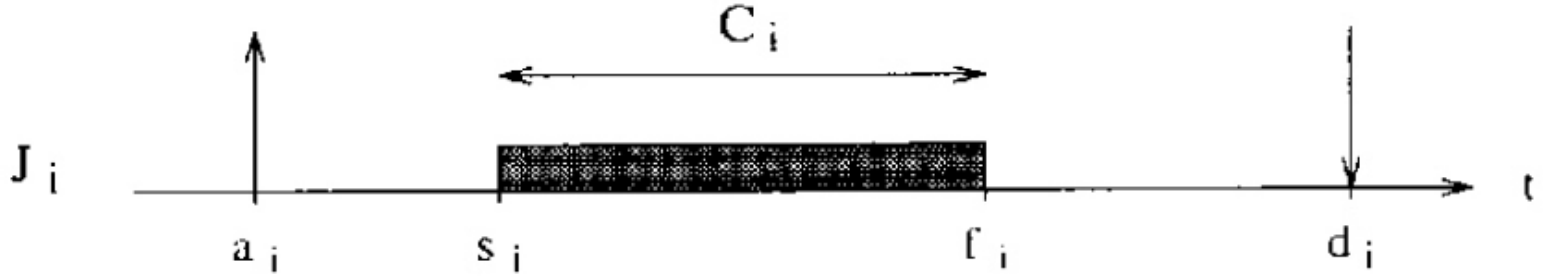
- ***Critical tasks: Critical task are those whos timely execution is critical; if deadline are missed, catastrophes occur. For example, the aircraft control system, nuclear control system, life-support systems.***
- ***Noncritical tasks: Noncritical real time (or soft real time) tasks are ,as the name implies, not critical to the application. However, they do deal with time-varying data and hence they are useless if not completed within a deadline. The goal in scheduling these tasks is thus to maximize the percentage of jobs successfully executed within their deadline.***

Classified by the consequences of not being executed on time

parameters about a task

- ***Arrival time*** or ***release time*** is the time at which a task becomes ready for execution.
- ***Computation time*** is the time necessary to the processor for executing the task without interruption.
- ***Deadline*** is the time at which a task should be completed.
- ***Start time*** is the time at which a task starts its execution.
- ***Finishing time*** is the time at which a task finishes its execution.

parameters about a task

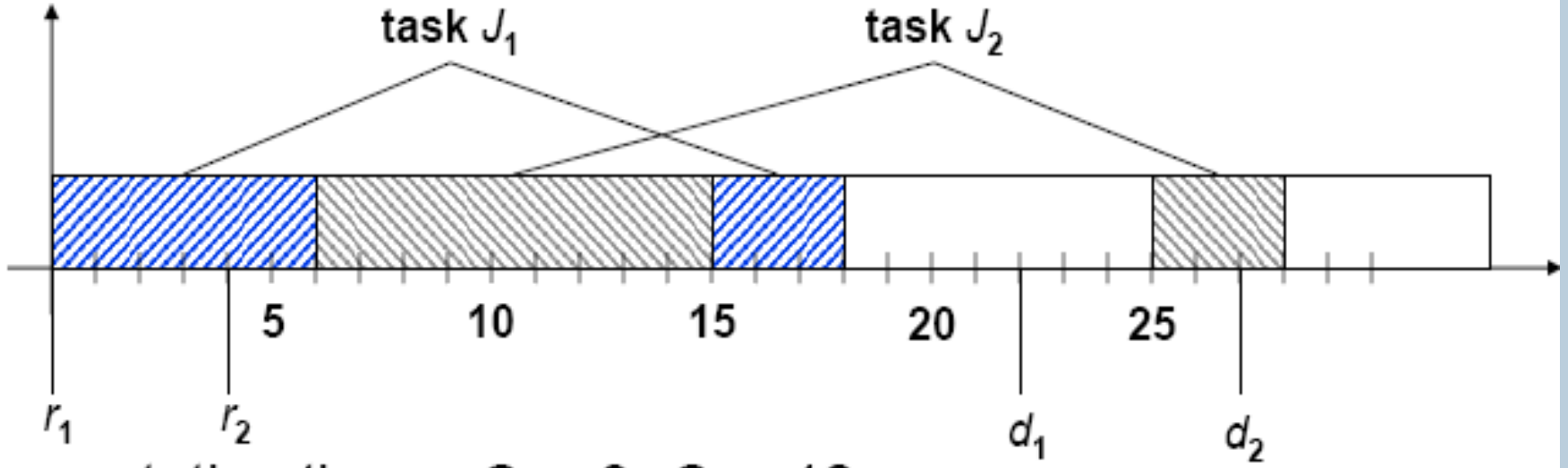


- Using the above definitions, we have $d_i \geq r_i + C_i$
- ***Lateness*** $L_i = f_i - d_i$ represents the delay of a task completion with respect to its deadline; note that if a task completes before the deadline, its lateness is negative.
- ***Tardiness or exceeding time*** $E_i = \max(0, L_i)$ is the time a task stays active after its deadline. 拖延（超出）时间
- ***Laxity or slack time*** $X_i = d_i - a_i - C_i$ is the maximum time a task can be delayed on its activation to complete within its deadline. 松弛时间

