Soft Real-Time Systems

EECE 494 – Real-Time Systems Design

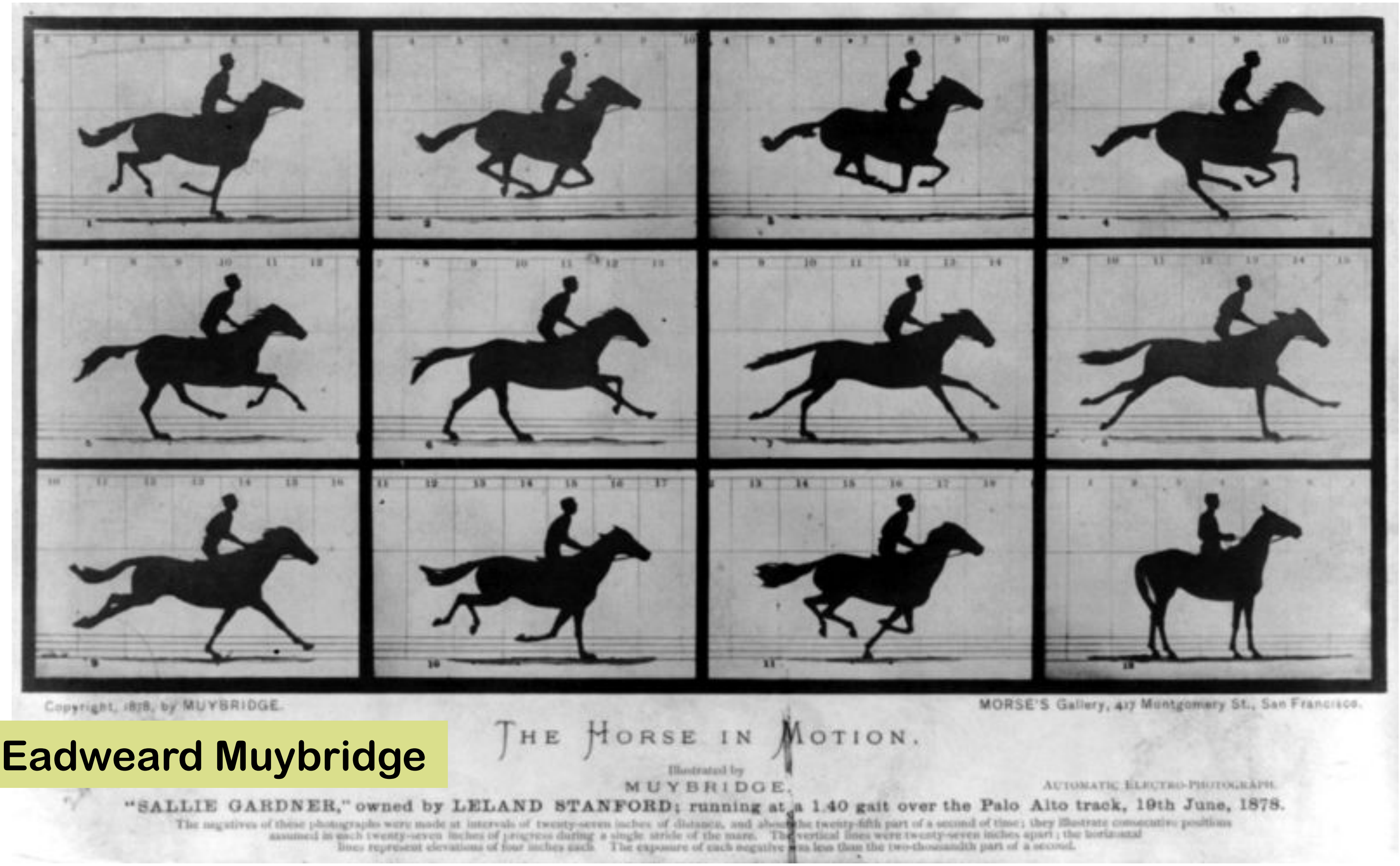
Elastic scheduling of real-time tasks

Ref: Elastic task model for adaptive rate control. Buttazzo, Lipari and Abeni. IEEE Real-Time Systems Symposium, 1998.

Many models for soft real-time systems

- Why?
- There are a variety of soft real-time systems.
- And there are a variety of ways in which their behaviour can be altered.
  - In a multimedia system…
    - Frames can be dropped, i.e., jobs can be skipped.
    - Alternatively, it is possible to reduce the frame rate.
      - ☐ Move from 60 fps to 30 fps or to 24 fps.
      - ☐ We are changing the periodicity of a task.

The elastic task model provides a natural abstraction for such tasks.



Eadweard Muybridge

Elastic task scheduling

- **What if we need all jobs to meet their deadlines... but we can allow tasks to execute at lower frequencies (higher periods)?**

- **Then we need to understand**
  - When should we increase task periods?
  - And increase the periods by what extent?
  - And for which tasks?

Task model for elastic scheduling

- For each task T_i :
  - There is an operating period range $[P_{i,min}, P_{i,max}]$
    - $P_{i,min}$ is the best possible period
  - There is also a value $P_{i,0}$ that represents a nominal period
    - This nominal period is preferred when we can not run the task at the best possible period

- For a hard real-time task:
    - $P_{i,min} = P_{i,max} = P_{i,0}$
- If $P_{i,0} = P_{i,max}$ and $P_{i,min} << P_{i,max}$, the task is very flexible (highly elastic)
- Only periods can be varied; execution times are constant

Task model for elastic scheduling

- **For each task T_i :**
  - There is also an elasticity factor k_i that represents the flexibility of the task

Additional parameters

Task	e_i	$P_{i,0}$	$P_{i,min}$	$P_{i,max}$	k_i
T_1	10	20	20	25	1
T_2	10	40	40	50	1
T_3	15	70	35	80	1

Schedulability analysis

- **We will assume utilization bounds are used for testing schedulability.**
  - Depending on the scheduling policy (EDF or RM), we know the utilization bound U_b
  - The goal of the elasticity is to ensure that the utilization of the set of tasks does not exceed U_b
  - Of course, we need to scale down task utilizations if the processor utilization threatens to exceed U_b (for instance, when we add a new task)
  - How do we scale down task utilizations?

An example

Task	e_i	$P_{i,0}$	$P_{i,min}$	$P_{i,max}$	k_i
T_1	10	20	20	25	1
T_2	10	40	40	50	1
T_3	15	70	35	80	1

- **Is this task set schedulable using EDF?**
  - If each task were to use its nominal period then:
    - $U = 10/20 + 10/40 + 15/70 = 0.964 < 1$.
  - To improve the QoS for T_3, we would like to use a period of 50 (between 35 and 80):
    - $U = 10/20 + 10/40 + 15/50 = 1.05 > 1$.
  - We could adjust the periods of T_1 (set to 22) and T_2 (set to 45):
    - $U = 10/22 + 10/45 + 15/50 = 0.977$.

The adjustment seems *ad hoc*. Can we systematically adjust task periods?

Period adjustment (rate adaptation)

- **At time instant t, let us suppose that task T_i is operating with period P_i**
  - Some of the tasks could be operating at the highest possible period $P_i = P_{i,max}$
  - We cannot increase the periods of these tasks
  - Denote this *set of tasks* by *M*

$$U_M = \sum_{T_i \in M} \frac{e_i}{P_{i,max}}$$

If the utilization bound is U_b, then the remaining tasks – those not in M – cannot have a combined utilization greater than U_b-U_M.

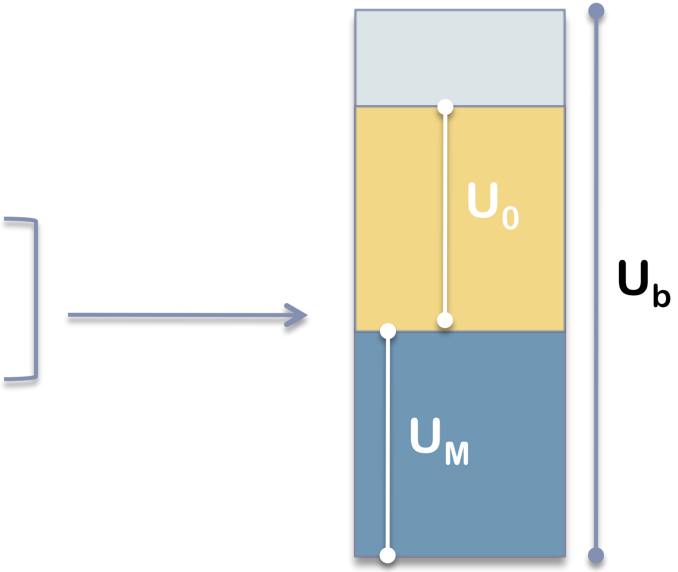
Period adjustment (rate adaptation)

- Let the set of tasks with variable/ adjustable periods be *V*.
- U_0 is the combined nominal utilization of tasks in set *V*.

If the utilization bound is U_b, then the remaining tasks – those not in M – cannot have a combined utilization greater than U_b-U_M.

$$U_0 = \sum_{T_i \in V} \frac{e_i}{P_{i,0}}$$

- If $U_0 = U_b - U_M$, set the periods of all tasks in V to their nominal periods.
- If $U_0 < U_b - U_M$, we can improve the QoS for some tasks.
- If $U_0 > U_b - U_M$, we need to reduce the QoS for some tasks.