parameters about a task

- ***Response time*** is the time taken to finish a task. Ri=Fi-Ri. Some time, it is the addition of the time that was preempted by higher priority task and the Computation time Ri=Ii+Ci.
- ***Utilization*** is a parameter to calculate the proportion of a task to occupy the processor. Ui=Ci/Di or Ui=Ci/Ti

An example



Computation times: $C_1 = 9$, $C_2 = 12$

Start times: $s_1 = 0$, $s_2 = 6$

Finishing times: $f_1 = 18$, $f_2 = 28$

Lateness: $L_1 = -4$, $L_2 = 1$

Tardiness: $E_1 = 0$, $E_2 = 1$

Laxity: $X_1 = 13$, $X_2 = 11$

Misconceptions I

- There is no science in real-time system design.
  - True 20 years ago, but as complexity grows, more and more theory is needed.
    - Real-time scheduling, formal verification, etc.
- Advances in hardware (Moore's Law) will take care of real-time requirements.
  - Other constraints (power, cost, reliability...) preclude using the latest and fastest chips in RTE systems
  - Worst-case timing is important, not average case
    - Modern CPU features, like cache, pipeline, super-scalar make worst-case guarantees difficult.

Misconceptions II

- Real-time computing is equivalent to fast computing.
  - Average-case vs. worst-case
  - A man was drawn in a average 1.2m depth lake!
- Real-time programming is assembly coding, priority interrupt programming, and device driver writing.
  - Assembly is OUT, except in high-performance DSP programming
  - Low-level programming unavoidable, but only a small portion.

Misconceptions III

- Real-time-systems research is performance engineering.
  - And a lot more!
    - Software engineering, distributed systems, programming languages, formal methods....
- The problems in real-time-system design have all been solved in other areas of computer science or operations research.
  - Queuing theory: average, not worst-case behavior
  - OR: one-shot, not recurring, periodic tasks.

Misconceptions IV

- It is not meaningful to talk about absolute guarantees, because we cannot guarantee that the hardware will not fail and the software is bug free
  - If you get on a plane, you have a 0.00001% probability of dying in a crash. It's a LOT better than a 1% probability!
- Real-time systems function in a static environment.
  - More and more dynamic, as complexity grows.

Research Challenges I

- Specification and verification.
  - Incorporate TIME
  - Lies between synchronous and asynchronous systems
  - We need quantitative analysis (deadlines, repetition rates) rather than the qualitative analysis (eventual satisfaction) that is typically handled by current verification techniques. Multicore or multiprocessor lead to much complicated problems!
  - Tackle state-space explosion.
- Real-time scheduling theory
  - Well-studied field of research
  - **Multicore or Multiprocessor bring in too many challenges**

Research Challenges II

- Real-Time Operating Systems
  - Hundreds of them

    Is there one support multicore?
  - Examples:
    - For hard real-time: VxWorks from WindRiver
    - For soft real-time: Windows CE from Microsoft
    - Many others for specialized application domains
- Real-time programming languages and design methodology.
  - C/C++, Java (real-time garbage collection), Ada (dead?), assembly
  - Support for the management of time.
  - Schedulability check.
  - Reusable real-time software modules.
  - Support for distributed programs and fault tolerance.

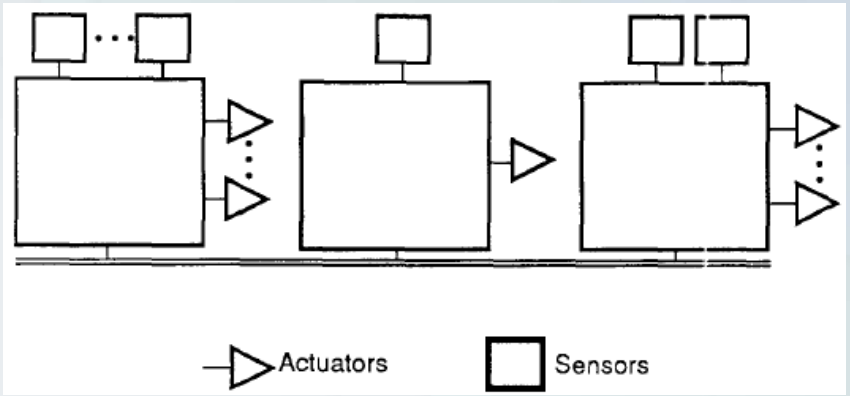
    Is there one support multicore?

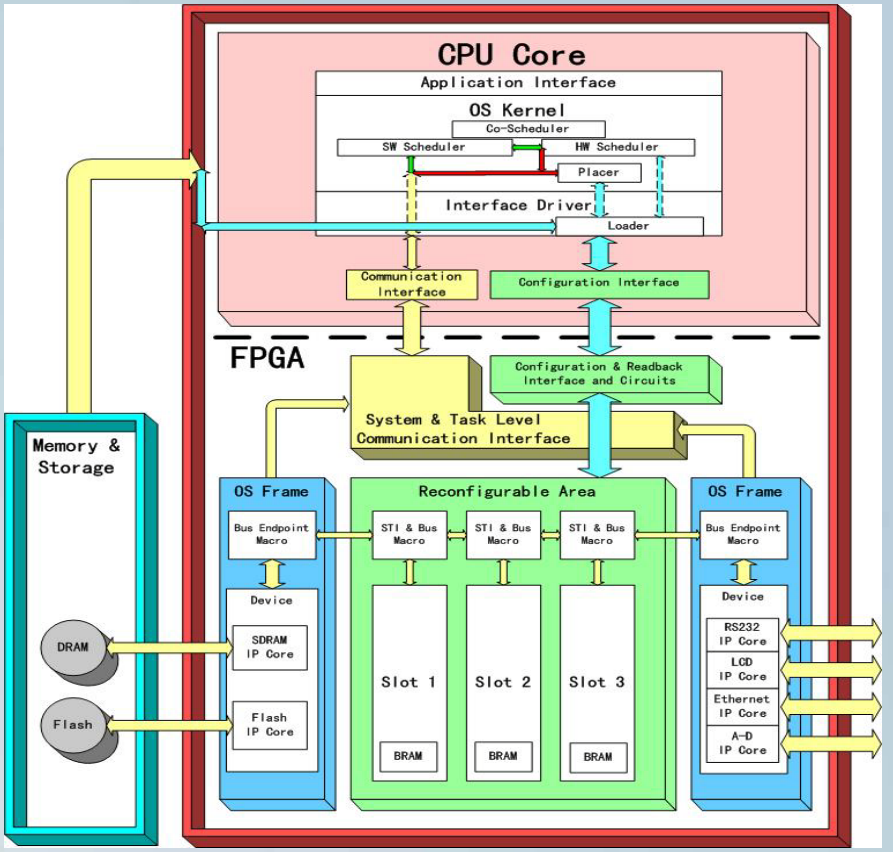
Research Challenges III

- Distributed real-time databases
  - Global serialization criteria need to be relaxed
  - Real-time concurrency control
- Artificial intelligence
  - AI relies on search, which is often NP-complete
  - Tradeoff between accuracy of results and timeliness
- Fault tolerance
  - Error/exception handling must consider timing constraints

Research Challenges IV

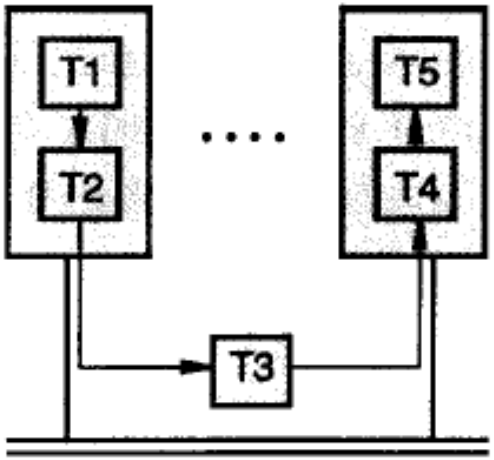
- Real-time-system architectures
  - More and more distributed
  - Distributed systems issues
  - New architecture: Multicore or reconfigurable system