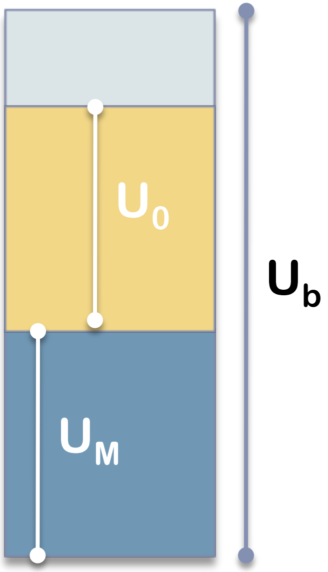


Period adjustment (rate adaptation)

- **If $U_0 < U_b - U_M$, we can improve the QoS for some tasks.**

$$U_0 = \sum_{T_i \in V} \frac{e_i}{P_{i,0}}$$

$$U_M = \sum_{T_i \in M} \frac{e_i}{P_{i,max}}$$



- **How do we adjust task utilizations?**
  - **Use the elasticity coefficients, k_i.**
- **The excess capacity is $[U_b - U_M] - U_0$. We need to distribute this among tasks in V.**

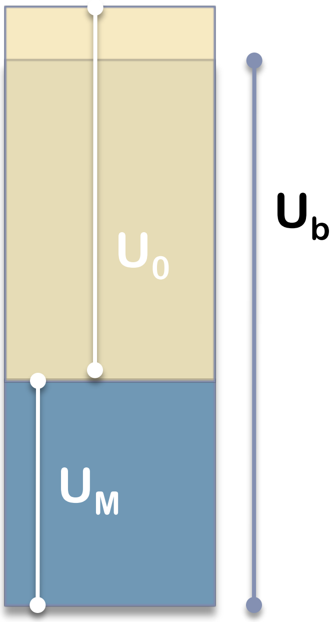
$$\forall T_i \in V : U_i = U_{i,0} + (U_b - U_M - U_0)\frac{k_i}{\sum_{T_j \in V} k_j}$$

Period adjustment (rate adaptation)

- **What if $U_0 > U_b - U_M$?**
- **We do not need a new solution!**

$$U_0 = \sum_{T_i \in V} \frac{e_i}{P_{i,0}}$$

$$U_M = \sum_{T_i \in M} \frac{e_i}{P_{i,max}}$$



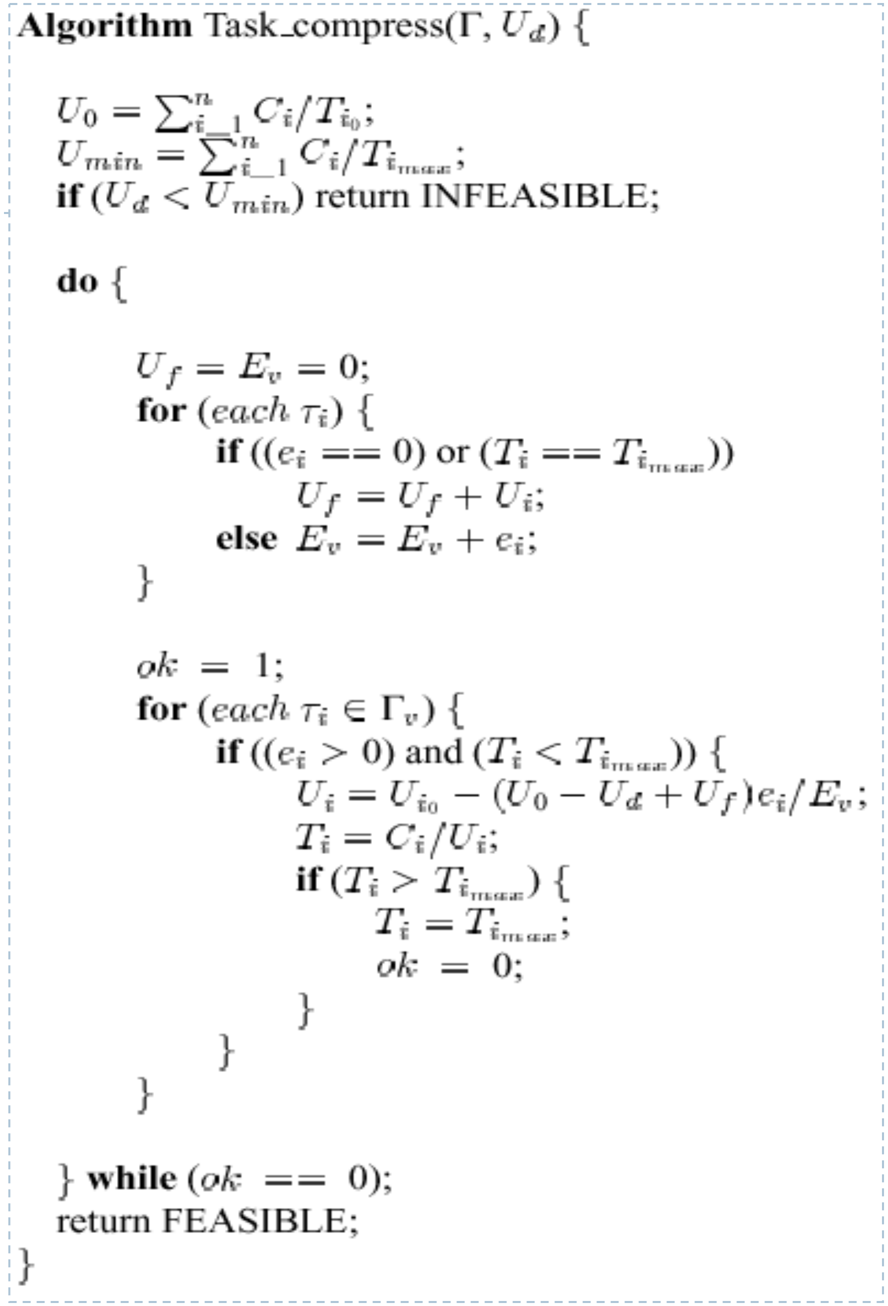
$$\forall T_i \in V : U_i = U_{i,0} + (U_b - U_M - U_0)\frac{k_i}{\sum_{T_j \in V} k_j}$$

Period adjustment (rate adaptation)

- **What do we do if the adjustment causes a task T_i to have a period greater then $P_{i,max}$?**

  - We have to set the period of that task to $P_{i,max}$,
  - Add this task to set M,
  - Remove the task from set V,
  - And try to adjust the periods once more for the task in V.

Detailed algorithm

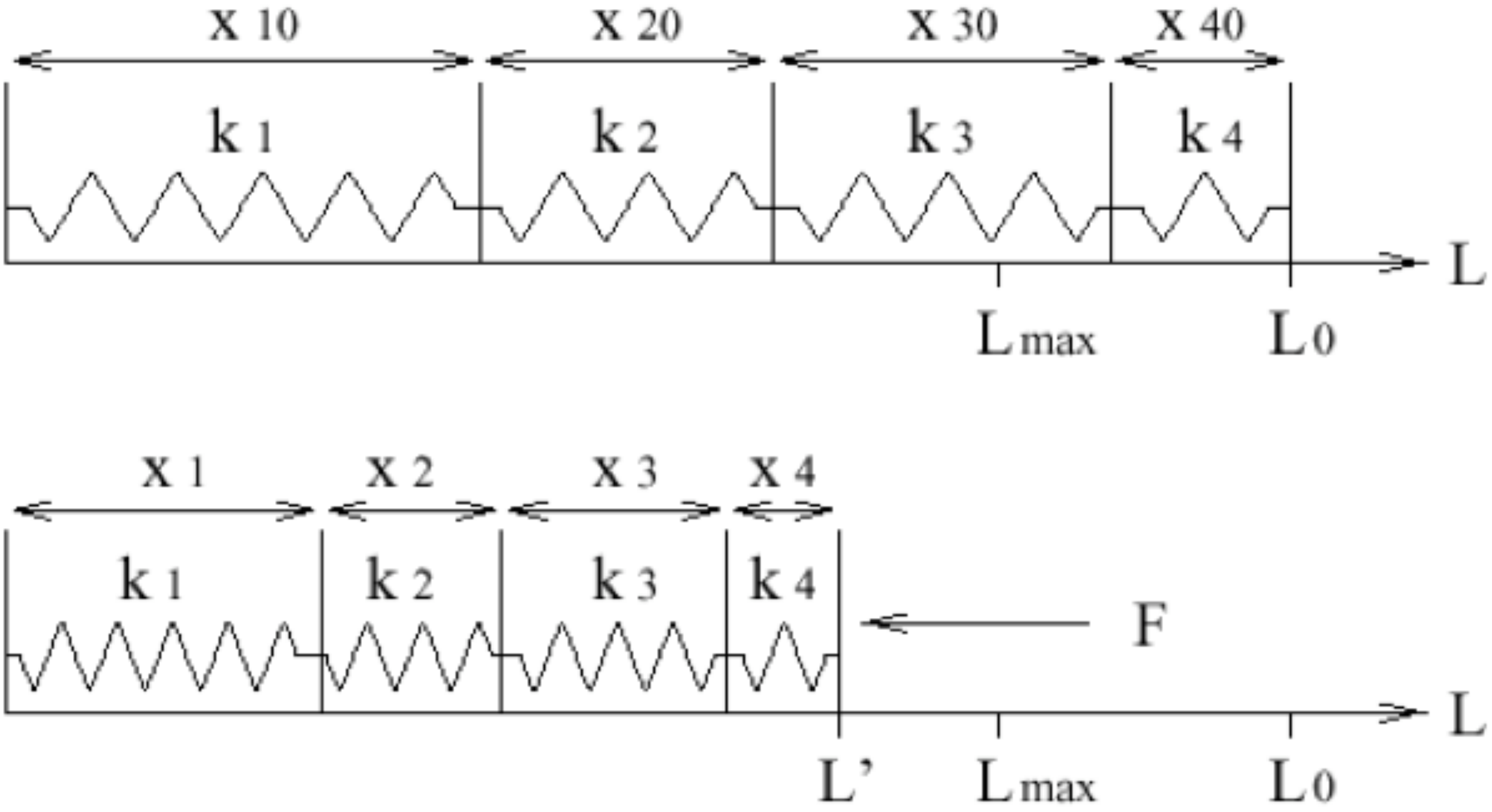


```
Algorithm Task_compress($\Gamma, U_d$) {

  $U_0 = \sum_{i=1}^{n} C_i/T_{i_0}$;
  $U_{min} = \sum_{i=1}^{n} C_i/T_{i_{max}}$;
  if ($U_d < U_{min}$) return INFEASIBLE;

  do {

      $U_f = E_v = 0$;
      for (each $\tau_i$) {
          if (($e_i == 0$) or ($T_i == T_{i_{max}}$))
              $U_f = U_f + U_i$;
          else $E_v = E_v + e_i$;
      }

      $ok = 1$;
      for (each $\tau_i \in \Gamma_v$) {
          if (($e_i > 0$) and ($T_i < T_{i_{max}}$)) {
              $U_i = U_{i_0} - (U_0 - U_d + U_f) e_i / E_v$;
              $T_i = C_i/U_i$;
              if ($T_i > T_{i_{max}}$) {
                  $T_i = T_{i_{max}}$;
                  $ok = 0$;
              }
          }
      }

  } while ($ok == 0$);
  return FEASIBLE;
}
```