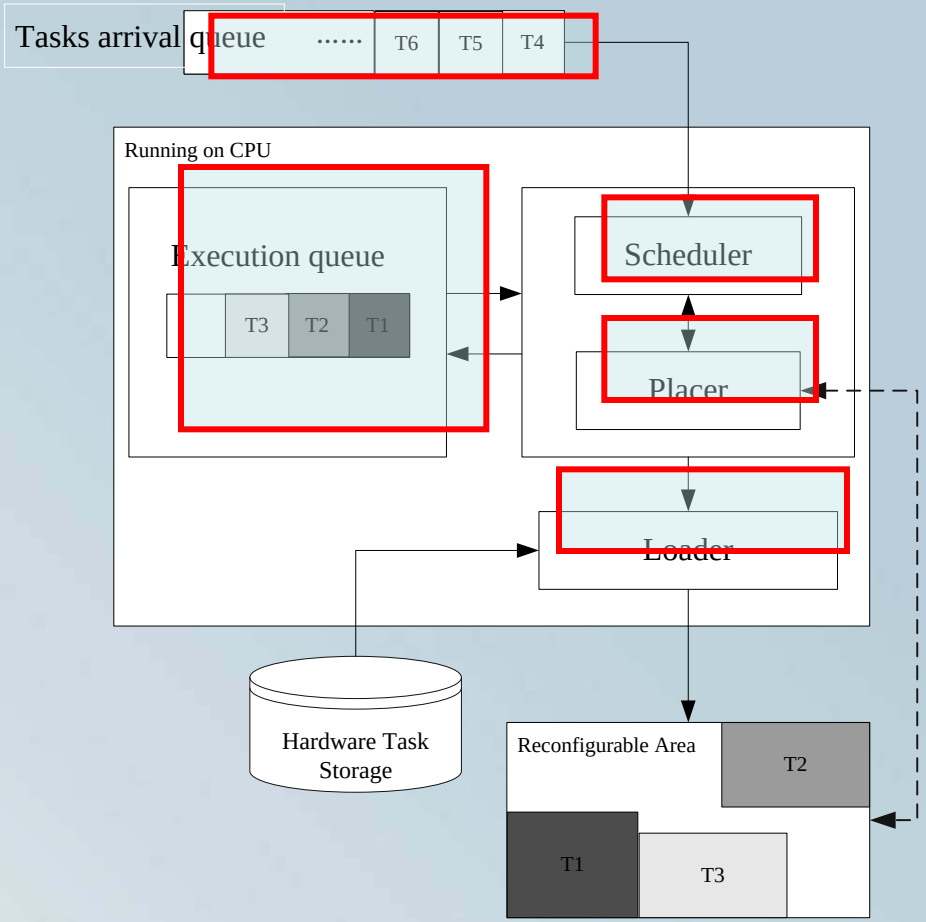
Research Challenges V

- Real-time communications
  - End-to-end timing constraints
  - Dynamic routing solutions
  - Network buffer management
  - Fault-tolerant communications.
  - Network scheduling that can be combined with processor scheduling to provide system-level scheduling solutions.
  - NOC for multicore



Hybrid system os

- HW Task Manager：
  硬件任务管理
  - Scheduler:
    任务调度器
  - Placer：
    任务放置及FPGA资源管理
  - Loader：
    硬件任务放置器
  - Others:
    辅助数据结构



Reference

- Book

《Real-time Systems》，Jane W.S. Liu，高等教育出版社，2002年

《Real-time Systems》，C.M.Krishna and Kang G.Shin，清华大学出版社，2004年

实时系统 (美)Jane W.S. Liu著 姬孟洛 ... [等] 译　高等教育出版社，2003

Reference

paper

- C.L.Liu, Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment
- J. A. Stankovic, UMass, "Misconceptions about real-time computing: a serious problem for next-generation systems, IEEE Computer, Vol 21, Issue 10, Oct 1998
- Liu Sha, etc. Real time scheduling theory: A historical perspective, Real time system 28,101-155, 2004
- Nan Guan, Wang Yi, Zonghua Gu, Qingxu Deng and Ge Yu, Improved Schedulability Analysis of Non-preemptive Scheduling on Multiprocessor Platforms, The 29th IEEE Real-Time Systems Symposium （RTSS2008） Barcelona, Spain.
- Nan Guan, Qingxu Deng, Zonghua Gu, Wenyao Xu, Ge Yu. “Schedulability Analysis of Preemptive and Non-preemptive EDF on Partially Runtime Reconfigurable FPGAs”, ACM Transactions on Design Automation of Electronic Systems (TODAES). (2008 Vol.13, No.4 Article 56:1-43)

Reference

- **Proceeding**
- **AREA: Hardware and Architecture**
- **Rank 1:**
- **ASPLOS: Architectural Support for Prog Lang and OS**
- **ISCA: ACM/IEEE Symp on Computer Architecture**
- **ICCAD: Intl Conf on Computer-Aided Design**
- **DAC: Design Automation Conf**
- **Rank 2:**
- **ISSS: International Symposium on System Synthesis**
- **DATE: IEEE/ACM Design, Automation & Test in Europe Conference**
- **Rank 3:**
- **ICA3PP: Algs and Archs for Parall Proc**
- **Unranked:**
- **International Symposium on System Synthesis**
- **International Symposium on Computer Design**
- **Asia Pacific Design Automation Conference**

Reference

- **Proceeding**
- **AREA: Programming Languages and Software Engineering**
- **Rank 1:**
- **FM/FME: Formal Methods, World Congress/Europe**
- **CAV: Computer Aided Verification**
- **AREA: Algorithms and Theory**
- **Rank 1:**
- **SPAA: ACM Symp on Parallel Algorithms and Architectures**
- **Rank 2:**
- **EMSOFT**

谢谢！

谢谢！