Elastic task model for adaptive rate control. Buttazzo, Lipari and Abeni. RTSS 1998.

The elasticity analogy

Do not worry about the terminology here.



Compression and decompression

- **When workload reduces, which may be because a task is complete and is removed from the task set, other tasks can expand or return to their nominal utilizations.**

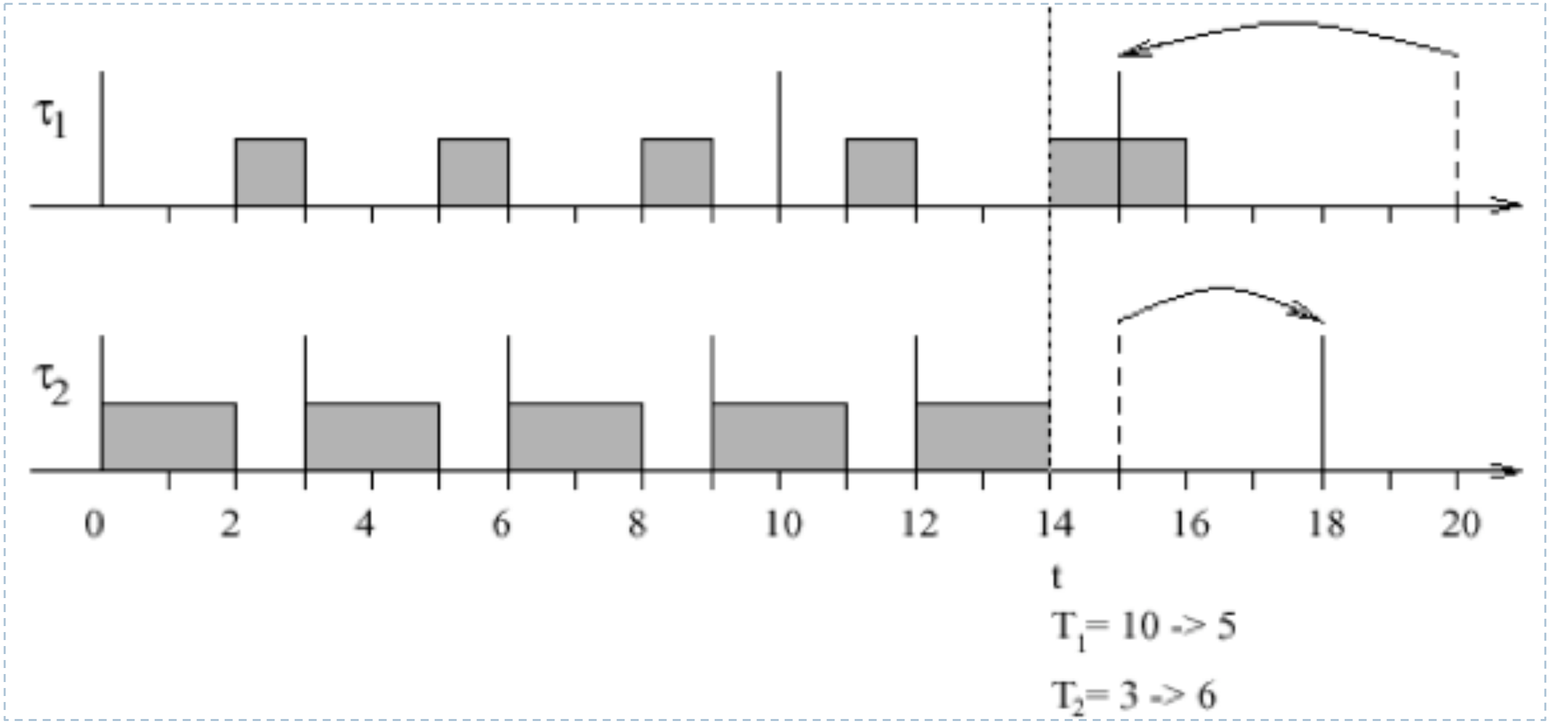
- **Compression/decompression refer to the utilization of the task.**
  - Increasing the period compresses the utilization.
  - Decreasing the period expands the utilization.

When do we compress/decompress?

- **Can we increase or decrease the period of a task at any time instant?**
  - Periods can be increased at any time (immediately).
  - Periods can be decreased (utilization increased) only at the next release time of the task.
  - [If you want the detailed proof, consult the reference article.]

Why decompress at specific instants?

- Originally, $U = 3/10 + 2/3 = 0.9666 < 1$.
- After changing periods at t=14: $U = 3/5 + 2/6 = 0.9666 < 1$.
- If the period of T_1 is changed at once (at t=14), T_1 misses a deadline.



Highlights

- **The elastic task model**

  - Allows period (rate) adaptation in a real-time system.
  - Analogous to physical spring systems.

  - Like skip-based scheduling, elastic scheduling is suitable for multimedia applications.
  - Also useful in manufacturing applications.
    - Silicon wafers processed in a semiconductor plant.
    - We can reduce the rate of processing but we cannot skip a wafer.

What you should know

- How do we adjust the periods of tasks?
- When can we adjust the periods of tasks?
- Period changes are performed, typically, when a new task is added to the system (may need to compress tasks) or when a task is removed (can decompress remaining tasks).
- The choice of which soft real-time model to adopt depends on the application and the expected behaviour.

Tasks with variable execution times

Ref: Algorithms for scheduling imprecise computations. Liu, et al. IEEE Computer, vol. 25, no. 5, May 1991.

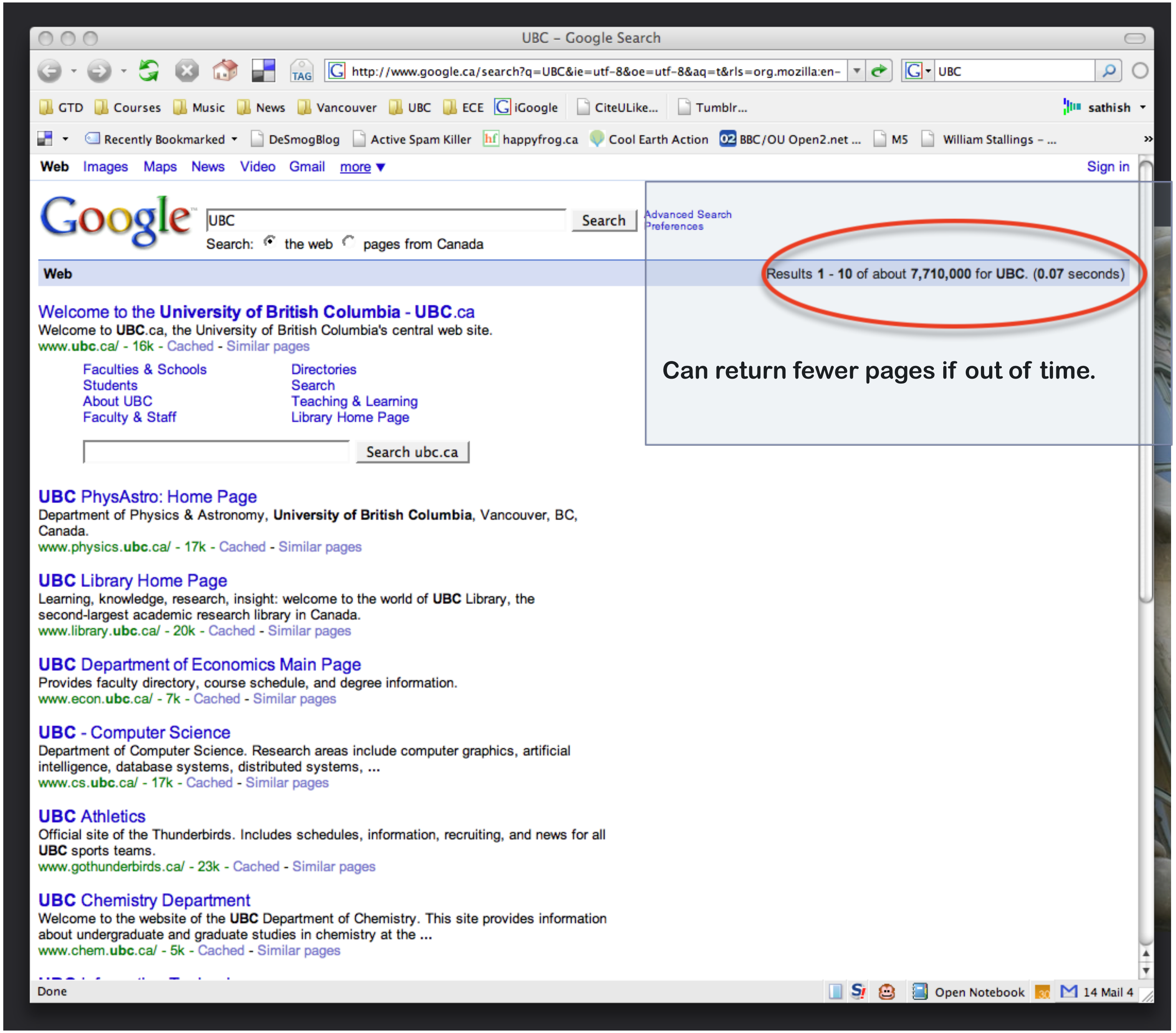
Lecture overview

- **Elastic scheduling allows us to adjust task periods at times of overload**

- **In this lecture, we will examine a third approach**
  - <u>Imprecise computation</u>, which trades accuracy of computation for schedulability
  - Assumes that the accuracy of computation is related to the execution time allotted to the task

Why is the imprecise computation model useful?

- The case for imprecise computations
- For specific applications, approximate results may suffice
  - Image processing (fuzzy frames)
  - Object tracking (location estimates rather than accurate location)
  - Artificial intelligence algorithms typically perform a search (shorter search time results in a lower quality result)
    - Google's search is not a bad example



Results **1** - **10** of about **7,710,000** for **UBC**. (**0.07** seconds)

Can return fewer pages if out of time.

Welcome to the **University of British Columbia** - **UBC**.ca
Welcome to **UBC**.ca, the University of British Columbia's central web site.
www.**ubc**.ca/ - 16k - Cached - Similar pages

- Faculties & Schools
- Students
- About UBC
- Faculty & Staff
- Directories
- Search
- Teaching & Learning
- Library Home Page

UBC PhysAstro: Home Page
Department of Physics & Astronomy, **University of British Columbia**, Vancouver, BC, Canada.
www.physics.**ubc**.ca/ - 17k - Cached - Similar pages

UBC Library Home Page
Learning, knowledge, research, insight: welcome to the world of **UBC** Library, the second-largest academic research library in Canada.
www.library.**ubc**.ca/ - 20k - Cached - Similar pages

UBC Department of Economics Main Page
Provides faculty directory, course schedule, and degree information.
www.econ.**ubc**.ca/ - 7k - Cached - Similar pages

UBC - Computer Science
Department of Computer Science. Research areas include computer graphics, artificial intelligence, database systems, distributed systems, **...**
www.cs.**ubc**.ca/ - 17k - Cached - Similar pages

UBC Athletics
Official site of the Thunderbirds. Includes schedules, information, recruiting, and news for all **UBC** sports teams.
www.gothunderbirds.ca/ - 23k - Cached - Similar pages

UBC Chemistry Department
Welcome to the website of the **UBC** Department of Chemistry. This site provides information about undergraduate and graduate studies in chemistry at the **...**
www.chem.**ubc**.ca/ - 5k - Cached - Similar pages

How do we achieve the imprecise computation model?

- **We assume that tasks are iterative**
  - The number of iterations suggests quality (fewer iterations imply lower quality)
  - Terminate the task after a few iterations with acceptable quality
  - Tasks may have a mandatory part
    - **Any computation beyond this mandatory portion improves the quality but the task is meaningless without the mandatory portion**

More examples

- Radar tracking: Get estimated target locations in a timely fashion rather than accurate information that is too late to be of use

- Multimedia systems: Transmit a low quality image in time rather than missing the deadline, e.g., to meet the 24 fps requirement

- Control systems: Produce an approximate result by a control law as long as the controlled system, e.g., cruise control system, remains stable

Scheduling imprecise tasks

- Tasks are periodic with known period (P_i) and an execution time range [$e_{i,min}$, $e_{i,max}$]
- $e_{i,min}$ represents the mandatory execution required by each task
- The accuracy of a task is highest when each job executes for $e_{i,max}$ time units

- First step
  - Ensure that the mandatory portions of all tasks are schedulable
  - If tasks are scheduled with rate monotonic priorities, we can use the Liu & Layland bound

$$\sum_{i=1}^{n} \frac{e_{i,min}}{P_i} \leq n(2^{1/n} - 1)$$

Example task set

Task	E_{min}	E_{max}	P
T1	2	7	10
T2	4	8	25
T3	6	10	30

$$\frac{2}{10} + \frac{4}{25} + \frac{6}{30} = 0.56 < 3(2^{1/3} - 1)$$

The mandatory portion of this task set is schedulable.

Scheduling imprecise tasks

- **Once we have ensured that the mandatory portions are schedulable, we have many options**
- **The loss in accuracy for each task can be specified by some error function *F***
  - Let e_i be the execution time of T_i then the error is some function of e_i and $e_{i,max}$
- **We could decide to minimize the (weighted) sum of errors**
  - How do we find the values for $\{e_i\}$ that minimize the error?

Objective function

$$\min \sum_{i=1}^{n} w_i F(e_i, e_{i,max})$$

Relative importance of the tasks

Subject to constraints

Linear program formulation

$$\begin{aligned} e_i &\geq e_{i,min}, \forall i \\ e_i &\leq e_{i,max}, \forall i \end{aligned}$$

Constraints on execution time

$$\sum_{i=1}^{n} \frac{e_i}{P_i} \leq n(2^{1/n} - 1)$$

Schedulability constraint

A simple error function

- $F(e_{i,max}, e_i) = e_{i,max} - e_i$ **is a simple error function**

- **Let us also assume that the weight of each task is 1.**

$$
\begin{aligned}
&\min \textstyle\sum_{i=1}^{n} (e_{i,max} - e_i) \\
&\text{subject to} \\
&e_i \geq e_{i,min}, \forall i \\
&e_i \leq e_{i,max}, \forall i \\
&\textstyle\sum_{i=1}^{n} \frac{e_i}{P_i} \leq n(2^{1/n} - 1)
\end{aligned}
$$

A simple linear program

Solving the linear program

$$\min \sum_{i=1}^{n} (e_{i,max} - e_i)$$
subject to
$$e_i \geq e_{i,min}, \forall i$$
$$e_i \leq e_{i,max}, \forall i$$
$$\sum_{i=1}^{n} \frac{e_i}{P_i} \leq n(2^{1/n} - 1)$$

Task	E_{min}	E_{max}	P
T1	2	7	10
T2	4	8	25
T3	6	10	30

Greedy algorithm

- Intuition: increasing the execution time of a task by x decreases error by x.
- The task with the largest period has the least utilization penalty.
  - Determine $1/P_i$ for each task. This is the utilization penalty for increasing the execution time of T_i by 1 time unit.
  - Simplifying assumption: execution times are integers (or can be represented as integers).
- Increase the execution time of the task with the smallest utilization penalty to the maximum extent possible.
- Then move to the task with the next smallest utilization penalty.
- Repeat until the utilization bound is reached or no further progress is possible.

Solving the linear program

Task	E_{min}	E_{max}	P
T1	2	7	10
T2	4	8	25
T3	6	10	30

- **The mandatory utilization is 0.56. The Liu & Layland bound for 3 tasks is 0.7797.**
- **The utilization penalty for T3 is 1/30=0.0333, for T2 is 1/25=0.04, for T1 is 1/10=0.1.**

- **We can increase the execution time of T3 by 4 units at a cost of 4 x 0.0333 = 0.1333**
  - The total utilization now becomes 0.6933.
- **We can increase the execution time of T2 by 2 units at a cost of 2 x 0.04 = 0.08**
  - The total utilization now becomes 0.7733.
- **We cannot make any further increases without violating the utilization bound.**
  - Thus we stop.
- ***An optimal solution is e_1=2, e_2=6, e_3=10.***

Highlights

- We examined another task model for providing good QoS for soft real-time systems.

- The <u>imprecise computation model</u> trades off execution time for accuracy.

- Different applications need different approaches to obtaining good *quality of service.*

- For each task parameter, we have studied some mechanism by which they can be controlled and the entire system behaves in a “predictable” manner.

What you should know

- **What is the imprecise computation model?**
- **Why, and where, is it useful?**
- **How do you decide execution times under this model?**
  - Solve for the simple case of linear error function.

- **Ponder: Can we use elastic scheduling when tasks have variable execution times? What should the parameters for such task sets be?**